FERRAMENTA DE GESTÃO DE REDES MESH

Pedro Nuno da Costa Rodrigues

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
2012

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Pedro Nuno da Costa Rodrigues, Nº 1040891, 1040891@isep.ipp.pt
Orientação científica: Prof. Dr. Jorge Botelho Mamede, jbm@isep.ipp.pt
Empresa: Nexttoyou – Network Solutions, www.nexttoyou.pt
Supervisão: Eng. Bruno Oliveira, boliveira@nexttoyou.pt

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
16 de novembro de 2012

Para ti Júlia, que acabaste de chegar!
Sê bem-vinda, um novo mundo te espera!

Agradecimentos

Antes de mais quero agradecer a todos o ter chegado a esta página. É o culminar de muitos anos seguidos a trabalhar e a estudar. Toda a gente que passou pela minha vida, e que de algum modo partilhou algum momento comigo, sabe que estará sempre presente na minha memória.

Agradeço em especial ao Prof. Doutor Jorge Mamede, que me proporcionou o desenvolvimento de um trabalho que me fez crescer enquanto engenheiro, na medida, que me fez procurar e ir mais além do que pensava ser possível. Pela sua orientação e disponibilidade, ao longo deste trabalho, mas também como professor pela sua competência, bom humor e capacidade de relacionamento.

Na empresa NextToYou agradeço em especial ao Eng. Bruno Oliveira, pela sua disponibilidade, pelas informações que disponibilizou e pela orientação necessária para a concretização do projeto.

No ISEP, quero agradecer a todos os meus professores e professoras que tanto na licenciatura como agora no mestrado sempre mostraram a sua competência e dedicação. A todos sem exceção apresento o meu agradecimento.

Agradeço aos meus colegas de curso, que tantas vezes me ajudaram. Eles também sabem o quão difícil é chegar ao fim do mestrado. Não me esqueço dos momentos hilariantes que passamos juntos. Espero que todos alcancem os seus sonhos. Agradeço ao departamento de Engenharia Eletrotécnica, especialmente ao Eng. Mário Felgueiras.

Em particular apresento o meu agradecimento ao fundador/administrador da rede Mvnet - Wireless de Moitas Venda - Cmsv, na realização deste trabalho, que me prestou uma preciosa ajuda quer quanto à partilha de informação, quer quanto à sua disponibilização para me esclarecer e informar.

Fica também aqui o meu agradecimento aos meus colegas de trabalho profissional, os que passaram e os que atualmente passam tantas e tantas horas a meu lado, todos os dias, com quem partilho uma boa relação de companheirismo.

Não posso esquecer os meus amigos e amigas, que estiveram sempre presentes ao longo da minha vida. Sem eles isto não tinha a mesma piada. Espero finalmente poder recuperar algum do tempo em que estive ausente da vossa companhia.

Quero agradecer também à minha família. À minha mãe sempre incansável, na procura do meu bem-estar. Ela sabe melhor do que ninguém o que custou fazer o caminho até chegar aqui. A ela lhe devo e agradeço do fundo do meu coração tudo o que fez e faz por mim.

Dedico também esta página ao meu pai, que já não se encontra entre nós e que estará bastante orgulhoso, onde quer que esteja.

Não posso esquecer a Célia, o Neca e o Hugo, minha irmã, meu cunhado e meu sobrinho, que estão sempre presentes na minha vida. Esta página também é vossa.

Ao Sr. José e à D. Glória, que sempre me acarinharam e de quem eu gosto muito.

À Zita e ao Rui, que este ano nos apresentaram a Clarinha! Também vos quero muito bem.

Por fim, à minha esposa Lina, que passou horas infindáveis à espera que eu chegasse do ISEP, que abdicou do tempo dela aos fins de semana para que eu estudasse, que tomou conta da nossa querida bebé Júlia de noite e de dia, para que eu conseguisse chegar a esta página. Sem ela nada faria sentido. Ainda bem que te encontrei na minha vida!

A todos, o meu OBRIGADO!

Resumo

O acesso às novas tecnologias usadas em edifícios de habitação (acesso à *internet*, comunicações, videovigilância, videoporteiro, gestão de condomínios, domótica, etc.) implica o uso da tecnologia IEEE 802.11, uma vez que os dispositivos usados atualmente, desde computadores portáteis a *smartphones*, estão equipados com *interfaces* 802.11. Devido à limitação de alcance de rádio da tecnologia 802.11 tem surgido novas soluções para aumentar a sua cobertura. As redes sem fios *mesh – Wireless Mesh Networks* (WMNs) são consideradas soluções flexíveis e económicas adequadas para serem usadas no contexto acima descrito.

O objectivo deste trabalho é criar uma ferramenta que possa ser integrada na solução *GateBox*, um dos produtos desenvolvidos e comercializados pela empresa *NextToYou*. Esta ferramenta de gestão deve permitir ao administrador da rede o controlo e a consulta do estado de uma rede *mesh* em tempo real.

O trabalho desenvolvido teve como base uma pesquisa quanto aos protocolos de *routing* e equipamentos utilizados, em soluções idênticas já existentes. Com base nesse estudo foi criada uma ferramenta de gestão para redes *mesh* que utilizem o protocolo de *routing Better Approach To Mobile Adhoc Networking advanced* (BATMAN-ADV).

A ferramenta desenvolvida permite a visualização gráfica e textual da localização, caracterização e estado das redes *mesh*. Permite também a configuração em tempo real das redes e dos seus nós, assim como permite a monitorização automática das redes e emite notificações de alerta, em caso de deficiências de desempenho.

Foram realizados testes à ferramenta de gestão desenvolvida que mostraram que a mesma responde com rapidez às solicitações efetuadas, assim como se revelou eficaz quanto à monitorização automática das redes. Devido à ferramenta de gestão desenvolvida, o produto *GateBox*, passa agora a estar habilitado a implementar redes *mesh* em edifícios de habitação, podendo controlar e monitorizar o funcionamento dessas redes em tempo real.

Palavras-Chave

Mesh, Ad-Hoc, BATMAN-ADV, servidor, *gateway*, cliente, nó, protocolo, *routing*, *wireless*.

Abstract

Access to new technologies used in residential buildings (internet access, communications, video surveillance, video intercom, property management, home automation, etc.) involves the use of IEEE 802.11 technology, once the devices used today, from laptops to smartphones, are equipped with 802.11 interfaces. Due to limited radio range of 802.11 technology has emerged new solutions to enhance its coverage. The Wireless Mesh Networks (WMNs) are considered appropriate economic and flexible solutions to be used in the context described above.

The aim of this work is to create a tool that can be integrated into the *GateBox* solution, a product developed and marketed by the company *NextToYou*. This management tool should allow the network administrator to control and query the status of a mesh network in real time.

The work was based on a survey regarding routing protocols and equipment used in existing identical solutions. Based on this study was created a management tool for mesh networks using the routing protocol BATMAN-ADV.

The developed tool provides a graphic and textual display of location, characterization and status of mesh networks. It also allows real-time configuration of networks and their nodes, as well as allows the automatic monitoring of networks and sends alert notifications in case of performance deficiencies.

Tests were conducted at management tool developed which showed that it responds quickly to requests made, and proved effective in the automatic monitoring of networks. Due to management tool developed, *GateBox* solution will now be able to deploy mesh networks in residential buildings, being able to control and monitor the operation of these networks in real time.

Keywords

Mesh, Ad-Hoc, BATMAN-ADV, server, gateway, client, node, protocol, routing, wireless.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

3G	–	3rd Generation
AES	–	Advanced Encryption Standard
AJAX	–	Asynchronous Javascript And Xml
AODV	–	Ad-hoc On-demand Distance Vector
AP	–	Access Point
API	–	Application Programming Interface
ARM	–	Acorn RISC Machine, posteriormente Advanced RISC Machine
ARP	–	Address Resolution Protocol
BATMAN	–	Better Approach To Mobile Adhoc Networking
BATMAN-ADV	–	Better Approach To Mobile Adhoc Networking - ADVanced
BGP	–	Border Gateway Protocol
BMX	–	Better approach to Mobile adhoc networking – eXperimental
BSSID	–	Basic Service Set IDentification
CFE	–	Common Firmware Environment
CPU	–	Central Processing Unit
DHCP	–	Dynamic Host Configuration Protocol
DNS	–	Domain Name System
DSDV	–	Destination-Sequenced Distance-Vector routing

DSL – Digital Subscriber Line

EIGRP – Enhanced Interior Gateway Routing Protocol

ESS – Extended Service Set

ETX – Expected Transmission Count

FSR – Fish-eye State Routing

FXS – Foreign eXchange Subscriber

FWCN – Free Wireless Community Network

GPL – General Public License

GPS – Global Positioning System

HNA – Host Neighbour Announcement

HSLS – Hazy-Sighted Link State

HTTP – HyperText Transfer Protocol

HWMP – Hybrid Wireless Mesh Protocol

ICMP – Internet Control Message Protocol

IP – Internet Protocol

ISEP – Instituto Superior de Engenharia do Porto

ISP – Internet Service Provider

LAN – Local Area Network

LED – Light Emitting Diode

MAC – Media Access Control

MAN – Metropolitan Area Network

MANET – Mobile Ad-hoc NETwork

MBSS – Mesh Basic Service Set

MC – Mesh Client

mDNS – Multicast DNS

MIMO – Multiple Input and Multiple Output

MIPS – Microprocessor without Interlocked Pipeline Stages

MMR – Mobile Multi-hop Relay

MPM – Multi-Point to Multi-Point

MPR – MultiPoint Relay

MTU – Maximum Transmission Unit

NTP – Network Time Protocol

OFLSR – Optimized Fish-eye Link State Routing

OGM – OriGinator Message

OLSR – Optimized Link State Routing protocol

OSI – Open Systems Interconnection

OSPF – Open Shortest Path First

p2p – Peer to Peer

PAN – wireless Personal Area Network

PDA – Personal Digital Assistant

PMT – Point to Multi-Point

PSK – Pre-Shared Key

PTP – Point to Point

QoS – Quality of Service

RAM – Random Access Memory

RREP – Route REPlies

RREQ – Route REQuests

RERR – Route ERRors

ROBIN – Routing Olsr Batman INside

RSL – Received Signal Level

SDK – Software Development Kit

SHA – Secure Hash Algorithm

SHWMP – Secured Hybrid Wireless Mesh Protocol

SSH – Secure SHell

SSID – Service Set IDentifier

TCP – Transmission Control Protocol

TFTP – Trivial File Transfer Protocol

TKIP – Temporal Key Integrity Protocol

TTL – Transistor-Transistor Logic

UCI – Unified Configuration Interface

UDP – User Datagram Protocol

USB – Universal Serial Bus

VLAN – Virtual Local Area Networks

VoIP – Voice over Internet Protocol

WAN – Wide Area Network

WDS – Wireless Distribution System

WEP – Wired Equivalent Privacy

Wi-Fi – Wireless Fidelity

WIMAX – Worldwide Interoperability for Microwave Access

WMN – Wireless Mesh Network

WMR – Wireless Mesh Router

WPA – Wi-fi Protected Access

WRM – Wave Radio Module

WSMN – Wireless Sensor Mesh Network

1. INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

A crescente utilização de novas tecnologias de comunicações sem fios e de computação móvel em edifícios de habitação e a crescente integração de diferentes valências nessas tecnologias (acesso à *internet*, comunicações, videovigilância, videoporteiro, gestão de condomínios, domótica, etc.) implica que sejam encontradas soluções integradas ajustadas às necessidades de quem as utiliza. Por vezes essas tecnologias só podem ser implementadas através da norma IEEE 802.11, também conhecida como Wi-Fi (*Wireless Fidelity*). Isso deve-se à impossibilidade de passagem de cabos através das paredes ou estruturas, sem as danificar ou porque se pretende uma eliminação da utilização de cabos por motivos estéticos ou apenas por simplificação da instalação. As infraestruturas Wi-Fi têm vindo a ser desenvolvidas um pouco por todo o mundo, devido à integração de Wi-Fi nos dispositivos portáteis desde *laptops* aos *smartphones*. No entanto as redes que utilizam Wi-Fi têm um alcance de rádio limitado. A cobertura de uma área geográfica, por exemplo num edíficio ou num ambiente residencial, implica a utilização de vários pontos de acesso – *Access Points* (APs), que por sua vez têm de estar ligados à rede central através de uma ligação por cabos *Ethernet*. Em alguns casos poderá ser necessário a instalação de um grande número de APs e de cablagem, o que implica um maior grau de complexidade, assim como custos financeiros acrescidos. Poderá também implicar um número maior de recursos técnicos e humanos para o desenvolvimento das redes. Outro efeito da necessidade da ligação de APs à infraestrutura cablada consiste na limitação da flexibilidade das redes.

A norma 802.11s especifica as redes sem fios *mesh – Wireless Mesh Networks* (WMNs). Esta solução é considerada uma solução flexível e económica adequada para ser usada no contexto acima descrito. Nas WMNs os nós colaboram uns com os outros de maneira que os dados possam ser transmitidos, sem a utilização de fios, desde a origem até ao destino. Estas redes autoconfiguram-se e auto-organizam-se de maneira a oferecerem os serviços disponibilizados da maneira mais eficiente e com o menor esforço de configuração por parte dos utilizadores, sendo adequadas para o uso em ambiente residencial.

O presente trabalho surge da necessidade da empresa *NextToYou*, integrar nos seus produtos a possibilidade de implementação e gestão de uma rede *mesh*.

Para isso é necessário proceder à implementação de uma ferramenta que permita controlar e verificar o estado de uma rede *mesh* em tempo real. Esta ferramenta será incluída num dos produtos desenvolvidos e comercializados pela empresa, a *GateBox*.

GateBox é um sistema de integração de serviços e de gestão de comunicações em edifícios residenciais ou empresariais, que proporciona, entre outras, as seguintes funcionalidades [1]:

- Sistema de videovigilância integrado;
- Sistema integrado de videoporteiro com gravação de mensagens;
- Central telefónica digital interna no edifício, ou equiparado;
- Serviços de rede comunitários (ex. página *Web* de condomínio, partilha de documentos, etc.);
- Acesso interno aos serviços em qualquer divisão da habitação;
- Acesso sem fios aos serviços nas áreas comuns (ex. jardim, piscina, sala de reuniões, etc.);
- Acesso externo aos serviços através de *Internet*.

A *NextToYou* é uma empresa de Telecomunicações que desenvolve um conjunto de sistemas modulares que integram diferentes tipos de soluções: comunicações, multimédia, segurança e gestão de edifícios.

1.2. OBJETIVOS

O objetivo principal deste projeto é criar uma ferramenta de gestão de redes *mesh* que possa ser instalada nas unidades *GateBoxes*. Para isso é necessário fazer o estudo e

comparação das tecnologias inerentes às redes *mesh*, das ferramentas de gestão utilizadas nessas redes, assim como proceder à implementação de uma rede *mesh*, que permita a realização de testes. A rede *mesh* criada deverá oferecer acesso à *internet* e a ferramenta de gestão deve possibilitar acesso a todos os recursos da rede, permitir a configuração desses recursos em tempo real, assim como enviar avisos e alarmes quando seja detetado alguma anomalia no funcionamento da rede. Dada a complexidade inerente a este objetivo, sentiu-se a necessidade de o subdividir em múltiplas tarefas de realização mais simples, tais como:

- Investigação, estudo e análise comparativa de redes *mesh*, suas arquiteturas e ferramentas de gestão utilizadas;
- Investigação estudo e análise comparativa de protocolos de *routing* utilizados;
- Investigação, estudo e conceção do *firmware* a utilizar;
- Instalação de *firmware* e configuração dos equipamentos utilizados;
- Identificação e caracterização dos requisitos funcionais da ferramenta de gestão;
- Implementação e desenvolvimento de um protótipo da ferramenta de gestão;
- Demonstração do funcionamento da ferramenta de gestão, testes e análise de resultados.

1.3. ORGANIZAÇÃO DO RELATÓRIO

Este documento encontra-se estruturado em 8 capítulos:

- O capítulo 1 introduz este trabalho.
- O capítulo 2 apresenta as redes *mesh*, arquiteturas e ferramentas de gestão utilizadas.
- No capítulo 3 serão apresentados e comparados os protocolos de *routing* usados nas redes *mesh*.
- O capítulo 4 descreve a arquitetura e funcionamento do protocolo *Better Approach To Mobile Adhoc Networking advanced* (BATMAN-ADV).
- No capítulo 5 é descrito o processo de criação, instalação, implementação e configuração do *firmware* usado, assim como são descritas considerações sobre o funcionamento da rede.
- No capítulo 6 é apresentada a solução proposta para a gestão da rede *mesh* criada, sua arquitetura e o funcionamento da ferramenta de gestão.
- No capítulo 7 são apresentados os testes e demonstrações efetuados.

- Por fim, no capítulo 8 são reunidas as principais conclusões e perspetivados futuros desenvolvimentos.

2. REDES MESH

2.1. REDES FLEXÍVEIS E DESCENTRALIZADAS

Estruturas descentralizadas poderão ser úteis em redes inspiradas na norma de redes sem fios IEEE 802.11, mais conhecidas como Wi-Fi [2]. Ao contrário da maioria das redes por cabo, as redes Wi-Fi estão constantemente a mudar a sua topologia. Qualquer agente externo (um edifício, uma árvore, chuva ou mesmo um pássaro) poderá mudar as suas propriedades. Se o administrador da rede atribuir o controlo dessa mesma rede apenas a um ou mais nós (*Masters*), e se algum agente externo causar uma mudança da topologia, um desses nós poderá ser afetado e consecutivamente o desempenho da rede. É por isso que uma estrutura não centralizada poderá ser útil numa rede Wi-Fi em ambiente exterior. O mesmo se aplica numa rede existente num ambiente interior, caso algum objeto ou estrutura se encontre entre dois nós, impossibilitando a sua ligação. Outra importante vantagem que carateriza uma rede *mesh* é a liberdade. A liberdade de se poder participar na rede. A liberdade de não ser um nó *Master*, nó esse que determina a prioridade e discriminação do tráfego que atravessa a rede.

2.1.1. MESH NETWORK E MOBILE AD-HOC NETWORK (MANET)

Uma rede *mesh* é uma rede onde todos os nós participantes são *routers*, o que quer dizer que todos eles aceitam e reenviam pacotes dos outros nós de acordo com as regras de

encaminhamento – *routing*. Graças a esta propriedade todos os nós têm que estar conectados a pelo menos uma ligação, sendo esta a única restrição relativamente à topologia da rede. Assim, para que qualquer participante, neste caso um *router*, se possa juntar à rede é necessário um sistema de coordenação. É neste contexto que surgem os protocolos de *routing*, protocolos que devem ser dinâmicos e que se tornam importantes na comunicação entre *routers*.

Quando se fala de redes descentralizadas e flexíveis, na realidade está-se a falar de redes MANET (Mobile Ad-Hoc NETwork). MANET são redes com propridades *mesh* que utilizam o modo de ligação Ad-Hoc 802.11, permitindo que os nós possam estar em movimento e possam ligar-se/desligar-se frequentemente sem qualquer problema.

2.2. CLASSIFICAÇÃO DAS REDES MESH – *WIRELESS MESH NETWORKS*

As redes sem fio podem ser classificadas conforme a conectividade dos seus elementos de rede [3], em redes *Point to Point* (PTP), *Point to Multipoint* (PMT) ou *Mutipoint to Multipoint* (MPM). Redes PTP funcionam bem, embora não sejam escaláveis e o seu nível de adaptabilidade seja baixo [4]. Redes PMT são moderadamente escaláveis, mas têm baixa adaptabilidade, como por exemplo *Worldwide Interoperability for Microwave Access* (WIMAX) ou *Bluetooth*. De maneira a ultrapassar estas limitações, redes MPM apresentam algumas características que permitem um melhor funcionamento que as redes anteriores, assim como uma maior adaptabilidade e escalabilidade para acomodar um grande número de utilizadores. De referir que conforme se aumentam os números de nós na rede, diminui a potência de transmissão que cada nó necessita para transmitir.

2.2.1. VANTAGENS DAS REDES *MESH*

Auto-organização e autoconfiguração: as WMNs são flexíveis quanto à arquitetura, ou seja, conseguem reorganizar-se e configurar-se por elas próprias. Estas características reduzem o tempo de *set-up*, assim como os custos de manutenção. Devido a estas características a rede é vocacionada para se alterar, expandir e adaptar, conforme as necessidades dos utilizadores finais.

Baixo custo de desenvolvimento: Os *routers mesh* são *routers* sem fios e têm a aptidão de servir em ambientes com muitos *hops* (saltos). Assim, o uso de *wireless routers* em grandes áreas é mais barato quando comparado com *routers* AP de *single hop*, que utilizam

conexões de cabo. Tipicamente as conexões de cabo são mais caras de instalar e manter, o que faz com que o desenvolvimento de WMNs seja mais rápido e mais fácil com um custo de operação menor.

Aumento da fiabilidade: Numa WMN existem múltiplos caminhos desde os nós de origem até aos nós de destino, o que permite caminhos alternativos em casa de falha. Por exemplo, em caso de congestionamento numa área da rede, poderá ser escolhido um caminho diferente. Esta característica é usada para equilibrar o tráfego na rede, o que permite uma diminuição dos engarrafamentos de tráfego, bem como o aumento do desempenho e da fiabilidade das WMNs.

Escalabilidade: Enquanto que nas redes tradicionais sem fios, quando o número de nós dessas redes aumenta, o desempenho é afetado negativamente, nas WMNs o aumento de nós aumenta a capacidade de melhor balanceamento de carga e o número de rotas alternativas. Os pacotes gerados localmente poderão chegar mais depressa ao seu destino do que pacotes gerados um ou dois saltos à frente. Para isso podem-se utilizar algumas configurações disponíveis nas WMNs através de protocolos que gerem as comunicações.

Interoperabilidade: As WMNs têm uma arquitetura MPM (Multi-Point to Multi-Point) que é compatível com normas existentes tais como WiMAX, *Cellular*, Wi-Fi, *Zigbee*, *Bluetooth*, *Sensor*, MANET, *Vehicular*, etc., o que torna atrativo o incremento do desenvolvimento e a reutilização das WMNs nas estruturas existentes. Todas as tecnologias mencionadas estão aptas, ou estarão aptas brevemente, para poderem ser configuradas como WMNs e comunicar umas com as outras. Os desenvolvimentos necessários para que qualquer tipo de redes comuniquem entre si melhoram as normas atuais e mantêm a interoperabilidade.

2.3. NORMAS DA INDÚSTRIA

IEEE 802.11s Wi-Fi Mesh: Atualmente a família da norma IEEE 802.11 é a mais aceite. *Extended Service Set* (ESS) e *Wireless Distribution System* (WDS) são definidos como IEEE 802.11s, para aplicações de *mesh multi-hop* e para autoconfiguração de caminhos entre *Wave Radio Modules* (WRM) [5]. A norma 802.11s define uma técnica de seleção de caminho (rota) conhecida por ser o seu protocolo de *routing*, que é *Hybrid Wireless Mesh Protocol* (HWMP).

IEEE 802.15.1 Bluetooth: *Bluetooth* é o nome comercial designado para esta norma que é especializada em *wireless Personal Area Networks* (PANs). O *taskgroup* 5 está a definir os mecanismos necessários na camada física e na camada *Media Access Control* (MAC) das redes sem fios PANs (WPANs), para que possam ser usadas como redes *mesh*. Existem duas possibilidades de topologias *mesh* em redes WPANs: topologia total *mesh* e topologia parcial *mesh* [6].Enquanto na topologia total cada nó está diretamente conectado a todos os nós, na topologia parcial, alguns deles estão ligados a todos os outros, enquanto os restantes só estão conectados aos nós que retransmitem dados.

IEEE 802.15.4 Zigbee: *Zigbee* foi inicialmente proposto pela empresa Motorola. Nas WPANs existem dois tipos de ligações, *single-hop* e *multi-hop*. *Zigbee* pode suportar *mesh* definindo um coordenador. Isso significa que esse coordenador é responsável por configurar a topologia da rede de uma forma *multi-hop*. Este método é muito apropriado para *Wireless Sensor Mesh Networks* (WSMNs).

IEEE 802.16 WiMAX: A norma IEEE 802.16j é uma norma para comunicações sem fios para *Metropolitan Area Networks* (MANs) que pretende proporcionar a funcionalidade *Mobile Multi-hop Relay* (MMR). Esta funcionalidade permite utilizar uma topologia *multi-hop* do tipo *mesh* numa rede WIMAX, usando, para isso,estações da própria rede, como estações de retransmissão. Assim uma instalação baseada nesta topologia permite uma eficiente extensão da cobertura da rede, e ao mesmo tempo uma poupança em termos de custos comparativamente com uma instalação de linhas fixas de transmissão. A norma IEEE 802.16j foi aprovada em 2009 como uma emenda à norma IEEE 802.16-2009 [7].

2.4. ARQUITETURA DAS WMNS

O objetivo da arquitetura das WMNs é fornecer acesso à *internet* com uma largura de banda significativa. As WMNs consistem em *Mesh Clients* (MC) e *Wireless Mesh Routers* (WMRs), que retransmitem entre si os pacotes de dados, através de uma arquitetura *multi-hop*. Outros elementos existentes serão as *gateways*, que normalmente estarão na fronteira das redes onde terão acesso a um *Internet Service Provider* (ISP). Cada nó WMR comunicará com outro WMR, que esteja ao seu alcance, o que permitirá que os dados possam percorrer uma longa distância através dos nós intermediários que terão a função de retransmitir a informação recebida. Esses nós, não só retransmitirão o sinal de rádio, como também retransmitirão os pacotes de dados provenientes de nós mais longínquos. Esta

arquitetura permite comunicações contínuas e reconfigura-se no caso de nós intermédios deixarem de existir. Se existir uma grande quantidade de tráfego, será escolhido um caminho alternativo caso ele exista, até que a informação chegue ao seu destino. A infraestrutura que suporta as redes WMNs é constituída pelos nós que fazem parte do *backbone*, sendo que compete a esses nós levar o sinal até às extremidades, estando em constante ligação ao ISP. Os MC podem aceder a esses serviços através dos WMRs. Os nós que fazem parte do *backbone* chamam-se *Internet Gateways*, que oferecem acessibilidade à *internet*. Uma rede típica WMN está ilustrada na Figura 1.

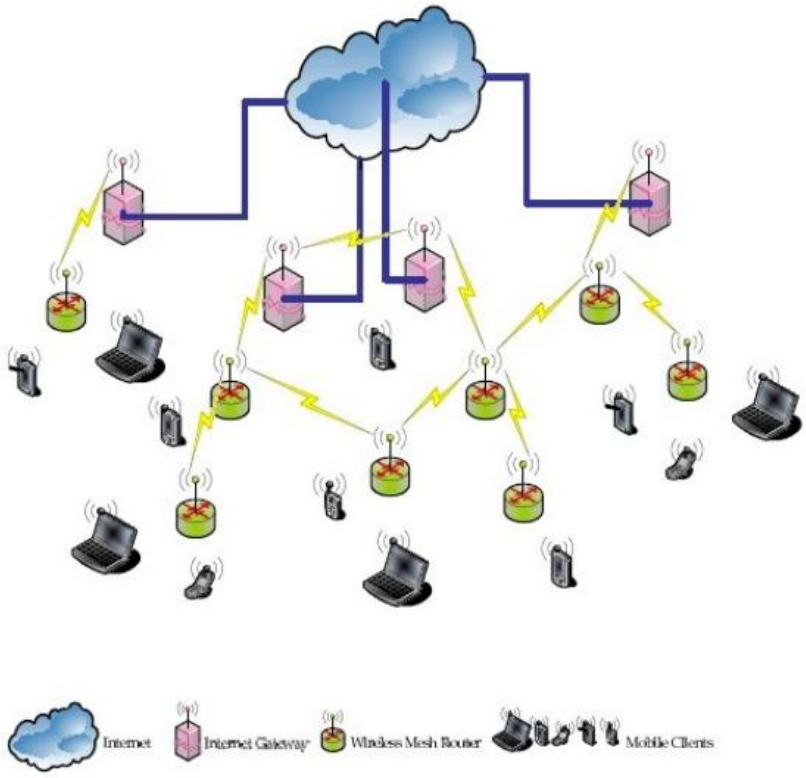


Figura 1 ***Wireless Mesh Network*.**

2.4.1. COMPONENTES DAS WMNS

Existem três tipos de nós numa WMN: cliente, *router* e *gateway*.

Clientes WMN são os utilizadores finais, tais como: *laptops*, *Personal Digital Assistants* (PDAs), *smartphones*, etc., que podem aceder à rede para uso de aplicações como por exemplo *e-mail*, *Voice over Internet Protocol* (VoIP), jogos, deteção de localização – *Global Positioning System* (GPS), etc. Todos estes aparelhos, supostamente são

dispositivos móveis, têm energia limitada, poderão ter capacidades de *routing* e poderão estar ou não sempre ligados à rede.

Routers WMN existem para encaminhar o tráfego existente na rede. As comunicações *multi-hop*, efetuadas pelos *routers*, provocam um baixo consumo de energia para transmissão de dados. Adicionalmente, o protocolo de acesso ao meio (MAC), num *router mesh* suporta múltiplos canais e múltiplas *interfaces*, de maneira a permitir a escalabidade num ambiente *mesh multi-hop.*

Gateways WMN são *routers* com acesso direto à infraestrutura cablada/*internet*. Os *gateways* ao possuírem múltiplas *interfaces* para se poderem ligar, quer às redes com fio, quer às redes sem fios, tornam-se caros. Assim existem um menor número de *gateways* WMN na rede. Além disso, o seu posicionamento na rede tem um impacto direto no desempenho dessa mesma rede.

2.5. APLICAÇÕES DAS WMNS

Existem aplicações que não podem ser suportadas senão através das redes WMNs. Daí o interesse no seu desenvolvimento. A seguir discriminam-se algumas aplicações possíveis.

Acesso *wireless* de banda larga: Hoje em dia o acesso à banda larga é um recurso importante para se aceder à informação. Fornece serviços em tempo real através de aplicações, como: *video telephony*, jogos *on-line*, *video on demand*, telecomunicações, etc. Cada nova aplicação tem um impacto na qualidade de vida, aumenta a produtividade, permite ganhar tempo. Nas áreas urbanas e suburbanas a escolha do acesso à banda larga é feita através de acesso por cablagens (cabo, *Digital Subscriber Line* (DSL) ou fibra ótica). Nas áreas rurais, como têm uma cobertura mais limitada, são usadas outras tecnologias, tais como satélites ou redes celulares. O acesso ao satélite tem dois inconvenientes: a tecnologia é cara e existe uma grande latência devido à distância do satélite. No caso das redes celulares, o custo da instalação de torres de rede é elevado e por vezes difícil. A falta de fornecedores de serviços e o grande custo do próprio serviço implica um menor uso da banda larga nestas zonas rurais.

WMNs permitem uma implementação mais fácil e um custo menor, proporcionando uma alternativa em áreas onde a televisão por cabo, a fibra óptica ou o DSL não estão

disponíveis. Podem ser rapidamente instaladas sem a necessidade de equipamento caro e o fornecedor de serviço poderá ter um rápido retorno do investimento. Além disso, devido ao seu baixo custo e fácil operacionalidade o acesso à banda larga aos residentes urbanos é também possível usando WMNs, aliás já existem muitas redes urbanas e tendem a aumentar.

Aplicações industriais: Num edifício existem muitos dispositivos que precisam de ser monitorizados e controlados como por exemplo dispositivos elétricos, incluindo a própria energia do edifício, luz, ar condicionado, elevadores, etc. Atualmente, as comunicações por cabo são a forma usada para essa monitorização e controlo. Isso torna-se muito dispendioso devido à complexidade e manutenção da rede de cabo. As redes Wi-Fi são uma opção para a redução de custos. Mas esta solução ainda não atingiu um desempenho satisfatório devido à cablagem de *Ethernet* que é necessária para os Wi-Fi *Access Points*. A substituição de APs por *routers mesh* resolve o problema. O processo de implementação será mais simples e os custos serão reduzidos.

Cuidados de Saúde: Num hospital ou num centro médico é necessário monitorizar e atualizar a informação dos pacientes como por exemplo o histórico médico, resultados de testes, informação de seguro de saúde, etc., que precisa de ser processada e transmitida de quarto para quarto. A capacidade de se poder ligar à rede é crucial para assegurar o acesso aos dados em cada sala de operações, nos serviços administrativos e nos laboratórios. Em muitos hospitais a transmissão de dados é feita em banda larga, devido às grandes quantidades de dados transmitidos, como por exemplo imagens médicas de alta resolução e monitorização periódica de informação. WMNs permitem o acesso ilimitado a qualquer dispositivo médico. Não necessita da existência de cablagens, eliminando pontos inacessíveis, permitindo um menor custo e simplicidade, que não podem ser oferecidos pelas redes cabladas.

Sistemas de transporte: O acesso à *internet*, utilizando-se as normas IEEE 802.11 e IEEE 802.16, é limitado nas estações e paragens de meios de transporte utilizados pelas pessoas. A tecnologia WMN pode ajudar a estender o acesso em autocarros, aviões, barcos e comboios. Assim, os passageiros a bordo de qualquer meio de transporte poderão aceder à *internet* enquanto viajam de um lugar para outro. Outros serviços como monitorização de veículos ou câmaras de segurança podem ser suportados também.

Hóteis e unidades de alojamento: Em hotéis e *resorts*, um dos serviços que costuma existir é a conectividade gratuita à *internet* de alta velocidade. WMNs são fáceis de instalar, de baixo custo, sem necessidade de alterar estruturas e tanto podem ser utilizadas dentro das instalações como no exterior dessas mesmas instalações.

Armazéns: Uma maneira de fazer controlo do *stock* que existe nos armazéns é usando um *scanner* de mão. É necessária conectividade na área. WMNs asseguram essa conectividade em armazéns modernos, como por exemplo para controlar a logística em armazéns de *shipping*, com menor esforço e custo.

Eventos temporários: Quando decorre uma construção de um edifício, os arquitetos e engenheiros, podem manter-se em contacto utilizando uma câmara e um sistema de intercomunicação. Assim, terão uma imagem real dos avanços do projeto. Outros eventos poderão ser, comícios políticos, mercados de rua, concertos ao ar livre, onde se podem colocar e retirar WMNs em minutos.

2.6. REDES COMUNITÁRIAS *WIRELESS* DE ACESSO LIVRE

Uma rede comunitária é uma rede feita e mantida pelos seus participantes. Ao contrário dos modelos utilizados pelas companhias de telecomunicações mundiais (cujo objetivo é o lucro), cada utilizador segue a filosofia do "faça você mesmo". Através de acordos e de organizações, os participantes são capazes de se ligar via *wireless* aos seus vizinhos e estes ligarem-se a outros vizinhos, e assim por diante. Estas comunidades são normalmente abertas e livres. A maioria destas redes usam a tecnologia Wi-Fi, porque é a maneira mais barata e fácil de desenvolver uma rede ao ar livre, associando protocolos *mesh* e sistemas MANET. Este modelo chamado de *Free Wireless Community Network* (FWCN), é atualmente usado em muitos locais e cresce de dia para dia [2]. No contexto português existem vários projetos, salientando-se o projeto "*mvnet*" implementado na localidade de Moitas Venda [8] que utiliza tecnologia *mesh* e que ambiciona vir a ligar a primeira rede *mesh* do mundo em dois continentes separados, nomeadamente América do Norte e Europa. A enciclopédia (*wiki*) que este projeto disponibiliza é a mais completa e dinâmica na língua portuguesa, no que concerne à tecnologia *mesh*. Alguns projetos de redes livres *wireless* em Portugal localizam-se em Olhão [9], em Torres Novas/Tomar [10], em Arruda dos Vinhos [11], no Vale do Sousa [12] e em Valongo [13].

2.7. PROJECTOS, INICIATIVAS E COMUNIDADES QUE ENVOLVEM SOLUÇÕES *MESH*

Algumas das maiores comunidades que usam WMNs na Europa são Freifunk na Alemanha com mais de 100 redes independentes e Guifi.Net em Espanha, na Catalunha com uma rede que consiste em mais de 12000 nós ativos [14].

A título de exemplo enumera-se alguns projetos e seus objetivos/significado [15]:

- Open Mesh Project – construção de uma rede *mesh* para o Egipto.
- Open Source Mesh – grupo que procura criar *software mesh*, *open-source*, fiável.
- B.A.T.M.A.N. – *Better Approach To Mobile Ad-hoc Networking; routing protocol for multi-hop Ad-Hoc mesh networks.*
- Roofnet – 802.11b/g rede *mesh* em desenvolvimento no MIT CSAIL.
- GNUnet – *framework* para p2p seguro através da rede que não utiliza qualquer serviço centralizado.
- Dot-P2P – sistema DNS aberto, livre e descentralizado.
- SMesh – rede *wireless mesh* sem "cortes" que está a ser desenvolvida em John Hopkins University.
- Coova – *software open source* para controlo de acesso a *captive portal* (UAM) e fornecimento de acesso a 802.1X.
- Babel – protocolo *loop-free distance-vector routing* para IPv6 & IPv4.
- SolarMESH – rede *wireless Local Area Network* (LAN) *mesh* IEEE 802.11 alimentada a energia solar e solução de retransmissão *infrastructure.*
- WING – rede *wireless mesh network* para a próxima geração de *internet*; parcialmente produzida em cima de *Roofnet*.
- Daihinia – ferramenta para Wi-Fi; transforma uma simples rede Ad-Hoc numa rede Ad-Hoc *multi-hop*.
- P2P DNS – construção de um sistema DNS p2p.
- Digitata.org – desenvolvimento de uma infraestrutura de baixo custo (terminais de *internet* de banda baixa) para permitir o acesso básico à *internet*, às crianças de países de África.
- Netsukuku – rede Ad-Hoc que usa conectividade Wi-Fi e usa endereços especificamente construídos, que permitem comunicações entre máquinas sem necessidade do protocolo HTTP.

- Tonika – projeto de rede *open source*; plataforma de administração livre para uma larga escala de redes (social), com robustez, segurança, anonimato, resiliência e desempenho garantidos.

Seguidamente enumeram-se algumas comunidades, cuja filosofia está inerente ao conceito *mesh*:

- We Rebuild - *cluster* de ativistas que se juntaram para colaborar em assuntos relacionados com o acesso livre à *internet*, sem vigilância intrusiva.
- Athens Wireless Metropolitan Network – comunidade *wireless* na Grécia.
- wlan ljubljana (na Eslovénia) – rede *wireless* livre, open Ljubljana.
- Redes comunitárias *Wireless* por região na wikipedia.

2.8. PLATAFORMAS COMERCIAIS DE GESTÃO DE REDES *MESH*

Após pesquisa e investigação das plataformas de gestão existentes para redes *mesh*, chegou-se à conclusão que de uma forma genérica, essas plataformas visam os mesmos objetivos e apresentam um conjunto similiar de funcionalidades base. Essas funcionalidades são:

- A possibilidade de criação de uma rede.
- A associação de nós à rede criada.
- A monitorização dos nós em tempo real.
- A possibilidade de configuração das características da rede e dos nós.
- A apresentação de aspetos de visualização gráfica da localização e estado dos nós, utilizando para isso, aplicações para apresentação de mapas.
- A monitorização automática de problemas existentes na rede, 24 horas por dia, com mecanismos de envio de notificações de alerta para o administrador da rede.
- A criação de registos do histórico da rede, nomeadamente do número de utilizadores, do tráfego de dados, de alterações efetuadas na rede, etc.
- A criação de gráficos de forma a interpretar todo o funcionamento da rede.
- Outras funcionalidades podem ser acrescentadas como por exemplo monitorização de sinal de vídeo e áudio.

Algumas soluções de gestão de redes *mesh* podem ser encontradas tanto em projetos comerciais, como em projetos de código aberto (*open source*). Alguns exemplos de

plataformas comerciais são: *Cloudtrax* [16] e *MeshConnect* [17], que fazem parte do projeto *Open-Mesh* [18], a plataforma *Wifi Mesh* [19], a plataforma da empresa *Nodalis* [20], a plataforma da empresa *Meraki* [21], entre outras.

Na Figura 2 pode-se ver um ecrã da plataforma *Cloudtrax*, onde se pode observar um mapa com a localização de 3 nós, assim como informações sobre o estado desses nós, o nome atribuído aos nós, o número de clientes, o tempo desde a última ligação, informações sobre *gateways*, latência, velocidade ou quebras de serviço.

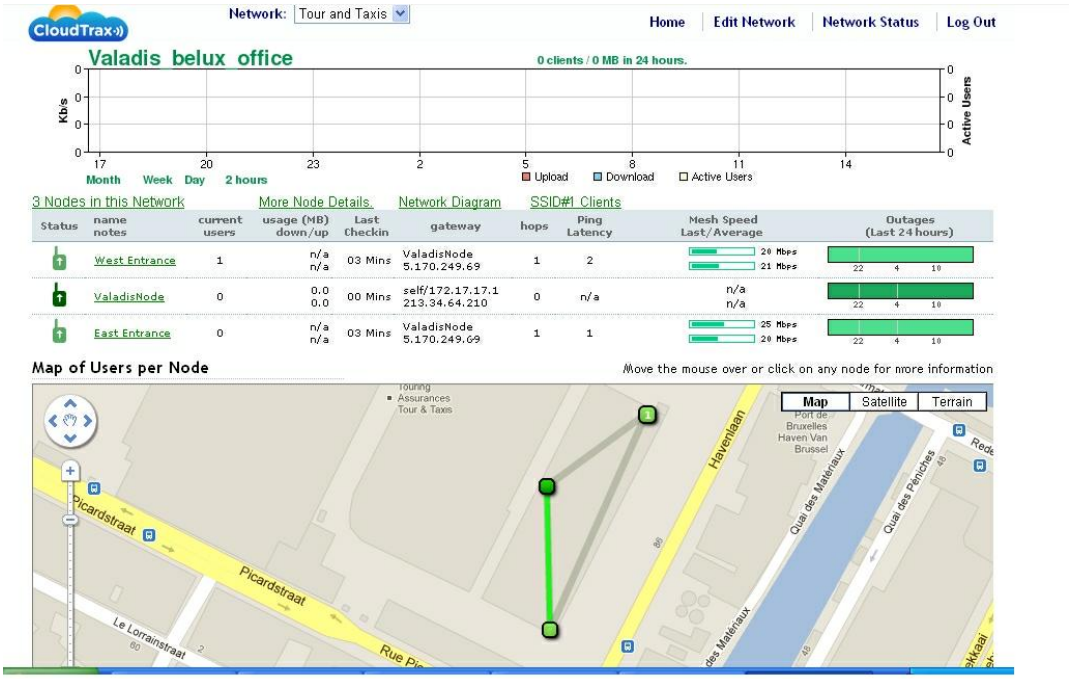


Figura 2 Écrã da plataforma de gestão *Cloudtrax*.

Na Figura 3, apresenta-se um ecrã da plataforma *Wifi Mesh*, onde se pode observar um mapa com a localização de 3 nós, assim como se pode observar que existe outras opções de gestão nos separadores, nomeadamente mapas, nós, clientes ou gráficos. Esta plataforma encontra-se *online*, onde pode ser criada uma rede, assim como podem ser adicionados nós.

A Figura 4 ilustra um ecrã da plataforma de gestão da empresa *Nodalis*, onde se pode observar um mapa com a localização de vários nós, assim como informações sobre o estado desses nós, o nome atribuído aos nós, os IPs, os MACs *addresses*, a distância em metros entre nós, indicador dos níveis da qualidade da ligação e os valores de potência associados aos sinais rádio.

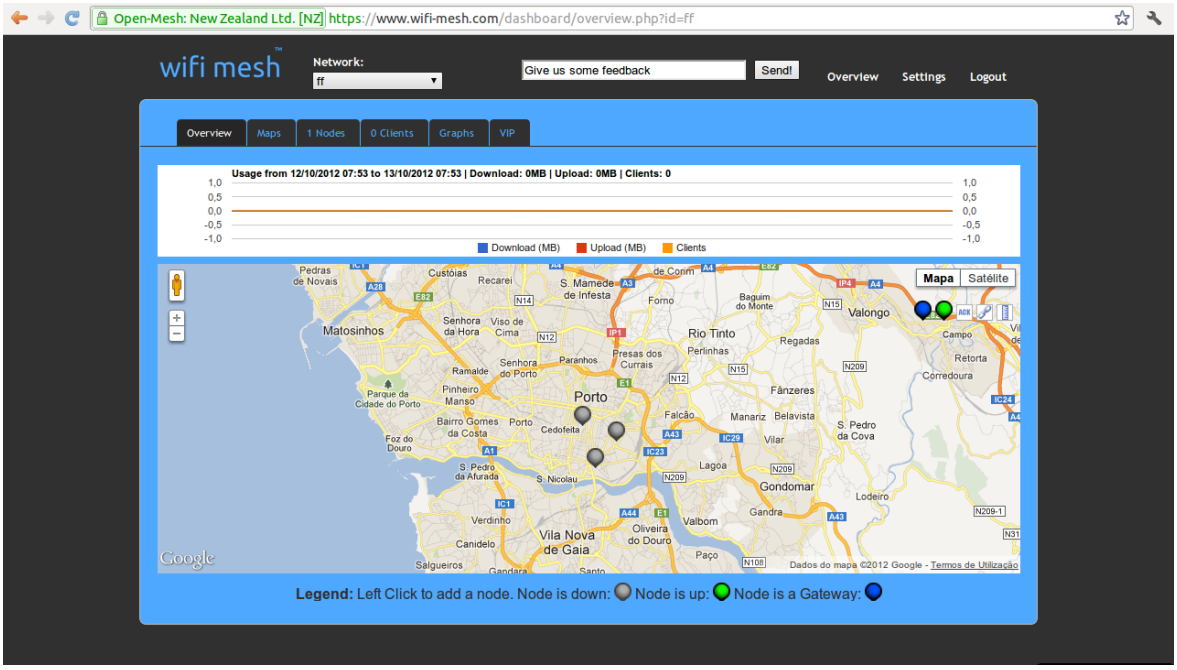


Figura 3 Écrã da plataforma de gestão *Wifi Mesh*.

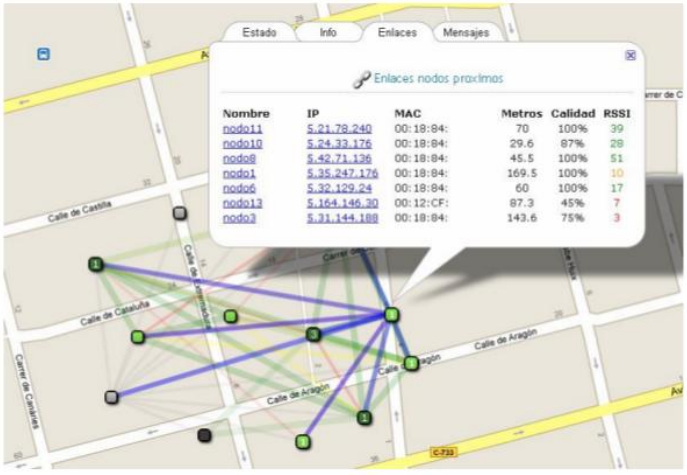


Figura 4 Écrã da plataforma de gestão da empresa *Nodalis*.

Nas Figuras 5 e 6 podem-se visualizar mapas onde são apresentados os números de utilizadores por nó de duas redes *mesh*, sendo para isso utilizadas as plataformas de gestão da empresa *Meraki*. Na Figura 6 pode-se observar a localização de vários nós, assim como informações visuais sobre o estado desses nós, nomeadamente se se tratam de nós *gateways* ou nós repetidores, as áreas de influência do sinal rádio, assim como o número de clientes de cada nó.

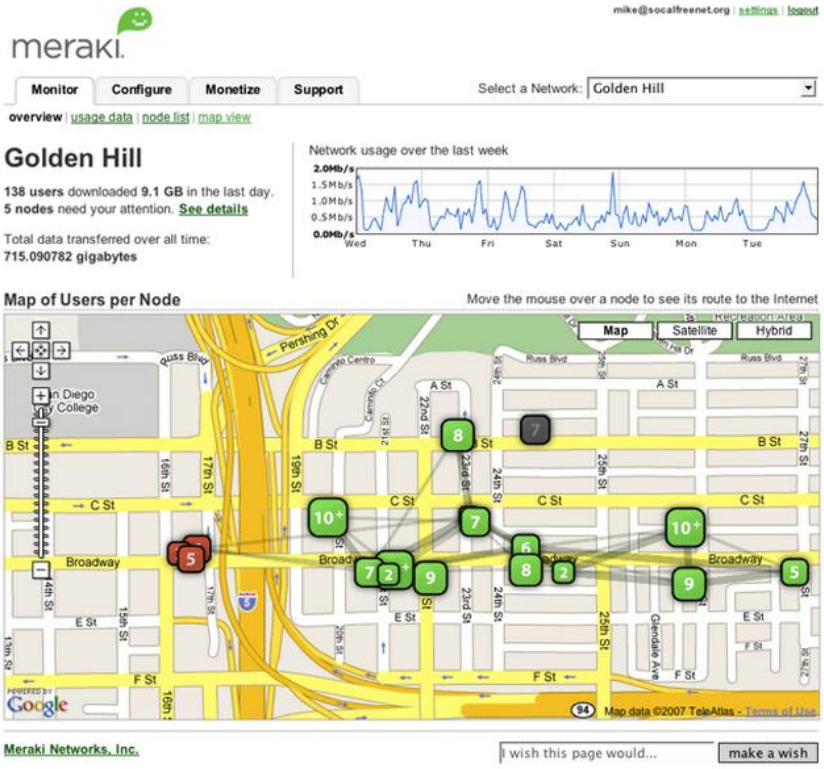


Figura 5 Écrã da plataforma de gestão da empresa *Meraki*.

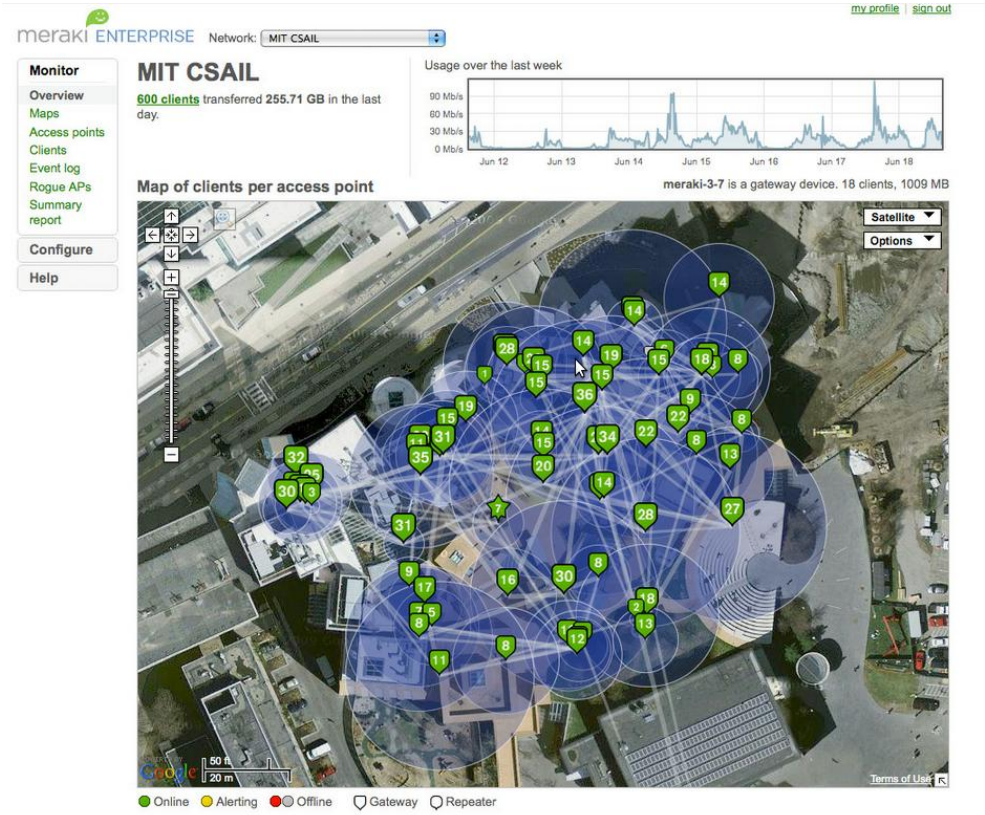


Figura 6 Outro écrã da plataforma de gestão da empresa *Meraki*.

2.9. PLATAFORMA DE GESTÃO DE REDE MESH *OPEN SOURCE* - *SOFTWARE ORANGEMESH*

Orangemesh [22] é uma plataforma de gestão *open source* e foi desenvolvida para a utilização em redes que têm como base de instalação o *firmware* ROBIN (*Routing Olsr Batman INside*), desenvolvido com base no *firmware OpenWRT* "*Kamikaze*" e "*Backfire*", que funciona principalmente em *routers Accton*, *La Fonera*, *Ubiquiti*, *Engenius* e *Wiligear* equipados com *chipsets Atheros*. O *firmware* ROBIN permite a instalação de BATMAN (L3) ou OLSR. Na Figura 7 pode-se observar um ecrã da plataforma de gestão *Orangemesh*.

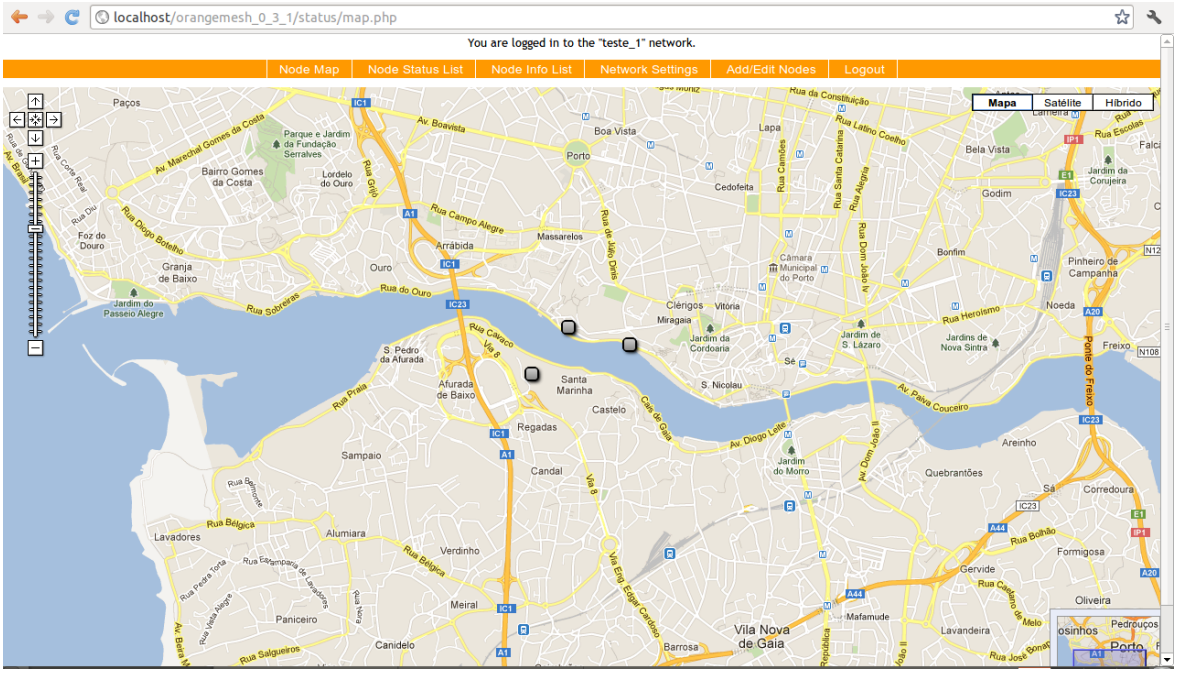


Figura 7 Écrã da plataforma de gestão *open source Orangemesh*.

Orangemesh é um *software* escrito principalmente em PHP, embora existe alguns guiões (*scripts*) escritos em *Javascript*, destinados à elaboração de mapas, usando para isso *Google Maps* API. Também é escrito em *Cascading Style Sheets* (CSS), que serve para estilizar o aspeto de apresentação visual, e em *HyperText Markup Language* (HTML). Um servidor central *Orangemesh* pode suportar várias redes ROBIN, cada uma associada a uma conta diferente. *Orangemesh* existe completamente separada da rede *mesh*, onde todas as comunicações entre os nós *mesh* e o servidor são feitas através da *internet* [22].

3. PROTOCOLOS DAS REDES *MESH*

Um dos padrões mais populares para ilustrar a arquitetura de uma rede é o modelo *Open Systems Interconnection* (OSI), desenvolvido pelo *International Standards Organization* [23]. O modelo OSI, especifica um conjunto de funções da rede, agrupado em camadas, conforme se pode observar na Figura 8.

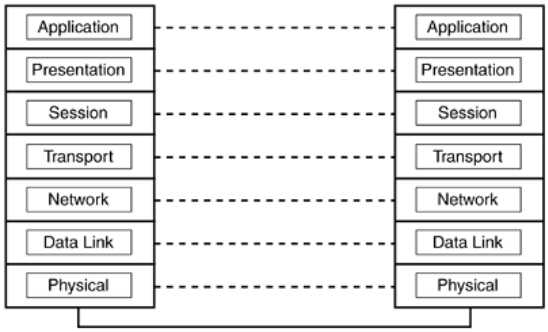


Figura 8 Camadas do modelo OSI.

A combinação das camadas da arquitetura da rede definem a funcionalidade de uma rede *wireless*, mas a implementação de uma rede *wireless* é feita apenas nas camadas inferiores do modelo.

3.1. CAMADA FÍSICA

A camada física é a camada mais baixa no modelo OSI e refere-se ao próprio meio físico no qual a comunicação ocorre. Pode ser um cabo de cobre, fibra ótica, ondas de rádio ou qualquer outro meio capaz de transmitir sinais.

As redes *wireless* usam as ondas de rádio, propagando o seu sinal pelo meio físico, que é o ar. A Figura 9 mostra o espectro disponível de frequências para a banda b/g da norma 802.11, onde se pode observar que os canais 1, 6 e 11 não interferem entre si.

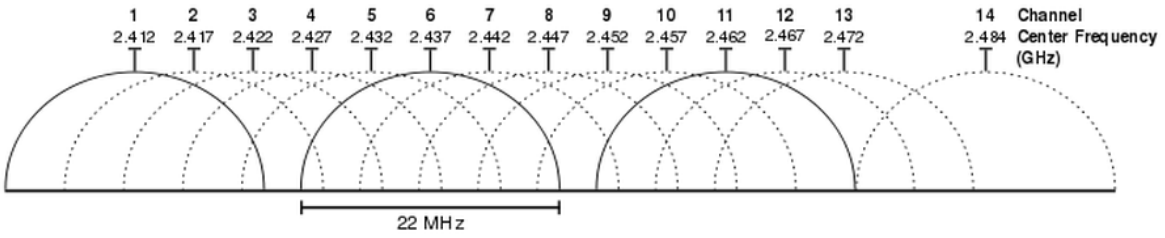


Figura 9 Canais Wi-Fi na banda de 2,4 *GHz*, usados pelos padrões IEEE 802.11 b/g.

A maioria dos modelos de *routers* em uso atualmente, têm a capacidade de transmitir o sinal usando múltiplos canais. Numa rede constituída por vários dispositivos que utilizem só um canal, a comunicação é *half-duplex*. Isto quer dizer que esses dispositivos não podem transmitir e receber um sinal simultaneamente. Assim a largura de banda transmitida é reduzida significativamente. Além disso quando um nó transmite, todos os outros nós têm que escutar e não podem transmitir, senão causariam uma colisão. Num ambiente constituído por dispositivos com capacidade de transmitir o sinal usando múltiplos canais, cada nó pode sintonizar a sua *interface* rádio em diferentes canais e assim aumentar a capacidade de transmissão. Essa capacidade de transmissão, ainda pode ser melhorada em ambientes com dispositivos constituídos por múltiplas *interfaces* rádio, com capacidade de transmissão usando múltiplos canais.

3.2. CAMADA DE ACESSO AO MEIO

Esta camada assegura o acesso ao meio, assim como controla a sincronização entre duas entidades. Exemplos de protocolos de usados para o acesso ao meio são *Ethernet, Token Ring, ATM* e os vários protocolos *wireless* 802.11 a/b/g/n, etc.

Nas redes *wireless*, esse controlo envolve a coordenação do acesso ao meio que é o ar, assim como a correção dos erros que possam ocorrer quando os dados são propagados desde a origem para o destino.

Nas WMNs, os melhoramentos realizados nos tradicionais protocolos de contenção, normalmente não são suficientes para melhorar a eficiência e a equitatividade no acesso ao meio. Os protocolos tradicionais de acesso ao meio são limitados. Por exemplo, dispositivos com múltiplas *interfaces* rádio que usam múltiplos canais para a transmissão do sinal, trazem novos problemas à atribuição de canais e ao acesso ao meio. Rádios *Multiple Input and Multiple Output* (MIMO) foram propostos para incrementar a capacidade das WMNs, de forma a mitigar a falta de equitatividade no acesso ao meio.

3.3. CAMADA DE REDE

Os protocolos de *routing* para redes *mesh* têm de ser concebidos tendo em mente os desafios da comunicação rádio [24]. As ligações *wireless* e a topologia de uma rede *mesh* são intrinsecamente instáveis. Os dispositivos da rede podem-se ligar e desligar, a largura de banda varia e as ligações muitas vezes falham com a consequente perda de pacotes.

Um protocolo de *routing* deve ser resiliente contra os erros de *routing* mesmo que as mensagens do protocolo cheguem atrasadas ou sejam perdidas. A largura de banda disponível e o desempenho dos nós *mesh* são limitados e não devem ser desperdiçados com decisões de protocolo e com tráfego de cabeçalhos.

Há algum tempo atrás só existiam alguns (poucos) protocolos de *routing* que se podiam utilizar na prática para redes *mesh*, como por exemplo *Optimized Link State Routing Protocol* (OLSR). Atualmente existe um leque variado de protocolos e implementações *mesh* e algumas dessas implementações encontram-se disponíveis nos pacotes de instalação de *OpenWRT*, um sistema operativo *Linux* usado em *routers* para encaminhamento de tráfego. A maioria dos protocolos de *routing* mais recentes (BABEL, *Better Approach To Mobile Adhoc Networking* - BATMAN, *B.a.t.M.a.n –eXperimental* - BMX, *version 6* - BMX6) encarregam-se de manter as tabelas de *routing* IPv4 e IPv6 em cada nó *mesh* adicionando, atualizando e removendo rotas. Estes protocolos *mesh* usam *routing* com base em IP. São protocolos *mesh* que utilizam a camada 3 do modelo OSI. O protocolo BATMAN-ADV é um protocolo relativamente novo que funciona na segunda camada do modelo OSI. Logo é um protocolo *mesh* de camada 2. Para as camadas

superiores (incluindo IP), BATMAN-ADV faz com que toda a rede *mesh* se apresente como um *switch*, onde todas as conexões são locais. O protocolo BATMAN-ADV é transparente para as camadas superiores do modelo OSI. Isto simplifica a configuração da rede *mesh*, uma vez que é possível usar *Dynamic Host Configuration Protocol* (DHCP), *multicast* DNS (mDNS) or MAC *bridging* conjuntamente com BATMAN-ADV. BATMAN-ADV é um módulo do *kernel* do sistema operativo *Linux*, que já está incorporado nas distribuições oficiais. Seguidamente far-se-á uma descrição mais pormenorizada de alguns protocolos de *routing* utilizados nas redes *mesh*. Poder-se-á consultar uma lista de protocolos de *routing* Ad-Hoc mais extensa e divida por tipos em [25].

3.3.1. BETTER APPROACH TO MOBILE ADHOC NETWORKING (BATMAN)

O protocolo *Better Approach To Mobile Adhoc Networking* (BATMAN) [26] é um protocolo proativo para determinar rotas *multi-hop* em redes Ad-Hoc móveis. O desenvolvimento do protocolo BATMAN começou no [RFC 3626], que não era completamente funcional em cenários práticos, particularmente onde existe uma larga distribuição de nós e em ambientes onde ocorriam falhas, nomeadamente de ligações ou de nós que por algum motivo ficavam inacessíveis. A aproximação desenvolvida pelo BATMAN consiste em espalhar o conhecimento dos melhores caminhos para o destino, pelos nós participantes. Nessa abordagem, cada nó percebe e mantém a informação acerca do "melhor próximo salto" (*best next hop*) em direção a todos os outros nós, o que evita conhecimento desnecessário acerca da totalidade da topologia e reduz a sinalização do cabeçalho.

Algoritmo: Cada nó *n* envia mensagens da sua origem para informar os nós vizinhos da sua existência. Estas mensagens são denominadas de *OriGinator Message* (OGM). Os vizinhos, por sua vez, retransmitem os OGMs para informar os seus próprios vizinhos acerca da existência do nó *n*, e por aí adiante. A rede é inundada com esses pequenos pacotes que contêm o endereço do nó original, o endereço do nó que está a retransmitir e um número de sequência que define o período de duração desses pacotes – *Time To Live* (TTL). Cada nó retransmite o OGM uma única vez, só se receber do nó considerado o melhor salto (*best hop*), o sinal para iniciar a transmissão. Assim a rede *mesh* é seletivamente inundada de mensagens OGMs. A descoberta de rotas e a seleção de vizinhos depende do número e da fiabilidade dos OGMs recebidos. Os números de

sequência servem para se perceber o quanto a informação se encontra atualizada na rede, pois qualquer mensagem recebida com um número de sequência inferior ao número de sequência recebido na mensagem anterior é deitada ao lixo. Os nós podem alterar o TTL dos seus OGMs de forma a limitar o número de saltos que as mensagens podem atravessar. Isto é útil para os nós pertencentes ao *backbone*, cujas funções consistem em assegurar a melhor conetividade e cobertura possíveis. Normalmente o *backbone* é constituído pelos nós centrais da rede, que costumam estar ligados ao terminal da rede e que prolongam esse terminal até onde for possível, sendo depois a propagação do sinal feita por nós da rede *mesh.*

O protocolo BATMAN ultrapassa os outros protocolos, em quase todas as métricas de desempenho, devido à sua conceção simplista. Ao não recolher mais informação do que a que efetivamente pode usar, e ao obter apenas informações acerca dos seus vizinhos, os nós podem calcular rotas de uma maneira mais eficiente. O cabeçalho de *routing* é significativamente menor do que por exemplo, o do OLSR, o que evidencia que aproximações complexas às vezes conduzem a um menor desempenho.

O protocolo BATMAN, inicialmente foi lançado como um protocolo clássico de camada 3 OSI (L3), que utiliza pacotes *User Datagram Protocol* (UDP) para trocar informação sobre *routing*. Mais tarde, uma extensão BATMAN ADVanced (BATMAN-ADV) foi desenvolvida para trabalhar na camada 2 (L2) OSI. Esta versão emula uma ponte *Ethernet* (*Ethernet bridge*), de modo que todos os nós pareçam estar ligados diretamente e todos os protocolos que operam no topo dessa camada não estejam conscientes da natureza *multi-hop* da rede subjacente. Os princípios de funcionamento do BATMAN-ADV são idênticos ao BATMAN clássico, com as adaptações para tratamento dos endereços da camada 2 (L2), em vez dos endereços IP. Em vez de o *deamon*, utilizado na versão original do BATMAN, funcionar no espaço do utilizador (*user* space), funciona no espaço do *kernel* (*kernel space*), evitando assim o custo de copiar pacotes de e para o espaço do utilizador [14]. BATMAN-ADV encapsula todo o tráfego que entra e sai funcionando como um *switch* entre todos os nós. Neste momento o protocolo suporta *bridging* e *roaming* de clientes não-*mesh.* Atualmente o BATMAN-ADV (L2) é incluído no *kernel* do sistema operativo *Linux* como um módulo. Para usá-lo apenas basta carregá-lo e pode-se controlá-lo através da ferramenta "*batctl*".

3.3.2. OPTIMIZED LINK STATE ROUTING PROTOCOL (OLSR)

[RFC 3626] [27]

O *Optimized Link State Routing* (OLSR) [28] é um protocolo proativo para redes móveis Ad-Hoc. Inclui um número de otimizações cujo objetivo é reduzir o custo da propagação de informação na rede. Em particular para cada nó existe um subconjunto de vizinhos, chamados de *Multi Point Relays* (MPR), cuja função é reduzir retransmissões duplicadas na mesma região [29].

Algoritmo: Cada nó escolhe o seu conjunto de MPRs entre os seus vizinhos que estão à distância de um salto de maneira a conseguirem dar cobertura até dois saltos de distância, usando esses mesmos MPRs selecionados. OLSR impõe que existam ligações bidirecionais para cada um desses nós. Cada nó pertencente à rede, periodicamente, difunde para a rede informação dos seus vizinhos selecionados como MPRs, que se encontram à distância de um salto. Ao receber esta lista de seleção de MPRs, cada nó calcula ou atualiza as suas rotas. A rota é então uma sequência de saltos através de MPRs. De maneira a detetar ligações bidirecionais com os vizinhos, cada nó envia, periodicamente, mensagens *hello*, que contêm a lista de vizinhos e o estado da sua ligação. As mensagens *hello* contêm a lista de endereços com quem o nó tem uma ligação bidirecional, assim como a lista dos vizinhos que consegue "ouvir". O conteúdo dessas mensagens permite cada nó conhecer a existência dos seus vizinhos até dois nós e a seleção dos seus MPRs, que também são indicados nas mensagens *hello*, e construir a sua própria tabela de seleção de MPRs. Cada nó difunde mensagens de controlo específicas chamadas de *Topology Control* (TC), de maneira a construir a tabela de encaminhamento (*routing*). As mensagens TC são enviadas periodicamente pelos nós de modo a declarar o conjunto de MPRs selecionados (se a lista de MPRs estiver vazia, a mensagem não é enviada). As mensagens TC são utilizadas para manter as tabelas de topologia para cada nó.

Uma vez que se trata de um protocolo de *routing* proativo, não existe um atraso quando é necessário descobrir uma rota. Embora o cabeçalho de *routing* seja maior do que o cabeçalho de um protocolo reativo, isso não implica que o cabeçalho aumente com o número de rotas em uso. Rotas por omissão e rotas existentes podem ser injetadas no sistema. Para que a informação esteja atualizada são utilizados valores de tempo e de validação da informação. O OLSR assume que a ligação está válida e a funcionar se um número de mensagens TC foi recebido recentemente.

Existem algumas extensões para caracterizar a qualidade da ligação [30], como por exemplo OLSRd (*deamon*) que é frequentemente usado nos *routers* com distribuições *Linux* e que foi vulgarmente chamado de *Radio-Aware* OLSR. Esta extensão foi incluída no *draft* da norma 802.11s e foi influenciada pelo *Hazy-Sighted Link State* (HSLS).

O protocolo OLSR produz imensa informação, o que pode causar alguma inconsistência relativamente aos estados das ligações, principalmente devido a perda de pacotes, o que origina a propagação de dados acerca de rotas que possivelmente não são usadas. Também o protocolo requer uma determinada energia para calcular os caminhos ótimos na rede, assim como provoca uma quantidade bastante grande de cabeçalhos de *routing* devido às mensagens de *Topology Control* (TC), que consomem demasiados recursos de largura de banda.

Na tentativa de resolver algumas das limitações aqui referidas, foi criado o projeto OLSR-NG. No OLSR original, assim como neste novo projeto OLSR-NG, no cálculo das rotas é usado o algoritmo de *Dijkstra*, sendo que para o OLSR original é utilizada a fórmula $O(n^2)$, enquanto no OLSR-NG é utilizada a fórmula O(n*(log(n)), onde O representa o pior caso possível em notação matemática e n representa o número de nós, o que se reflete num menor número de cálculos para OLSR-NG.

3.3.3. AD-HOC ON-DEMAND DISTANCE VECTOR (AODV)

[RFC 3561/2003] [31]

O *Ad-hoc On-demand Distance Vector* (AODV) [32] é um protocolo reativo que cria e mantém rotas só quando são solicitadas. Em cada nó, na tabela de *routing*, é armazenada informação acerca do próximo *router* (*next hop*) para o destino desejado e um número de sequência recebido desse mesmo destino, preservando a informação atualizada.

Algoritmo: Tipos de mensagens do AODV são: *Route REQuests* (RREQ), *Route REPlies* (RREP) e *Route ERRors* (RERR). Quando solicitada, a descoberta da rede é feita através de uma mensagem RREQ difundida pelos vizinhos que possam reencaminhar essa mensagem ao destino desejado. Cada nó que recebe o pedido incrementa a métrica de saltos (*hops*) e atualiza a sua própria tabela. Entradas que não sejam usadas na tabela de *routing* são removidas após um certo tempo. Quando uma ligação falha, é devolvido ao nó transmissor um erro de *routing* e o processo repete-se. Os nós reagem às quebras de

ligação e às mudanças de topologia, oportunamente. Se uma ligação falha, um erro de *routing* é enviado ao nó transmissor. Assim o protocolo AODV notifica o conjunto de nós afetados, de maneira a que esses nós possam invalidar as rotas onde conste essa ligação perdida. O AODV é um protocolo *loop free*, ou seja, os pacotes não deambulam indefinidamente pela rede. O AODV oferece uma rápida convergência quando a topologia de rede muda, como por exemplo quando um nó se move, evitando o problema de Bellman-Ford - "*counting to infinity*" [32]. O AODV possibilita que o *routing* seja dinâmico, regenerador e com alcance de múltiplos saltos (*multi-hop*) entre nós móveis numa rede Ad-Hoc. Protocolos reativos como AODV tendem a reduzir os cabeçalhos das mensagens usados para controlo de tráfego, embora isso acarrete o custo de um incremento da latência quando procura novas rotas.

3.3.4. OPTIMIZED FISH-EYE LINK STATE ROUTING (OFLSR)

O protocolo *Optimized Fish-eye Link State Routing* (OFLSR) combina dois protocolos existentes: OLSR e *Fish-eye State Routing* (FSR).

O protocolo OLSR já foi abordado na subsecção 3.3.2. O protocolo FSR pertence à classe de protocolos Ad-Hoc proativos, também denominados de *table-driven protocols*, cujos mecanismos têm por base o protocolo *Link State Routing* usado nas redes com fio. Esse protocolo tenta minimizar o cabeçalho usando a técnica de olho-de-peixe (*fish-eye*). Os intervalos das atualizações sobre o estado da ligação para os nós mais distantes são mais longos do que para os nós que se encontram na vizinhança, tendo como referência o olho-de-peixe (área circular à volta de um determinado nó). Assim, o protocolo FSR origina uma boa diferenciação entre zonas de redes Ad-Hoc móveis e suporta altas taxas de mobilidade.

O protocolo OLSR origina uma quantidade grande de cabeçalhos de *routing* devido ao encaminhamento da mensagens *Topology Control* (TC), sendo que o protocolo OLFSR limita este fluxo de mensagens TC, adotando a técnica de olho-de-peixe, uma vez que a fonte apenas precisa de saber rotas aproximadas para os destinos mais longínquos [33]. Com a redução do tamanho das mensagens do estado das ligações, o desempenho do protocolo OFLSR poderá ser melhorado [29].

3.3.5. DYNAMIC SOURCE ROUTING (DSR)

[RFC 4728/2007] [34]

O protocolo *Dynamic Source Routing* (DSR) é um protocolo simples e eficiente concebido especificamente para ser usado em redes Ad-Hoc sem fios (*wireless*) *multi-hop* de nós móveis, pertencente à classe de protocolos reativos. O protocolo DSR permite que a rede se auto-organize e autoconfigure, sem a necessidade da existência de uma administração de infraestrutura da rede. O DSR usa IP *routing*, ou seja, todos os pacotes enviados usam o protocolo DSR que contém a lista completa dos nós que os pacotes têm de atravessar. Existem dois mecanismos principais: descoberta de rotas e manutenção de rotas, que trabalhando em conjunto permitem que os nós descubram e mantenham rotas para vários destinos da rede. Quando é solicitada uma operação de entrega de pacotes, o cabeçalho de *routing* dos pacotes é ajustado automaticamente para que sejam alteradas só as mudanças necessárias nas rotas em uso, de maneira a que os pacotes sejam entregues. O protocolo permite que sejam encontrados múltiplos caminhos para o destino, no entanto permite ao nó emissor que selecione a rota com base em alguns critérios, tais como o balanceamento da carga, o que permite o controlo das rotas usadas para envio dos pacotes.

Este protocolo evita a necessidade de atualização da informação das rotas nos nós intermediários, assim como reduz o cabeçalho de controlo, eliminando as mensagens periódicas de atualização de tabelas e guardando em memória a informação disponibilizada pelos outros nós. O atraso na configuração das conexões é maior do que nos protocolos proativos (*table-driven*). O protocolo DSR tem um bom desempenho em ambientes estáticos e de baixa mobilidade, mas o desempenho degrada-se rapidamente com o incremento da mobilidade. O cabeçalho de *routing* é proporcional ao tamanho do caminho, devido ao mecanismo de *routing* utilizado no DSR.

3.3.6. BATMAN EXPERIMENTAL VERSION 6 (BMX6)

O protocolo *BatMan eXperimental version 6* (BMX6) é baseado no código do BATMAN 0.3 (L3), mas agora na versão 6 foi totalmente reescrito. Inicialmente foi uma extensão do BATMAN mantido por Axel Neumann, com alguns melhoramentos no algoritmo de *routing*. É provavelmente o protocolo mais parecido com o BATMAN original, uma vez que usa pacotes UDP para a descoberta de vizinhos declinando a ideia do estado de ligação. Alguns extras foram desenvolvidos, como por exemplo uma pequena mensagem

embutida que permite enviar qualquer informação para o resto dos nós usando os mesmos OGMs. Pode-se caracterizar o protocolo BMX6 como sendo proativo, ou seja, mantém e atualiza a tabela de *routing* periodicamente, distribuindo informação pela rede que é uma rede *loop-free* e utiliza o algoritmo *Destination-Sequenced Distance-Vector* (DSDV). Os nós não sabem a topologia inteira da rede, mas apenas o vizinho que oferece melhores condições para um destino específico [2].

Como o seu antecessor BATMAN, o BMX6 envia periodicamente pacotes OGMs para os vizinhos, que contêm o endereço do destinatário, normalmente a cada meio segundo. A mensagem flui pela rede, e graças a este reencaminhamento, um nó fica a conhecer a existência de todos os participantes da rede *mesh*. BMX6 não usa como identificador dos nós os endereços IP. Em vez disso usa identificadores globais *Secure Hash Algorithm* (SHA), baseados em SHA2, encriptados, e usa pequenos identificadores locais para representar os nós e *interfaces* relacionadas na rede *mesh*. Como resultado disto o cabeçalho é bastante reduzido devido ao impacto marginal que o tamanho do endereço IP acarreta.

Uma característica interessante que o BMX6 apresenta é uma extensão (p*lug-in*), de nome SMS. Esta extensão usa os pacotes de *routing* para transmitir qualquer informação de um nó para toda a rede. A propagação funciona, mesmo que não exista caminho para os dados. Ainda que a rede Wi-Fi esteja sujeita a más condições, como por exemplo ruído, distância entre nós, etc., os dados serão propagados. O funcionamento consiste na clonagem de um ou mais ficheiros enviados por um nó para todos os outros nós. Todos os nós podem proceder da mesma forma. O funcionamento é simples, tendo cada nó duas diretorias localizadas em */var/run/bmx6/sms/send* e */var/run/bmx6/sms/rcvd*. Os ficheiros colocados na pasta *send* serão clonados para todos os outros nós, que os receberão na pasta *rcvd*. Esta funcionalidade está a ser utilizada para gerar informação quanto ao posicionamento da rede no mapa, e pensa-se que poderá servir para outros propósitos como por exemplo estatísticas, *captive portals*, filtragem de MACs, etc.

3.3.7. BABEL

O protocolo BABEL [35] é um protocolo proativo baseado no protocolo de *routing distance vector*. Este protocolo é mais recente do que OLSR e BATMAN. Foi concebido com base no *Destination-Sequenced Distance-Vector routing* (DSDV) [36]. A técnica de

utilizar números de sequência é replicada do protocolo DSDV de maneira a prevenir uma contagem até ao infinito dos pacotes que circulam na rede. O BABEL também adota o *Enhanced Interior Gateway Routing Protocol* (EIGRP) – protocolo que permite prevenir os *loops* (voltas) de *routing* e assim, rapidamente, consegue convergir caminhos livres de *loops* (*loop-free path*) [37]. O protocolo BABEL utiliza a métrica *Expected Transmission Count* (ETX – utiliza a taxa de perda de pacotes para determinar o melhor caminho), utilizada também pelo protocolo OLSR. Este protocolo tem como característica principal a otimização do mecanismo de transmissão. Para isso, utiliza um histórico para selecionar rotas. Se aconteceram problemas em determinadas rotas, que tenham inviabilizado o uso dessas mesmas rotas, poderá optar por outros caminhos que apresentem a mesma qualidade de ligação. Assim, será escolhida uma rota que já tenha sido utilizada com sucesso em detrimento de outra rota alternativa. O protocolo BABEL executa uma atualização reativa e força um pedido de informação de *routing* quando deteta que uma ligação a um dos seus vizinhos preferidos falhou. Num estudo publicado, onde foram realizados testes de consumo de energia, testes de atraso (*delay*), testes de perdas de pacotes, testes de cabeçalhos e testes de percentagem de pacotes fora de ordem [38], sendo os protocolos comparados AODV, OLSR, BATMAN (L2) e BABEL, verificou-se que o protocolo BABEL apresentou os melhores resultados, tendo o melhor desempenho comparativamente aos outros protocolos.

3.3.8. HYBRID WIRELESS MESH PROTOCOL (HWMP)

O Hybrid Wireless Mesh Protocol (HWMP) é o protocolo por omissão das WMNs [39]. A norma IEEE 802.11s apresenta um tipo de topologia de rede chamado *Mesh Basic Service Set* (MBSS). MBSS pode ter os seguintes três tipos de entidades: uma estação *mesh* (*Mesh station*), que se trata de uma normal estação 802.11s com funcionalidades acrescidas de descoberta de rotas e encaminhamento de pacotes; um *mesh Access Point*, que é uma estação *mesh* que providencia conectividade aos utilizadores finais; e um portal *mesh* que interconecta a WMN com outras redes diferentes de 802.11, como por exemplo 802.3. A norma 802.11s especifica o HWMP que funciona na camada MAC, como sendo o protocolo obrigatório para a seleção de caminhos/rotas.

Algoritmo: Os nós podem usar dois modos de operação: podem construir uma árvore de caminhos a pedido (*On-demand-Mode*) ou proativamente. O *On-demand-Mode* é baseado no protocolo AODV [32], como já vimos anteriormente funciona na camada MAC. Neste

caso existem três pacotes de controlo diferentes: *Path REQuest* (PREQ), *Path REPly* (PREP) e *Path ERRor* (PERR), sendo o seu funcionamento em tudo análogo ao AODV. Quando um nó *n* quer enviar informação, envia PREQ para toda a rede. Cada PREQ tem um número de sequência (análogo a situações já descritas anteriormente), que permite que os nós percebam se se trata de uma mensagem antiga ou recente. Os nós intermediários que recebem o PREQ criam ou atualizam o caminho para a fonte, dependendo desse número de sequência; se não existir caminho, simplesmente reencaminham o pedido até que este chegue ao destino. Quando o caminho se encontra estabelecido, guarda-o na memória e os subsequentes PREQs não são reencaminhados durante algum tempo. Quando o destinatário recebe o PREQ, envia um PREP via *unicast* de volta para o nó *n*.

No modo proativo de construção da árvore, um dos nós funciona como *ROOT* (raiz), neste caso nó *r*. O nó *r* periodicamente envia para toda a rede PREQs. O campo de endereço desses PREQs é um endereço *broadcast* (difundido para toda a rede), assim todos os nós que recebem os PREQs, enviam um PREP para o nó *r*. Desta forma é construída uma árvore proativa e o nó *r* tem a tabela de *routing* preenchida com todos os possíveis destinos da rede.

No modo híbrido, tanto os componentes proativos como os reativos concorrem no seu funcionamento. Extensões do HWMP permitem escolher qualquer métrica de *routing* ou combinação de métricas. O *on-demand routing* oferece uma grande flexibilidade em redes que variam de topologia. Já a árvore construída proativamente é muito eficiente em redes *mesh* fixas. A combinação de ambos faz com que o HWMP seja apropriado para uma variedade de diferentes configurações de rede. A métrica usada por omissão é baseada na métrica *airtime*, que é similar á métrica ETX, já referida anteriormente. Pode-se combinar com outras métricas para se obter melhor desempenho.

O *on-demand routing* no HWMP permite que os nós rapidamente obtenham rotas para novos destinos. Não obriga a que rotas para destinos que não apresentem comunicação ativa sejam mantidas. Para a descoberta de rotas, o *on-demand routing* utiliza um método para limitar o excesso de pacotes de *routing*, denominado de algoritmo *expanding ring search* [40]. São utilizados os tipos de mensagens já vistos anteriormente no protocolo AODV – *Route REQuests* (RREQ), *Route REPlies* (RREP) e *Route ERRors* (RERR) – para as definições de rotas. No que concerne à manutenção de rotas é utilizado também o tipo de mensagens *Route ERRors* (RERR). Todos os nós da rede possuem e mantêm o número

de sequência de destino, o que garante *loop-freedom* de todas as rotas relativamente a esse nó específico. Os *Mesh Points* monitorizam as ligações e podem alterar as ligações usando RERRs, evitando a recontrução da árvore. A perda de ligação provoca também um RERR, o que faz com que os nós decidam/selecionem outros caminhos de recurso.

3.3.9. SECURED HYBRID WIRELESS MESH PROTOCOL (SHWMP)

O HWMP é vulnerável a vários tipos de ataques [29], pelo que foi proposta uma versão *Secured Hybrid Wireless Mesh Protocol* (SHWMP) [41] como sendo a versão segura do HWMP, isto é, uma extensão de segurança da norma 802.11s. O SHWMP opera de uma forma similar ao HWMP, mas usa extensões de criptografia de maneira a assegurar autenticidade e integridade nas mensagens de *routing* e previne manipulação não autorizada nos campos de informação de *routing*. É uma forma robusta de segurança contra ataques e providencia uma taxa maior de entrega de pacotes quando comparado com o tradicional HWMP. Tendo em conta o sistema de troca de chaves existente na norma 802.11s (os nós possuem uma chave e uma palavra-passe para se autenticarem e reconhecerem), evita-se uma sobrecarga de troca de chaves [42]. Os campos mutáveis e os não mutáveis na mensagem de *routing* são identificados, protegendo-se a parte não mutável através de encriptação de chave simétrica. É utilizada a abordagem *Merkle-Tree* (árvores de dispersão ou árvores de *Merkle* – transformam a chave numa sequência de caracteres encriptados) para autenticar a informação mutável. Uma vez que só executa operações com chaves simétricas, a computação torna-se eficiente.

3.4. COMPARAÇÃO DE PROTOCOLOS

O protocolo de *routing* é provavelmente o componente mais importante de uma rede *mesh*. Quando se gere um pequeno grupo de nós, torna-se uma tarefa simples e transparente, uma vez que a rede funciona relativamente bem, independentemente do protocolo utilizado. Atualmente a maioria das redes utilizam protocolos como *Border Gateway Protocol* (BGP) ou *Open Shortest Path First* (OSPF), que serão provavelmente uma boa escolha para ambientes estáticos, principalmente devido à sua simplicidade e estabilidade [2]. Contudo, nos cenários atuais, e principalmente na próxima geração de redes, a rede estará em constante mutação e os protocolos estáticos não são adequados para estes casos. É necessário um protocolo mais complexo. Outras métricas, para além do número de saltos,

devem ser tidas em consideração, como por exemplo a perda de pacotes, saturação dos caminhos, qualidade das ligações, etc.

A pergunta óbvia é: qual o protocolo a adotar?

É difícil selecionar um protocolo, uma vez que alguns protocolos encontram-se em constante desenvolvimento e mudam frequentemente. Muitos deles nasceram em ambientes de *hackers* onde os seus criadores os desenvolveram apenas por gozo ou para poderem manter as suas próprias redes. Atualmente um dos protocolos mais estáveis é o OLSR, provavelmente devido ao RFC que já tem quase dez anos, o que faz com que a implementação esteja bastante "madura". Mas essa maturidade também se poderá considerar em vias de desatualização, uma vez que as redes e os aparelhos mudaram bastante. Os protocolos BABEL e BATMAN são protocolos relativamente novos e foram concebidos para redes comunitárias. Apresentam características interessantes baseadas na experiência das pessoas das comunidades. A escolha do protocolo deve ser bem ponderada, pois significa que a rede dependerá sempre dele. Se as pessoas que os desenvolvem decidirem deixar de os desenvolver (porque a partir de um dado momento se torna um produto comercial e fechado ou porque apenas se desinteressam e abraçam novos desafios) fará com que a rede fique desatualizada, obsoleta e obrigará à mudança de sistema em cada nó. Se porventura algum nó se encontrar num local de difícil acesso ou mesmo inacessível, poderá acontecer que não possa ser alterado. Estas são algumas das considerações a ter em conta na seleção do protocolo.

Alguns estudos, testes e publicações podem ajudar nessa escolha, embora os critérios para a diferenciação possam sempre ser discutíveis. Apresenta-se aqui algumas considerações.

Tanto o protocolo AODV como OFLSR funcionam bem nas WMNs com pouca carga de tráfego. O protocolo AODV deixa de ser escalável à medida que a carga aumenta [29]. Por outro lado, OFLSR proporciona um melhor desempenho em termos de taxa de entrega de pacotes, débito, latência e cabeçalhos de *routing* debaixo de diferentes condições de tráfego e mobilidade [33]. A implementação atual do "rascunho" corrigido da norma 802.11s pelo *open802.11s* (*open-source*) [43] e pelo BATMAN-ADV (L2) em ambientes estáticos evidencia que BATMAN-ADV apresenta melhor desempenho onde *open802.11s* apresenta instabilidade. Contudo *open802.11s* recupera mais rapidamente, em caso de falha de nó. O BATMAN-ADV tem alguma dificuldade em restabelecer a comunicação

depois de uma interrupção abrupta [44]. Alguns resultados confirmam que o cabeçalho do OLSR é maior do que o do BATMAN [45]. O BATMAN atinge o maior nível de estabilidade e de entrega de pacotes, enquanto BABEL oferece a maior largura de banda em ambiente *multi-hop* e o menor tempo de reparação de rotas em caso de falha. Tanto o BATMAN como BABEL apresentam melhor desempenho que OLSR, em várias métricas examinadas [46]. Outro estudo sugere que OLSR e BATMAN são similares [47]. Estudos teóricos sugerem que BATMAN supera OLSR em quase todas as métricas de desempenho, devido à sua aproximação simplista, no entanto ainda será necessário aprofundar os testes, para se poder provar essa ideia. Em redes Ad-Hoc *multi-hop*, o cabeçalho de *routing* existente nos protocolos tem um largo impacto no débito da rede. O protocolo BABEL proporciona um alto débito em redes pequenas, no entanto tem de ser melhor testado em redes maiores. Testes realizados com BABEL criam algum incentivo na prossecução do estudo de BABEL [47], uma vez que este protocolo apresenta um desempenho bastante interessante. Muitos estudos indicam que os sistemas híbridos apresentam um melhor desempenho quando comparados com sistemas não híbridos [48]. O protocolo híbrido HWMP é bastante escalável para WMNs. Já o SHWMP, que envolve cálculos de encriptação de chaves simétricas, supera o HWMP [41].

3.4.1. *WIRELESS BATTLE MESH*

Anualmente acontece um evento denominado de "*Wireless Battle Mesh*", que pretende juntar pessoas de todo o mundo para testar o desempenho de diferentes protocolos usados em redes Ad-Hoc, tais como BABEL, BATMAN, BMX, OLSR e *routing* estático [49].

É um torneio com caráter social, onde além de se testarem os protocolos na tentativa de se aferir qual será o melhor, aproveita-se para testar novas implementações, ferramentas, *drivers*, *scripts* e *hardware*, entre outras coisas. Este evento já se encontra na sua quinta edição, tendo sido a última este ano de 2012 na Grécia. Infelizmente os resultados desta última edição não se encontram disponíveis. A próxima ocorrerá em abril de 2013, na cidade de Aalborg, na Dinamarca. Nas versões anteriores, que decorreram em Itália, em 2010, e em Espanha, em 2011, os resultados foram publicados.

3.4.2. *WIRELESS BATTLE MESH IV* – ESPANHA - 2011

O cenário de testes consistiu num total de 33 nós (*routers*), distribuídos por um grande complexo (aproximadamente 2000 metros quadrados), com as seguintes especificações [2]:

- *Hardware*: Fonera 2100 e Fonera 2200
- Um rádio 2.4 Ghz (802.11bg)
- Uma porta *Fast Ethernet*
- *OpenWRT* como sistema operativo

O teste é feito debaixo de más condições, porque todos os *routers* usam o mesmo canal, o *hardware* não é poderoso (são usados cinco protocolos em cada nó, o que faz com que o CPU esteja sempre ocupado) o rádio/antena foi concebido para uso interno (*indoor*), a ligação entre nós afastados é má. Contudo estas condições nem são uma menos valia, pois assim obriga os protocolos a demonstrarem as suas melhores funcionalidades.

Os testes foram de um nó para todos os outros nós, conforme se pode ver na Figura 10.

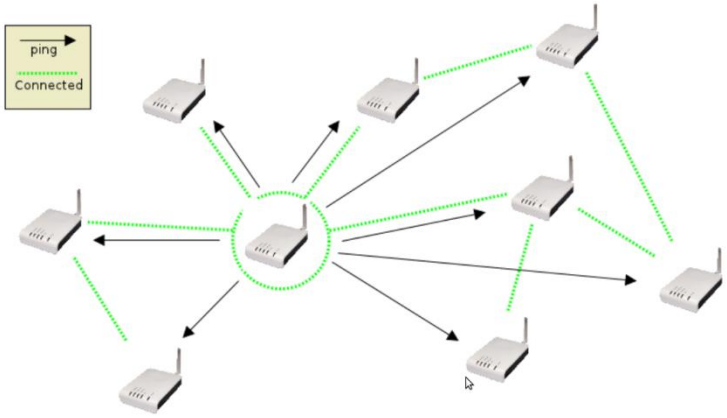


Figura 10 Cenário de *Battle Mesh*.

Desde o nó central foram realizadas as seguintes provas:

1. 50 *pings* com um intervalo de 0,1 *ms* de maneira a obter o *Address Resolution Protocol* (ARP) dos seus vizinhos.
2. 5 testes de 20 *pings* com um intervalo de 0,2 *ms* para o nó de destino.

Foram realizados um total de 3 testes, identificados pela hora de início:

1. T519 (05:19)
2. T630 (06:30)

3. T744 (07:44)

Conforme se pode constatar, cada teste demorou cerca de uma hora a ser concluído.

Os protocolos foram divididos em dois diferentes grupos:

- A: Babel, OLSR, BMX6
- B: Batman, Batman-adv

O número de nós que constituem o cenário do grupo B é de 25 (em vez dos 33 nós que constituem o grupo A), porque não estão a usar a *interface Ethernet* [2].

3.4.3. RESULTADOS

Na Tabela 1 são apresentados o número total de nós vistos por cada protocolo.

Tabela 1 Número total de nós vistos por cada protocolo.

Protocol	T519	T630	T744
Babel (A)	18	22	20
OLSR (A)	23	24	28
BMX6 (A)	26	27	28
Batman (B)	15	19	18
Batman-adv (B)	24	20	23

Como foram criados dois grupos (A e B), foi necessário comparar os protocolos em função da percentagem de nós vistos a partir do nó central, conforme apresentado na Tabela 2.

Tabela 2 Percentagem de nós vistos por cada protocolo.

Protocol	Available nodes	T519	T630	T744
Babel (A)	33	55%	67%	61%
OLSR (A)	33	70%	73%	85%
BMX6 (A)	33	79%	82%	85%
Batman (B)	25	60%	76%	72%
Batman-adv (B)	25	96%	80%	92%

O número de nós vistos a partir do nó central é ilustrado na Figura 11.

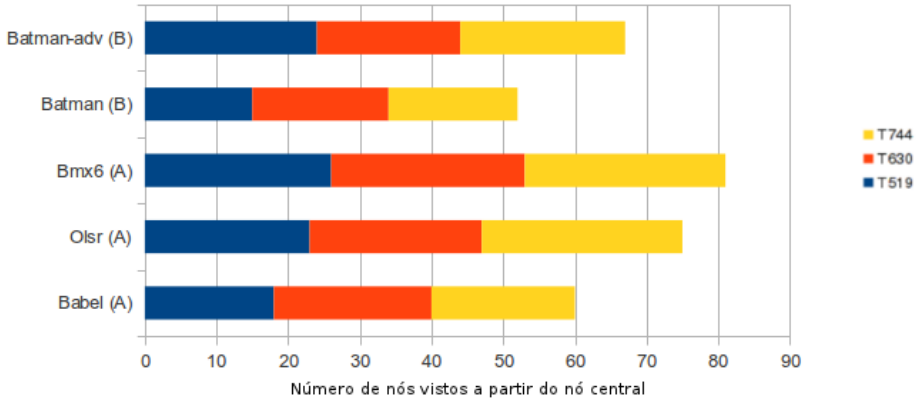


Figura 11 Relação entre os nós disponíveis e os vistos pelo nó central (mais é melhor).

Na Tabela 3 são apresentados os valores percentuais relativamente à taxa de sucesso de *pings* bem sucedidos por cada protocolo (um *ping* bem sucedido significa que houve uma resposta de pelo menos um *ping* em vinte pedidos, valor de cada teste).

Tabela 3 Percentagem de *pings* bem sucedidos por cada protocolo.

Protocol	T519	T630	T744
Babel (A)	97.78	88.18	94.00
OLSR (A)	96.52	97.5	94.29
BMX6 (A)	98.46	100	98.57
Batman (B)	94.67	89.47	100
Batman-adv (B)	100	97.04	94.57

Na figura 12, pode-se observar o resultado de *pings* bem sucedidos.

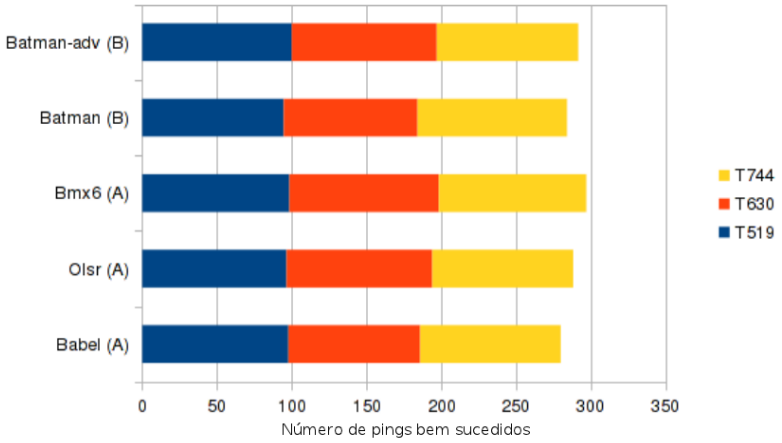


Figura 12 *Pings* bem sucedidos (mais é melhor).

Na Tabela 4 são apresentados os valores de latência por protocolo. Os valores são apresentados em milisegundos.

Tabela 4 Valores de latência por protocolo em milisegundos.

Protocol	T519	T630	T744
Babel (A)	89	137	110
OLSR (A)	221	40	86
BMX6 (A)	62	131	56
Batman (B)	45	164	81
Batman-adv (B)	39	37	39

Os resultados referentes à latência, podem ser visualizados na Figura 13:

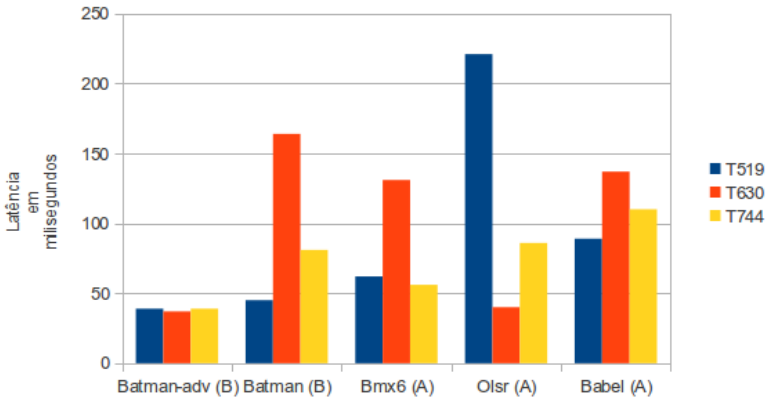


Figura 13 Latência por protocolo em milisegundos (menos é melhor).

3.4.4. CONCLUSÕES

Não é completamente clara a conclusão acerca destes testes. Apenas fornece uma ideia sobre o funcionamento dos protocolos. Mais tipos de provas são necessários para se chegar a conclusões relevantes, como por exemplo a utilização de um protocolo de transporte persistente como *Transmission Control Protocol* (TCP), que tente que os pacotes sejam entregues, uma vez que neste teste foram usados pacotes *Internet Control Message Protocol* (ICMP).

Outra observação pertinente pode ser feita relativamente às WMNs. Efetivamente nestes testes não foi considerada a mobilidade, o cenário é estático e muitos dos protocolos implementados apresentam características interessantes para ambientes móveis.

Contudo algumas considerações podem ser retiradas. Tendo em conta que foram criados dois grupos de protocolos e que foi utilizado como meio de comparação os valores percentuais dos testes realizados, verifou-se que no teste de nós vistos a partir do nó central, BMX6 e BATMAN-ADV tiveram os melhores resultados, cada qual no seu grupo. A percentagem de nós detetados para cada um é de 82% e 89%, respetivamente, sendo bastante mais elevada que a dos outros protocolos. Também quanto à comunicação entre nós parece ser bastante estável, verificando-se a percentagem de 99% para BMX6 e de 97% para BATMAN-ADV (são deduzidas da prova de *pings* bem sucedidos). Temos de ter em conta que o cenário não é dos mais fáceis e estes números são a prova que estes protocolos têm uma boa implementação e que conseguem operar em más condições. E não se deve esquecer o OLSR nem o BABEL. Os números de OLSR estão perto de BMX6 e atrás já foi referido que BABEL é bastante rápido e eficiente a recuperar de quebras de ligações e a encontrar rotas alternativas rapidamente, assim como apresenta algumas extensões bastante interessantes, como por exemplo "SMS".

Os protocolos que apresentam resultados mais interessantes do ponto de vista de desempenho e de continuidade no futuro próximo são os protocolos que se encontram em constante evolução, com imensas pessoas a colaborar a partir de toda a parte do mundo, usando para isso as *mailing lists* que estão disponibilizadas na *internet*. Para se poder participar basta fazer um registo e pode-se trocar informações ou colocar dúvidas e participar na conceção de código.

Tendo em conta que:

- Existe uma comunidade interessada e que trabalha diariamente no seu desenvolvimento;
- O projeto é amplamente divulgado e já faz parte das distribuições oficiais do sistema operativo *Linux* e também do *firmware OpenWRT*;
- É um protocolo inovador quanto à filosofia de *routing*, nomeadamente por ser um protocolo de camada 2 OSI, que utiliza endereços MAC para ligar a rede *mesh*, e que permite utilizar todas as aplicações da camada 3 OSI, de uma forma transparente;

Optou-se por escolher o protocolo BATMAN-ADV como protocolo utilizado na conceção da ferramenta de gestão de rede *mesh*, proposto no âmbito desta dissertação de mestrado.

4. O PROTOCOLO *BATMAN-ADV*

Uma vez que é necessário proceder à configuração do protocolo BATMAN-ADV, importa perceber melhor este protocolo quanto ao seu modo de funcionamento e de implementação, de uma maneira mais pormenorizada. O protocolo ao longo dos anos sofreu várias alterações e foi sujeito a variados testes. O núcleo do algoritmo evoluiu e neste momento encontra-se na IV geração [50]. Seguidamente serão expostos alguns aspetos do protocolo escolhido.

4.1. ARQUITECTURA E FUNCIONAMENTO

Nesta secção será descrito como o BATMAN-ADV está estruturado quando configurado para uma rede *mesh*.

4.1.1. ESTRUTUTURA DO BATMAN-ADV

O protocolo BATMAN-ADV é um protocolo de *routing* de camada 2 e é implementado como módulo do *kernel Linux* desde março de 2011 [14]. Encontra-se integrado numa camada entre a *interface* dos *drivers* da rede e a *interface* virtual da rede, que é usada pelas aplicações para comunicar através da rede *mesh*. Uma representação da implementação é

apresentada na Figura 14. Os pacotes são entregues ao módulo BATMAN-ADV, quer através da interface virtual *mesh*, quer pela interface de rede. O módulo verifica o destino do pacote e reencaminha-o para o próximo melhor nó ou entrega-o na camada superior.

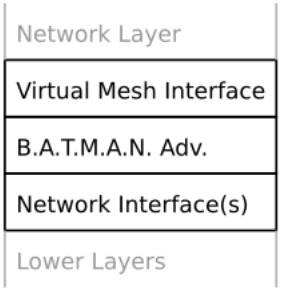


Figura 14 Configuração da estrutura do protocolo BATMAN-ADV num único nó.

As camadas superiores à *interface* virtual nada precisam de saber acerca de BATMAN-ADV ou da rede *mesh*. Esta estrutura faz com que a inteira rede *mesh* atue como um *switch* virtual que transporta pacotes de uma *interface* virtual para outra *interface* virtual.

4.1.2. ENVIO DE PACOTES

De forma a poder ter conhecimento dos nós e dos hóspedes da rede, cada nó BATMAN-ADV mantém três tabelas:

originators – A tabela *originator* guarda informação acerca de cada nó da rede *mesh* e tanto é usada para *routing* de pacotes locais como para *routing* de pacotes a reencaminhar.

transtable_global – Guarda informação acerca dos hóspedes da rede e dos nós aos quais estão conectados, ou seja, nós de fim de rede. A *transtable_global* é consultada quando um pacote local é transmitido, de modo a encontrar uma conexão que leve o pacote ao seu destino.

transtable_local – Guarda informação dos hóspedes conectados ao próprio nó. A *transtable_local* é usada quando são enviados os OGMs para toda a rede, onde as entradas da tabela relativas a hóspedes são apensas ao OGM de maneira a informar a rede da existência desses hóspedes.

A *interface* virtual *mesh* é o local de entrada e saída para a rede *mesh*. Quando um pacote é transmitido através da *interface*, é entregue ao BATMAN-ADV como sendo uma trama

MAC, cujo endereço da *interface* virtual é o endereço de origem. O destino pode ser uma *interface* virtual de outro nó, um hóspede ou um endereço de grupo (*multicast* ou *broadcast*). Os passos efetuados antes da transmissão dos pacotes encontram-se ilustrados na Figura 15.

Figura 15 Procedimento para transmissão de pacote usando o protocolo BATMAN-ADV.

Quando é solicitado o envio de pacote para um destino é escolhido o nó que oferece melhores condições de entrega. Então o pacote é construído quer com a informação original quer com a informação do nó que oferece melhores condições de entrega, antes de ser entregue ao *driver* do dispositivo para transmissão física.

Para reencaminhar o pacote, o protocolo BATMAN-ADV procura o endereço de destino na tabela *transtable_global* e encontra o nó que está conectado ao destino do pacote. O pacote BATMAN-ADV é construído com a informação original e com a informação adicional encontrada.

Os pacotes que são enviados através da *interface* virtual (que podem ser enviados para uma camada superior, uma aplicação ou um nó noutra rede que esteja ligada através de uma ponte) são distinguidos dos pacotes que são reencaminhados na rede *mesh*. Durante esta secção, os pacotes enviados através da *interface* virtual são denominados de pacotes locais.

4.1.3. REENCAMINHAMENTO DE PACOTES

Os pacotes provenientes da rede *mesh* são recebidos pelos nós através de um dispositivo de rede e são processados como sendo pacotes BATMAN-ADV, isto no caso de o tipo de

pacote de *Ethernet* condizer com o tipo de pacote BATMAN-ADV (0x4305). Se o campo da origem do pacote não condiz com o endereço do nó, o pacote é reencaminhado.

A aproximação é similar quando se transmitem pacotes locais, excepto no que diz respeito ao campo que identifica a origem, que já está construído. Para reencaminhar o pacote, é procurado o nó que oferece melhores condições e o pacote é transmitido.

4.1.4. RECEÇÃO DE PACOTES

Quando um pacote chega ao nó de destino, o cabeçalho BATMAN-ADV é retirado e é entregue na *interface* virtual. A *interface* pode estar ligada através de uma ponte e nesse caso o pacote pode ser transmitido para o hóspede conforme descrito anteriormente.

4.2. TIPOS DE PACOTES

O protocolo BATMAN-ADV tem seis tipos de pacotes diferentes, que são usados para a descoberta da rede. O protocolo possui diferentes tipos de transporte de dados e uma ferramenta de visualização da rede [14]. Cada tipo é sucintamente descrito aqui.

Pacote BATMAN-ADV – O pacote BATMAN-ADV, também conhecido como OGM, trata-se do pacote primário do protocolo BATMAN-ADV. É usado para a descoberta de nós e rotas na rede e cada nó na rede cria e difunde um OGM, num intervalo fixo de tempo. Conforme ilustrado na Figura 16, cada nó atualiza a sua tabela de *routing* e retransmite os pacotes OGMs, de maneira que os nós mais distantes também "aprendam" acerca do nó originário.

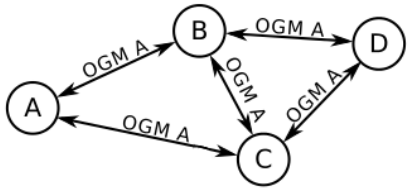


Figura 16 O nó A envia OGM para os seus vizinhos, que atualizam a sua tabela de *routing* e, por sua vez, retransmitem novamente os pacotes.

Os OGMs servem dois propósitos principais: anunciar a presença do nó, a possibilidade de esse nó anunciado servir como próximo salto no envio de pacotes e medir a qualidade da rota através do nó anunciado. Ambos serão descritos mais tarde nesta secção.

Os pacotes contêm a seguinte informação, acerca do nó originário e do nó seguinte:

1. Endereço do nó originário – identifica o nó que criou o OGM.
2. Número de sequência – usado para medir a qualidade da ligação e detetar pacotes duplicados.
3. Transmissão de qualidade, em inglês, *Transmit Quality* (TQ) – descreve a qualidade da ligação da rota total para o nó originário.
4. Endereço do nó antecessor que enviou o OGM – usado para descartar OGMs já enviados e novamente recebidos pelo nó recetor.
5. *Time To Live* (TTL) – limita o número de nós que um OGM pode atravessar.
6. Contagem *Host Neighbour Announcement* (HNA) – descreve o número de nós não-BATMAN-ADV (nós exteriores à rede) que podem ser alcançados através do nó originário. Informações acerca de cada HNA estão apensas ao OGM.
7. Sinalização de *Gateway* – usada se o nó oferece uma conexão para outra rede.

Internet Control Message Packet – Os pacotes *internet control message* são usados para suportar um subconjunto de funções fornecidas pela versão IP de ICMP. Como o protocolo BATMAN-ADV opera na camada de ligação, os nós na rede nem sempre podem ser alcançados pelos seus endereços IP (por exemplo quando acontece uma falha num nó), assim o BATMAN-ADV suporta pedidos de *ping*, respostas e falhas.

Pacote *Unicast* – Pacotes *unicast* encapsulam dados *unicast* das camadas superiores. Um pacote *unicast* contém, em adição aos dados reais a transmitir (*payload*), o endereço de destino e o campo TTL.

Pacote *Unicast* fragmentado – Devido ao encapsulamento do *payload* feito pelo BATMAN-ADV, o comprimento dos pacotes *unicast* podem exceder a unidade máxima de transmissão – *Maximum Transmission Unit* (MTU) permitida para a ligação utilizada, e os pacotes têm de ser fragmentados e agregados no destino. Os pacotes *unicast* de fragmentação transportam partes do *payload* e têm um campo de sequência, que é necessário para agregar o pacote *unicast* original. Têm também um campo que se refere à origem do pacote e que serve para identificar a sequência e *flags* para sinalizar o fim da sequência.

Pacote *Broadcast* – Para suportar o envio de pacotes para toda a rede, e não apenas o envio para vizinhos que estejam ao alcance de um nó, os pacotes *broadcast* são reencaminhados. De maneira a evitar a duplicação de pacotes, os pacotes *broadcast* possuem um número de sequência, o endereço da origem e o TTL.

Pacote de visualização – O protocolo BATMAN-ADV suporta a visualização gráfica da rede. De maneira a obter-se a informação necessária para a visualização, um servidor pode ser iniciado num ou em mais nós na rede. Um servidor anuncia a sua presença através dos pacotes de visualização e cada nó reporta informação de volta para o servidor.

4.3. DESCOBERTA DE NÓS

Como descrito anteriormente, todos os nós numa rede BATMAN-ADV transmitem periodicamente OGMs para todos os outros nós da rede. Quando um nó recebe um OGM, realiza os seguintes passos:

1. Verifica se o remetente do OGM é ele próprio. Nesse caso o emissor é um vizinho direto e atualiza a tabela de *routing* de acordo com essa informação.

2. Verifica se o campo "remetente anterior" (*previous sender*) é ele próprio. Se assim for, o OGM já foi processado e é descartado.

3. Determina a origem do OGM. Se não existir na tabela de *routing*, então é criada.

4. A classificação do nó originário do OGM é atualizada.

5. Os campos da *Transmission Quality* (TQ) e do TTL são atualizados e o OGM é enviado outra vez.

Também são feitas verificações de maneira a evitar que os pacotes deambulem pela rede, assim como são feitas verificações para evitar duplicação de OGMs.

4.4. ESTIMATIVA DE QUALIDADE DE LIGAÇÃO

De maneira a estimar a qualidade de ligação a um nó, a informação *Transmission Quality* (TQ) existente num OGM é mudada durante o caminho percorrido através da rede. A TQ descreve a probabilidade de um pacote chegar ao seu destino e é calculada conforme se descreve a seguir. O valor da qualidade de ligação é guardado numa sequência de 8 *bits*, cuja representação do valor se encontra entre 0 e 255, que permite aferir a granularidade da TQ, sem incrementar o tamanho do pacote.

Qualidade de receção – No cenário ilustrado na Figura 17(a) [14], o nó B transmite OGMs que são recebidos no nó A, que calcula a qualidade de receção – *Receive Quality*

(RQ) do nó B. Para esse cálculo é usada uma janela deslizante de tamanho *n* (por omissão 128), onde o número de sequência dos últimos *n* OGMs é recordado. A TQ é então calculada como a percentagem dos OGMs recebidos.

Qualidade de eco – Quando o nó A envia o OGM, o nó B reenvia o OGM recebido, que chega novamente ao nó A – Figura 17(b). Analogamente a RQ, uma janela deslizante é usada para calcular a qualidade de eco – *Echo Quality* (EQ).

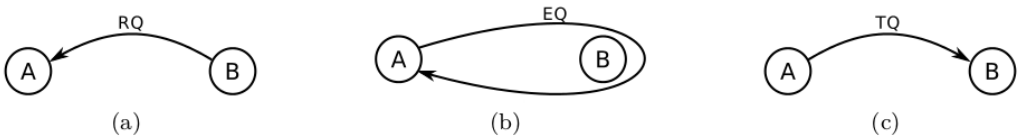


Figura 17 Três qualidades de ligação usadas no BATMAN-ADV em função do ponto de vista do nó A. *Receive Quality* (RQ) em (a), *Echo Quality* (EQ) em (b) e *Transmit Quality* (TQ) em (c).

Qualidade de transmissão – A qualidade de transmissão é a probabilidade de um pacote enviado do nó A ser recebido corretamente no nó B – Figura 17(c). Uma vez que a EQ é a probabilidade de um pacote ser entregue corretamente desde o nó A para o nó B e desde o nó B de volta para o nó A, a qualidade de transmissão – *Transmission Quality* (TQ) pode ser definida em função de RQ e de EQ:

$$EQ = RQ \cdot TQ$$
$$TQ = EQ / RQ$$

4.5. PROPAGAÇÃO DA TRANSMISSÃO DE QUALIDADE

Para informar os outros nós da rede acerca da TQ para o nó A, o valor de TQ é adicionado ao OGM. A cada nó atravessado pelo OGM, a TQ global para o nó A é atualizada com os valores locais da TQ em relação ao último nó por onde passou o OGM, conforme ilustrado na Figura 18 [14].

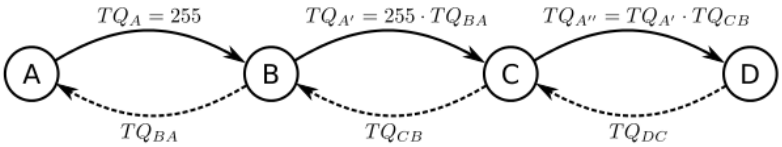


Figura 18 O valor de *Transmission Quality* (TQ), existente em cada OGM, é atualizado em cada nó que atravessa.

4.6. SELEÇÃO DE ROTA

Enquanto alguns protocolos de *routing* calculam a inteira topologia de rede quando escolhem rotas, BATMAN-ADV guarda apenas a informação acerca de qual nó é o melhor para chegar ao nó desejado. Quando um pacote é recebido num nó, esse nó volta a escolher novamente o melhor nó para entregar o pacote no destino.

Um cenário de *routing* é ilustrado na Figura 19, onde o nó F transmite um pacote para o nó A. A única informação que o nó F possui acerca do nó A, são dois TQs globais: um através do caminho pelo nó D e outro através do caminho do nó E. Neste caso o caminho pelo nó D é melhor e o pacote é transmitido pelo nó D, que conhece apenas os TQs globais para o nó A através do nó B, nó C e nó E. Qualquer nó que seja selecionado terá de fazer a mesma escolha. Neste caso o nó C é escolhido, que por sua vez escolhe o caminho direto para o nó A.

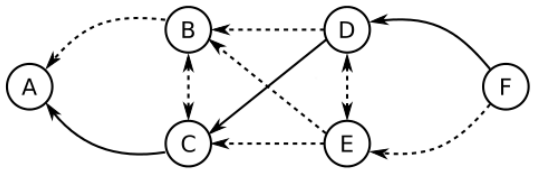


Figura 19 Quando é transmitido um pacote do nó F para o nó A, cada nó escolhe o melhor caminho com base na TQ global (caminho de linhas sólidas).

4.7. ANÚNCIO DE HÓSPEDE

Nos OGMs é incluído o anúncio de hóspedes alcançáveis através dos nós BATMAN-ADV, campo denominado de *Host Neighbour Announcements* (HNA). Como esses hóspedes nada sabem acerca do protocolo BATMAN-ADV, o nó que os alcança tem de informar a rede da existência desses vizinhos. Para isso é adicionada uma lista de endereços desses hóspedes ao OGM transmitido pelo nó BATMAN-ADV.

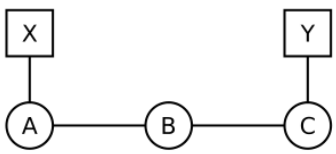


Figura 20 Os hóspedes na rede *mesh*.

Um cenário simples é retratado na Figura 20, onde o nó A está conetado ao vizinho X e o nó C está conetado ao vizinho Y. Quando o nó A recebe um pacote do vizinho X, que é dirigido ao vizinho Y, sabe que o pacote tem de ser transmitido para o nó C da rede *mesh*.

4.8. TRATAMENTO DE LIGAÇÃO ASSIMÉTRICA E ATRIBUIÇÃO DE PENALIDADE

De maneira a assegurar que as melhores ligações são escolhidas, um valor adicional é introduzido de forma a penalizar as ligações que têm uma pobre qualidade de receção – *Receive Quality* (RQ) [51].

Esta penalização de ligação assimétrica (f_{asym}) é um valor ponderado e é determinado através da seguinte fórmula, onde RQ é a *Receive Quality*:

$$f_{asym} = (100\% - (100\% - RQ)^3)$$

Esta penalidade tem uma enorme influência no valor de TQ para ligações com grande taxa de perdas de pacotes e uma pequena influência em ligações com baixa taxa de perdas de pacotes.

Um nó que usa o protocolo BATMAN-ADV só conhece o melhor nó (*best next hop*) para chegar a um destino e não a rota na sua totalidade. Assim, o nó não sabe quantos saltos fazem parte da rota. Em algumas redes poderá ser desejável escolher o caminho mais curto para o destino, ou seja, o que apresenta o menor número de saltos, de maneira a reduzir a latência e a poupar largura de banda. Por este motivo uma penalidade de salto (*hop_penalty*) foi também introduzida por cada salto que o OGM faz, decrescendo a TQ num determinado valor.

Ambas as penalizações são consideradas no cálculo do valor de TQ, cada vez que é recebida uma mensagem OGM. O valor final de TQ é calculado em função da seguinte fórmula:

$$TQ = TQ_{received} . TQ_{local} . f_{asym} . hop\_penalty$$

Na Figura 21 pode-se observar o formato do OGM, na geração BATMAN IV.

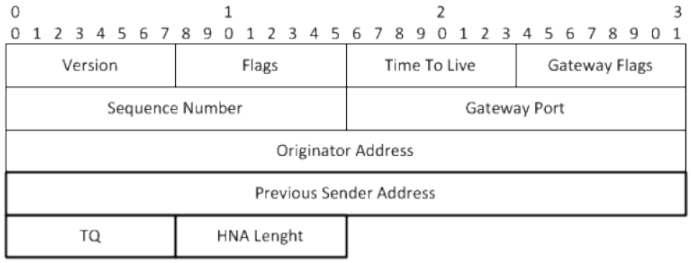


Figura 21 Formato do OGM na geração BATMAN-ADV IV.

A Tabela 5 apresenta alguns dos atributos mais importantes e valores por omissão usados no protocolo BATMAN-ADV.

Tabela 5 Atributos do protocolo BATMAN-ADV e os seus valores por omissão.

Atributos	Valores por omissão	Descrição
DEFAULT VERSION	4	Versão BATMAN.
MAX SEQUENCE NUMBER	65535	Número máximo de sequência permitido.
DEFAULT TTL	50	Valor de *Time To Live* por omissão.
OGM BROADCAST INTERVAL	1 seg.	Intervalo periódico entre envio de OGMs, em segundos.
PURGE TIMEOUT	200 seg.	Tempo de vida de um nó, na lista de nós que não foi atualizada recentemente, em segundos.
HOP PENALTY	5	Penalidade considerada no cálculo de TQ por cada salto na rede.
MAX TQ VALUE	255	Valor máximo de TQ possível.

5. INSTALAÇÃO E CONFIGURAÇÃO DOS *ROUTERS* BATMAN-ADV

O objetivo de uma WMN é proporcionar aos utilizadores finais acesso à *internet* de alta velocidade, entre outros serviços que podem ser disponibilizados localmente. Para se atingir este objetivo, a arquitetura da rede deve ser pensada cuidadosamente. É preciso proporcionar *Quality of Service* (QoS) aos utilizadores finais, assim como um bom desempenho de toda a rede. Planear uma WMN pressupõe determinar o número de *gateways*, otimizar a localização dessas *gateways*, determinar o uso da largura de banda e minimizar os custos de implementação. Existem, tipicamente, dois tipos de implementação [4]. A implementação numa nova área, onde tudo pode ser pensado de raiz, podendo-se pensar melhor na topologia, o que permite retirar todo o proveito e capacidade da rede WMN, por outro lado existe a implementação numa área já existente, ou seja, a WMN será implantada usando-se já as estruturas existentes. Neste caso, o arquiteto da rede terá as opções da topologia a usar, mais limitadas. Até se chegar à rede *mesh* final, é necessário resolver uma série de questões e tomar algumas decisões.

Para a implementação da rede *mesh* utilizada neste trabalho foi necessário:

- Criar o *firmware* com incorporação do protocolo BATMAN-ADV.
- Instalar o *firmware* criado nos *routers*.
- Proceder à configuração dos parâmetros dos *routers*.
- Solucionar a questão do posicionamento dos nós.
- Assegurar a segurança da rede.

Os equipamentos colocados à disposição, por parte do Instituto Superior de Engenharia do Porto (ISEP), para a realização deste trabalho foram *routers Linksys WRT54GL*. Contudo e uma vez que esses equipamentos durante o ano letivo foram necessários para a docência no estabelecimento de ensino, foi necessário proceder à sua devolução com o *firmware* original instalado, sendo que estariam disponíveis ao fim de algum tempo novamente. Em virtude desta limitação, foi decidido adquirir cinco *routers*, de maneira a evitar que o trabalho fosse interrompido, e uma vez que já se tinha iniciado o trabalho naqueles equipamentos, optou-se por adquirir equipamentos semelhantes de maneira a potenciar o trabalho já desenvolvido.

Ao realizarem-se testes, por lapso danificou-se um dos *routers* do ISEP, o que reforçou ainda mais a decisão da aquisição dos *routers*, já que foi necessário por várias vezes soldar pinos nas portas série e JTAG, para proceder ao desbloquemento dos *routers*, que estando sujeito à experimentação de imagens de *firmware*, acabaria inevitavelmente por bloquear, não permitindo a sua recuperação através dos botões de *reset*. A abertura e soldadura dos equipamentos do ISEP, assim como a instalação de outros *firmwares*, provocaria a perda da garantia oferecida pelos fabricantes, o que tornou inviável o prosseguimento da utilização dos equipamentos disponibilizados pelo ISEP.

Na realização do trabalho aqui apresentado, os equipamentos utilizados para a instalação do *firmware*, foram *routers Linksys* WRT54GS, equipados com *chipsets Broadcom*, com velocidades de CPU superiores a 200 *MHz*, com 8 *MB* de *flash* e com 32 ou 64 *MB* de Random Access Memory (RAM). Foram configuradas quatro unidades, que constituíram uma pequena rede *mesh*.

O *firmware* escolhido para a prossecução deste trabalho foi o *firmware OpenWRT*. Foi necessário abordar a questão do *hardware* indicado para a utilização do *firmware*, assim

como foi necessário explorar e descrever todo o processo inerente à criação do *firmware*, desde as opções que podem ser feitas até à sua instalação e configuração.

Foi necessário criar doze versões de *firmware*, até chegar à versão final estável e funcional, que está instalada nos equipamentos, versão essa usada para fazer os testes da rede BATMAN-ADV, apresentados neste documento.

5.1. *FIRMWARE OPENWRT*

OpenWRT é uma distribuição *Linux* para sistemas embebidos (principalmente *routers*) [52]. O projeto *OpenWRT* começou quando a empresa *Cisco* publicou debaixo de *General Public License* (GPL) o *firmware* do seu *router* sem fios *Linksys* WRT54G. Uma comunidade de criadores/programadores usaram este código para criar versões derivadas. Inicialmente, o suporte era limitado às séries WRT54G, mas com o tempo, foi estendida a muitos outros *chipsets* e arquiteturas. Foi um acontecimento importante para as redes de comunicação sem fios com fins não comerciais porque estas distribuições oferecem muitas funções que não são encontradas nos *routers* comuns [2].

Algumas das suas mais importantes características são:

- Sistema de ficheiros gravável, que permite aos utilizadores adicionar, remover ou modificar qualquer ficheiro.
- Um sistema de gestão de pacotes "*opkg*", similar ao "*dpkg*" ou "*pacman*" para sistemas operativos *Linux*.
- Um repositório de pacotes de instalação com cerca de 2000 pacotes.
- Um sistema de atualização – *Sysupgrade*, que preserva as configurações quando se fazem atualizações do *firmware*.
- Uma ferramenta que visa simplificar a configuração de todo o sistema – *Unified Configuration Interface* (UCI).
- A possibilidade de configurar a rede em *Virtual Local Area Networks* (VLANs).
- A capacidade de filtrar, manipular, atrasar e rearranjar pacotes de rede, através de métodos customizáveis.
- Uma *interface web Asynchronous Javascript and XML* (AJAX), graças as projeto *LuCI*.
- A possibilidade de configuração dos botões de *hardware*.

- Atualizações e reparações de *patches* (*patch* – excerto de código criado para reparar/solucionar problemas detetados posteriormente), mesmo em aparelhos que já não são suportados pelos seus fabricantes.
- Suporte para dispositivos de *3rd Generation* (3G), impressão, partilha de ficheiros, áudio/vídeo, *webcam*, etc.

A decisão de usar *OpenWRT*, como base do sistema operativo da rede *mesh* criada neste trabalho em detrimento de outras opções, foi devido à existência de vários fatores a favor. É o sistema que a maioria das comunidades *wireless* utiliza. É o sistema operativo mais difundido neste tipo de ambientes, o que faz com que exista um grande número de pessoas que o mantém, reportando falhas e adicionando novas funções ao *firmware*. Como consequência o repositório de pacotes está constantemente a ser atualizado. Também o *Software Development Kit* (SDK) fornecido pelo *Open*WRT, permite compilar fontes para diferentes arquitecturas como por exemplo x86, *Microprocessor without Interlocked Pipeline Stages* (MIPS) ou *Acorn RISC Machine*, posteriormente *Advanced RISC Machine* (ARM), o que é uma mais valia, uma vez que a maior parte dos *routers* usam um variado tipo de arquiteturas, que não a arquitetura x86. De maneira a funcionar bem nesses sistemas de baixo desempenho, o sistema é muito otimizado. Os binários são normalmente muito pequenos porque são compilados através da livraria *uClibc*, sendo libertados de funções desnecessárias, assim como as aplicações são cuidadosamente selecionadas e algumas aplicações são modificadas para serem mais pequenas. Por fim, o suporte de *wireless drivers* é provavelmente o melhor de todas as distribuições que têm como base o *Linux*. Como exemplo, no ano de 2011 foi o primeiro sistema que suportou *Independent Basic Service Set* (IBSS), vulgo modo Wi-Fi Ad-Hoc, em 802.11n e em HT40 (MIMO com canais 2x20 *MHz*) para dispositivos equipados com o *framework mac80211*, que são normalmente os mais comuns (*mac80211* é o responsável no sistema *Linux* pela implementação funcional de *hardware*/*firmware* no que toca à norma 802.11).

Existem outras opções ao *firmware OpenWRT* para a implementação de uma rede *mesh*. Muitas dessas opções são derivações do próprio *OpenWRT*. Algumas são especificamente concebidas para serem utilizadas em redes *mesh*, associadas a projetos que visam a implementação e gestão de redes *mesh*, como por exemplo *Freifunk* [53], já referido anteriormente como sendo um projeto de comunidades *mesh* na Alemanha. Outros exemplos de derivação de *OpenWRT* são *tomato* [54], *Gargoyle* [55] ou *Routing Olsr*

Batman INside (ROBIN) [56], que no âmbito deste trabalho aqui apresentado, terá uma forte influência quanto à plataforma de gestão da rede criada, conforme será descrito mais adiante. Uma distribuição de *firmware* que originalmente surgiu como uma alternativa ao *firmware* que trazia o WRT54G é *DD-WRT* [57]. O *firmware DD-WRT é* uma distribuição concebida para utilizadores finais. Permite o uso em modo Ad-Hoc, e pode ser utilizado em redes *mesh*, mas só utiliza o protocolo OLSR.

5.2. *ESCOLHA DE HARDWARE*

Existe um grande número de dispositivos que podem ser convertidos num *router mesh*. Por outro lado, o processo de converter o *firmware OpenWRT* para um *firmware mesh* através do sistema de gestão de pacotes tornou-se menos transparente, uma vez que acontecem mais erros de compilação.

Alguns fabricantes de *routers* Wi-Fi apresentam os dispositivos já com *OpenWRT*, como sendo o *firmware* de fábrica, como por exemplo *Mesh-Potato* [58], *Dragino MS-12* [59] ou *Allnet 0305* [60]. Por exemplo o *Mesh-Potato* é um dispositivo de baixo consumo de energia para ambientes exteriores concebido para redes *mesh* com uma porta *Foreign eXchange Subscriber* (FXS) (telefone analógico), de maneira a que se possa ligar um telefone analógico e fazer chamadas telefónicas, via rede *mesh*. O *Mesh-Potato* usa o protocolo BATMAN (L3). Um segundo *firmware* chamado de SECN (*Small Enterprise / Campus Network*) também está disponível para o *firmware Mesh-Potato*, que utiliza o protocolo BATMAN-ADV (L2), mas estes não são os únicos dispositivos que se devem considerar para redes *mesh*. O *OpenWRT* suporta uma grande variedade de *routers wireless*. Substituir o *firmware* que o *router* traz de fábrica pelo *OpenWRT*, poderá não ser fácil. Poderão ocorrer erros e os *routers* ficarem bloqueados. A lista de *routers* disponíveis é grande e não existe um método único para se fazer o upgrade de *firmware* que funcione para todos os dispositivos. A tabela de *hardware* suportada pelo *OpenWRT* é extensa e continua a expandir. Pode ser consultada em [61]. Este sítio deverá servir como base de consulta, antes de se adquirir qualquer *router* para instalação do *OpenWRT*. De salientar que os fabricantes de *hardware* podem mudar os *chips* dos dispositivos sem o declararem explicitamente. Não é garantido que novas versões de *hardware* funcionem, sem que alguém as tenha testado e reportado na *wiki OpenWRT*.

A questão da escolha dos *routers* que vão fazer parte da rede *mesh* é uma questão delicada. Em primeiro lugar, deve-se definir se se pretende uma rede 802.11 b/g ou 802.11n. A primeira permite velocidades até 54 *Mbit/s* e a segunda permite velocidades até 300 *Mbit/s.* Após esta escolha deve-se ter em conta que a memória *flash*, ou seja a memória onde vai ser guardado o *firmware*, nunca deve ser menor que 2 *MB*, de preferência deve ser sempre acima de 4 *MB*, isto porque poucos *firmware* cabem em 2 *MB*, e com menos de 4 *MB* poucos pacotes de *software* adicional podem ser instalados. Uma memória *flash* de 8 *MB* é uma mais valia e se for superior é excelente. No entanto, é possível aumentar tanto as memórias *flash* como as memórias RAM, desde que se encontrem *chips* compatíveis, as ferramentas necessárias e aptidão para substituí-las. No caso de se proceder à alteração de memórias, é necessário alterar os parâmetros do *Common Firmware Environment* (CFE), que se trata do *bootloader* que faz arrancar o sistema operativo do *router*. Relativamente aos *chips*, existem vários no mercado, no entanto dois dos mais relevantes são *Atheros*, já referido anteriormente e *Broadcom*. Ambos têm suporte *OpenWRT*. Relativamente a este tema, as opiniões dividem-se. Por um lado, os utilizadores das placas *Atheros* são unânimes ao considerarem os *chips* rádio mais potentes que *Broadcom*, no entanto para *chips Broadcom* o suporte *Linux* é muito sólido e é garantida a estabilidade na implementação. A memória RAM no mínimo deve ser de 16 *MB.* 32 *MB* é bom e acima de 64 *MB* é excelente. Quanto ao CPU, a frequência mínima do relógio deve ser de 200 *MHz*, acima de 240 *MHz* é bom e acima de 400 *MHz* é excelente. Após estas considerações é que se deve proceder à escolha do *router*. Por razões económicas e técnicas, uma opção poderá ser o *Linksys* WRT54G com chip *Broadcom* e 4 *MB* de *flash*. Outras boas soluções poderão pertencer às marcas *Asus* e *Buffalo*. Um router *Linksys WRT54GL* em Portugal poderá custar € 45 (euros) e um *Buffalo WHR-HP-G300N* poderá custar € 55 (euros). O *WRT54GL* tem *chip Broadcom*, 4 *MB* de memória *flash*, 16 *MB* de memória RAM e 200 *MHz* no CPU, enquanto que o *Buffalo WHR-HP-G300N* (que já traz instalado o *firmware DD-WRT*, o que facilita a mudança para o *firmware OpenWRT*) tem *chip Atheros*, 4 *MB* de memória *flash*, 32 *MB* de memória RAM e 400 *MHz* no CPU. Estas considerações são determinantes quando se pretende construir uma rede *mesh*, de maneira a calcular o custo dos equipamentos que farão parte dessa rede. A escolha é vasta e estes exemplos devem ser tidos em consideração no sentido do raciocínio a fazer na aquisição de *routers* para equipar uma rede *mesh.*

5.3. CRIAR O *FIRMWARE OPENWRT*

O *firmware OpenWRT* pode ser construído conforme as necessidades de quem o vai utilizar. Embora sejam disponibilizadas imagens padrão do *firmware* já construídas para várias arquiteturas e modelos de *routers*, este pode ser personalizado. Para isso é disponibilizado o *OpenWRT Buildroot* que se trata de um conjunto de *makefiles* e *patches* que permitem gerar uma ferramenta de compilação integrada com um sistema de ficheiros de raiz para *routers* sem fios.

Atualmente é possível compilar a versão estável *Backfire10.03.2*, ou a versão *trunk - Attitude Adjustment12.09-beta*, que se trata da versão em testes experimentais. O primeiro passo consiste em fazer o *download* de uma cópia de uma das derivações das fontes do *OpenWRT*, através do seu repositório *Subversion*, muitas vezes abreviado SVN [62].

Para a compilação do *OpenWRT* é utilizado um sistema onde se pode selecionar qual a coleção de pacotes a integrar essa mesma compilação. Chama-se *feeds system* e está localizado num local comum (utiliza um servidor remoto ou um sistema de ficheiros local) [2]. Para gerir esses pacotes de instalação, o SDK providencia uma utilidade localizada em *scripts/feeds* que permite instalar, remover ou atualizar todos os pacotes de instalação.

Na Figura 22, apresenta-se o ecrã de abertura de compilação da versão experimental do *OpenWRT Attitude Adjustment* na sua versão atual, onde se podem observar os menus principais de configuração disponibilizados. Estes menus crescem ou decrescem conforme os pacotes solicitados no *feeds system*, referido anteriormente.

Conforme se pode observar é possível escolher a arquitetura do *chip* do *router*, onde a imagem vai funcionar, é possível determinar que formato de imagem se pretende compilar, assim como os pacotes de utilitários a incorporar e os pacotes que fazem parte do sistema base, também se podem incluir utilitários para IPv6, assim como a *interface* gráfica *LuCI*.

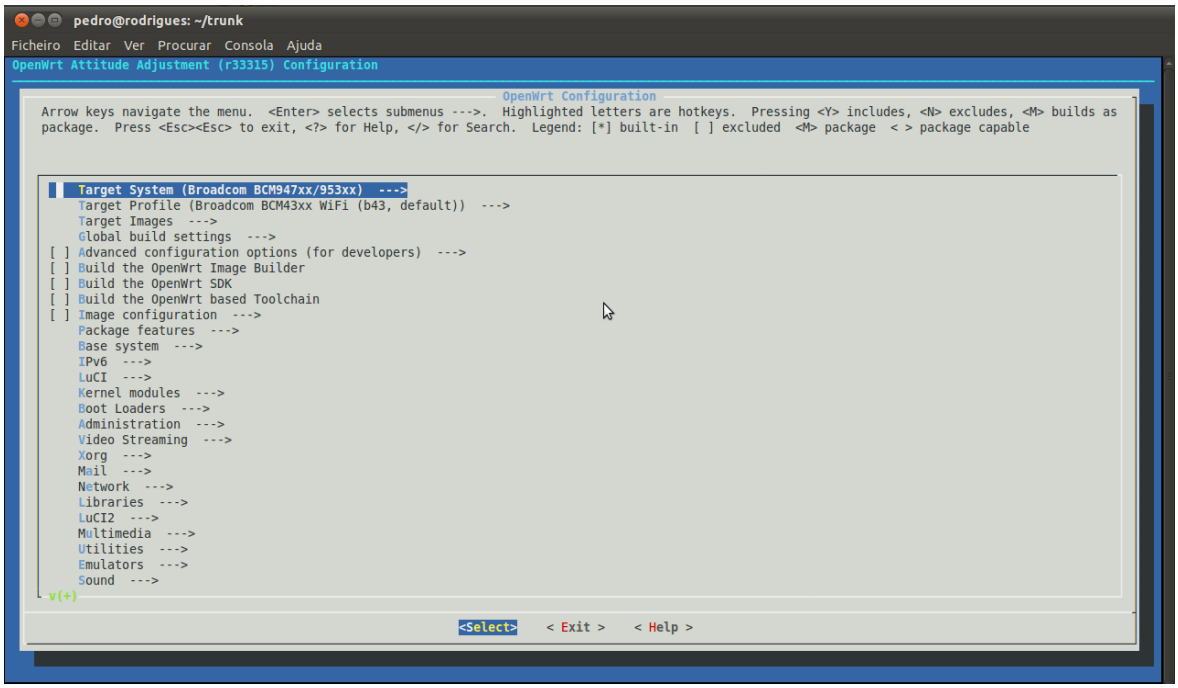


Figura 22 Ecrã de entrada do sistema de configuração do *firmware OpenWRT*.

Entre aplicações de criptografia, módulos de gestão de *Light Emitting Diodes* (LEDs), suporte *Universal Serial Bus* (USB), vídeo ou VoIP, é na opção referente a *Kernel modules* que se encontra o suporte para o protocolo BATMAN-ADV. Em *Administration* é possível incluir aplicações de administração de redes como *zabbix* ou *nagios*. Opções para tratamento de *e-mail*, também são disponibilizadas, assim como uma panóplia de ferramentas para *routing*. Em *Network* encontram-se as opções para suporte dos protocolos BABEL, BATMAN (L3), BMX6 ou OLSR. Ainda é possível incluir interpretadores de linguagens de programação como por exemplo *Erlang*, JAVA, *Lua*, PHP, *Perl*, *Python*, *Ruby*, etc. Todo o tipo de utilitários podem ser incluídos na imagem a compilar, com o consequente aumento de tamanho e também com a consequente degradação do desempenho do *router*. A escolha deve ser ponderada e devem ser pensados os custos/benefícios na inclusão de pacotes para compilação.

Na Figura 23 apresenta-se o ecrã, onde é possível selecionar vários componentes de protocolos *mesh*.

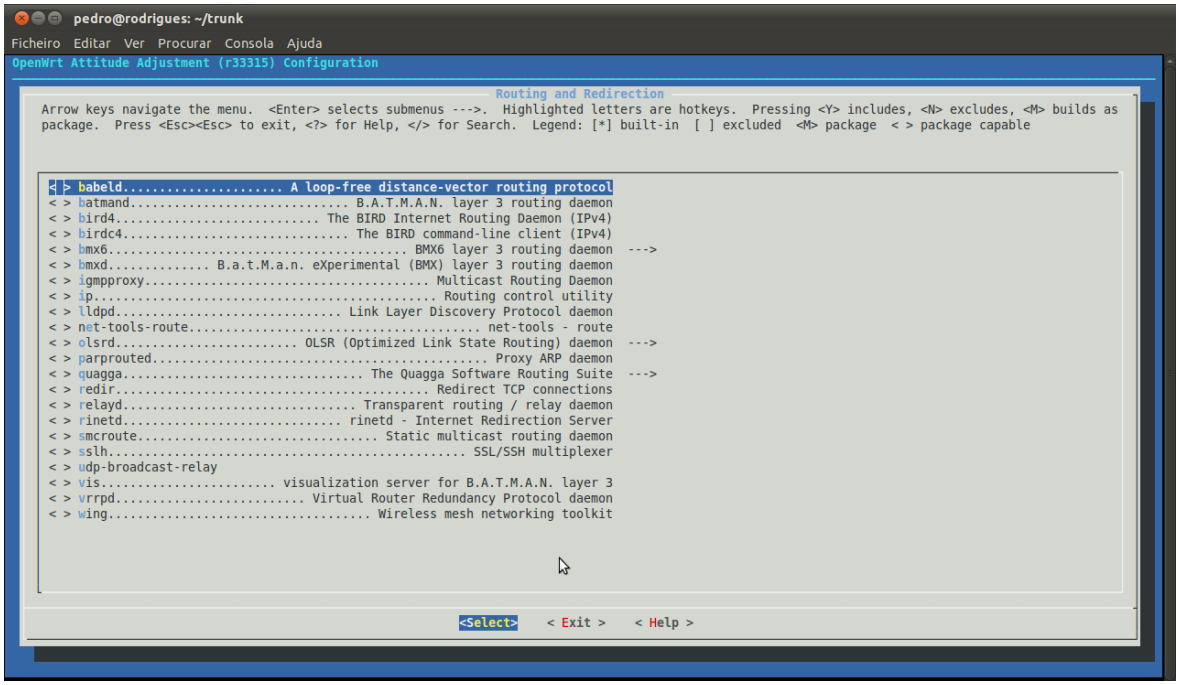


Figura 23 Ecrã de *Routing and Redirection* de *OpenWRT*.

No anexo A é descrito o processo de criação de imagem do *firmware OpenWRT*, pormenorizadamente [A].

5.4. *INTERFACE* GRÁFICA DO *OPENWRT*

Tanto o *OpenWRT Backfire10.03.2*, como *Attitude Adjustment 12.09-beta*, dispõem de uma *interface* gráfica de nome *LuCI*.

A *interface* gráfica *LuCI* trata-se de uma *Application Programming Interface* (API) escrita na linguagem de programação *Lua*. Esta *interface* permite configurar a rede e disponibiliza o uso de algumas ferramentas como *traceroute* ou *ping*, assim como gráficos de tráfego ou de carga em tempo real. A Figura 24 apresenta um ecrã deste *interface* gráfico.

Para administração do protocolo BATMAN-ADV, esta *interface* não disponibiliza qualquer ferramenta. Assim a administração tem de ser feita através de uma *shell*, usando para isso a aplicação terminal/consola existente nos sistemas operativos, através do protocolo de rede *telnet* ou *Secure SHell* (SSH), conforme se pode observar na figura 25.

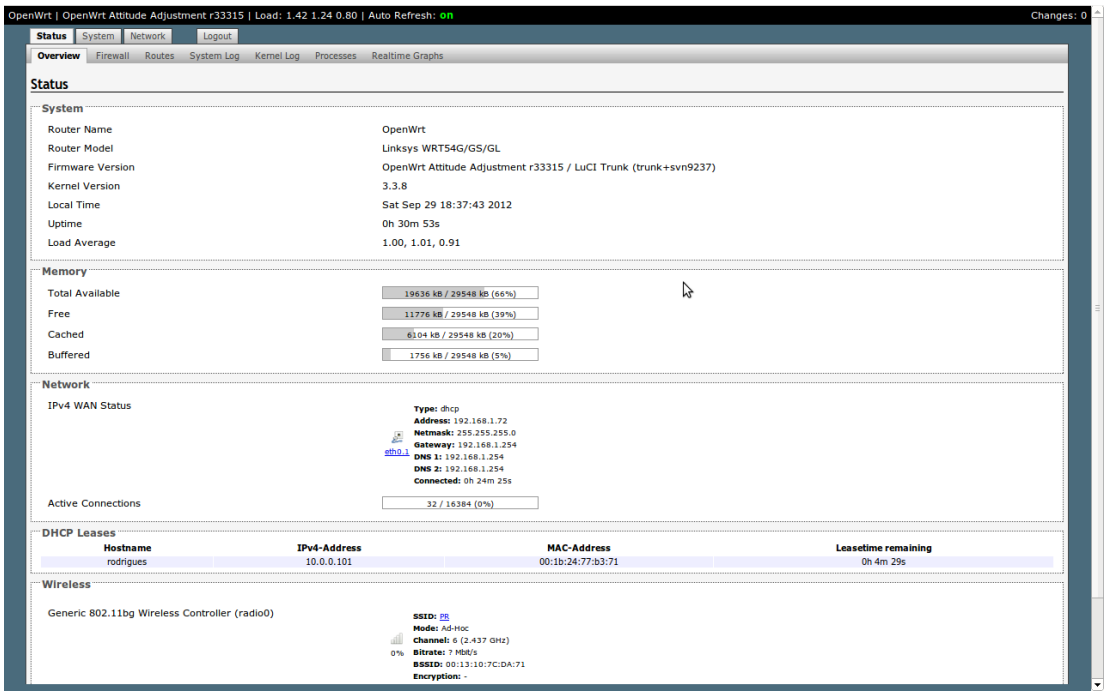


Figura 24 Ecrã de entrada da *interface* gráfica *LuCI*.

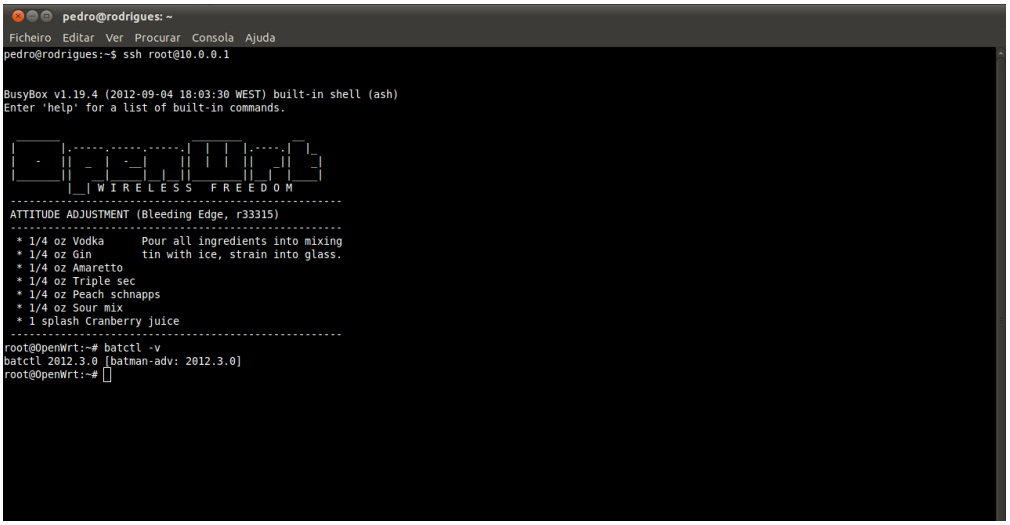


Figura 25 Ecrã de acesso ao *router* através de SSH, com instalação *OpenWRT*.

5.5. INSTALAÇÃO DO *FIRMWARE OPENWRT*

Tendo sido criada a imagem a instalar no *router*, é preciso proceder à substituição do *firmware* que o *router* possui. Qualquer *firmware* estará apto para que se possa fazer uma atualização (*upgrade*). Para isso, por norma, encontra-se disponível uma página que normalmente pode ser acedida através de um navegador (*web browser*), onde essa

operação estará disponível. Se ao tentar-se atualizar o *firmware* de origem para o *firmware OpenWRT* for apresentada alguma mensagem a indicar essa impossibilidade ou a indicar que a extensão do ficheiro não é compatível, poderá ser necessário fazer o *upgrade* para um *firmware* "intermédio" e então depois, a partir desse *firmware*, fazer o *upgrade* final para o *firmware OpenWRT*. Um *firmware* que funciona muito bem para ser utilizado nesse passo intermédio é o *firmware DD-WRT*, que se trata de um *firmware* baseado no *OpenWRT*, mas que permite uma mais fácil instalação, quando se trata de uma primeira alteração do *firmware* de origem.

Se este método não resultar então poderá ser necessário fazer a atualização do *firmware*, usando uma ferramenta de transferência de ficheiros para o *router*. Uma ferramenta disponível para os sistemas operativos *Linux* e *Microsoft Windows*, chama-se *Trivial File Transfer Protocol* (TFTP). Quando o *router* é ligado, normalmente fica à "escuta", ou seja, à espera de receber ficheiros. É uma opção que é definida em função do tempo, e quando esse tempo passa, se o *router* não receber dados, então procede ao arranque normal do seu próprio sistema instalado. Essa opção de tempo de espera existente nos *routers*, antes que arranquem o seu sistema operativo, pode ser alterada. É possível aumentar o tempo de espera, mas por omissão o tempo é suficiente para se enviar a imagem do *firmware OpenWRT*. Para o envio da imagem é utilizado TFTP. Convém assegurar que se possui uma cópia da imagem original do *router* que vai ser atualizado, de modo a poder-se recuperar o *router*, se o *upgrade* falhar. As empresas que comercializam os *routers* costumam disponibilizar nos seus sítios na *internet*, as imagens do *firmware* instalado nos seus produtos.

Mais cedo ou mais tarde, bloquear-se-á um *router* com testes de *firmware* ou com atualizações de *firmware*. Na maior parte dos casos é possível recuperar o *router* e retomar o desenvolvimento/testes. Para tal são necessárias determinadas ferramentas e técnicas. Existem várias formas de desbloquear ou aceder a um *router*, no entanto quando tudo o resto falha e não se consegue aceder ao *router* a partir da consola do computador resta-nos a porta série ou JTAG. Nem todos os *routers* possuem estas portas, alguns possuem só uma delas e outros equipamentos não possuem nenhuma. Foi necessário desbloquear *routers* várias vezes, devido a variadas instalações de *firmware*, até se chegar à solução final apresentada neste trabalho.

No anexo B é descrito o processo de *upgrade* de imagem do *firmware OpenWRT*, utilizando TFTP [B].

5.6. CONFIGURAÇÃO DO *OPENWRT*

Existem vários parâmetros a configurar de maneira a criar uma rede *mesh*. As configurações podem ser variadas.

Numa primeira fase foi necessário proceder à configuração e instalação das chaves de autenticação, de maneira que sempre que se solicite uma conexão com qualquer um dos nós, usando SSH, não seja necessário introduzir a *password*.

Os ficheiros a configurar encontram-se no sistema de ficheiros do *OpenWRT*, mais precisamente em */ect/config/*. Nesta diretoria será necessário configurar um conjunto de ficheiros, nomeadamente os ficheiros *batman-adv, dhcp, network, wireless* e *system*.

Existem nós servidores/*gateways* e nós clientes. As configurações dos nós são diferentes em função da sua natureza.

Foram definidas as seguintes configurações do protocolo BATMAN-ADV:

- A agregação de OGMs.
- O isolamento de APs.
- O modo de alternância de *interfaces* (*bonding*).
- O modo de fragmentação de pacotes.
- A largura de banda.
- O valor do intervalo de omissão de OGMs.
- O modo de visualização de gráficos.
- A habilitação do mecanismo para evitar voltas de pacotes desnecessárias (*bridge loop avoidance*).

A configuração da rede é feita no ficheiro *network*, onde é definido o IP de cada nó e criada a *interface* BATMAN-ADV. No ficheiro *dhcp* é definido a atribuição de IPs, no caso dos nós servidores.

No ficheiro de configuração *wireless*, são definidos:

- O canal da rede *mesh*.

- As configurações da *interface* rádio quanto a definição de *Service Set IDentifier* (SSID) e *Basic Service Set IDentification* (BSSID).
- O modo de operação da rede e de encriptação.

Por fim foi configurado o ficheiro *system* de forma a habilitar o *Network Time Protocol* (NTP), permitindo que todo os nós sincronizem data e hora.

No anexo C, é descrito o procedimento de configuração de chaves de autenticação [C].

No anexo D, é descrito o procedimento pormenorizado de configuração do *firmware OpenWRT* [D].

5.7. CONFIGURAÇÃO DOS APS

Para os utilizadores da rede *mesh* BATMAN-ADV se poderem ligar via *wireless* à rede, não o poderão fazer diretamente através dos nós BATMAN-ADV. Para se obter uma ligação à rede é necessário fazer a ligação a um AP, que estará agregado a um nó BATMAN-ADV. Cada nó BATMAN-ADV poderá ter um AP agregado, mas para o próprio funcionamento da rede *mesh* isso não será necessário, pois os nós BATMAN-ADV constituem a rede, por si só. A colocação de um AP agregado a um nó BATMAN-ADV só faz sentido se existir na sua área de influência utilizadores que acedam à rede via *wireless*. É possível cada nó BATMAN-ADV fornecer ligação via *Ethernet* a cinco utilizadores, usando para isso as cinco portas do *router* ou até mais utilizadores, através da utilização de um *switch*. A utilização de um AP não é obrigatória, para se ter acesso à rede. Mesmo em ambientes onde não seja possível passar cabos, o sinal poderá ser transmitido via *wireless* através dos nós BATMAN-ADV e podem localmente ser ligados utilizadores através de cabo.

No anexo E, é descrito o procedimento de configuração de APs, pertencentes à rede BATMAN-ADV [E].

5.8. CONSIDERAÇÕES SOBRE POSICIONAMENTO DOS NÓS

No âmbito deste trabalho foi necessário ponderar a localização dos nós da rede *mesh*. Seguidamente são apresentadas algumas considerações que determinaram o posicionamento dos nós da rede.

Para se definir o posicionamento dos nós é necessário perceber o alcance dos seus sinais, de maneira a que nenhum nó fique isolado e que o alcance do sinal chegue onde é expectável. Esta perceção é importante em termos de custos e de eficácia. Se uma rede for bem planeada, não serão precisos mais nós do que os estritamente necessários, assim como será garantido o serviço ao utilizador que se encontra fisicamente mais distante do alcance da rede.

Existem vários tipos de *software* para deteção e análise de redes *wireless* [8]. Para *Microsoft Windows* existe o *NetStumbler* ou *VistSlumber*. Para *Linux* existe o *Kismet* ou *Gkismet*. Para *Mac OS X* existe *KisMAC* e *iStumbler*. O *software* utilizado na conceção da rede criada neste trabalho foi o *inSSIDer*, que funciona na plataforma *Microsoft Windows*. Sabendo que o sinal mínimo, tipicamente, deverá se encontrar entre -75 a -95 *dBm* [24], para garantir que um nó BATMAN-ADV consegue comunicar com outro nó BATMAN-ADV, procedeu-se a implementação da rede *mesh*. Na Figura 26 pode-se observar que as potências de sinal emitidas pelos nós BATMAN-ADV, referenciado com o SSID "PR" e pelo nó AP, referenciado com o SSID "batman-adv-----AP", encontram-se no limiar do valor considerado aceitável para a receção do sinal. Neste local onde foram registados estes valores, seria necessário colocar outro nó BATMAN-ADV, assim como outro AP, se se pretendesse fazer chegar o sinal via *wireless*, a utilizadores mais afastados desta área.

Em espaços exteriores, terá que se ter em consideração outros fatores. Poderá ser necessário instalar antenas com ganhos de potência maiores. Para isso poderá ser calculado o custo da ligação (em inglês, *link budget*) [24]. Este cálculo terá que ter em consideração as características dos equipamentos e a perda de sinal. As características a ter em conta são:

- A potência de transmissão que é expressa em *mW* ou em *dBm*.
- O ganho da antena, expresso em *dBi*.
- O nível mínimo de sinal recebido – *Received Signal Level* (RSL), que é expresso em *dBm* (*-dBm*). O valor mínimo típico encontra-se entre -75 a -95 *dBm*, conforme já referido anteriormente.

As perdas nos cabos, dependem dos cabos usados e do seu comprimento. Por exemplo, as perdas nos cabos coaxiais com pouco comprimento, incluindo conectores são bastante baixas, na ordem dos 2 a 3 *dB*.

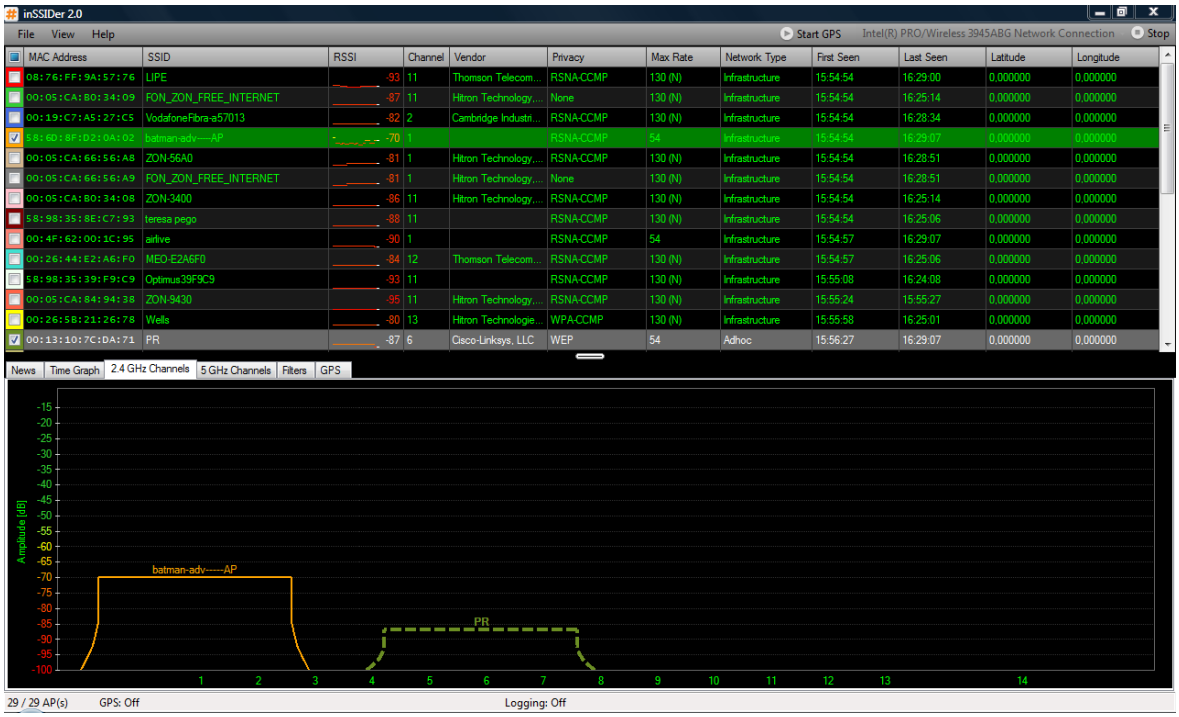


Figura 26 Utilização do programa *inSSIDer*, na análise do sinal de rádio emitido por um nó BATMAN-ADV (PR) e por um nó AP (batman-adv-----AP).

Outros fatores a ter em conta são as perdas de sinal no caminho a percorrer (perdas em espaço livre), a atenuação e a dispersão de sinal. A potência do sinal diminui com a propagação geométrica da onda frontal, fenómeno conhecido como "*free space lost*". Este fenómeno acontece devido à expansão da energia do sinal erradiado em função da distância ao transmissor [24]. Outro aspeto a ter em conta é a atenuação, que é causada pela absorção de potência quando as ondas passam através de objetos sólidos, como por exemplo árvores, paredes, janelas ou pisos de edifícios. Já a dispersão é um fenómeno que tem origem no desfasamento da receção da energia da rádio-frequência no recetor. Isto deve-se ao espalhamento da energia quando esta deixa a antena transmissora, provocando que a energia seja refletida no chão e noutros obstáculos e chegue com um atraso à antena recetora. Este fenómeno pode provocar a dispersão do sinal, ou seja, o sinal pode ser diminuído ou anulado completamente, o que por vezes impossibilita a sua recuperação.

Se for montada uma rede *mesh* que vise cobrir áreas maiores de extensão e ao ar livre, será necessário fazer um estudo mais aprofundado, tendo em conta estas considerações.

5.9. CONSIDERAÇÕES SOBRE SEGURANÇA

No âmbito deste trabalho foram ponderadas questões de segurança da rede *mesh*. Seguidamente são apresentadas algumas considerações que tiveram de ser analisadas e que determinaram as opções tomadas em relação à rede criada.

Existem várias hipóteses de se tentar proteger uma rede *wireless* de intrusos. No caso apresentado neste trabalho, verificou-se que o *firmware OpenWRT* (utilizado nos nós BATMAN-ADV) permite encriptação WEP para redes Ad-Hoc. A encriptação WEP, embora seja facilmente decifrada, oferece alguma proteção. Contudo, utilizando esta encriptação, não se pode evitar de todo, a intrusão na rede *mesh*. Replicando a instalação do *firmware OpenWRT* e associando o MAC *address* definido para a rede, que pode ser facilmente capturado, pode-se associar um novo nó BATMAN-ADV à rede, sem conhecimento do administrador. A solução passa pela alteração da chave de encriptação frequentemente, assim como uma monitorização mais atenta da rede. Quanto à encriptação utilizada no AP, usou-se WPA2, com TKIP+AES, que até à data ainda não foi quebrada, oferecendo segurança quanto à autenticação dos utilizadores da rede.

Existem outras soluções comuns para segurança de redes *wireless*. Uma delas é a filtragem de MACs [24]. Esta filtragem pode ser utilizada nos APs, ou seja, quando um utilizador tenta aceder à rede, se o seu MAC *address* não estiver registado, o acesso é negado. Ou pode-se restringir utilizadores, permitindo o acesso a todos os outros, exceto aos utilizadores que tenham o seu MAC registado na filtragem. No entanto, os MACs *addresses* podem ser forjados, alterando o CFE ou simulados por aplicações de *software*. A aplicação de filtragem de MACs *addresses* está disponível no *firmware OpenWRT*, no entanto essa opção não foi testada nos nós BATMAN-ADV. Outra solução possível consiste em esconder o SSID da rede, originando o desconhecimento desta, por parte de quem não sabe da sua existência. Ao inibir-se o SSID, não são enviados "*beacons*" (sinalização de presença) pelos nós. Isto previne que utilizadores casuais tomem conhecimento da rede. No entanto, a opção de esconder o SSID pode provocar que inconscientemente seja instalada uma rede, precisamente no canal da rede escondida, o que pode provocar interferências em ambas as redes com prejuízo comum a ambas as partes. Também podem ser detetadas pelos programas de deteção e análise de rede, tramas que sejam enviadas do utilizador para o AP, tramas essas que necessariamente contêm o nome da rede, o que pode provocar que seja tentado o acesso à rede por utilizadores estranhos.

Outra solução existente é conhecida como portal de entrada na rede (*captive portal*). Estes portais servem para a autenticação do utilizador e utilizam um *web browser* que oferece a oportunidade de introdução de credenciais de acesso. Ao utilizar um *web browser* para autenticação, os *captive portals* funcionam como qualquer outro sistema de autenticação usado na *internet* e nos sistemas operativos. Este sistema baseia-se no MAC e IP dos utilizadores, mas conforme já foi dito, estes identificadores podem ser manipulados. Em ambientes públicos ou semi-públicos, as encriptações WEP e WPA são desadequadas, uma vez que se torna impossível partilhar chaves a todos os utilizadores, sem comprometer a segurança. Neste contexto a solução *captive portal* faz sentido, ao fornecer um nível de serviço algures entre um sistema de autenticação completamente aberto e um sistema de autenticação completamente fechado.

Estas considerações de segurança, na conceção de uma rede *mesh*, são pertinentes. Consoante o tipo de rede, seja privada, semi-privada ou pública, poder-se-á optar por alguma solução em particular ou juntar várias soluções de modo a reforçar a sua segurança.

6. SOLUÇÃO PROPOSTA PARA A GESTÃO DA REDE *MESH*

Nesta fase do trabalho e após a configuração dos nós BATMAN-ADV, tornou-se necessário administrar e gerir uma, ou mesmo várias redes *mesh*. Essa administração e gestão terá de ser automatizada, de forma a que, a partir de uma só localização e usando uma só plataforma, seja possível em tempo real, gerir redes *mesh*. Para isso o administrador (ou administradores) necessita de uma ferramenta que lhe pemita configurar a rede no seu todo e os nós individualmente. Além disso precisa de um mecanismo que o ajude a fazer essa gestão, informando-o quando acontece algum problema na rede.

Neste capítulo será apresentado o trabalho realizado para administrar a rede criada. Desenvolveu-se uma ferramenta que poderá ser replicada e instalada onde existam redes *mesh*, cujo protocolo de *routing* seja BATMAN-ADV. Esta ferramenta pode continuar a ser desenvolvida e adaptada ajustando-se ao desenvolvimento das próprias redes. Apesar de se ter tentado cobrir vários aspetos da gestão, poder-se-á implementar novas funcionalidades, utilizando sempre como base de trabalho, a ferramenta aqui apresentada.

6.1. PLATAFORMA DE GESTÃO DE REDE BATMAN-ADV, BASEADA NO *SOFTWARE ORANGEMESH*

Tendo por base o *software Orangemesh* [22], desenvolveu-se uma ferramenta de gestão para as redes *mesh*, que utilizam o protocolo BATMAN-ADV. Este *software Orangemesh* foi o único *software open source* encontrado, publicado debaixo de *General Public License* (GPL), sendo o seu código disponibilizado de uma forma gratuita. Em virtude de apresentar algumas funcionalidades já desenvolvidas aproveitou-se a base desse código, a sua arquitetura, nomeadamente os *scripts* de criação da rede, os *scripts* de adição e remoção de nós, os *scripts* de manipulação de dados desses mesmos nós e os *scripts* de criação de mapas para a criação da ferramenta de gestão de redes *mesh* realizada neste trabalho. Outros *scripts* foram criados de raiz, como por exemplo o *script* de comunicação com os nós BATMAN-ADV, ou o *script* que permite a atualização dos valores de utilização da rede a cada 24 horas ou o *script* de encriptação da rede. Foi necessário criar a *interface* de administração dos nós BATMAN-ADV, assim como foi necessário reformular a página de administração da rede e a página de apresentação das características e de estado dos nós. Foi ainda preciso pensar numa alternativa à colocação de nós no mapa, como por exemplo num edíficio de muitos andares, onde o espaço de implementação de nós no mapa é exíguo para poder aglomerar todos os nós. Foi necessário pensar em como apresentar graficamente o estado das ligações e o estado dos *routers*, quanto à sua carga de trabalho, uma vez que esta função na versão disponibilizada do *software Orangemesh*, não existia. Em relação à base de dados criada no *My Structured Query Language* – MySQL, foi necessário reformular vários campos da base de dados, em função das necessidades da rede BATMAN-ADV e em função das variáveis criadas. Foi criado também um sistema de registos (*logs*) de todas as comunicações entre o servidor da plataforma de gestão e os nós BATMAN-ADV e das alterações efetuadas nesses mesmos nós e na rede.

A filosofia da gestão da rede *mesh* com o *firmware* ROBIN, utilizado no *software Orangemesh*, consiste numa comunicação periódica por parte dos nós, onde cada nó faz o "*check in*" de cinco em cinco minutos, fazendo um pedido HTTP ao *script* que reside no servidor, onde está instalada a plataforma de gestão e a base de dados. O pedido HTTP inclui um conjunto de variáveis *Common Gateway Interface* (CGI) (por exemplo, MAC *addresses*, número de saltos para a *gateway*, etc.), enviadas através do método GET. O *script* que reside no servidor grava as informações de estado dos nós na base de dados e envia uma resposta HTTP para o nó, que contém informação sobre as configurações da

rede, como por exemplo o valor limite de *download* por nó. Nesse caso concreto o nó receberia essa informação e atualizaria a sua configuração interna de maneira a assegurar que esse limite seria cumprido [22].

Para a plataforma de gestão da rede BATMAN-ADV, optou-se por criar um *script* no servidor que comunica com os nós de vinte em vinte minutos. Os nós respondem ao servidor, cada vez que lhe é solicitada essa intervenção. Essa informação é gravada na base de dados e interpretada de maneira a ser apresentada. O código mantém-se escrito em PHP, *Javascript*, CSS e HTML. Mantêm-se a utilização do *Google Maps* API, do servidor *web Apache*, do interpretador de linguagem PHP e da base de dados MySQL.

Foi designado como nome da plataforma de gestão de redes *mesh* criada: *batman-adv-admin*.

6.1.1. INSTALAÇÃO E IMPLEMENTAÇÃO DA BASE DE DADOS, INTERPRETADOR DE PHP E SERVIDOR *WEB*

No projeto *Orangemesh* é utilizada uma compilação de *software open source* denominada *XAMPP*, que consiste na base de dados MySQL, no servidor *web Apache* e nos interpretadores para linguagens de *script*: PHP e *Perl*. No entanto verificou-se que com a utilização de *XAMPP*, na manipulação de dados da rede BATMAN-ADV ocorriam bastantes erros, devido a problemas de permissões relativamente a leitura e escrita de ficheiros. Assim para a implementação da base de dados e da sua manipulação e interação com a linguagem PHP, foi utilizada a ferramenta *phpMyAdmin*, que é uma ferramenta escrita em PHP e que visa o tratamento da base de dados MySQL usando para isso um *web browser*. Também foi necessária a instalação do servidor *Apache* para a visualização das páginas *web* da plataforma de gestão. A comunicação efetuada entre o servidor e os nós é feita através de SSH, usando chaves de autenticação, conforme foi descrito anteriormente.

Na Figura 27 pode-se observar o ecrã de entrada da ferramenta *phpMyAdmin*.

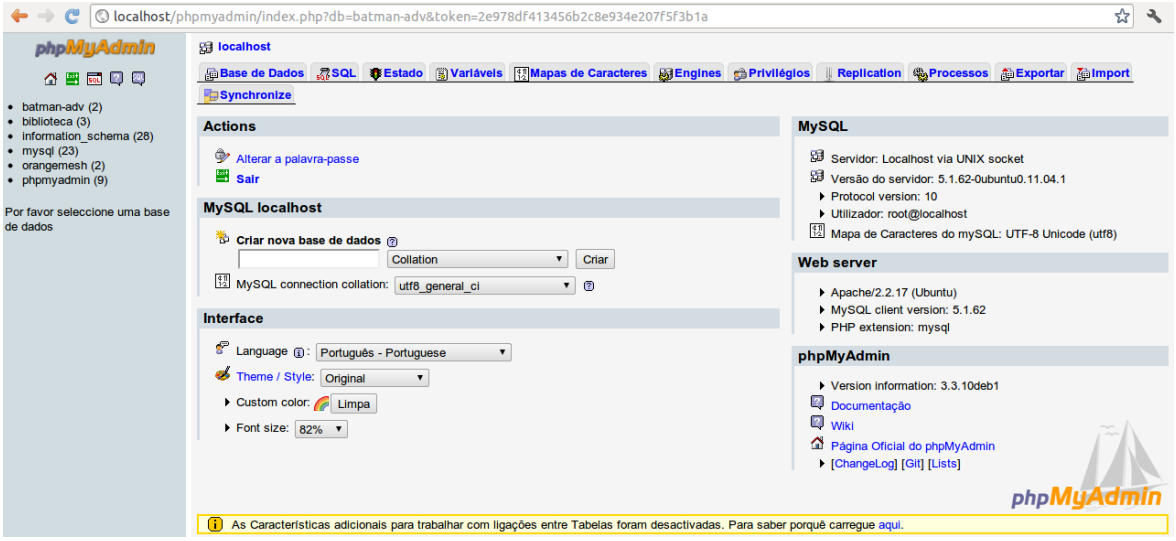


Figura 27 Écrã de entrada de *phpMyAdmin*.

6.2. ARQUITETURA DA PLATAFORMA DE GESTÃO

Em analogia com a plataforma *Orangemesh*, a plataforma *batman-adv-admin* é uma arquitetura constituída por *scripts* que utilizam os dados inseridos na plataforma de gestão. Uma parte desses *scripts* são apresentados de uma forma visual do ponto de vista do administrador (*scripts* de visualização), a outra parte são *scripts* que processam os dados enviados pelos nós, e atualizam a base de dados (*scripts* de processamento). Cada módulo é dividido em duas partes, a parte de visualização e a parte de processamento.

Na Figura 28 é ilustrado o esquema da arquitetura da plataforma de gestão de *batman-adv-admin*, onde as caixas dos *scripts* de visualização apresentam a cor verde e as caixas dos *scripts* de processamento apresentam a cor vermelha.

Os *scripts* de visualização geram uma representação visual dos dados gravados na base de dados. De uma forma geral, qualquer *script* que gere uma saída visual, será um ficheiro de visualização.

As tarefas dos *scripts* de visualização são:

- Solicitar à base de dados informação, de maneira a ser apresentada.
- Proporcionar formulários para criação e alteração de configurações.
- Apresentar dados e relacionamentos entre dados existentes na base de dados.

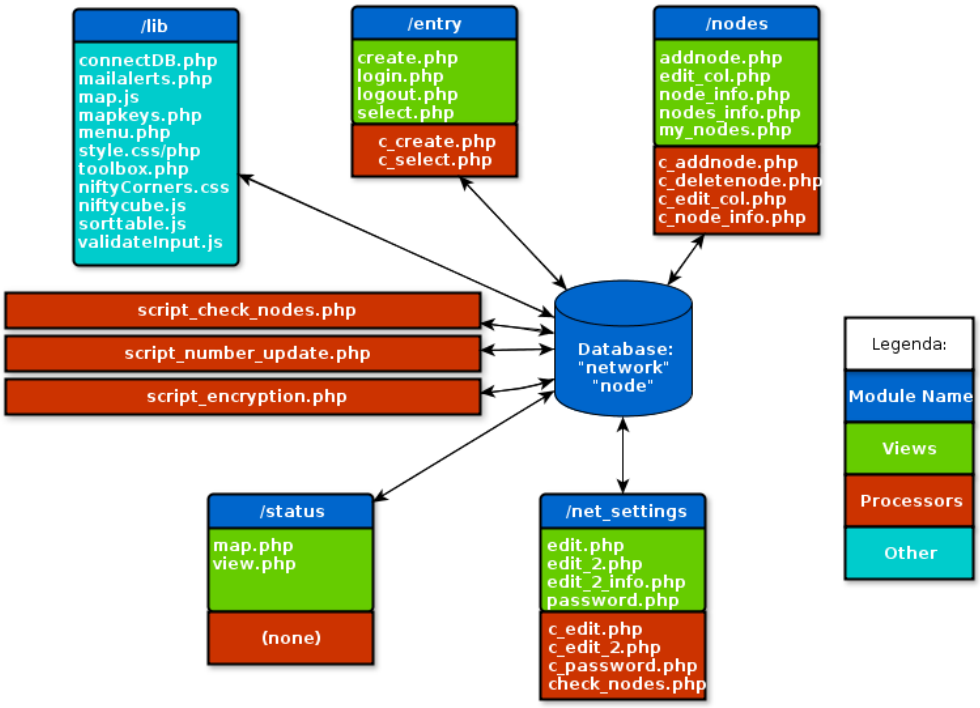


Figura 28 Arquitetura da plataforma *batman-adv-admin*.

Os *scripts* de processamento, são *scripts* para manipulação da base de dados. Alguns dos mais relevantes são o *check_nodes.php*, que é o *script* que estabelece conexão com os nós e que foi criado de raiz. Este *script* solicita informação aos nós, processa essa informação e envia também informações de atualização para os nós. Outro *script* relevante é o *c_edit_2.php*, que foi criado propositadamente, para se proceder às alterações de configurações, do protocolo BATMAN-ADV.

As tarefas dos *scripts* de processamento são:

- Iniciar contacto com os nós receber a informação solicitada e apresentá-la aos *scripts* de visualização.
- Atualizar os nós com alterações de configurações da rede.
- Alterar configurações dos nós.
- Receber dados de formulários, e escrevê-los na base de dados.

De salientar que uma plataforma pode suportar um número arbitrário de redes *mesh*.

6.2.1. COMPONENTES DA PLATAFORMA *BATMAN-ADV-ADMIN*

Conforme se pôde observar na Figura 28, a plataforma é constituída por uma base de dados com duas tabelas. A tabela *network* e a tabela *node*. A tabela *network* contém a informação acerca das redes existentes na base de dados, como por exemplo o nome da rede, a sua *password*, o *e-mail* do administrador da rede, a sua localização, etc. A tabela *node* contém informação acerca dos nós existentes em cada uma das redes, como por exemplo a descrição do nó, sua localização, suas características, seu estado, etc.

Os *scripts* de visualização e processamento estão divididos em módulos, em função das suas tarefas. Cada módulo é responsável pela criação de várias páginas *web* na plataforma de gestão. Isto permite que cada módulo possa ser modificado independentemente de outro módulo, assim como facilita a adição de *scripts* de visualização, de uma forma modular. Cada módulo encontra-se numa diretoria diferente. A organização dos módulos é a mesma existente no *software* original. Para a ferramenta utilizada neste trabalho criaram-se novos *scripts* e modificaram-se *scripts* já existentes de maneira a integrar todo o funcionamento dos nós com a plataforma. A diretoria onde está instalada a plataforma de gestão localiza-se em */var/www/*.

Os módulos que integram a plataforma *batman-adv-admin* são:
entry – Trata da seleção de rede e da autenticação do utilizador, assim como da criação de rede.
status – Permite a visualização do estado da rede. Inclui a listagem dos nós pertencentes a cada rede, assim como permite a visualização do mapa.

- Neste módulo foi alterada a informação visual e textual dos parâmetros e do estado dos nós no mapa, de forma a serem integrados com a informação obtida dos nós BATMAN-ADV. Foi criada uma representação gráfica para a qualidade de ligação entre nós, através de segmentos de reta coloridos.

net_settings – Este módulo permite o tratamento das configurações da rede e dos nós.

- Foi criado um mecanismo para se fazer *reboot* à rede automaticamente, podendo-se definir o momento em que isso acontece. Foi criada uma secção numa página *web*, onde se apresentam os valores de *downloads*, *uploads* e número de clientes referentes às últimas 24 horas. Foi criada a página *web* que permite configurar individualmente cada nó BATMAN-ADV existente na rede, quer quanto à configuração do protocolo de *routing*, quer quanto a outras configurações. Também

foi criado neste módulo um repositório de ficheiros que são necessários para se proceder à configuração dos nós (usados para edição e posteriormente enviados para os nós). Foi criado um mecanismo de aviso de impossibilidade de comunicação com um nó específico e respetiva pagina *web*. É neste módulo feita a atualização inicial, quer da base de dados, quer dos nós, sempre necessária após a criação da rede, tendo sido criado para isso um *script* de comunicação com os nós.

node – Neste módulo é possível adicionar nós individuais, assim como obter informação individualizada dos nós.

- Foi criado neste módulo a possibilidade de inserção de nós coletivos (convenção que representa vários nós individuais), para obtenção de informação e gestão de nós concentrados em altura, como por exemplo num edifício de muitos andares (devido à impossibilidade de representação desses nós no mapa). Foi criado um *script* para se fazer a gestão desses nós coletivos

lib – Providencia *scripts* para o funcionamento da plataforma. Alguns desses *scripts* são responsáveis pelo tratamento das conexões com a base de dados e pelo processamento de informação entre módulos. Também aqui é definido o estilo da plataforma e as imagens que a compõem. Os *scripts* escritos em *Javascript* residem neste módulo, como por exemplo o *Javascript map.js*, para tratamento do *Google Maps* API.

- Neste módulo foi acrescentada uma função que verifica a validade dos IPs inseridos quanto à sua forma e gama, uma vez que só existia para MAC *addresses*. Foi criada uma função que converte *bytes* para um formato mais facilmente interpretável. Foram criadas funções para a edição e manipulação de ficheiros, necessários para a configuração dos nós. Foi criada uma função para o envio e *e-mails* automáticos. Foi também alterado o menu de administração de maneira a permitir a execução das novas funcionalidades criadas. Foi alterado o *script* refererente aos estilos utilizados nas páginas *web* geradas. Foi alterado o *script* responsável pela apresentação gráfica dos nós, de maneira a que os nós individuais apresentem cores conforme o seu estado e os nós coletivos apresentem a cor azul. Também aqui foi alterada a função que apresenta informações sobre os nós.

Foram criados ainda os seguintes *scripts*:

script_check_nodes.php – É o *script* responsável pela recolha de informação e atualização dos nós periódica (de 20 em 20 minutos), assim como é responsável pelo início do procedimento de envio de *e-mails*, cuja função é a notificação de alertas.

script_number_update.php – *Script* que trata de atualizar dados relativos à utilização da rede a cada 24 horas (*downloads*, *uploads* e número de clientes).

script_encryption.php – *Script* encarregue de alterar periodicamente a chave de encriptação da rede *mesh*.

Foi criado um mecanismo de registo de todas as operações efetuadas na rede, através da criação de *logs*. Para isso foi incorporado nos *scripts* uma função que efetua a criação de uma pasta com o nome da rede e regista no seu interior todas as informações periódicas emanadas pelos *scripts*, assim como regista as alterações efetuadas por iniciativa do administrador.

Na Tabela 6 apresenta-se um resumo das funcionalidades usadas e das funcionalidades desenvolvidas na ferramenta de gestão *batman-adv-admin*.

Tabela 6 Resumo das funcionalidades usadas e desenvolvidas.

Módulo	Funcionalidades usadas	Funcionalidades desenvolvidas
entry	Formulário para criação de rede	
status	Listagem e visualização dos nós no mapa	Informação individual do estado e parâmetro de cada nó Representação e tratamento da qualidade das ligações entre nós
net_settings	Configuração da rede e dos nós (nome da conta, mudar a *password* da conta, nome da rede, *e-mail* do administrador)	*Reboot* geral à rede Apresentação de valores de *download*/*upload* e número de clientes em cada 24 horas Configuração individual de cada nó da rede, e do protocolo BATMAN-ADV Apresentação de página *web* em caso de erro de comunicação com nó *Script* de comunicação com os nós – versão integral
node	Inserção de nós individuais	Inserção e tratamento de nós coletivos
lib	Conexão com base de dados API Google Maps *Cascading Style Sheets* – CSS Funções para tratamento da informação (validação de MAC *addresses*, manipulação do tempo, etc.)	Função para validação de IPs Função de conversão de *bytes* num formato mais fácil de interpretar Função de manipulação de ficheiros de configuração dos nós Função de envio automático de *e-mails* Apresentação gráfica dos nós individuais e coletivos
		Script de comunicação com os nós e envio de alertas – versão *light*
		Script de atualização de valores da utilização da rede a cada 24 horas
		Script de atualização de chave de encriptação a cada 24 horas
		registo de *logs*

6.3. FUNCIONAMENTO DA FERRAMENTA DE GESTÃO *BATMAN-ADV-ADMIN*

Nesta secção será descrito todo o procedimento para a criação da rede e sua gestão utilizando a plataforma de gestão criada – *batman-adv-admin*. Será também descrito o funcionamento da plataforma, explicando os *scripts*, que a compõem. A apresentação dos *scripts* será feita em função dos módulos a que pertencem, uma vez que o código escrito foi dividido em módulos. Os resultados serão apresentados ao longo da secção, através de comentários e de figuras.

Na Figura 29 pode-se observar o ecrã de entrada do *batman-adv-admin*.

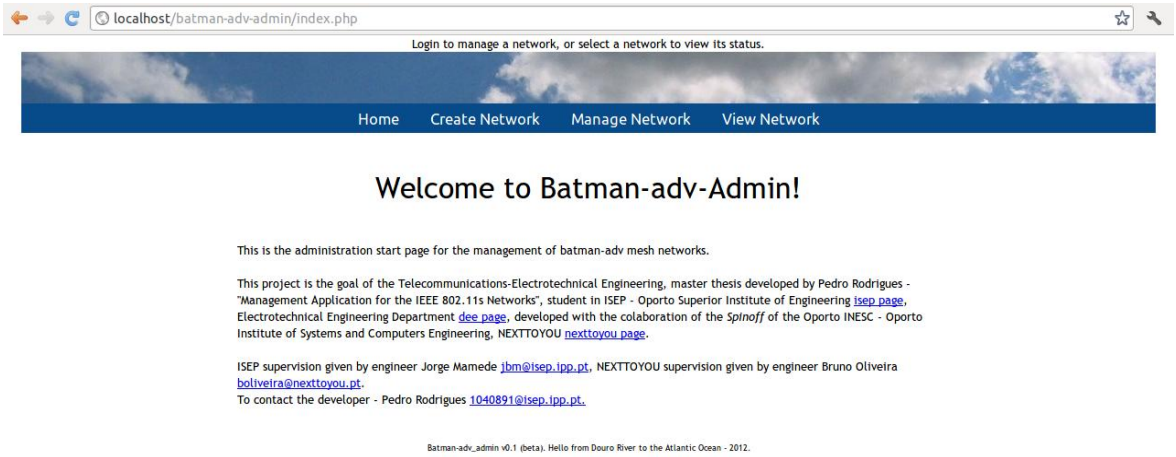


Figura 29 Écrã de entrada da ferramenta de gestão *batman-adv-admin*.

O ecrã de entrada do *batman-adv-admin* apresenta quatro opções no separador localizado no topo, nomeadamente *Home*, *Create Network*, *Manage Network* e *View Network*. O separador *Home* tem como função o redirecionamento para a página de entrada, ou seja a que se encontra apresentada na Figura 29. O separador *Create Network* tem como função a criação de uma rede. O separador *Manage Network* tem como função a administração da rede. Este separador solicita o nome da rede a ser administrada, assim como a *password* de acesso. O separador *View Network* permite ver alguns aspetos da rede, embora não permita que se faça a sua administração. Este separador solicita o nome da rede que se pretende visualizar e não solicita *password*.

6.3.1. MÓDULO *ENTRY*

Neste módulo encontra-se o *script create.php*. Este *script* é ativado quando se pretende criar uma nova rede, através do separador *Create Network* e disponibiliza um formulário onde é solicitado o preenchimento dos dados da rede.

Na Figura 30 apresenta-se o ecrã gerado pelo *script create.php*.

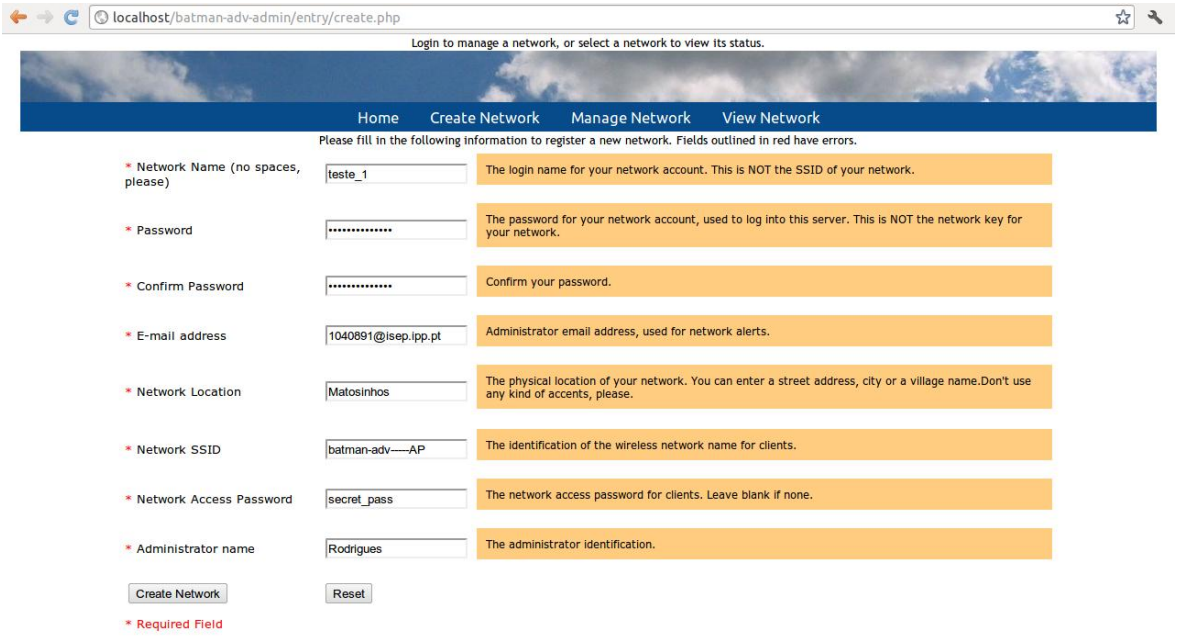


Figura 30 O separador *Create Network* gera um ecrã, que permite a criação de uma nova rede.

Utilizando este *script* foi criada a rede "teste_1", localizada na localidade de Matosinhos. Foi definido o SSID da rede, transmitido pelo AP, como "batman-adv-----AP", tendo sido definido também a *password* de acesso à rede (aqui o nome da rede e *password*, ficam registados na base de dados, só para consulta). O nome do administrador escolhido foi "Rodrigues" e o *e-mail*, foi o associado à conta do ISEP.

Depois de criada a rede com sucesso através do separador *Create Network*, ou desde que seja validada a autenticação através do separador *Manage Network*, é efetuado o redirecionamento para a página de administração da rede, conforme se pode observar na Figura 31.

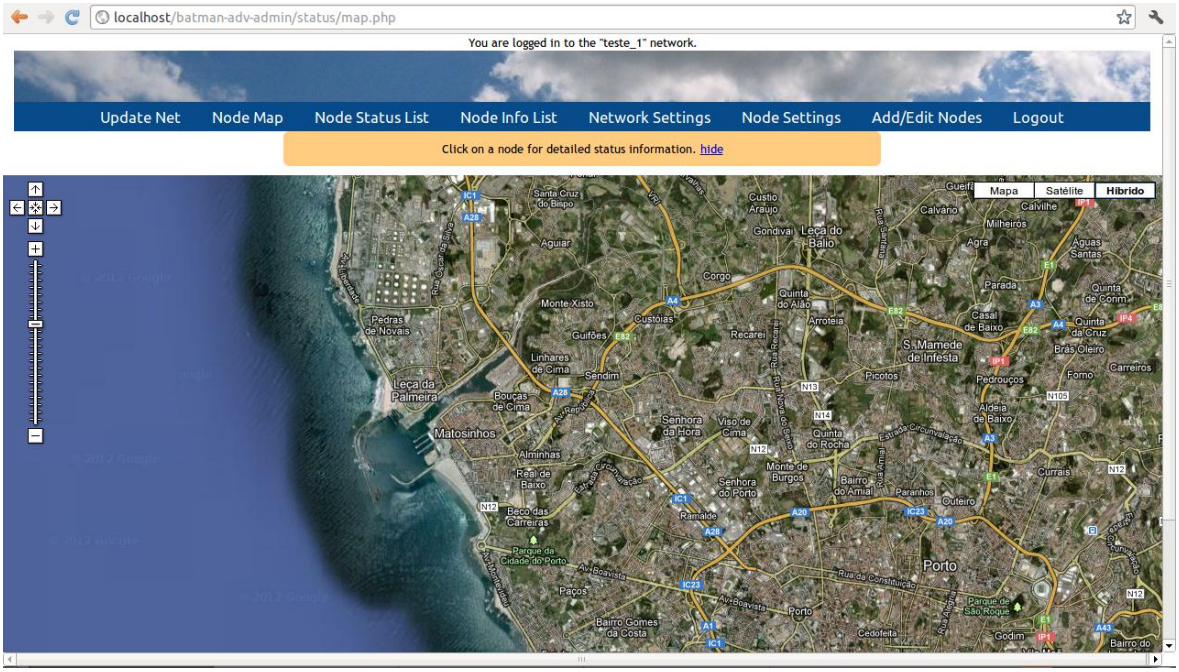


Figura 31 Écrã de administração de rede, após a criação de rede "teste_1".

Além dos separadores já existentes foi criado o separador *Update Net* que permite fazer a atualização da rede, sendo feita a extração dos dados e atualização dos parâmetros dos nós, assim como a atualização da base de dados. Foi também criado o separador *Node Settings* que apresenta informações dos nós e permite configurá-los individualmente.

Na página inicial de entrada (Figura 29) também existe um separador *View Network*. Este separador permite que um utilizador possa visualizar e ter acesso a informações sobre uma rede, sem poder administrá-la. Este separador ativa o *script select.php*.

Neste módulo existe ainda o *script* de *login.php* utilizado para a autenticação do administrador. E o *script logout.php* utilizado para limpar todas as variáveis de sessão de maneira a forçar uma nova autenticação, caso assim seja solicitado.

6.3.2. MÓDULO *NODES*

Tanto um administrador, através do separador *Manage Network*, como um utilizador através do separador *View Network*, têm à sua disposição a opção *Add Nodes*. O *script* ativado quando selecionada essa opção é o *script addnode.php*. Este *script* permite a inserção de nós, associados a uma localização no mapa. Para isso é utilizado *Javascript*, mais concretamente o *script map.js*, que é uma linguagem baseada no conceito de objetos. Ou seja, de maneira a inserir um nó, o *script addnode.php* primeiro prepara o objeto *GMap*

e depois fica à escuta que ocorra um evento, neste caso o evento será um "*click*" do rato no mapa, que adiciona um nó.

Para o *batman-adv-admin* foi determinado existirem dois tipos de nós, os nós singulares e os nós coletivos. Quando não é possível adicionar vários nós no mapa devido à insuficiência do espaço disponível, como por exemplo, quando num determinado local existem vários andares e em cada um dos andares existem vários nós, que se sobrepõem, torna-se impossível a associação de nós no mapa de uma forma intelígivel. Para ultrapassar essa dificuldade foi criada uma opção que permite inserir um nó que representa vários nós. Quando ocorre o evento "*click*", surge no mapa um formulário com o preenchimento automático da latitude e longitude, associado ao local do mapa. Um dos campos desse formulário é uma caixa de seleção (*checkbox*) denominada "*Colective Node*", *checkbox* essa que determina se o nó é colectivo ou singular, consoante o mesmo seja marcado ou não. Na Figura 32 apresenta-se a imagem desse formulário de inserção do nó servidor/*gateway*, que faz parte da rede criada.

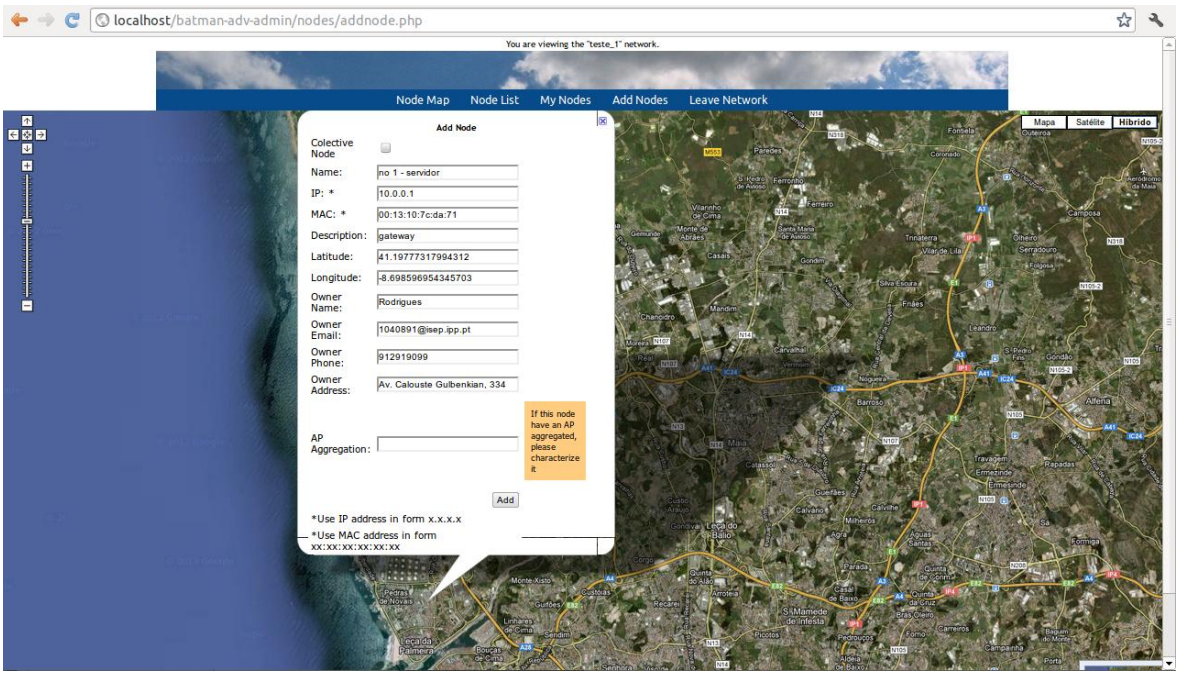


Figura 32 Formulário para adição de nós, após "*click*" no mapa.

No formulário foi adicionado a *checkbox* referente a sinalização de nó coletivo e foi adicionado o campo denominado de *AP Aggregation*. Conforme já foi referido anteriormente, cada nó de uma rede *mesh* BATMAN-ADV não necessita obrigatoriamente de ter um AP agregado. Apenas é necessário agregar um AP ao nó BATMAN-ADV,

quando existem utilizadores susceptíveis de serem servidos pela rede sem fios. Este campo permite indicar se o nó criado é um desses nós, assim como permite caracterizar o AP agregado.

Após o preenchimento do formulário e da sua submissão, que é efetuada através do botão "*Add*", o *script addnode.php* reúne toda a informação existente na base de dados, processa-a num formato que possa ser facilmente interpretado e cria o objeto *nodeMarker* para cada linha da tabela da base de dados *node*, que pertence à rede que está a ser visualizada. Cada objeto *nodeMarker*, correspoderá a um nó e será criado conforme a sua especificidade, podendo alterar a cor, conforme a sua carga de trabalho e o tamanho, conforme o número de clientes que está a servir.

Se for marcada a *checkbox* de nó coletivo, o formulário servirá para caracterização do primeiro nó dum conjunto de nós coletivos, que será denominado de "*Master*" (M). Pode-se observar no código seguinte, extraído de *addnode.php*, a parte respeitante ao *checkbox Colective Node*, apresentado na Figura 32:

```
<td><input type="checkbox" name="flag_colective_node"
value="M"></td>
```

Se for adicionado um nó coletivo, será feito um reencaminhamento para uma nova página, só com formulário e sem qualquer mapa, pelo *script* criado para o efeito – *edit_col.php*. De salientar que todos os outros nós que venham a ser associados a este nó coletivo terão a mesma latitude e longitude do nó inicial e serão denominados de "*Server*" (S), conforme se pode observar no seguinte código pertencente a *c_edit_col.php*, que insere o nó coletivo na base de dados:

```
$latitude=$_POST["latitude"]; // is always the same
$longitude=$_POST["longitude"]; // is always the same
. . .
else    // if is a new node, insert it
      {
      $status="A";
      $flag_colective_node_="S";          //flag Server
      if($ap_ == "" || $ap_ =="none") {$ap_="none";$flag_ap_=0;}
else {$flag_ap_=1;}
      //add the node
      $fields =
array('name_','ip_','mac_','description_','owner_name_','owner_ema
il_','owner_phone_','owner_address_');
      $values = getValuesFromPOST($fields);
      array_push($values, $ap_);
      array_push($values, $flag_ap_);
      array_push($values, $flag_colective_node_);
      array_push($values, $netid);
```

```
      array_push($values, $latitude);    //latitude
      array_push($values, $longitude);   //longitude
      array_push($values, $status);
      $fields_2 =
array('name','ip','mac','description','owner_name','owner_email','
owner_phone','owner_address','ap','flag_ap','flag_colective_node',
'netid','latitude','longitude','approval_status');
      insert('node',$fields_2,$values);//Insertion of coletive node
      }
```

Na Figura 33 apresenta-se um ecrã gerado pelo *script edit_col.php*.

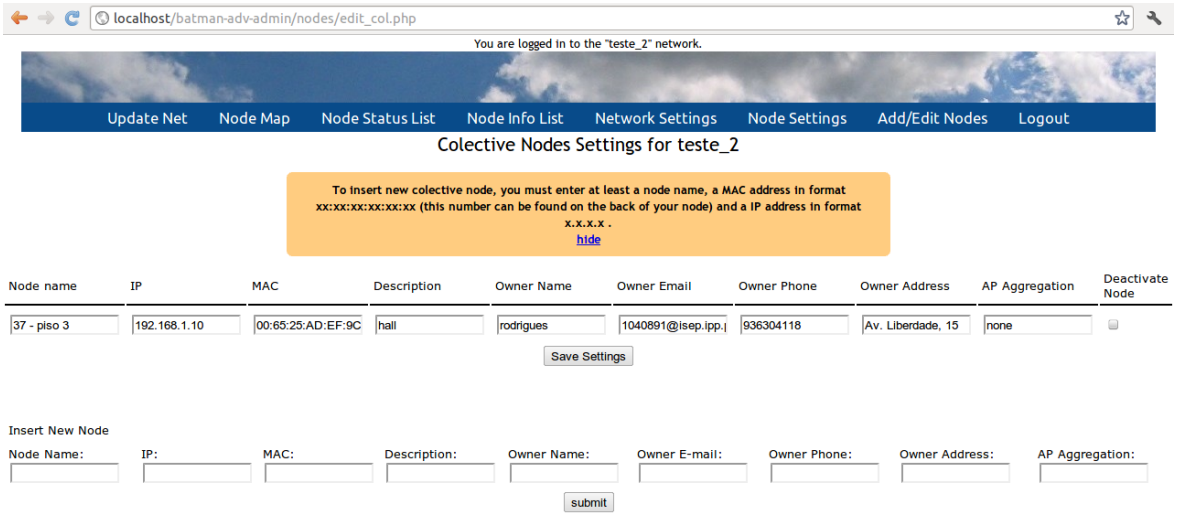


Figura 33 Écra gerado pelo *script edit_col.php*.

O *script* que trata as entradas dos nós colectivos, tendo sido criado para o efeito, é o *scritp c_edit_col.php*. Este *script* é análogo ao *script c_add_node.php*, ou seja o *script* que trata os dados inseridos dos nós individuais. No que toca à inserção de nós, o *script c_edit_col.php* executa um procedimento de verificação, no sentido de validar os IPs e MACs *addresses*. Verifica se os dados inseridos encontram-se na gama e formato corretos. Se não estiverem corretos, o nó não é inserido. Se já existem nós coletivos com o IP ou MAC *address* inseridos, atualiza os dados desses nós. Também é possível desativar os nós coletivos marcando a *checkbox* "*Deactivate Node*".

Outros *scripts* existentes neste módulo são:

nodes_info.php – Só está disponível na página de administração, e apresenta a lista de nós ativados e desativados;

node_info.php – Permite que os nós sejam ativados, desativados ou apagados;

c_deletenode.php – *Script* que trata de desativar os nós através da mundança da *flag* de "A" (*Activated*) para "D" (*Deactivated*), na base de dados;

my_nodes.php – Apresenta uma lista de nós associados ao um *e-mail*.

6.3.3. MÓDULO *STATUS*

O *script map.php* usa o mesmo processo que o *script addnode.php*. Cria objetos *nodeMarker* com base na informação da base de dados. Ao "clicar" num nó são apresentadas numa janela de visualização, informações referentes a esse nó, como por exemplo o tempo decorrido desde a última verificação do nó, o número de utilizadores do nó, a quantidade de *downloads* e *uploads* efetuados. Também é apresentado o número de saltos para chegar à *gateway*, a qualidade de transmissão, assim como se o nó tem um AP agregado. Nessa mesma janela existe outro separador que fornece informações do proprietário do nó (*e-mail*, telefone, etc.) Se for um nó servidor, é indicado que se trata de um servidor/*gateway*. Na Figura 34 pode-se observar a janela de informação do nó 3 da rede criada, pertencente ao "andar superior" e que tem um AP agregado.

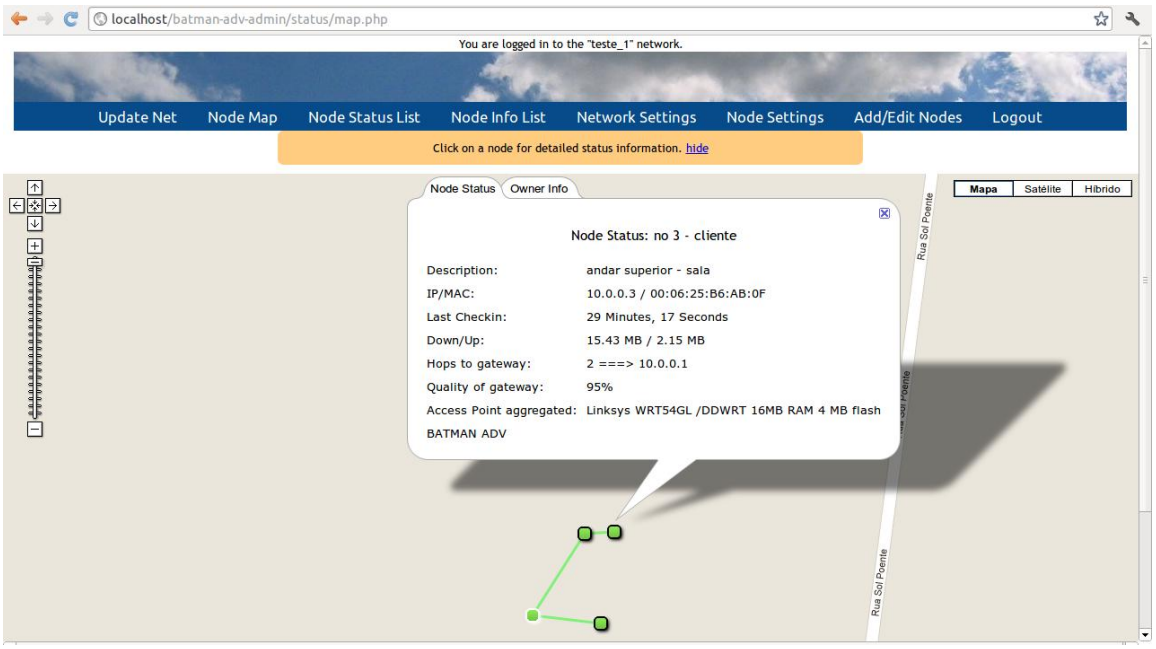


Figura 34 Ao "clicar" no nó, o *script map.php* apresenta informações variadas.

O *script map.php* consulta na base de dados, os valores definidos como limites para a qualidade da ligação. O valor da qualidade de transmissão varia entre 0 e 255. Para

apresentar a qualidade de transmissão no mapa, foi criado um objeto *GPolyline*, que constrói um segmento de reta entre dois pontos, cuja função definida em *map.php* é a seguinte:

```
var polyline = new GPolyline([  new GLatLng("$latitude",
"$longitude"),  new GLatLng("$lat2" , "$long2")], "$RGB",
"$width");
```

O método utilizado tem por base a seguinte metodologia:
As coordenadas (x,y) do início do segmento da reta a ser desenhada no mapa são sempre as coordenadas associadas à latitude e longitude do nó, no código, definidas como "*$latitude*" e "*$longitude*".
Se o nó for um servidor/*gateway* as coordenadas (x,y) de chegada são as mesmas de início, ou seja não existe um segmento de reta, mas sim um ponto.
Se o nó for um nó cliente, as coordenadas (x,y) de chegada serão as coordenadas do nó (*best next hop*), determinado pelo protocolo BATMAN-ADV. Desta forma é desenhado um segmento de reta entre dois nós.
As coordenadas de chegada são definidas por "*$lat2*" e "*$long2*".

Para especificar a qualidade da ligação, determinou-se existirem três cores: verde, amarelo e vermelho (definidas através da variável "*$RGB*"), associados a três espessuras do segmento de reta, nomeadamente, grossa, média e fina (definidas através da variável "*$width*"). A cor verde representa uma ligação de boa qualidade, a cor amarela representa qualidade inferior ao desejável e cor vermelha representa má qualidade de transmissão. Na Figura 35 apresenta-se um exemplo da representação gráfica dos segmentos de reta criados entre nós.

Para se obter os segmentos de reta amarelo e vermelho foi alterada a configuração dos valores considerados como limites da qualidade de transmissão dos nós 2 (andar inferior – sala) e nó 3 (andar superior – sala), para valores mais elevados que os definidos por omissão.

Foi determinado que os nós servidores/*gateways* são os nós que apresentam a borda branca e os nós clientes apresentam a borda negra. Para isso tiveram que ser criadas as figuras dos nós servidores/*gateways*, uma vez que não existiam.

O *script view.php* apresenta esta mesma informação, de uma forma textual. O *script* consulta a base de dados e por cada nó pertencente à rede que se encontre ativado,

apresenta informações sobre o seu estado, configurações e características. Se o nó for um nó servidor/*gateway*, o seu nome aparece a negrito.

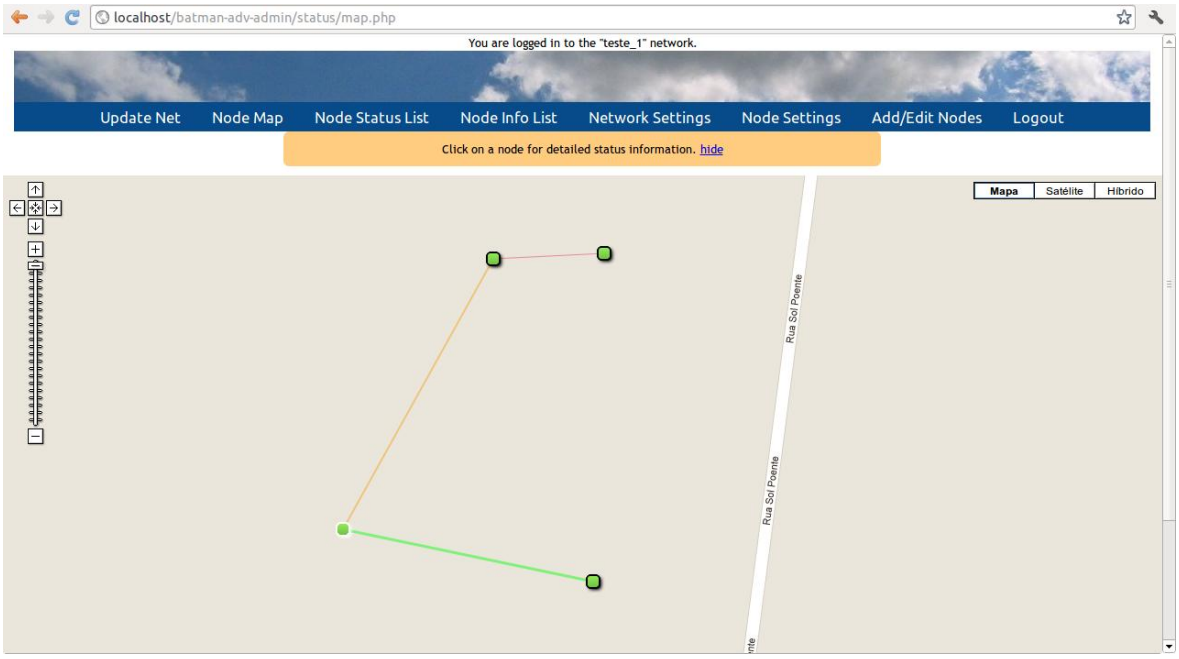
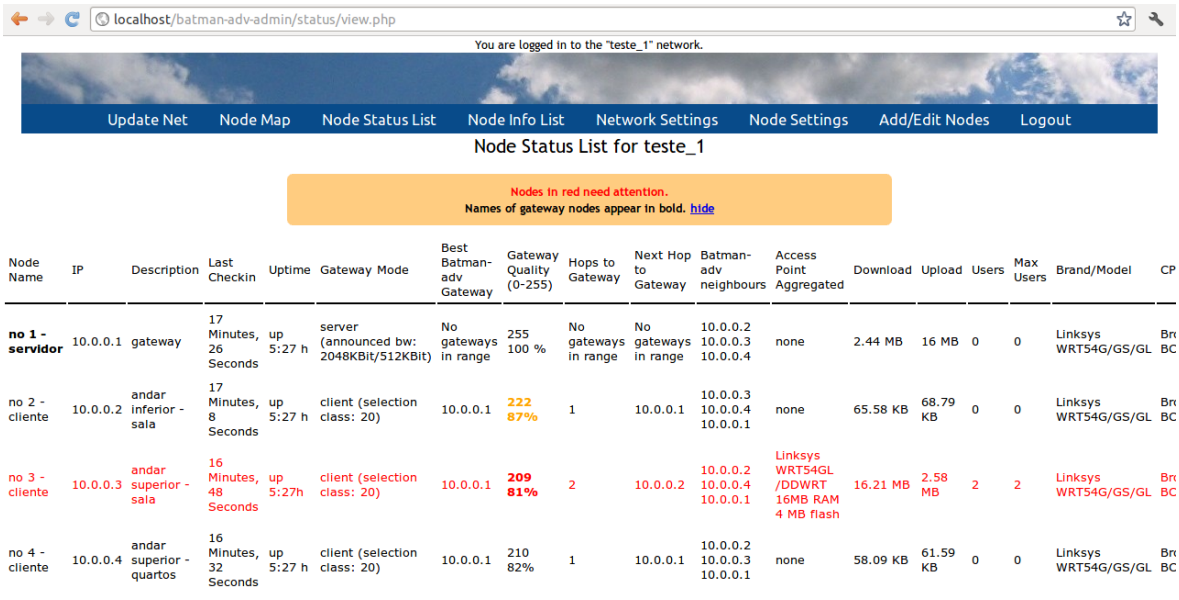


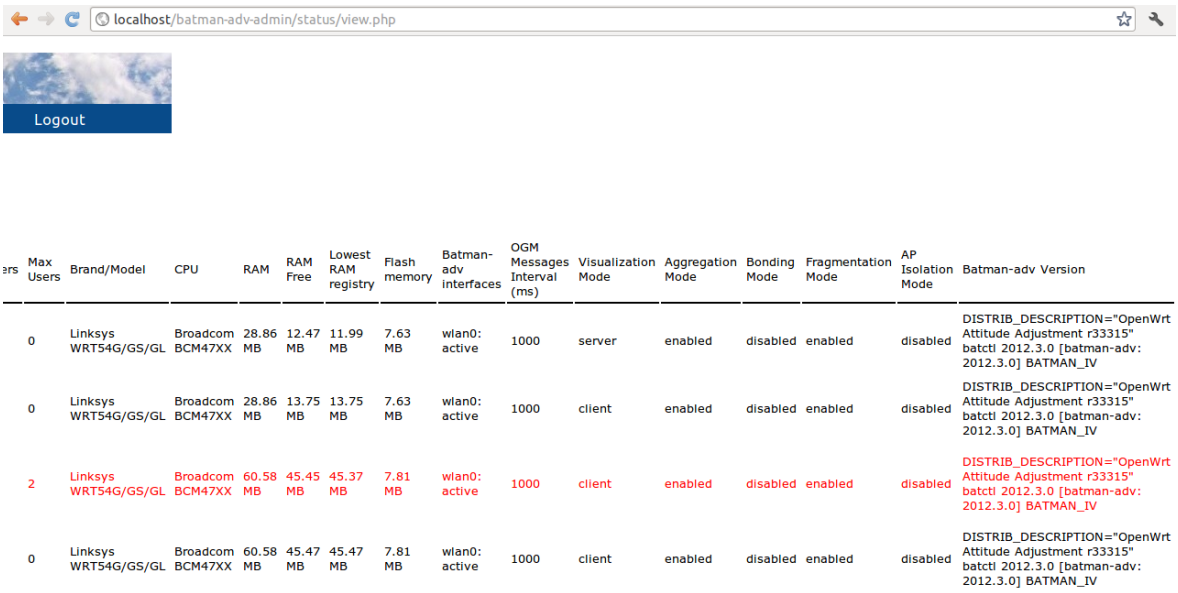
Figura 35 Qualidade de transmissão entre nós, representada por segmentos de reta coloridos.

Na Figuras 36 e 37 é ilustrada uma apresentação desta informação em modo textual, da rede "teste_1", criada.



Node Name	IP	Description	Last Checkin	Uptime	Gateway Mode	Best Batman-adv Gateway	Gateway Quality (0-255)	Hops to Gateway	Next Hop to Gateway	Batman-adv neighbours	Access Point Aggregated	Download	Upload	Users	Max Users	Brand/Model	CP
no 1 - servidor	10.0.0.1	gateway	17 Minutes, 26 Seconds	up 5:27 h	server (announced bw: 2048KBit/512KBit)	No gateways in range	255 100 %	No gateways in range	No gateways in range	10.0.0.2 10.0.0.3 10.0.0.4	none	2.44 MB	16 MB	0	0	Linksys WRT54G/GS/GL	Br BC
no 2 - cliente	10.0.0.2	andar inferior - sala	17 Minutes, 8 Seconds	up 5:27 h	client (selection class: 20)	10.0.0.1	222 87%	1	10.0.0.1	10.0.0.3 10.0.0.4 10.0.0.1	none	65.58 KB	68.79 KB	0	0	Linksys WRT54G/GS/GL	Br BC
no 3 - cliente	10.0.0.3	andar superior - sala	16 Minutes, 48 Seconds	up 5:27h	client (selection class: 20)	10.0.0.1	209 81%	2	10.0.0.2	10.0.0.2 10.0.0.4 10.0.0.1	Linksys WRT54GL /DDWRT 16MB RAM 4 MB flash	16.21 MB	2.58 MB	2	2	Linksys WRT54G/GS/GL	Br BC
no 4 - cliente	10.0.0.4	andar superior - quartos	16 Minutes, 32 Seconds	up 5:27 h	client (selection class: 20)	10.0.0.1	210 82%	1	10.0.0.1	10.0.0.2 10.0.0.3 10.0.0.1	none	58.09 KB	61.59 KB	0	0	Linksys WRT54G/GS/GL	Br BC

Figura 36 O *script view.php* apresenta os dados referentes a cada nó da rede "teste_1".



ers	Max Users	Brand/Model	CPU	RAM	RAM Free	Lowest RAM registry	Flash memory	Batman-adv interfaces	OGM Messages Interval (ms)	Visualization Mode	Aggregation Mode	Bonding Mode	Fragmentation Mode	AP Isolation Mode	Batman-adv Version
	0	Linksys WRT54G/GS/GL	Broadcom BCM47XX	28.86 MB	12.47 MB	11.99 MB	7.63 MB	wlan0: active	1000	server	enabled	disabled	enabled	disabled	DISTRIB_DESCRIPTION="OpenWrt Attitude Adjustment r33315" batctl 2012.3.0 [batman-adv: 2012.3.0] BATMAN_IV
	0	Linksys WRT54G/GS/GL	Broadcom BCM47XX	28.86 MB	13.75 MB	13.75 MB	7.63 MB	wlan0: active	1000	client	enabled	disabled	enabled	disabled	DISTRIB_DESCRIPTION="OpenWrt Attitude Adjustment r33315" batctl 2012.3.0 [batman-adv: 2012.3.0] BATMAN_IV
	2	Linksys WRT54G/GS/GL	Broadcom BCM47XX	60.58 MB	45.45 MB	45.37 MB	7.81 MB	wlan0: active	1000	client	enabled	disabled	enabled	disabled	DISTRIB_DESCRIPTION="OpenWrt Attitude Adjustment r33315" batctl 2012.3.0 [batman-adv: 2012.3.0] BATMAN_IV
	0	Linksys WRT54G/GS/GL	Broadcom BCM47XX	60.58 MB	45.47 MB	45.47 MB	7.81 MB	wlan0: active	1000	client	enabled	disabled	enabled	disabled	DISTRIB_DESCRIPTION="OpenWrt Attitude Adjustment r33315" batctl 2012.3.0 [batman-adv: 2012.3.0] BATMAN_IV

Figura 37 O *script view.php* apresenta a linha vermelha, se o nó tem desempenho inferior ao estipulado.

As cores das linhas desta página alteram para vermelho quando algum dos limites estabelecidos, que podem ser qualidade de transmissão entre nós, tráfego de *download* ou *upload* ou memória RAM inferior a 25% da memória total do nó, são ultrapassados, conforme se pode verificar no seguinte código extraído do *script*, referente à ultrapassagem do limite definido para *download* ou *upload*:

```
while($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    if ($row["approval_status"] == "A") {   //show only activated
nodes
. . .
else if ($row["download_bytes"] >= $download_limit_2 ||
$row["upload_bytes"] >= $upload_limit_2){
echo "<tr class=\"down\">";
}
. . .
```

A classe "*down*" é definida no *script style.css*, e implica que o que for apresentado no ecrã seja de cor vermelha, neste caso será toda a linha referente ao nó.

Alguns dos campos das linhas podem ficar a amarelo quando se trata apenas de desempenho ligeiramente abaixo do expectável. Na linha é sempre realçado o campo a negrito, cujo valor ultrapassou o limite definido.

6.3.4. MÓDULO *NET_SETTINGS*

O *script edit.php* gera uma página onde são apresentadas informações que constam na base de dados *network*. Contém algum texto de ajuda no preenchimento dos campos e chama o *script c_edit.php* para o processamento do formulário.

A página gerada apresenta duas áreas distintas. A área superior permite que se possa proceder a alterações nos dados e configurações da rede, submetendo para isso um formulário. A parte inferior é apenas de leitura. Na Figura 38 apresenta-se a parte superior relativa ao formulário.

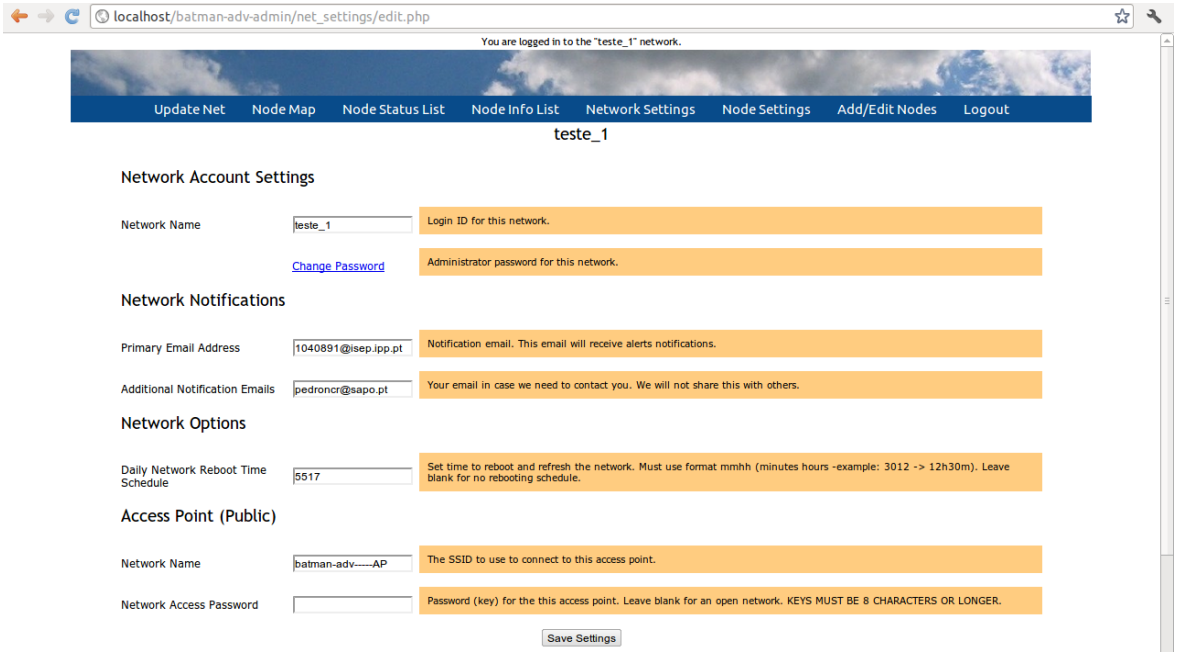


Figura 38 Parte superior da página gerada pelo *script net_settings.php*.

É possível editar o nome da rede, mudar a *password* de acesso na ferramenta de gestão *batman-adv-admin*, alterar os *e-mails* associados a esta conta (de salientar que o *Primary Email Address* será o *e-mail* utilizado para envio de notificações de alertas da rede), assim como é possível editar o SSID emitido pelos APs e a *password* de acesso para os utilizadores da rede (neste caso o SSID e a *password* não produzem alterações nas configurações dos nós, apenas servem para informar o administrador, qual o SSID e *password* associados à rede em questão).

Foi criado o campo *Daily Network Reboot Time Schedule*. Este campo permite fazer um *reboot* diário nos nós, a determinada hora. É editado um ficheiro de nome *root*, onde é inserida a hora de *reboot*, que é enviado para todos os nós pertencentes à rede, o que fará

com que os nós se desliguem e reiniciem todos ao mesmo tempo. Isto permite fazer um refrescamento das memórias dos nós e reiniciar os contadores de tráfego e de utilizadores, se assim o administrador o entender.

O *script c_edit.php* processa a informação dos formulários e atualiza a base de dados, devolve uma *flag* de sucesso à variável de sessão e redireciona novamente para a página criada pelo *script edit.php*. No código seguinte extraído de *c_edit.php*, pode-se observar o procedimento de criação do ficheiro *root*, ficheiro que é enviado para os nós e que provocar o *reboot* em cada nó num determinado momento:

```
$data_2=$data." "."$data_1"." * * * /sbin/reboot";// create the
text to write in the "root" file
$filename_1 = '/var/www/batman-adv-
admin/net_settings/files/'.$net_name.'_'.$net_location.'/';//if
the net directory doesn't exist, create one
mkdir($filename_1, 0777);
$filename_2= '/var/www/batman-adv-
admin/net_settings/files/'.$net_name.'_'.$net_location.'/'.'root';
file_put_contents($filename_2, $data_2); // put the text created
into the "root" file
```

A variável "*$data*" corresponde às horas inseridas (hh) e a variável "*$data_1*" corresponde aos minutos inseridos (mm), a que se pretende fazer o *reboot* à rede. Se a diretoria com nome e localização da rede não existir é criada, se já existir esta linha é ignorada. Esta diretoria serve para registo de todas as alterações efetuadas na rede e nos nós. Posteriormente é criado o ficheiro *root* e é gravado o texto "*hh mm * * * /sbin/reboot*", que posteriormente será enviado para cada um dos nós que pertençam à rede em questão e que estejam ativos, que provocará um *reboot* à hora e minuto especificado.

Na parte inferior da página, foi criada uma secção onde são apresentados valores de utilização da rede e informações sobre configurações, conforme ilustrado na Figura 39.

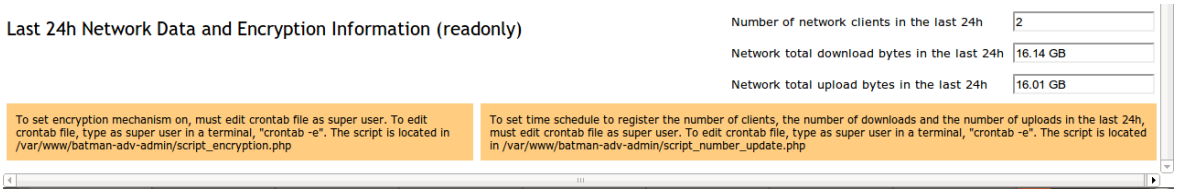


Figura 39 Parte inferior do ecrã gerado pelo *script net_settings.php*. Esta secção é só de leitura.

Conforme se pode observar na figura são apresentados os valores diários do número de clientes da rede, da quantidade de *downloads* e *uploads* da rede. Estes valores são atualizados a cada 24 horas.

Mais abaixo são dadas instruções em como habilitar os mecanismos de encriptação e de registo dos valores diários da rede (clientes, *downloads* e *uploads*). Estes dois mecanismos fazem parte de dois *scripts* distintos que foram criados para a ferramenta de gestão. E podem estar ativos ou não, depende da preferência do administrador da rede. Pode ativá-los se pretende algum nível de encriptação na rede BATMAN-ADV (não confundir com a rede de acesso *wireless* para clientes, essa está sempre encriptada) ou se pretende que seja feita uma atualização dos valores da rede num intervalo periódico de tempo. Aqui propôs-se 24 horas, mas pode-se definir o intervalo de tempo que se quiser, desde minutos a dias. Poder-se-ia, simplesmente ter colocado mais duas *checkboxes* com as opções habilitar/desabilitar encriptação e habilitar/desabilitar atualização de valores da rede, mas isso implicaria editar o ficheiro "*crontab*" existente no sistema operativo, e voltar a reiniciá-lo. Essa operação poderia causar alguma desconfiguração no sistema automático de execução de outras tarefas já agendadas, uma vez que teria que se apagar e reescrever o ficheiro. Optou-se por fazê-lo manualmente, uma vez que é uma tarefa feita uma única vez, na criação da rede (este ficheiro "*crontab*" encontra-se em */var/spool/cron/crontabs*). Na Figura 40 pode-se observar o ecrã de edição de "*crontab*", onde se podem observar três *scripts batman-adv-admin* habilitados para funcionarem periodicamente.

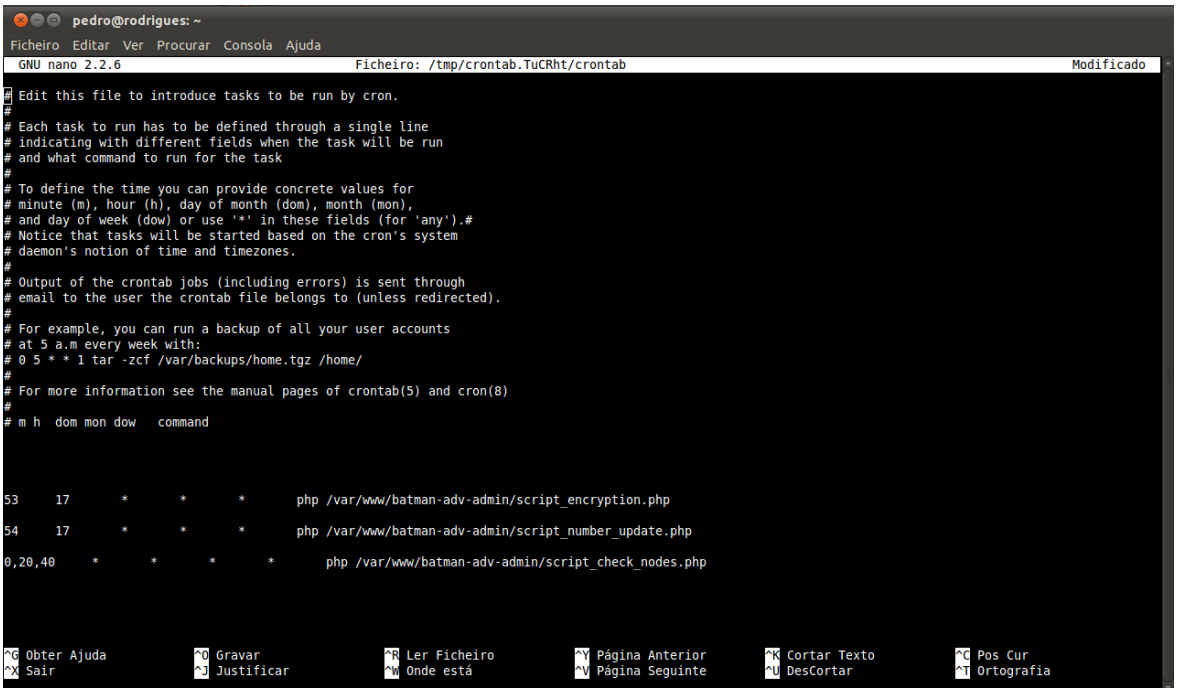


Figura 40 Écrã relativo ao agendamento de tarefas relacionadas com a gestão da rede.

Os *scripts* de encriptação (*script_encryption.php*) e o *script* de atualização de valores da rede (*script_number_update.php*) funcionam de 24 em 24 horas (às 17h53m e 17h54m, respetivamente). O *script* de verificação de nós (*script_check_nodes.php*) funciona de 20 em 20 minutos (aos 0, 20 e 40 minutos). Estes *scripts* serão explicados mais adiante e foram todos criados de raiz para a ferramenta *batman-adv-admin*.

Para a gestão dos limites de tráfego de *download* e *upload* e dos limites de qualidade de transmissão, assim como para a gestão das configurações individuais foi criado o *script edit_2.php*. Nas Figuras 41 e 42 apresenta-se o ecrã gerado pelo *script edit_2.php*.

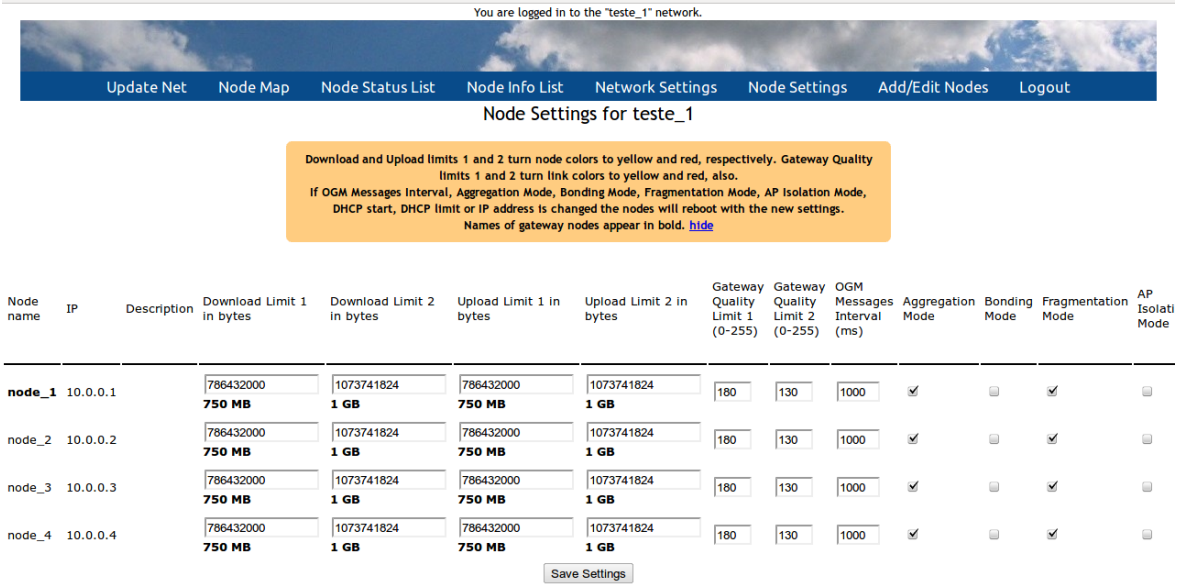


Figura 41 Página do *script edit_2.php*, onde se pode configurar os nós BATMAN-ADV.

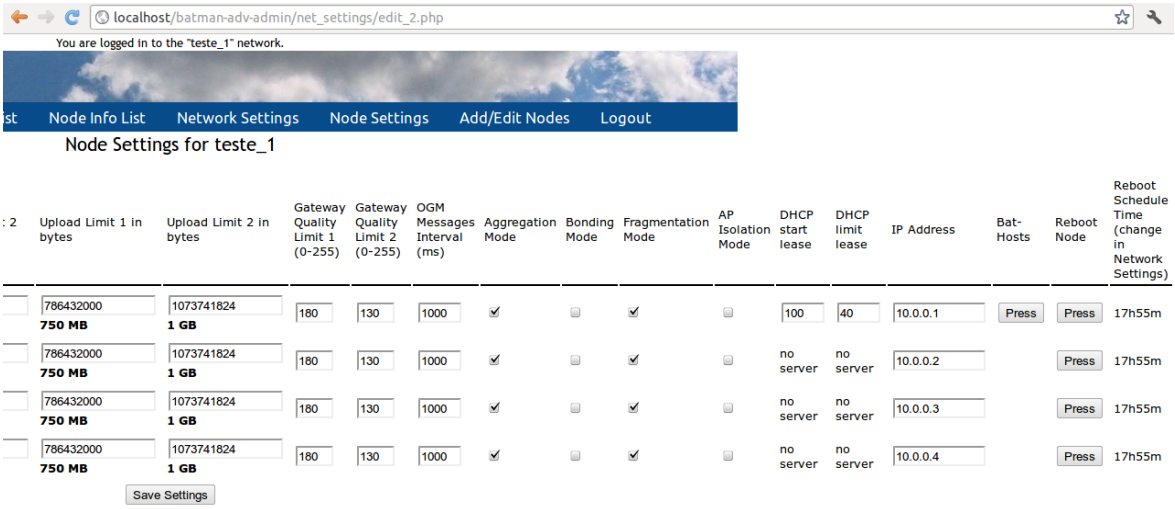


Figura 42 Continuação da página de configuraração dos nós da rede BATMAN-ADV.

É neste *script* que são definidos aos valores de *download*, *upload* e qualidade de transmissão. Os valores dos campos associados ao "*Limit 2*" (limite nível 2), implicam o envio de *e-mails* de notificação de alertas ao administrador, caso o valor definido seja ultrapassado e estão associados à cor vermelha dos nós e segmentos de reta. Os valores dos campos associados ao "*Limit 1*" (limite nível 1), estão associados à cor amarela dos nós e segmentos de reta.

Os campos e *checkboxes* relativos ao protocolo BATMAN-ADV, assim como os campos de atribuição e alteração de IPs, quando submetidos, produzem efeitos nos nós.

Além dos campos referidos, existem outro tipo de informações apresentadas na página, como o nome, IP e descrição dos nós e é apresentada hora de *reboot* dos nós, caso tenha sido configurada pelo administrador da rede. São disponibilizados ainda botões que permitem fazer o *reboot* individualmente a cada nó (botão *Reboot Node*), e existem ainda botões para a apresentação gráfica da rede BATMAN-ADV, que é gerada através de um conjunto de grafos, que o próprio protocolo disponibiliza e que se pode ativar através do botão *Bat-Hosts*. Este botão está disponível para todos os nós servidores/*gateways*. Ao pressionar este botão é aberta uma nova janela com a representação gráfica da rede, conforme se pode observar na Figura 43.

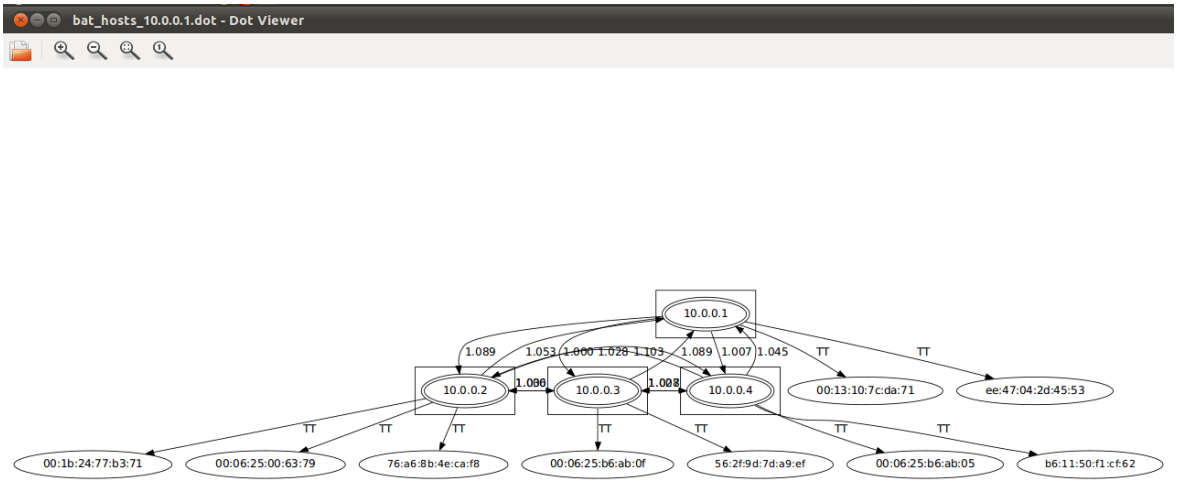


Figura 43 Pressionando o botão *Bat-Hosts* é apresentada graficamente a rede.

Todas as opções foram testadas e funcionam corretamente.

O *script* criado para o tratamento das alterações efetuadas no *script edit_2.php* chama-se *c_edit_2.php*. Este *script* é responsável por armazenar toda a informação submetida. Uma vez que é possível submeter várias alterações ao mesmo tempo, este *script* trata sequencialmante a informação recebida, quer seja para atualização da base de dados ou para alteração das configurações dos nós.

Apresenta-se uma parte do código existente neste *script* que implementa a metodologia utilizada, de modo a que as alterações submetidas produzam efeitos nos nós:

```
if ($new_interval_ogm != $old_interval_ogm)
      {
      $flag_1=1;
      if (file_exists($filename))    //$filename=batman-adv
            {
            $field="'orig_interval'";
            $data="'orig_interval' '";
            $buffer = file_get_contents($filename);
            $value="$new_interval_ogm";
            $data_6=batman($field, $data, $buffer, $value);
            file_put_contents($filename, $data_6);
            }
       else
            {
            $buffer = file_get_contents($filename_2);
            $data_2=explode("'orig_interval'",$buffer);
            $data_3=$data_2[0]."'orig_interval'
"."'"."$new_interval_ogm"."'".$data_2[1];
            mkdir($filename_10, 0777);
            touch ($filename);
            chmod($filename, 0777);
            file_put_contents($filename, $data_3);
            }
. . .
{
ssh2_scp_send($ssh, $filename_3, '/etc/config/batman-adv', 0777);
}
. . .
{
$COMMAND_1='reboot';
ssh2_exec($ssh,'export PATH=/bin:/sbin:/usr/bin:/usr/sbin
&&'.$COMMAND_1);
}
```

Neste código é verificado se o valor submetido relativamente ao valor de OGM de um determinado nó é igual ou diferente ao valor registado na base de dados. Se o valor for igual, esta porção de código é ignorado, senão:

1. Verifica se existe um ficheiro "*batman-adv*" guardado. Se existe já houve alguma alteração no protocolo BATMAN-ADV, não se trata da primeira vez.

2. Coloca o texto completo desse ficheiro numa variável "*$buffer*" e chama a função *batman* onde passa como argumentos, sequências de texto a encontrar no ficheiro *batman-adv*, neste caso "*$field*" e "*$data*", o texto integral do ficheiro *batman-adv* e o novo valor do OGM submetido.

3. Por sua vez a função *batman* retorna o ficheiro com o novo valor de OGM, que é gravado no mesmo local onde se encontrava previamente.

4. Por fim o ficheiro é enviado para o nó correspondente e o *router* é reiniciado para que a alteração produza efeitos (através das funções *ssh2_scp_send( )* e *ssh2_exec( )*, respetivamente).

Se não existe um ficheiro "*batman-adv*" guardado, quer dizer que nunca se procedeu a qualquer alteração do protocolo BATMAN-ADV nesse nó. Para este caso foi criado um ficheiro padrão que é utilizado para se proceder à primeira alteração de configuração efetuada num nó, sendo o procedimento idêntico ao descrito. Neste procedimento é criada uma diretoria com o nome da rede e localização (se ainda não existir), e é guardado o ficheiro *batman-adv* nessa diretoria para posterior utilização.

Para que se possa configurar todos os parâmetros apresentados foi necessário criar um repositório de ficheiros de configuração. Esse repositório foi criado na diretoria localizada no módulo /*net_settings* ao qual foi atribuído o nome "*files*".

Nessa diretoria localizada em */var/www/batman-adv-admin/net_settings/files* são disponibilizados os ficheiros necessários para as configurações iniciais dos nós, quando os nós são configurados através do *script edit_2.php*.

Os ficheiros são os seguintes:

- *batman-adv*, *batman-adv_server* e *batman-adv_client* – Os ficheiros *batman-adv* são utilizados para configuração do protocolo BATMAN-ADV.
- *dhcp_server* – O ficheiro *dhcp_server* é utilizado para as atribuições de IPs.
- *wireless* – O ficheiro *wireless* é utilizado para alteração de chave de encriptação (se for ativado o respetivo *script*).
- *network_v1* e *network_v3* – Os ficheiros network* são utilizados para a alteração de IPs.
- *root* – O ficheiro *root* é usado para habilitação/desabilitação de *reboot* geral da rede

Convém salientar que as configurações de portas dos *routers Linksys WRT54GS* variam conforme a versão dos *routers*. Como foram utilizadas as versões v1 e v3, foi necessário diferenciar os ficheiros de configuração *network*, salvaguardando que as portas WAN e LAN são configuradas como pretendido. Esta questão leva-nos a outro patamar de configuração: por cada *router*, seja de que marca e modelo for, que seja utilizado nesta rede, será necessário estudar os seus ficheiros de configuração e casuisticamente adicionar, se houver necessidade disso, mais ficheiros de configuração a esta diretoria, o que já se vem verificando na criação de *firmware OpenWRT*, ou seja é necessário indicar os *chipsets* para os quais se pretende criar a imagem a instalar, e mesmo assim é disponibilizada *n* imagens para *chipsets* iguais, variando só o modelo do *router*, análogo ao caso aqui enunciado.

No final desta operação de edição e gravação dos ficheiros no *router* é atualizada a base de dados com as novas configurações e é reiniciado o *router*. Caso não seja possível a comunicação com o *router*, as alterações não seram atualizadas na base de dados e será transmitida uma mensagem de erro, indicando o IP do *router* e o motivo do insucesso, tendo sido para isso criado o *script edit_2_info.php*, que origina a página que se pode observar na Figura 44.

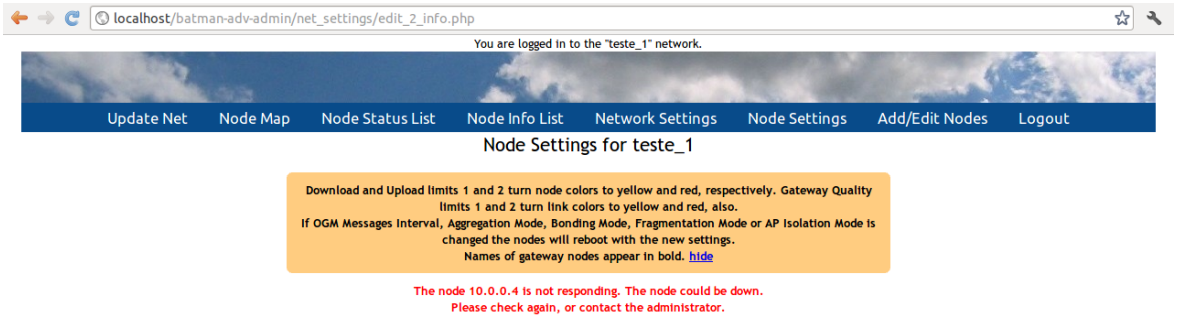


Figura 44 Este *script*, quando não se consegue aceder ao nó, apresenta o motivo do insucesso.

Para se obter a imagem da Figura 44, foi desligado o nó 4 (andar superior – quartos) e procedeu-se à alteração do valor de OGM de 1000 milisegundos para 2000 milisegundos, usando a página de configuração dos nós. Como o nó estava desligado, o valor não foi alterado no nó, não foi alterado na base de dados e foi gerada a página de erro, indicando o possível motivo do insucesso da operação.

O motivo porque se decidiu colocar estes *scripts* que permitem a configuração individual dos nós, neste módulo e não no módulo associado à configuração do nós – */nodes*, foi devido à criação da diretoria *files*, que além de ter como função a disponibilização de ficheiros para configuração inicial dos nós, será também a localização para registos de *logs* de todas as operações efetuadas na rede. Foi optado localizar tudo numa diretoria e não em sítios diferentes e optou-se por localizá-la neste módulo */net_settings*, uma vez que existe muita informação genérica acerca da rede. Posteriormente será abordada o tema dos registos de *logs*.

Outros *scripts* existentes neste módulo são:

password.php – Permite alterar a *password* de acesso à conta da rede na ferramenta de gestão;

c_password.php – Verifica se a antiga *password* está correta e guarda a nova na base de dados, encriptada (utiliza o algoritmo de encriptação *md5*);

O último *script* existente neste módulo é o *script check_nodes.php*, que será abordado mais adiante.

6.3.5. MÓDULO *LIB*

O *script map.js* é utilizado na conceção e manipulação dos mapas, conforme já foi referido anteriormente. Este *script* contém toda a informação necessária para criar e trabalhar os mapas. Inclui funções AJAX para adicionar/atualizar e desativar nós de uma rede através do uso do mapa, assim como é aqui que é definido o objeto *nodeMarker* (usado para apresentar o nó no mapa). Também aqui se encontra a função para o arrastamento dos nós no mapa, quando é necessário reposicioná-los.

Foi determinado, para a ferramenta de gestão desenvolvida, que os nós coletivos serão de cor azul e os nós individuais poderão apresentar três cores – verde, amarelo e vermelho – em função da quantidade de *downloads* e *uploads*, estipulados na página gerada pelo *script edit_2.php*. Quando se "clica" nos nós coletivos (azuis), é ativado o script *edit_col.php*, redirecionando para a página de edição de nós coletivos. Ainda é neste *script map.js*, que é escolhida a imagem do nó individual, em função do número de utilizadores ligados a esse nó. O número de clientes é descrito no interior do objeto *nodeMarker* até um limite de 9. Se estiverem 10 clientes ou mais será apresentado "10^{+}". O tamanho desse objeto também

varia conforme o número de clientes. Na Figura 45 é ilustrado um ecrã com um nó coletivo, a título representativo, uma vez que esta rede não existe.

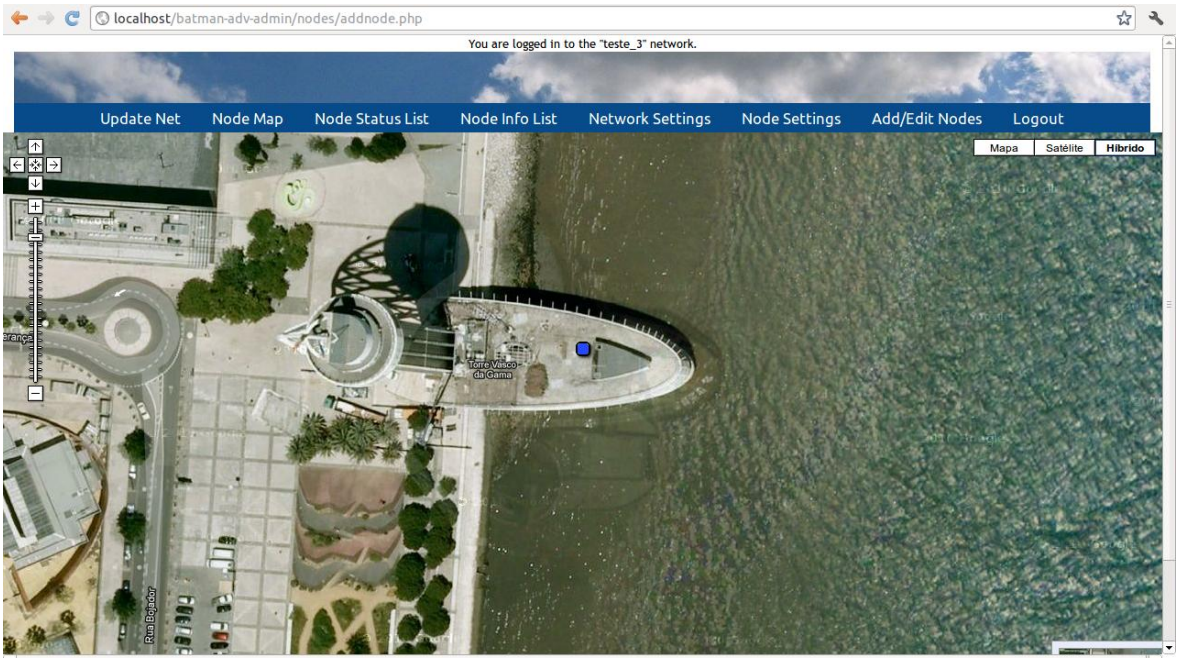


Figura 45 Pormenor de objeto *nodeMarker*, utilizado para sinalização de nó coletivo.

"Clicando" sobre o nó é ativada a página de adição/desativação de nós gerada pelo *script edit_col.php*. Este tipo de nós é indicado para edifícios e outros espaços que tenham muitos nós sobrepostos em altura, o que por vezes não permite a utilização de nós individuais devido à escala do mapa. Na Figura 46 é ilustrado o ecrã com os nós da rede criada "teste_1", onde se pode observar o número de clientes e as cores possíveis que podem ter.

Pode-se observar na Figura 46, que o nó 2 (andar inferior – sala) tem 2 clientes e a cor amarela do nó indica que foi ultrapassado o valor de *download* ou *upload* de nível 1. O nó 4 (andar superior – quartos) tem 3 clientes. A cor vermelha indica que foi ultrapassado o valor de *download* ou *upload* de nível 2, o que implica o envio de um *e-mail* de alerta ao administrador. Para se isolar o nó 4 do nó servidor, foi colocado um armário de metal por baixo do *router*, impedindo a transmissão de sinal entre os dois nós. A ligação do nó 4 para o nó 3 (andar superior – sala), está a vermelho, o que implica também o envio de *e-mail* de alerta.

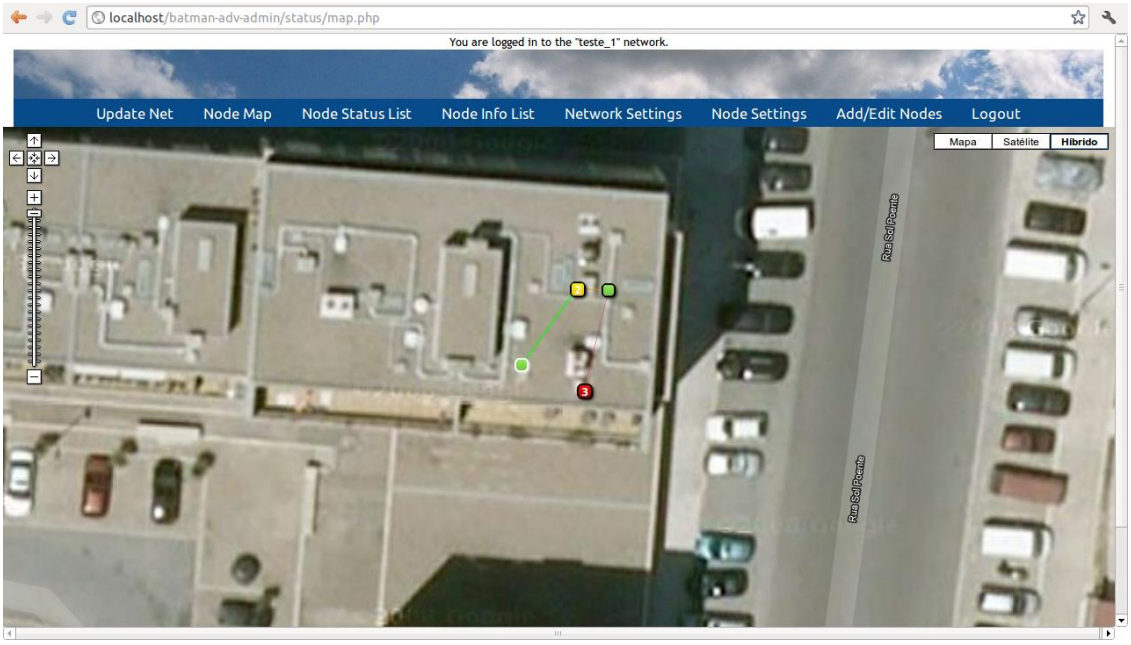


Figura 46 Vários objetos *nodeMarker*, que representam nós individuais.

Os nós podem ser colocados no mapa e arrastados para a localização que se pretender. O exemplo da Figura 47 mostra isso mesmo. Os nós da rede "teste_1" foram arrastados para outra localização geográfica. A rede "teste_1" poderia estar localizada fisicamente em qualquer parte do mundo. A gestão da rede é sempre possível desde que se tenha acesso a um dos nós que constituem a rede, o que pode ser feito remotamente, por exemplo, através de um túnel SSH.

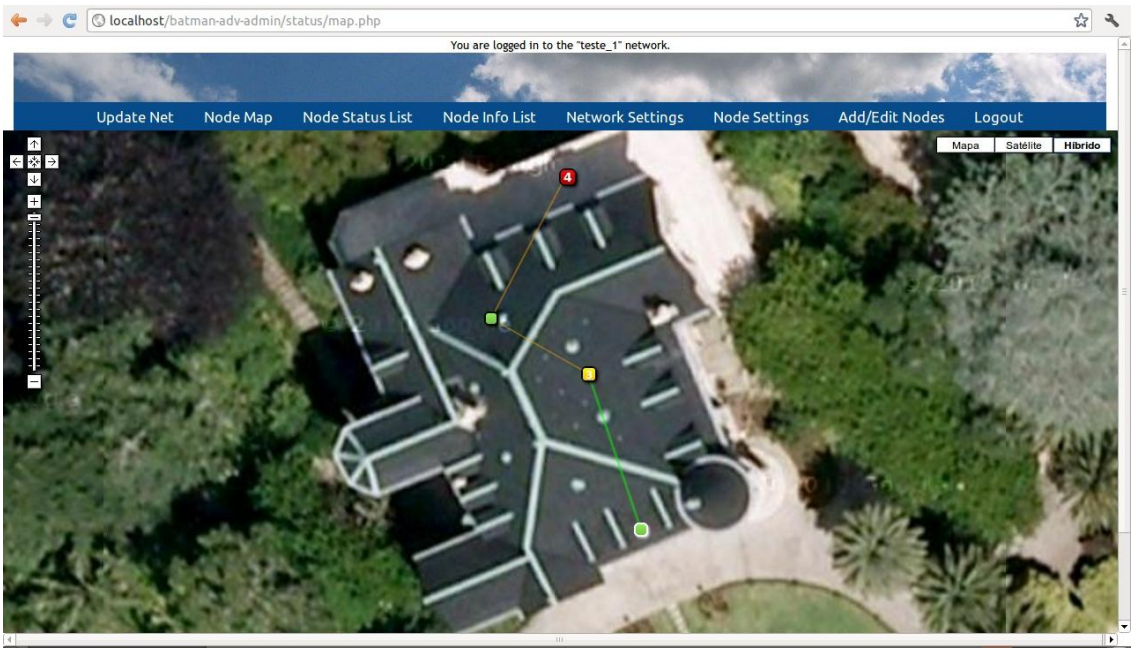


Figura 47 Exemplo de arrastamento dos nós da rede, para outra localização geográfica.

Quando se ativa o *script map.php* ou o *script addnode.php*, a função que passa os argumentos para que os nós sejam criados é a função *nodeMarker*( ), conforme se pode observar no código seguinte:

```
var marker = new nodeMarker(map, "$net_name", point, "$name",
"$notes", "$mac", "$gateway", "$gw_metric", "$up", "$draggable",
"$users","$is_gateway","$download_bytes","$upload_bytes","$downloa
d_limit_1","$download_limit_2","$upload_limit_1","$upload_limit_2"
,"$flag_node_down","$flag_fake_authentication","$flag_colective_no
de");
```

Esta função teve de ser alterada em função da função original. Os novos argumentos passados através da função que vão influenciar os nós escolhidos são:

$users – O nó terá um número e um tamanho em função do número de utilizadores. Se não tiver utilizadores não terá número;

$is_gateway – Em função desta variável, o nó terá borda branca se for servidor ou borda negra se for cliente;

*$download_bytes** e *$upload_bytes** – Em função das variáveis de quantidade de tráfego, o nó será verde, amarelo ou vermelho;

$flag_node_down e *$flag_fake_authentication* – Se o nó estiver em baixo (incontactável) ou se o não for reconhecida a chave de autenticação, o nó será vermelho;

$flag_colective_node – Se for um nó coletivo, será azul;

Os nós são criados no *script map.js* através da função *createIcon*( ), em função dos valores passados.

Para que a ferramenta de gestão desenvolvida pudesse funcionar, foi necessário criar algumas funções que se juntaram às funções já existentes no *script toolbox.php*, que se trata de um *script* que contém funções úteis no processamento dos dados. As funções criadas são:

- ***is_ip*( )** – esta função assegura que o formato e gama dos IPs se encontra correto quando são inseridos os nós.
- ***bytesToSize*( )** – converte *bytes* para um formato facilmente interpretável.
- ***batman*( )** – manipula ficheiros da seguinte forma: procura uma palavra, parte o ficheiro nesse ponto, apaga ou acrescenta informação nova e volta a reconstruir o ficheiro na sua totalidade. É usado para proceder a alterações de configurações nos nós.

- ***cron*( )** – utilizado para a apresentação das horas e minutos do *reboot* da rede, no formato (hh mm) na página creada pelo *script edit_2.php*.
- ***send_alert*( )** – utilizado para enviar notificações de alerta através de *e-mail*.

Outros *scripts* utilizados:

connectDB.php – Contém as configurações necessárias para a conexão à base de dados e estabelece a ligação.

sorttable.js – É utilizado para que as listagens constantes na plataforma de gestão possam ser ordenadas por coluna, bastando para isso "clicar" no seu título, como por exemplo na página gerada pelo *scrip view.php*.

validateInput.js – Valida os dados submetidos no formulário. Indica os campos de preenchimento obrigatório colocando uma caixa vermelha à volta e verifica se os dados estão de acordo com um determinado formato. Por exemplo no formulário de criação de rede, se o *e-mail* não tiver o formato especificado, não permite a sua introdução.

menu.php – É o *script* responsável pelo menu (separadores) que é apresentado em todas as páginas no topo do ecrã. Gera dinamicamente o menu, com base em quem o usa (administrador ou utilizador). Este *script* inicia a função *session_start*( ), que vai transportar as variáveis de sessão entre páginas até se fazer o *Logout*.

style.php – Contém o estilo da informação. Utiliza o *script style.css*, para introduzir em cada écrã a figura do céu azul e das nuvens, ou o estilo e cores do texto apresentado.

mapkeys.php – Contém a chave para acesso ao *Google Maps* API. Para atribuição de uma chave é necessário criar uma conta no *Google* e depois ativar a versão 2 de *Google Maps* API, que deve ser copiada para este *script*.

Por fim, é na diretoria */lib/images* que se encontram todas as imagens usadas na plataforma de gestão, incluindo as imagens dos nós.

6.3.6. OUTROS *SCRIPTS*

O *script check_nodes.php*, localizado no módulo *net_settings* só está disponível para o administrador e aciona-se através do separador *Update Net*. Tanto o *script* como o

separador foram criados de raiz. Este *script* tem a função de estabelecer comunicação com os nós pertencentes à rede. O *script* executa as seguintes funções:

- Obtém a identificação da rede a atualizar, através da variável de sessão correspondente.
- Consulta a base de dados *node*, e linha a linha verifica se o nó pertence à rede em questão.
- Se o nó pertence, verifica se o mesmo se encontra ativado.
- Se o nó estiver disponível e se o nó reconhecer a chave de autenticação prossegue com a comunicação.
- Atualiza a base de dados, com a data e hora do sistema em que é feita a comunicação com o nó.

Extrai e atualiza na base de dados, todos os parâmetros necessários à caracterização e estado do nó. Os parâmetros são os seguintes:

- A marca e modelo do *router*.
- O modelo do CPU.
- A quantidade de memória RAM.
- A quantidade de memória *flash.*
- O tempo decorrido desde *power on*.
- O modo *gateway* ou cliente.
- A quantidade de *downloads* efetuados
- A quantidade de *uploads* efetuados
- O valor livre de memória RAM. Se o valor é inferior ao menor valor conhecido é atualizado também o campo referente a esse dado.
- A versão BATMAN-ADV.
- As *interfaces* em uso BATMAN-ADV.
- A lista de vizinhos BATMAN-ADV.
- A lista de nós *gateways*.
- A qualidade de transmissão para o nó *gateway*.
- A rota para o nó *gateway* e conta os saltos até lá chegar.
- O intervalo de tempo em milisegundos da emissão de OGMs.
- O modo de visualização gráfica BATMAN-ADV.
- O modo de *aggregation* BATMAN-ADV.
- O modo de *bonding* BATMAN-ADV.

- O modo de *fragmentation* BATMAN-ADV.
- O modo de AP *isolation* BATMAN-ADV.
- O IP do próximo nó, de maneira a alcançar o nó *gateway* (*best next hop*).
- O número de clientes do nó. Se o valor é superior ao maior valor conhecido é atualizado também o campo referente a esse dado. Atualiza também os valores do nós que entretanto perderam clientes.
- O valor inicial e o número máximo de atribuição de IPs (só para nós *gateway*).
- O IP.
- O MAC *address*.
- A combinação de portas do *router*.
- BATMAN-ADV MAC *address*. Esta operação implica a criação do ficheiro *bat-hosts.* Este ficheiro *bat-hosts* é análogo ao ficheiro */etc/hosts* existente nos sistemas operativos. Contém por cada linha, um MAC *address* e o IP associado. O MAC *address* é substituído pelo IP, no que concerne a toda a manipulação de MAC *addresses*, tanto para a representação gráfica como manipulação de dados. Se o ficheiro *bat-hosts* não existir é criado. Se já existir é adicionada uma nova linha com o novo MAC *address* e o respetivo IP associado.

Apresenta-se a porção de código respeitante à extração BATMAN-ADV MAC *address*:

```
/// 28 /////////----extracts the batman-adv mac address from node

$COMMAND_32='ifconfig | grep HWaddr | awk \'/wlan0/ {print $5}\'';
$stream = ssh2_exec($ssh,'export
PATH=/bin:/sbin:/usr/bin:/usr/sbin &&'.$COMMAND_32);
stream_set_blocking ($stream, true);
$data = '';
while($buffer = fread($stream, 4096))
      {
      $data .= $buffer;}
      fclose($stream);
      $data = substr($data, 0, -1);
      $filename = '/var/www/batman-adv-
admin/net_settings/files/'.$net_name.'_'.$net_location.'/';
      $filename_2 = '/var/www/batman-adv-
admin/net_settings/files/'.$net_name.'_'.$net_location.'/bat-
hosts';

      if (file_exists($filename))
            {
            $buffer = file_get_contents($filename_2);
            $pos = strpos($buffer,$data);
            if($pos === false)
                  {
                  $data_2="";
```

```
                $data_2=$data_2.$data." "."$data_3"."\n";
                file_put_contents($filename_2, $data_2,
FILE_APPEND);
                }
            }
            else
                {
                $data_2="";
                $data_2=$data_2.$data." "."$data_3"."\n";//$data_3
                mkdir($filename, 0777);       // refers to node
                touch ($filename_2);          // IP, comes from
                chmod($filename_2, 0777);     // another extration
                file_put_contents($filename_2, $data_2);
                }
                $field = "bat_hwaddr";
                mysql_query("UPDATE node SET $field='$data' WHERE
ip='$row_ip' AND netid='$netid'");
                $log_42= "Batman-adv MAC Address: $data\n";
```

Basicamente o que é feito nesta porção de código é o seguinte:

1. É executado o comando 32 através da função *ssh2_exec*( ), definido na variável *$COMMAND_32*, que vai extrair o BATMAN_ADV MAC *address* do nó. Este MAC encontra-se associado à interface "*wlan0*", e é guardado na variável *$data*.

2. Se o ficheiro *bat-hosts* já existe nos registos do servidor e já lá consta o MAC *address* extraído avança para o ponto 4, mas se ainda lá não constar este MAC *address* extraído, então acrescenta uma nova linha ao ficheiro com o novo MAC e respetivo IP.

3. Se não existe a diretoria dos registos da rede, então é criada, assim como o ficheiro *bat-hosts*.

4. É guardado o MAC *address* extraído na base de dados *node* no campo "*bat_hwaddr*".

5. É guardado o MAC *address* extraído na variável *$log_42*, que será posteriormente usada no registo de *log* desta extração.

No final da extração são enviados os seguintes ficheiros para o nó:
O ficheiro *bat-hosts* atualizado e o ficheiro *root*, caso se tenha estipulado fazer um *reboot* a todos os nós da rede. Caso não se pretenda fazer o *reboot*, é apagado o ficheiro *root* existente no nó (se lá existir algum). De seguida apresenta-se este excerto do código:

```
$COMMAND_29='/etc/init.d/cron restart';
$COMMAND_30='rm /etc/crontabs/root';

///  update  ///--- send the bat-hots file up to date to all nodes
$filename = '/var/www/batman-adv-
admin/net_settings/files/'.$net_name.'_'.$net_location.'/'.'bat-
hosts';
ssh2_scp_send($ssh, $filename, '/etc/bat-hosts', 0777);
////////////////////////////////////////////////////////

///  update  ////////---- send the time for node reboot using node
crontab / or delete the cronjob if the reboot time is blank
$filename = '/var/www/batman-adv-
admin/net_settings/files/'.$net_name.'_'.$net_location.'/'.'root';
if ( '' != file_get_contents( $filename))
      {
      ssh2_scp_send($ssh, $filename, '/etc/crontabs/root', 0777);
      ssh2_exec($ssh,'export PATH=/bin:/sbin:/usr/bin:/usr/sbin
&&'.$COMMAND_29);
      }
else  {
      ssh2_exec($ssh,'export PATH=/bin:/sbin:/usr/bin:/usr/sbin
&&'.$COMMAND_30);
      }
////////////////////////////////////////////////////////
```

Na primeira parte do código é enviado para todos os nós o ficheiro *bat-hosts*.

Na segunda parte do código é verificado se o ficheiro *root* não está vazio em "*if (' ' != file_get_contents($filename))*" e caso seja verdade envia também o ficheiro *root*. Se o ficheiro *root* estiver vazio significa que não existe pretensão de fazer *reboot* à rede e é apagado o ficheiro *root* no nó, caso ele exista.

De salientar que os contadores de *downloads* e *uploads*, quando chegam a 4 *GB* voltam a zero, por imposição dos *routers* utilizados. Assim foi necessário fazer a verificação e ajustes para que as leituras sejam fidedignas. A solução encontrada consiste na verificação do valor lido. Se o valor lido for inferior ao último valor registado, isso significa que o contador deu a volta, conforme se pode observar no código correspondente:

```
if ($data_5 < $last_regist)
      {
      $flag_counter=1
      $GB_4=4294967296;
      $rest_div=fmod($last_regist,$GB_4);
```

Se o valor lido for inferior ao valor do último registo que existe na base de dados, significa que o contador deu a volta. Assim com a utilização da função *fmod*( ) existente em PHP, achamos o resto da divisão entre o último registo e os 4 *GB*. Se o valor for 0, quer dizer

que temos ou 8, ou 12, ou 16 *GB*, etc., ou seja um múltiplo dos 4 *GB*, o que implica que o valor do contador seja a soma do último valor lido mais o valor extraído.

Se o resto for diferente de 0 significa que o valor do resto terá de ser tido em conta. Assim o valor do contador será a soma do último registo mais o último valor lido, ao qual se somará a diferença entre os 4 *GB* e o resto.

Por fim é feito um registo de *log* por cada nó e gravado na diretoria */net_settings/files/*.

Este *script*, que extrai toda a informação dos *routers*, é um *script* que não é executado automaticamente. É utilizado na criação da rede e após a primeira inserção de todos os nós na plataforma de gestão, quer sejam coletivos ou individuais. Só após a inserção de todos os nós, com o devido preenchimento dos formulários é que se executa este *script*. E executa-se várias vezes seguidas até que todos os nós tenham recebido o ficheiro *bat-hosts* atualizado, de maneira a poderem associar os MAC *addresses* aos IPs correspondentes. Como as ligações das transmissões de qualidade, são desenhadas com base no IP do melhor próximo salto (*next best hop*) é necessário que esse campo da base de dados, seja alterado de MAC *address* para IP. Deve-se verificar que na coluna *Next Hop to Gateway*, que é apresentada no separador *Node Status List*, não constem MAC *addresses*, mas sim IPs. Será necessário executar este *script* através do separador *Update Net*, as vezes necessárias até que isso aconteça. Esta tarefa só é necessária repetir, de cada vez que seja adicionado um nó na rede. Trata-se portanto de uma tarefa inicial, feita pelo administrador sempre que é criada uma rede ou inserido um ou mais nós.

Outra situação em que deve ser utilizado este *script*, é quando se altera o campo que permite fazer *reboot* a todos os nós da rede, que se encontra no separador *Network Settings*, com o nome de *Daily Network Reboot Time Schedule*. Se se proceder a alterações neste campo, deve-se proceder uma única vez à execução do *script*, para que seja atualizada esta informação em todos os nós.

O *script_check_nodes.php* (localizado em */var/www/batman-adv-admin*) é análogo ao *script* descrito anteriormente (*check_nodes.php*), embora existam algumas diferenças e também foi criado de raiz para a plataforma de gestão desenvolvida.

Primeiro este *script* é executado automaticamente de 20 em 20 minutos, através do programa *cron*, conforme se demonstrou na Figura 40. Em segundo lugar, não executa

todas as tarefas do *script* anterior (*check_nodes.php*), como por exemplo, a extração da marca e do modelo do *router*, ou a extração da sua memória *flash*, ou a extração da memória RAM inicial, etc., porque esses dados já foram extraídos na criação da rede. Como essas informações já estão disponíveis, poupa-se tempo na extração dos dados dos *routers* e largura de banda, não as solicitando outra vez. Assim este *script* (na sua versão mais reduzida) só vai extrair os dados considerados necessários, como por exemplo o atual valor de memória RAM disponível, o tempo decorrido desde que o nó está ligado, a quantidade de *downloads/uploads*, a lista de vizinhos BATMAN-ADV, a qualidade de transmissão, etc. No final da sua execução vai novamente enviar o ficheiro *bat-hosts* para atualização dos nós, relativamente ao binómio MAC *address* – IP.

Este *script* executa a comunicação com todos os nós, independemente da rede a que pertencem, contrariamente ao *check_nodes.php* que executa a comunicação com os nós pertencentes à rede que está a ser administrada naquele momento. O único critério para que a comunicação ocorra consiste na verificação da atividade do nó. Se o nó estiver desativado, avança para a linha seguinte da base de dados *node*. Devido a este procedimento, podem existir várias redes numa mesma plataforma de gestão e numa mesma base de dados. Não podem existir nós com o mesmo IP. Esta limitação implica que os nós não tenham IPs repetidos, mesmo que pertençam a redes diferentes, porque a comunicação efetuada entre o servidor e os nós é realizada através de SSH, onde é usado o IP para distinção do nó a comunicar. Se existirem dois IPs repetidos, o que acontece é que o primeiro nó a ser encontrado (cujo IP seja igual ao procurado), será considerado como sendo o nó correto, o que por vezes poderá não corresponder à verdade. Como cada servidor poderá gerir várias redes, e como a gama de IPs disponíveis para uso interno é alargada, não será necessária a repetição de IPs. Se isso acontecer, poder-se-á dividir fisicamente a rede e implementar um novo servidor para essa rede.

Além disso o *script_check_nodes.php* é responsável pelo envio das notificações de alerta através de *e-mail* para o administrador. Este *script* ao ser executado chama outro *script* de nome *mailalerts.php* localizado no módulo *lib* que verifica, antes de mais, se existe algum nó de alguma rede que não responda há mais de 30 minutos. Se existir é criada uma mensagem, onde consta a localização e nome da rede, a descrição do nó (se foi inserida), e o tempo decorrido desde a última verificação. Um dos nós da rede "teste_1" foi desligado durante várias horas, o que originou o envio de vários *e-mails*, idêntico ao da Figura 48.

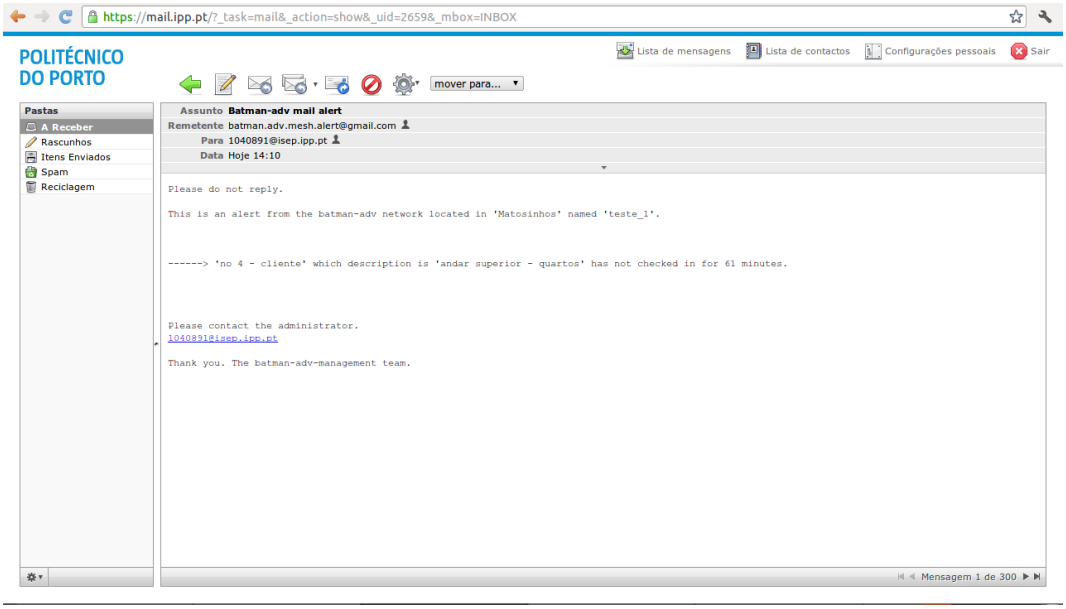


Figura 48 Envio de *e-mail*, devido ao "nó 4 – cliente" não responder há 61 minutos.

De seguida o *script* verifica se os limites de *downloads*, *uploads* e qualidade de transmissão, ultrapassaram o limite de nível 2, e se assim for, também envia notificações de alerta. Para o envio de *e-mails* utilizou-se a função *mail*( ) existente para *scripts* escritos em PHP. A função *send_alert*( ) que se encontra no *script /lib/toolbox.php* é a função utilizada para esse efeito. Para executar o envio utilizou-se o programa *ssmtp*, que se trata de um emulador do programa *sendmail*, numa versão muito mais leve e mais simples, que utiliza o protocolo *Simple Mail Transfer Protocol* (SMTP). Para a emissão automática de *e-mails* criou-se uma conta de *e-mail* no *Google* de nome *batman.adv.mesh.alert@gmail.com*, utilizando-se o *Gmail* como servidor de *e-mails*.

Outro exemplo de notificação de alerta é apresentado na Figura 49, devido ao "nó 4 – cliente" ter ultrapassado o valor de *download*, definido como limite.

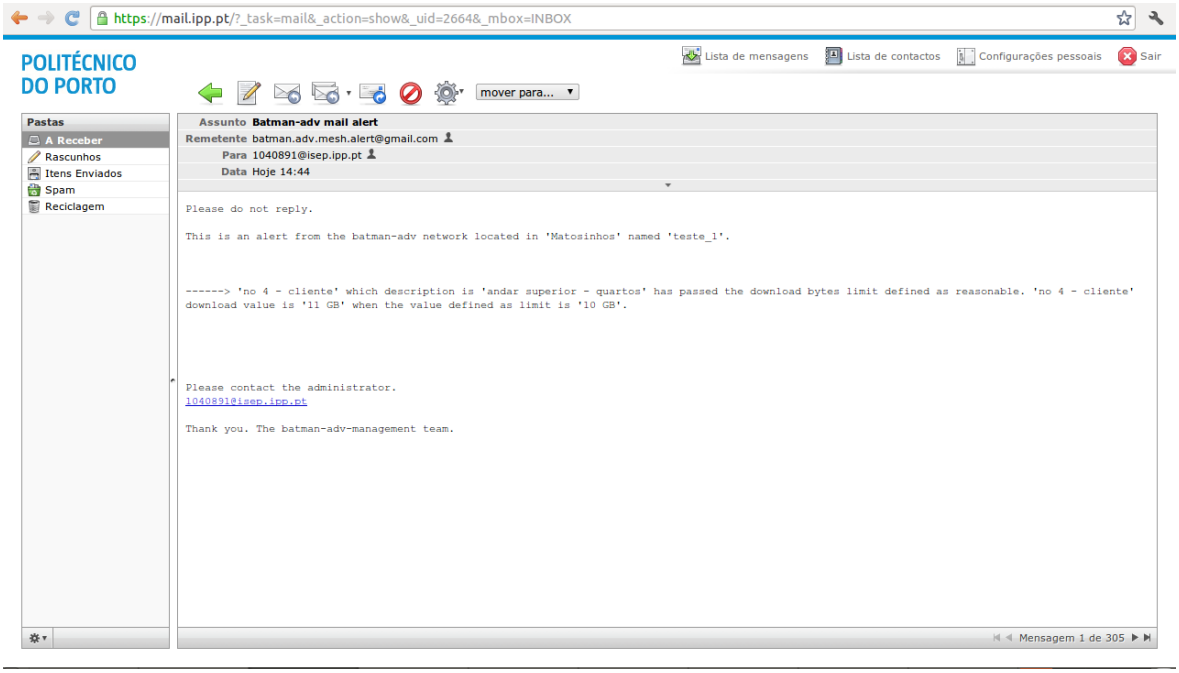


Figura 49 Envio de *e-mail*, devido ao "nó 4 – cliente" ter ultrapassado 10 *GB* de *download*.

Apresenta-se o código que trata do procedimento de envio de notificação de *e-mail*:

```
if ($RX_bytes>$download_limit_2 && $return_2 != "true") // calls
the function 'signal' and send an alert e-mail
      {
      $alert="1";
      $return_2=signal($alert);
      }
```

Conforme se pode observar é verificado se o valor lido relativo à quantidade de *downloads* é superior ao valor estabelecido como limite (nível 2), e se ainda não tiver sido enviado o *e-mail* por este motivo (*$return_2 != "true"*), então chama a função *signal*( ) e como argumento indica que o texto a construir deve ser referente a *downloads*.

No anexo F é descrito o procedimento de instalação e configuração de *ssmtp* [F].

O *script_number_update*, localizado em */var/www/batman-adv-admin* é um *script* criado para esta plataforma de gestão, que é executado automaticamente de 24 em 24 horas, através do programa *cron*, conforme se demonstrou na Figura 40. Este *script* recolhe os valores dos *downloads*, *uploads* e número de clientes de cada nó, soma-os e atualiza a base de dados. Esta atualização é efetuada por cada rede existente na base de dados. No final os valores são apresentados na página correspondente ao separador *Network Settings* e os valores da base de dados são convertidos em zero, para se recomeçar uma nova contagem.

Foi salvaguarda a seguinte condição: se se optar por não fazer *reboot* periódico à rede, o *script* verifica se a contagem é superior à última contagem efetuada. Se assim for, apresenta a diferença, ou seja o valor da contagem desde a última verificação. Se a contagem é inferior, isso significa que foi feito um *reboot* à rede, sendo considerado o valor total da contagem, e será esse o valor apresentado. Por fim é registado em *log* os valores periódicos de *downloads*, *uploads* e número de clientes das redes.

O último *script* também criado para a plataforma de gestão *batman-ad-admin* chama-se *script_encryption.php* e está localizado em */var/www/batman-adv-admin*. A função deste *script* é alterar periodicamente a chave de encriptação WEP da rede *mesh*. Não se trata da rede de acesso aos APs por parte dos clientes. Essa rede está encriptada com WPA2, com TKIP+AES.

Conforme já foi referido, rapidamente se pode quebrar a chave e configurar um nó BATMAN-ADV, para se poder pertencer à rede. Para redes Ad-Hoc, o *firmware OpenWRT* só disponibiliza, para já, a possibilidade de encriptação WEP. Nesse sentido criou-se um *script* que que é executado automaticamente de 24 em 24 horas, através do programa *cron*, conforme se demonstrou na Figura 40. Este *script* gera uma chave aleatória e envia essa chave para todos os nós da rede. Também produz um registo de *log*, onde é indicada a chave de encriptação e caso algum nó não a receba, regista essa informação no *log* e envia uma notificação de alerta através de *e-mail* ao administrador da rede.

O comando que gera a chave aleatória é:

```
$COMMAND_1='dd if=/dev/random bs=1 count=13 2>/dev/null|xxd -ps';
//command to generate a random WEP key for wireless ad-hoc
encryption
```

O resto do procedimento é análogo ao já descrito. O *script* verifica se já existe a diretoria correspondente à rede em questão. Verifica se já existe uma cópia do ficheiro *wireless*. Se existir é editado com a nova chave. Se não existir é criado um novo ficheiro *wireless* com a nova chave e é enviado para todos os nós. Por fim é feito *reboot* automático à rede para que se proceda à alteração da chave de encriptação.

Por fim, faz parte da plataforma de gestão o ficheiro relativo à conceção da base de dados *batman-adv_v2.sql*, que se encontra localizado em */var/www/batman-adv-admin*. Este ficheiro contém a estrutura da base de dados *batman-adv-admin*.

6.3.7. REGISTO DE *LOGS*

São guardado registos de todas as operações efetuadas em */var/www/batman-adv-admin/net_settings/files*.

O *script_check_nodes.php*, que se trata do *script* que comunica com todos os nós de 20 em 20 minutos, regista na diretoria acima especificada o resultado de toda a comunicação que efetua com cada nó, ou seja a resposta a cada pedido efetuado via SSH, respostas essas, que serão guardadas nos campos da base de dados. Este *log* chama-se *log_check_nodes_global* e trata-se de um ficheiro de texto. Se este ficheiro não existir é criado. Se já existir é apensado o novo conteúdo, no final do ficheiro. Como este *script* é responsável pelo envio de *e-mails*, regista também nesse *log* todo o conteúdo do *e-mail* enviado, assim como se o *e-mail* foi enviado com sucesso ou se pelo contrário, falhou o seu envio. Na Figura 50 pode-se observar um exemplo do *log_check_nodes_global*, onde foram registadas as respostas enviadas pelo "nó 4 – cliente", assim como também foi registado o envio de notificação de alerta através de *e-mail*, com sucesso.

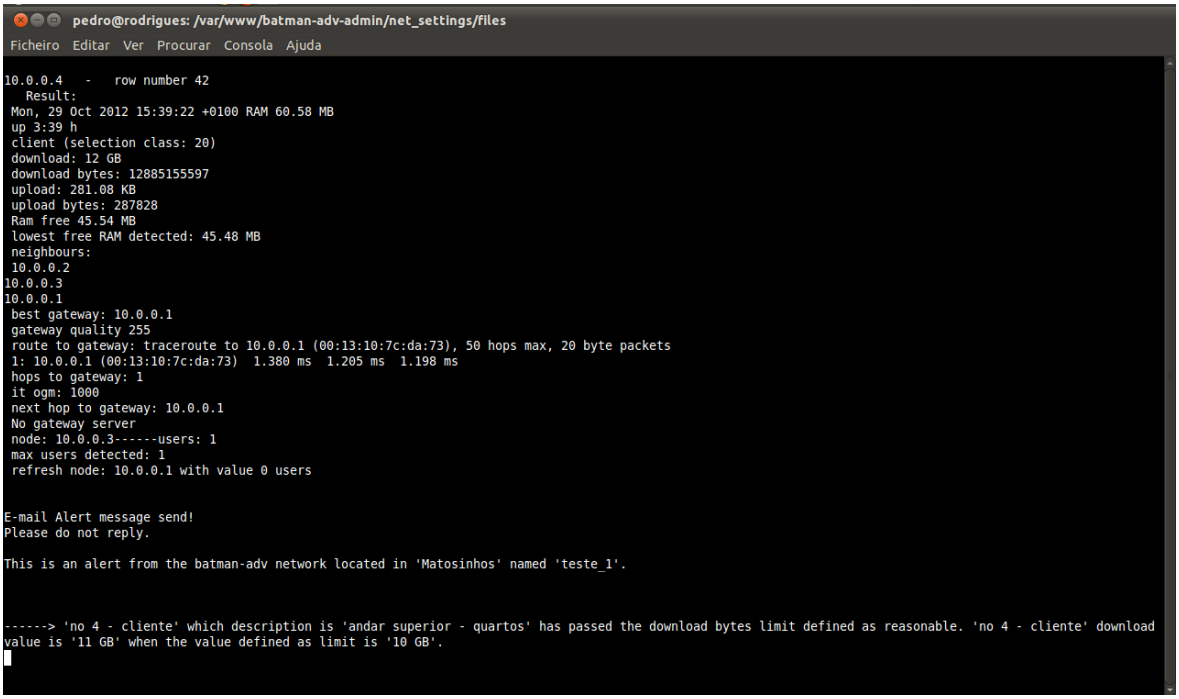


Figura 50 Exemplo extraído do *"log_check_nodes_global".*

Por cada rede que é criada, na primeira iteração que corresponda a uma alteração dos parâmetros iniciais da rede, é criada um pasta com o nome da rede e localização, em */var/www/batman-adv-admin/net_settings/files/"pasta_da_rede_x".*

Qualquer alteração efetuada fica registada em ficheiro de texto (ex. alteração do ficheiro *bat-hosts*, alteração de IPs, alteração de DHCP *limits*, etc.). O *script check_nodes.php*, que se trata da versão integral de comunicação com o nó, cria um *log* de nome *log_check_nodes* dentro da pasta da rede a que pertence, registando também todo o resultado da comunicação dos nós dessa rede específica.

O script *script_number_update.php* também cria um *log* de nome *number_update_log_"nome_da_rede_localização"*. Este *log* é gravado na pasta da rede correspondente. Este ficheiro se não existir é criado. Se existir o resultado é apensado no final no seu final. Na Figura 51 pode-se observar um excerto do *log*.

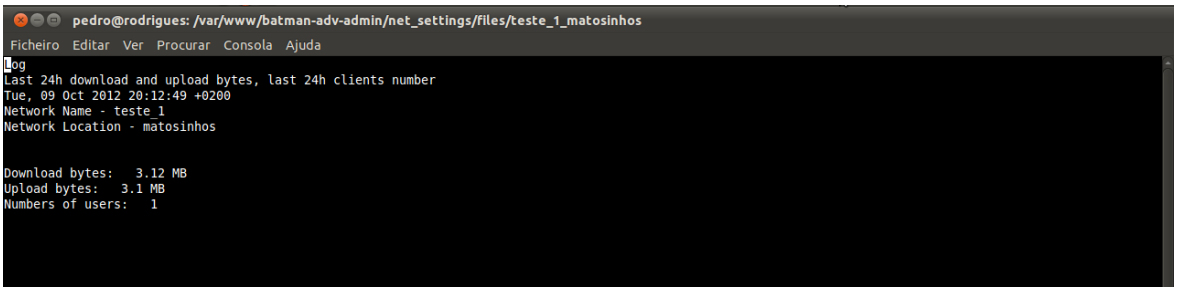


Figura 51 Exemplo extraído do *log "number_update_log_teste_1_matosinhos".*

Por fim o *script script_encryption.php* cria um *log* de nome *encryption_key_log_"nome_da_rede_localização"*, cada vez que é gerada uma nova chave de encriptação. Se não for possível proceder à alteração da chave de encriptação em algum nó, é reportado o erro no *log*, onde é registada a identificação do nó, assim como é registado o envio e conteúdo do *e-mail* de alerta para o administrador. Análogo aos *logs* anteriores, se o ficheiro não existir é criado, se existir é atualizado. Na Figura 52 apresenta-se um exemplo deste *log*.

A ferramenta de gestão foi experimentada nos *web browsers Mozilla Firefox e Google Chrome*, com sucesso.

Faz parte integrante deste relatório, o código da ferramenta de gestão *batman-adv-admin*, em formato de CD.

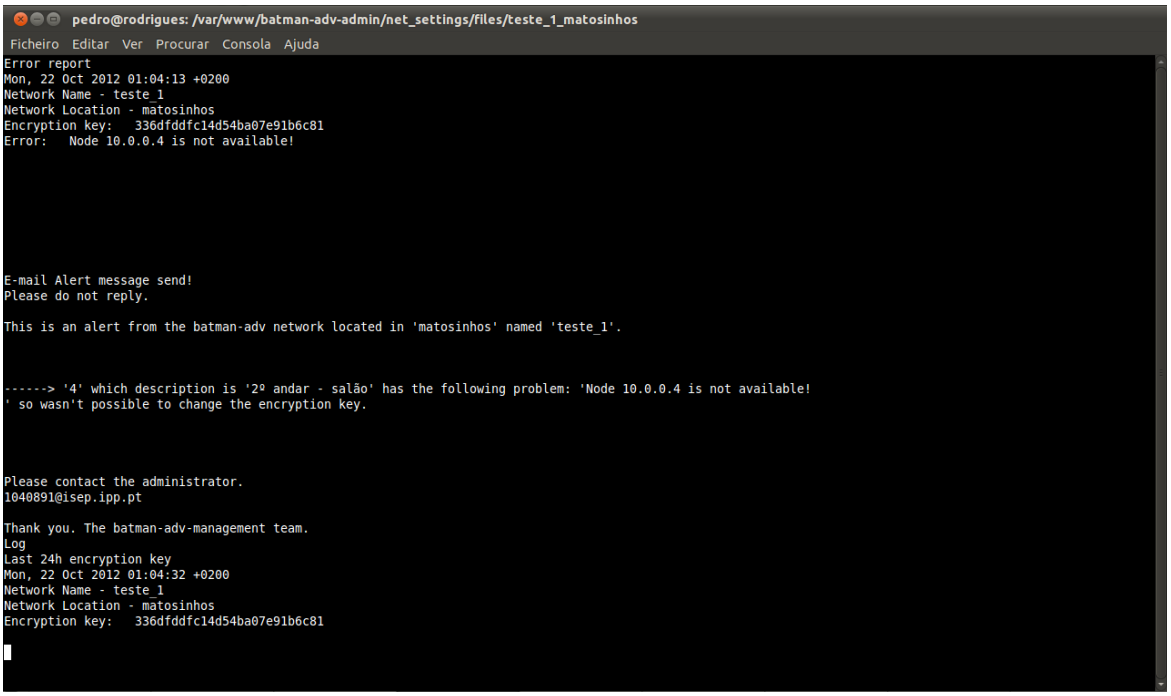


Figura 52 Exemplo extraído do *log "encryption_key_log_teste_1_matosinhos".*

6.4. GESTÃO DOS APS

A gestão dos APs poderá ser efetuada através de monitorização dos *routers*, utilizando-se para isso SNMP. O *firmware DD-WRT* disponibiliza essa opção de monitorização, conforme se pode ver na Figura 53, referente ao AP pertencente à rede BATMAN-ADV.

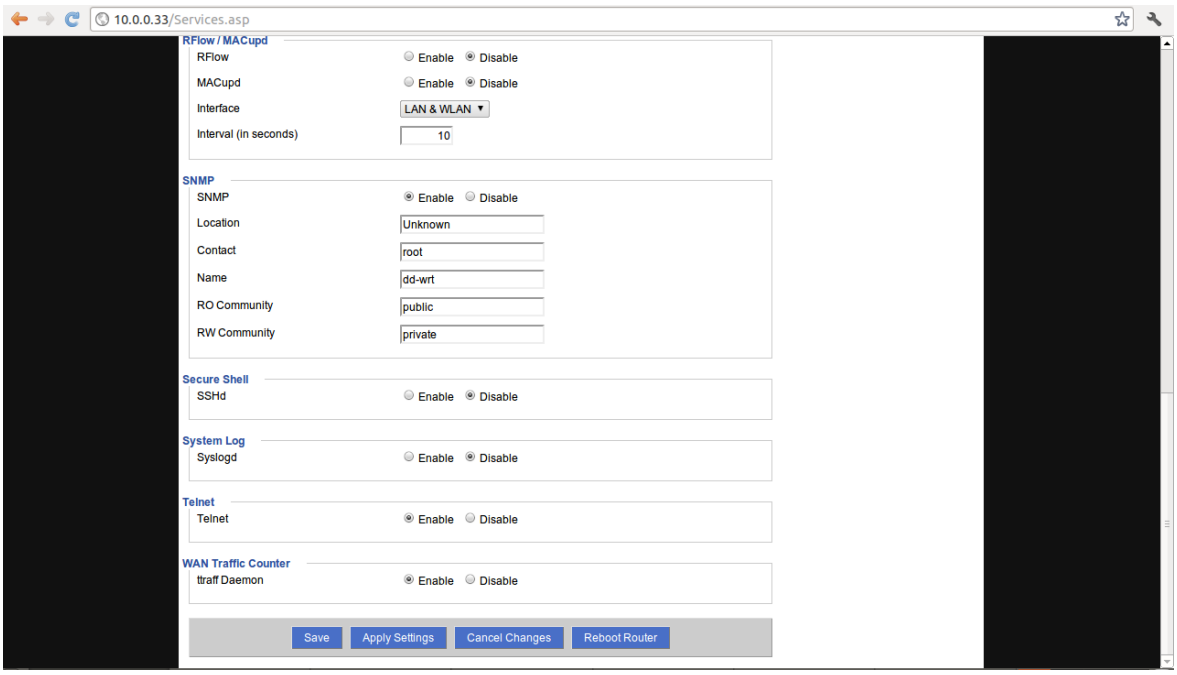


Figura 53 O *firmware DD-WRT* disponibiliza a utilização de SNMP.

No âmbito do trabalho de mestrado "Sistema *Web* de gestão e monitorização remota de elementos de rede baseado em SNMP" desenvolvido no ISEP (Instituto Superior de Engenharia do Porto), cujo estágio foi realizado na empresa *NextToYou – Network Solutions, Lda*., foi criada uma ferramenta que permitiu flexibilizar a gestão, monitorizar e configurar remotamente equipamentos instalados numa infra-estrutura de rede IP comunitária, usando como base o protocolo SNMP [69].

Assim e uma vez que a ferramenta de gestão *batman-adv-admin* foi desenvolvida para poder ser integrada nas *GateBoxes*, poder-se-á aproveitar o sistema de gestão e monitorização desenvolvido para monitorização dos APs, podendo o mesmo vir a ser integrado na ferramenta de gestão da rede BATMAN-ADV.

7. TESTES E VALIDAÇÃO DA PLATAFORMA DE GESTÃO *BATMAN-ADV-ADMIN*

Para o teste e avaliação da plataforma *batman-adv-admin* desenvolvida foi criada uma rede constituída por 4 nós, sendo um dos nós configurado como nó servidor/*gateway* e os restantes 3 configurados como nós clientes.

Neste capítulo será apresentado a caracterização da rede implementada, o resultado dos testes efetuados, assim como algumas considerações finais.

7.1. CARACTERIZAÇÃO DA REDE *MESH* IMPLEMENTADA

A rede BATMAN-ADV foi implementada num ambiente residencial, para uso em interiores. Na Figura 54 pode-se observar a planta onde foram implementados 2 dos nós BATMAN-ADV.

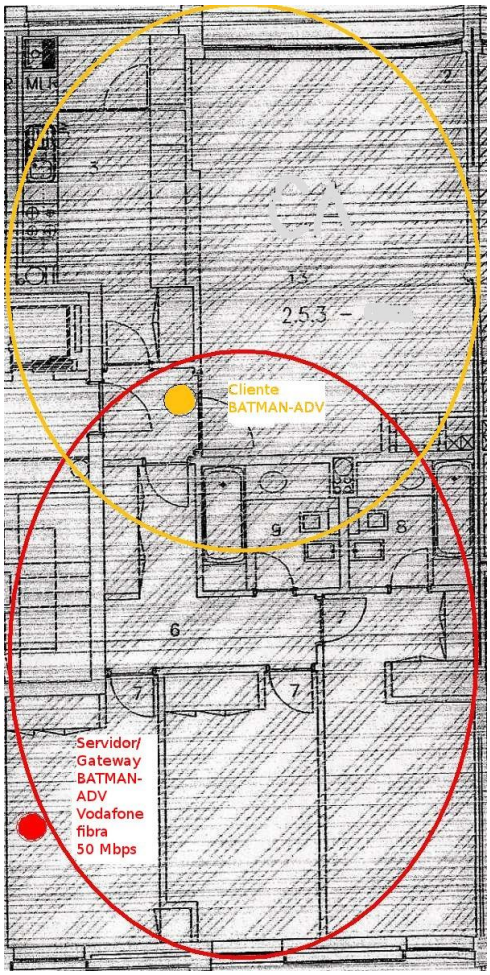


Figura 54 Localização do nós BATMAN-ADV servidor/*gateway* e de nó cliente.

A área da implementação dos 2 nós descritos na Figura 54, tem cerca de 130 m^2 e localiza-se num apartamento. O objetivo de cobertura de sinal está representado pela linha vermelha e pela linha amarela, referentes ao nó servidor/*gateway*, e ao nó cliente, respetivamente. Os outros 2 nós clientes foram instalados no andar imediatamente superior a este, que se trata de um apartamento semelhante. Os nós foram posicionados sensivelmente na mesma localização apresentada na Figura 54, pretendendo de igual forma cobrir a área total do apartamento superior, que têm a mesma área, perfazendo uma cobertura total de aproximadamente 260 m^2. Na Figura 55 apresenta-se a localização dos 4 nós, vista no mapa da plataforma de gestão criada. O nó servidor/*gateway* tem a borda branca. Os nós clientes têm a borda negra. No mapa, o cursor aponta para o nó cliente (no 2 – cliente), que se encontra no mesmo piso do nó servidor. Os outros 2 nós apresentados, encontram-se no piso superior. A rede BATMAN-ADV estende-se por 2 pisos.

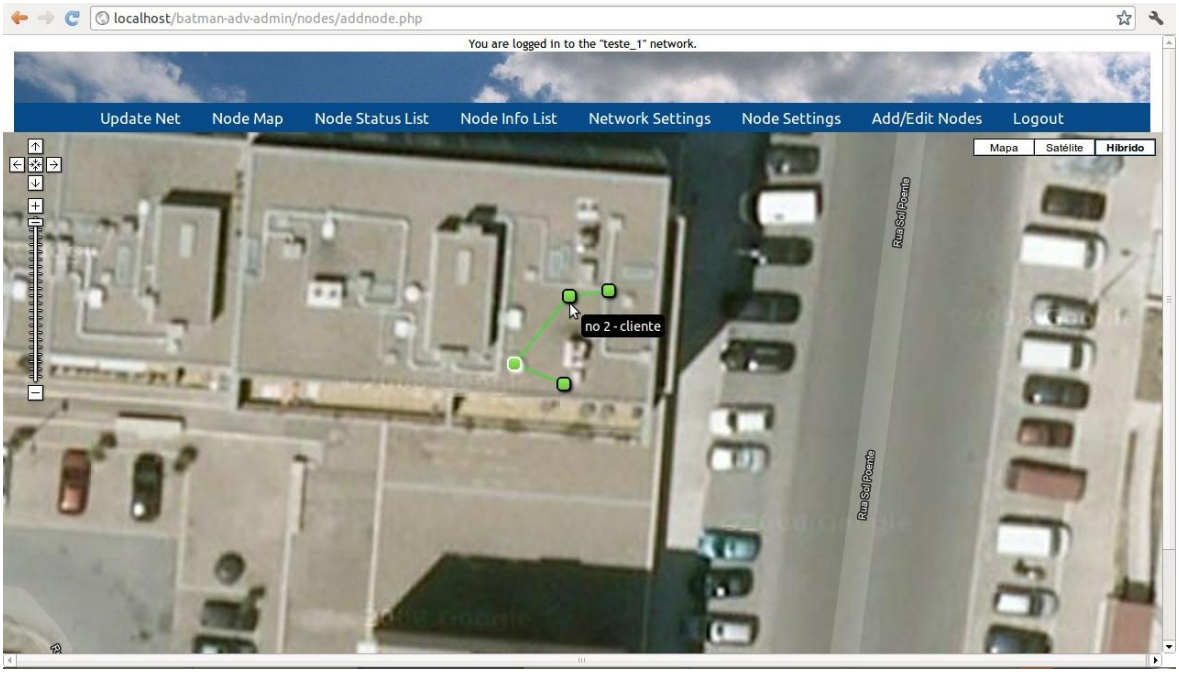


Figura 55 Localização no mapa dos 4 nós BATMAN-ADV.

O posicionamento dos nós foi decidido em função do nível mínimo de sinal recebido (RSL). O nó servidor/*gateway* BATMAN-ADV ficou posicionado no local onde termina a fibra ótica. O serviço fornecido pelo ISP apresenta como capacidade de transmissão 50 *Mbps*. A ligação do nó servidor/*gateway* BATMAN-ADV ao ISP é efetuada através do *router* já existente nesse local do apartamento, utilizando-se um cabo *Ethernet*, que liga a porta WAN do nó BATMAN-ADV servidor/*gateway* à porta LAN do *router* fornecido pelo ISP. O nó 2 encontra-se no mesmo piso do nó 1 (servidor). O nó 3 e nó 4 encontram-se no piso superior. O nó 3 não tem conetividade com o nó 1 (servidor/*gateway*), devido à distância e obstáculos existentes entre os dois. O nó 3 tem um AP agregado, que permite o acesso de clientes via *wireless*. Para o posicionamento dos nós foi utilizado o programa *inSSIDer*, de maneira a aferir o RSL mínimo para a receção de sinal.

Na Figura 56 apresenta-se o esquema da rede BATMAN-ADV.

Os traços contínuos representam ligações através de cabo. Os traços a tracejado representam ligações sem fios.

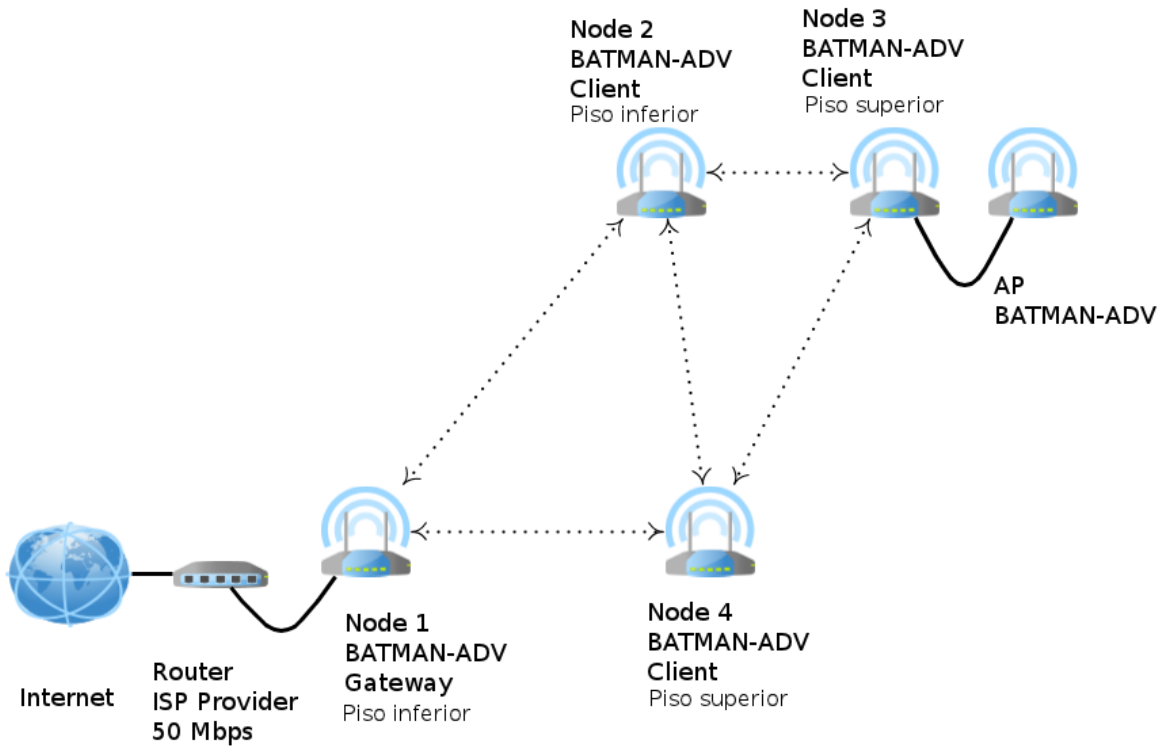


Figura 56 Esquema da rede BATMAN-ADV instalada.

Na tabela 7 apresenta-se a relação de nível de sinal (*dbm*) entre os nós da rede.

Tabela 7 Relação de sinal entre os nós da rede BATMAN-ADV (*dbm*).

Relação (*dbm*)	Node 1	Node 2	Node 3	Node 4
Node 1	---------	-79	---------	-65
Node 2	-79	---------	-59	-86
Node 3	---------	-59	---------	-80
Node 4	-65	-86	-80	---------

7.2. TESTES DE DESEMPENHO

De maneira a garantir o RSL mínimo (-80 *dbm*) entre os *routers*, a distância máxima entre nós não pôde ser superior a cerca de 10 metros de comprimento em linha reta. A distância foi aferida tendo em conta o programa *inSSIDer* e em função dos obstáculos existentes no ambiente residencial testado, tendo sido registados os valores de nível de sinal entre nós apresentados na Tabela 7. Noutros ambientes residenciais terá de se ter em conta a perda do sinal em função do material de construção das paredes, assim como dos objetos que se encontram entre os nós, o que poderá fazer variar as distâncias entre os mesmos.

7.2.1. TESTE DE DÉBITO

Foi realizado o seguinte teste:

O nó que oferece melhor débito na rede *mesh* é o nó que está na fronteira da rede, ou seja o nó servidor/*gateway*. Este nó foi o escolhido, para a realização deste teste, por ser o que oferece melhor serviço em termos de débito.

Na rede "teste_1" efetuou-se um *download* de 4 *GB* utilizando para o efeito o nó servidor/*gateway* e um computador portátil, ligado através de cabo *Ethernet*. O tempo que demorou a fazer o *download* foi de 120 minutos para uma ligação de 50 *Mbps* fornecida pelo ISP. Extrapolando, se tivéssemos um serviço de 100 *Mbps*, demoraria cerca de 60 minutos a fazer o mesmo *download* e para um débito de 400 *Mbps*, atualmente o mais poderoso oferecido em Portugal, demoraria cerca de 15 minutos. Na Figura 57 apresenta-se este raciocínio, através da apresentação de um gráfico.

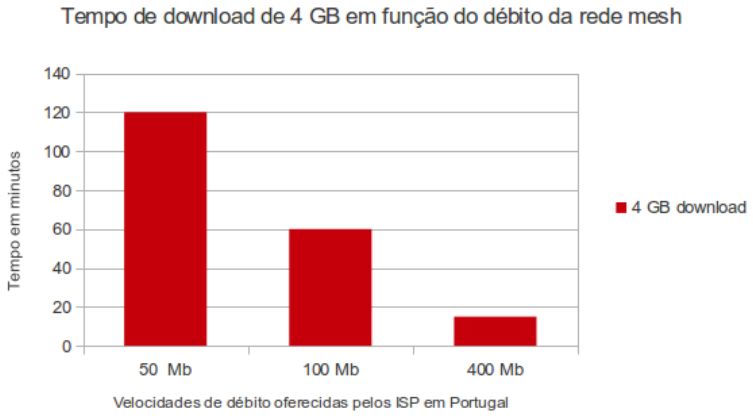


Figura 57 Tempo de *download* de 4 *GB* em função do débito oferecido pelo ISP, utilizando o nó servidor/*gateway*.

Tendo em conta que o *script_check_nodes.php*, entre outras coisas verifica se foi ultrapassado o valor de 4 *GB* referente aos contadores de *downloads* e *uploads* dos nós, teria o *script* de correr de 15 em 15 minutos, para uma velocidade de 400 *Mbps* oferecida pelo ISP. Isto de modo a garantir que interpretava corretamente o valor desses contadores, sem correr o risco de os contadores atingirem o seu limite 2 vezes consecutivas, entre verificações de *script*. Para a ligação utilizada (50 *Mbps*), o *script* teria de correr pelo menos de 120 em 120 minutos para evitar que isso acontecesse. No entanto o *script* corre de 20 em 20 minutos de maneira a enviar alertas se os nós estiverem mais de 30 minutos sem responder.

7.2.2. TESTE DE TEMPO DE RESPOSTA

Outro teste realizado na rede "teste_1" consistiu em verificar o tempo que demora cada nó a responder ao *script_check_nodes.php*. O *script_check_nodes.php* solicita 13 respostas a cada nó (por exemplo o valor de RAM, a lista dos vizinhos, etc.), assim como é o responsável pelo envio das notificações de alerta em caso de deficiência ou de mau desempenho de cada nó. Foram realizados 5 testes e registados os valores referentes aos tempos de resposta de cada nó. Nestes 5 testes estavam ligados apenas 2 utilizadores à rede. Na Tabela 8 apresenta-se os valores dos tempos de resposta dos nós:

Tabela 8 Tempos de resposta dos nós com 2 clientes na rede, em segundos.

Tempo de resposta em segundos – 2 clientes na rede	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
nó 1- servidor	5	5	7	6	8
nó 2 – cliente	6	7	6	8	6
nó 3 – cliente	7	7	8	7	6
nó 4 – cliente	7	8	8	7	7
Média de tempo dos 4 nós	6,25	6,75	7,25	7	7,25

Após 5 testes e estando a rede com 2 utilizadores apenas, verificou-se que para terminar o processo de recolha de dados para os 4 nós que constituem a rede, cada nó demora em

média, cerca de 7 segundos a responder. No entanto não foi possível extrapolar para uma rede com muitos clientes, em virtude de não ter existido essa possibilidade. Contudo foi testado com 7 clientes ligados à rede e verificou-se que o tempo de verificação dos nós aumentou. Após vários testes, o valor médio de resposta por nó demorou cerca de 18 segundos, conforme se pode observar na Tabela 9.

Tabela 9 Tempos de resposta dos nós com 7 clientes na rede, em segundos.

Tempo de resposta em segundos – 7 clientes na rede	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
nó 1- servidor	17	15	17	16	16
nó 2 – cliente	16	18	19	19	17
nó 3 – cliente	17	19	20	17	19
nó 4 – cliente	19	16	19	18	17
Média de tempo dos 4 nós	17,25	17,00	18,75	17,50	17,25

Relativamente a esta questão não foi possível aferir o impacto real dos procedimentos de verificação do estado da rede num cenário real. A degradação da QoS aumenta com o número de clientes. Para se perceber melhor seriam necessários testes numa rede com mais utilizadores.

7.2.3. TESTE DE FALHA DE NÓ SERVIDOR/*GATEWAY*

Foi testada uma outra arquitetura de rede *mesh*, usando novamente os 4 nós, conforme ilustrado na Figura 58. Alterou-se a arquitetura da rede para 2 nós servidores e 2 nós clientes. Ligaram-se os 4 nós e constituiu-se a rede BATMAN-ADV. Cada nó cliente escolheu automaticamente o nó servidor/*gateway*, de acordo com os critérios já referidos anteriormente.

Desligou-se um dos nós servidores/*gateways* e verificou-se que os nós clientes demoram cerca de 200 segundos a encontrar nova rota para o nó servidor/*gateway* remanescente. Durante esses 200 segundos os utilizadores da rede (ligados ao nó cliente, que ficou sem nó servidor) não conseguem aceder à internet. Só quando o *gateway* é excluído da lista

existente no protocolo, é que os nós alteram/atualizam as rotas, o que implica um tempo de espera demasiado longo. Existe outro acontecimento que inviabiliza a mudança. Por exemplo, se o cabo que traz o sinal fornecido pelo ISP à porta WAN do servidor, se desconecta por algum acaso, o nó cliente nunca procurará uma nova rota. O servidor continua a atribuir IP ao nó cliente, mas não tem sinal vindo do ISP. Nesse caso o cliente não muda a rota, pois continua a ser servido pelo nó servidor/*gateway*. Este é um problema grave do protocolo BATMAN-ADV. Só poderá ser resolvido após a tomada de conhecimento do problema, pois não foi desenvolvido neste trabalho maneira de proceder à deteção da falha.

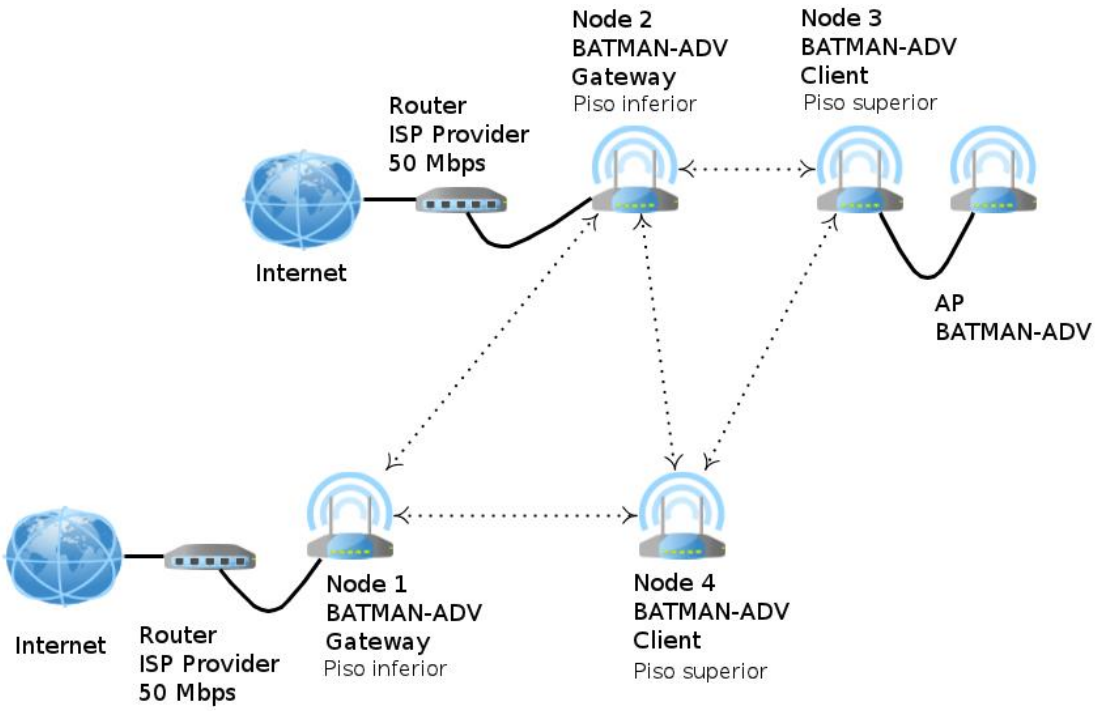


Figura 58 Rede BATMAN-ADV com 2 *gateways* e 2 clientes.

7.2.4. TESTE DE GESTÃO DE DUAS REDES INDEPENDENTES

Foram criadas duas redes independentes, constituídas por um nó servidor e um nó cliente. Ligaram-se essas duas redes a um *switch*, onde também se ligou o servidor onde reside a plataforma de gestão *batman-adv-admin* e verificou-se que a plataforma de gestão funcionou normalmente, independentemente do número de redes. Para se separar as duas redes, alterou-se o BSSID e a chave de encriptação de cada rede.

7.2.5. TESTE DE COBERTURA DE SINAL

Procedeu-se à montagem da rede BATMAN-ADV, conforme demonstrado na Figura 59.

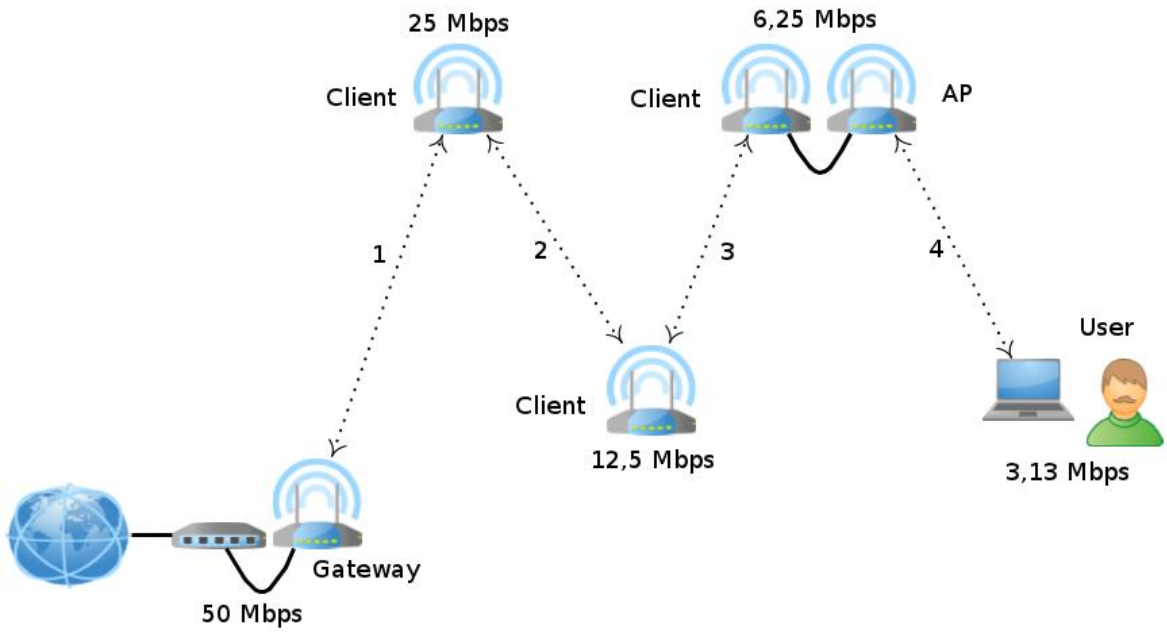


Figura 59 Esquema de teste, efetuado na rede BATMAN-ADV.

Prolongou-se a extensão da rede, utilizando os 4 nós BATMAN-ADV, e verificou-se que por cada salto, a velocidade de *download* decresce em 50 %. Conforme se pode observar, a velocidade de *download* fornecida pelo ISP é de 50 *Mbps*. Após 4 saltos, essa velocidade foi reduzida a 3,13 *Mbps*, velocidade de *download* oferecida a um utilizador da rede via *wireless*.

Este teste mostra que não será possível estender os nós clientes, indefenidamente. No máximo poder-se-á estender a ramificação a 4 nós clientes, embora 3 seja o número ideal de nós clientes, de maneira a obter-se a velocidade de *download* para clientes sem fios de 3 *Mbps*, o que permite por exemplo, a visualização de vídeos na *internet* sem interrupções. Na Figura 60 é apresentada a solução proposta para este problema, para uma rede que seja servida com uma velocidade de *download* fornecida pelo ISP de 50 *Mbps*.

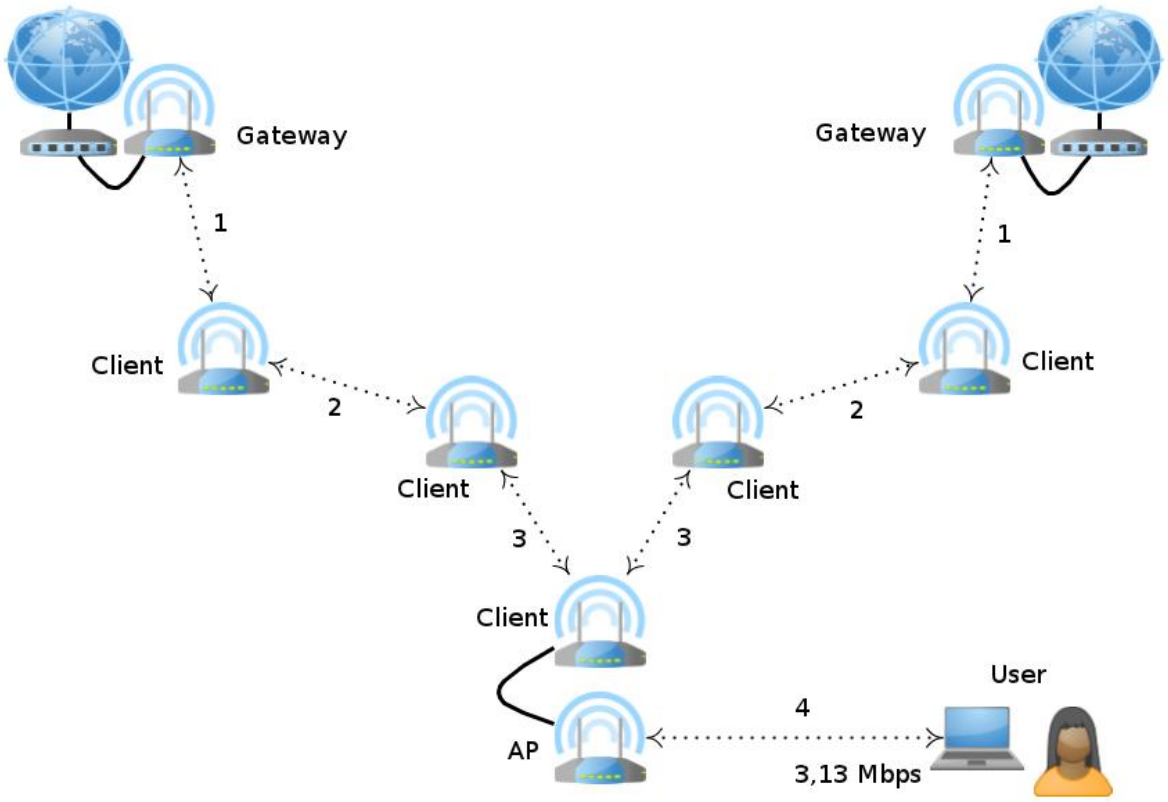


Figura 60 Solução proposta para arquitetura da rede BATMAN-ADV, para 50 *Mbps* fornecidos pelo ISP.

Cada nó servidor/*gateway* nunca deverá servir extensões de nós clientes com mais do que 3 saltos, de forma a garantir qualidade de serviço.

7.3. CONSIDERAÇÕES FINAIS

Quanto a estatísticas de tráfego, utilização dos nós, etc., poder-se-á compilar a imagem de *firmware LuCI*, que integra o *firmware OpenWRT*, e que disponibiliza gráficos e valores individualizados de todas as *interfaces* do *router*. A contrapartida é o aumento do tamanho da imagem final. Como os *routers* usados na rede "teste_1", apresentam uma capacidade razoável de memória RAM e *flash*, a inclusão da *interface LuCI* não causaria qualquer problema de degradação de desempenho, podendo esses dados servirem para efeitos estatísticos.

8. CONCLUSÕES

8.1. REVISÃO DO TRABALHO REALIZADO E PRINCIPAIS CONCLUSÕES

O trabalho desenvolvido teve como objetivo a implementação de uma rede *mesh*, e a criação de uma ferramenta de gestão dessa mesma rede, de maneira a poder ser implementada na solução comercial *GateBox* da empresa NextToYou.

Após estudo de soluções de redes *mesh* e comparação de protocolos de *routing* utilizados, foi decidida a implementação de uma rede *mesh* com base no protocolo BATMAN-ADV, protocolo de *routing* inovador, que trabalha na camada 2 OSI, utilizando as tramas *Ethernet*, para a constituição da rede. Para a sua implementação foi necessário proceder à criação e instalação do *firmware OpenWRT* nos *routers* que constituiram a rede.

A ferramenta de gestão foi desenvolvida tendo por base código *open source*, que teve de ser completamente reformulado de maneira a poder integrar o código desenvolvido de raiz, que serviu para a comunicação com os nós e para a administração e gestão da rede. Aproveitou-se a parte gráfica e organizativa, tendo-se reformulado a parte respeitante à comunicação com os *routers* e reformulado a base de dados em conformidade.

Os objetivos específicos da ferramenta de gestão da rede *mesh* foram alcançados, nomeadamente:

- A obtenção da topologia da rede, e obtenção de informação centralizada, usando-se um protocolo de *routing*;
- A obtenção de valores de funcionamento periódico da rede, respeitantes ao tráfego gerado e transportado;
- A visualização da topologia numa *interface* gráfica ou em forma de texto, de fácil interpretação;
- A aplicação de alterações de configuração da rede e dos nós individualmente;
- A implementação exequível da ferramenta de gestão no produto *GateBox*;
- O envio de notificações de alerta através de *e-mail* para o administrador da rede;
- A possibilidade de integração de novas funcionalidades, conforme forem desenvolvidas, através da inserção de *scripts* nos módulos do código;

O funcionamento da rede e da ferramenta foi validado e apresentou resultados satisfatórios. Pode-se dividir a validação em duas partes distintas: os testes de conetividade da rede e de verificação do funcionamento de protocolo BATMAN-ADV, que funcionou conforme esperado, e os testes à ferramenta de gestão criada.

A ferramenta de gestão da rede permite configurar a rede em tempo real e os *scripts* de verificação do estado da rede e de atualização da base de dados revelaram-se eficazes, apresentando a informação recolhida de uma forma facilmente perceptível. Foram emitidos alertas de notificação automáticos, através do envio de *e-mails*, em função dos parâmetros de configuração determinados na ferramenta de gestão, nomeadamente os limites de *download*, *upload* e de qualidade de transmissão.

É possível integrar nesta ferramenta qualquer protocolo utilizado em redes *mesh*. Para isso basta criar novos *scripts* de comunicação com os nós, em função dos protocolos *mesh* desejados, e extrair os dados, à semelhança do *script* criado para o protocolo BATMAN-ADV. O tratamento e armazenagem dos novos dados recolhidos na base de dados, para posterior tratatamento, é em tudo análogo ao trabalho já desenvolvido, aproveitando-se o trabalho já realizado.

8.2. TRABALHO FUTURO

Um dos aspetos a melhorar consiste na alteração da versão utilizada do *Google Maps* API, que é a versão 2, para a versão 3, uma vez que a versão 2 irá ser descontinuada.

A integração e gestão de novos nós na rede, de marcas e modelos diferentes aos usados, implica que seja compilada nova imagem de *firmware*, que poderá implicar que se tenha que acrescentar ou alterar o código da plataforma de gestão. Esse é um trabalho contínuo e conforme forem associados novos modelos de *routers*, será necessário proceder a testes de validação do seu funcionamento.

Outros aspetos a considerar serão o desempenho da rede com o protocolo BATMAN-ADV e da ferramenta de gestão num cenário real. Para se verificar a viabilidade de se utilizar todo este conjunto de *hardware* e *software* integrado, seria necessário instalar a rede e submetê-la a testes num cenário real.

Em relação ao funcionamento do protocolo, está provado que o mesmo satisfaz as necessidades dos seus utilizadores, conforme se pode aferir pelos projetos idênticos existentes, e pelas comunidades a nível mundial, que o usam.

Em ambientes reais não se consegue aferir se será possível estabelecer comunicação em tempo útil com todos os nós de uma rede, ou o tempo que isso levará. Se a rede estiver a ser utilizada com imensas solicitações e sujeita a uma carga de trabalho considerável, poderá não ser possível fazer as verificações em tempo real. Uma opção que chegou a ser ponderada e que vai de encontro a outras opções existentes noutros projetos, consiste na transferência da iniciativa de comunicação do servidor para os nós, ou seja o *script* que corre no servidor e que inicia a comunicação com os nós, correria nos nós, devidamente adaptado. Passariam os nós a enviar a informação periodicamente para o servidor, que acumularia essa informação numa fila de espera e procederia ao seu tratamento, conforme a sua disponibilidade. Para se proceder a essa alteração, seria necessário testar a ferramenta de gestão num cenário real, e aferir o seu comportamento de forma a tomar a decisão pela alteração da arquitetura pensada para a gestão da rede.

Em termos de redução de custos, poder-se-ia optar por substituir os *routers* utlizados, por *routers* equipados com duas placas de rádio, o que permitiria a eliminação dos APs

agregados aos nós clientes. Existem vários equipamentos no mercado com estas características, que podem ser experimentados, procedendo-se às devidas adaptações.

A implementação do registo de *logs* permite desenvolver novas funcionalidades como por exemplo a criação de gráficos estatísticos, uma vez que a informação necessária é toda guardada e facilmente poderá ser usada para esses fins. Será uma funcionalidade a desenvolver no futuro.

Este trabalho poderá ser melhorado e reformulado quanto às suas funcionalidades e poderá tornar-se numa ferramenta poderosa de administração e gestão de redes *mesh*, desenvolvendo a ferramenta de gestão *batman-adv-admin* para poder ser utilizada com qualquer protocolo de *routing*, desenvolvendo e aperfeiçoando o código criado.

De uma forma geral o trabalho revelou-se desafiante e gratificante. Está apto a funcionar e a ser integrado com qualquer rede *mesh*, cujo protocolo de *routing* seja o protocolo BATMAN-ADV, assim como pode ser integrado na solução *GateBox*. O trabalho desenvolvido corresponde aos objetivos estabelecidos e constitui uma mais-valia para empresa NextToYou.

Referências Documentais

[1] *NextToYou*, http://www.nexttoyou.pt/ [Novembro 15, 2012].

[2] GARCIA PAU ESCRICH, «Quick deployment network using MANET», Universitat Politècnica de Catalunya, 2012, http://hdl.handle.net/2099.1/14103 [Novembro 15, 2012].

[3] YASIR DRABU,«Gateway Placement And Fault Tolerance In QoS Aware Wireless Mesh Networks», PhD thesis, Kent State University, College of Arts and Sciences, Department of Computer Science, 2010.

[4] SEYEDZADEGAN MOJTABA, OTHMAN MOHAMED, MOHD ALI BORHANUDDIN and SUBRAMANIAM SHAMALA, «Wireless Mesh Networks: WMN Overview, WMN Architecture», in *International Proceedings of Computer Science and Information Technology*, vol. 19, 2011, Article 3, published in *International Association of Computer Science and Information Technology Press*, Singapore, 2011.

[5] *Wireless LAN Working Group*, http://www.ieee802.org/11/, [Novembro 15, 2012].

[6] *Wireless Personal Area Network (WPAN) Working Group*, http://www.ieee802.org/15/, [Novembro 15, 2012].

[7] *Broadband Wireless Access Working Group*, http://www.ieee802.org/16/, [Novembro 15, 2012].

[8] MVNet Wireless de Moitas Venda, http://wirelesspt.net/moitasvenda/, [Novembro 15, 2012].

[9] Comunidade *Wireless* de Olhão, http://mwp-olhao.net/, [Novembro 15, 2012].

[10] Comunidade *Wireless* localizada em Fungalvaz, Torres Novas/Tomar, Portugal, http://fungus-wireless.blogspot.pt/, [Novembro 15, 2012].

[11] Comunidade *Wireless* de @rruda dos Vinhos, http://cwav.com.sapo.pt/, [Novembro 15, 2012].

[12] MWP Vale do Sousa, http://mwpvaledosousa.no.sapo.pt/, [Novembro 15, 2012].

[13] Valongo *on Wireless*, http://www.valongowireless.org/, [Novembro 15, 2012].

[14] HUNDEBØLL MARTIN and LEDET-PEDERSEN JEPPE, «Inter-Flow Network Coding for Wireless Mesh Networks», Master Thesis in *Networks and Distributed Systems*, Aalborg University, Spring 2011.

[15] *Projects & Initiatives Building Ad-Hoc Wireless Mesh Networks*, http://emergentbydesign.com/2011/02/11/16-projects-initiatives-building-ad-hoc-wireless-mesh-networks/, [Novembro 15, 2012].

[16] *CloudTrax* – Plataforma comercial de gestão de rede mesh, https://cloudtrax.com/, [Novembro 15, 2012].

[17] *MeshConnect* – Plataforma comercial de gestão de rede mesh, http://wifimesh.trigmax.com/en/index.php, [Novembro 15, 2012].

[18] *Open-Mesh* – Projeto comercial de redes mesh, http://www.open-mesh.com/index.php/, [Novembro 15, 2012].

[19] *Wifi Mesh* – Plataforma comercial de gestão de rede mesh, https://www.wifi-mesh.com/, [Novembro 15, 2012].

[20] *Nodalis* – Projeto comercial de redes mesh, http://www.nodalis.es/, [Novembro 15, 2012].

[21] *Meraki* – Projeto comercial de redes mesh, http://www.meraki.com/, [Novembro 15, 2012].

[22] *Orangemesh* – Plataforma *open source* para gestão de rede mesh, http://orangemesh.sourceforge.net/, [Novembro 15, 2012].

[23] *International Organization for Standardization*, http://www.iso.org/iso/home.html, [Novembro 15, 2012].

[24] FLICKENGER ROB, AICHELE CORINNA, BÜTTRICH SEBASTIAN, DREWETT LAURA M., ESCUDERO-PASCUAL ALBERTO, BERTHILSON LOUISE, FONDA CARLO, FORSTER JIM, HOWARD IAN, JOHNSTON KYLE, KRAG TOMAS, KUPFERMANN GINA, MESSER ADAM, NEUMANN JUERGEN, PIETROSEMOLI ERMANNO, RENET FRÉDÉRIC and ZENNARO MARCO, «Wireless Networking in the Developing World», version 3, Published 2011-11-01.

[25] *List of ad hoc routing protocols*, http://en.wikipedia.org/wiki/List_of_ad_hoc_routing_protocols#table-driven_.28Pro-active.29_routing, [Novembro 15, 2012].

[26] NEUMANN AXEL, AICHELE CORINNA and LINDER MAREK, «Better Approach To Mobile Ad hoc Networking (B.A.T.M.A.N)», in *IETF draft*, October 2008.

[27] *Optimized Link State Routing Protocol* (OLSR), http://www.ietf.org/rfc/rfc3626.txt, [Novembro 15, 2012].

[28] JACQUET, P., MUHLETHALER P., CLAUSEN T., LAOUITI A., QAYYUM A. and VIENNOT L., «Optimized link state routing protocol for ad hoc networks», in *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings.IEEE International*, págs.62-68, 2001.

[29] VENKAT MOHAN. S. and DR. KASIVISWANATH. N., «Routing Protocols for Wireless Mesh Networks», in *International Journal of Scientific & Engineering Research*, Volume 2, Issue 8,August-2011.

[30] DEARLOVE CHRISTOPHER, CLAUSEN THOMAS and JACQUET PHILIPPE, «The Optimized Link State Routing Protocol version-2», in *IETF draft*, September 2009.

[31] *Ad hoc On-Demand Distance Vector* (AODV) *Routing*, http://www.ietf.org/rfc/rfc3561.txt, [Novembro 15, 2012].

[32] HUHTONEN ALEKSANDR, «Comparing AODV and OLSR Routing Protocols», in *Publications in Telecommunications Software and Multimedia*, Helsinki University of Technology Telecommunications Software and Multimedia Laboratory, Seminar on Internetworking, Spring 2004.

[33] SINGH SIKANDER, SINGH SUKHWINDER and DR. CHAND TIRLOK, «Performance Comparison of AODV, OLSR and OFLSR in Wireless Mesh Networks», in *Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008)*, RIMT-IET, Mandi Gobindgarh, March 29, 2008.

[34] *Dynamic Source Routing Protocol* (DSR), http://www.ietf.org/rfc/rfc4728.txt, [Novembro 15, 2012].

[35] CHROBOCZEK JULIUSZ, «The Babel Routing Protocol», in *Internet-Draft*, April 2009.

[36] PERKINS CHARLES E. and BHAGWAT PRAVIN, «Highly Dynamic Destination-Sequenced Distance Vector (DSDV) routing for mobile computers», in *SIGCOMM Computer Communications Review*, vol. 24, no. 4, págs. 234-244, 1994.

[37] ALBRIGHTSON BOB, GARCIA-LUNA-ACEVES J. J. and BOYLE JOANNE, «EIGRP-A Fast Routing Protocol Based on Distance Vectors», in *Net-World/Interop94*, 1994.

[38] FRIGINAL JESÚS, ANDRÉS DAVID DE, RUIZ JUAN-CARLOS and PEDRO GIL, «Towards benchmarking routing protocols in wireless mesh networks», in *ELSEVIER - Ad Hoc Networks - Recent advances on practical aspects of Wireless Mesh Networks*, Volume 9, Issue 8, Págs. 1371-1488, November 2011.

[39] WANG XUDONG and LIM AZMAN O., «IEEE 802.11s wireless mesh networks: Framework and challenges», in *ELSEVIER - Ad Hoc Networks*, Volume 6, Issue 6, Págs. 970-984, August 2008.

[40] JOHNSON D. and MALTZ D., «Dynamic source routing in ad hoc wireless networks», in *Mobile Computing*, edited by *T. Imlellnski and H. Korth, Eds. Kluwer*, págs 153-181, 1996.

[41] ISLAM M. S., HAMID M. A. and HONG C. S., «A Secure Hybrid Wireless Mesh Protocol for 802.11s Mesh Network», in *SPRINGER Computational Science and Its Applications - ICCSA 2008 - International Conference, Perugia, Italy, June 30 - July 3, 2008, Proceedings, Part I*, págs. 972-985, 2008.

[42] HARKINS, D., «Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks», in *Second International Conference on Sensor Technologies and Applications, 2008. SENSORCOMM '08*, págs. 839-844, 2008.

[43] *open80211s* – Implementação *open-source* da norma IEEE 802.11s corrigida, http://open80211s.org/open80211s/, [Novembro 15, 2012].

[44] GARROPPO ROSARIO G., GIORDANO STEFANO and TAVANTI LUCA, «Experimental evaluation of two open source solutions for wireless mesh routing at

layer two», in *International Symposium on Wireless Pervasive Computing (ISWPC), 2010 5th IEEE*, págs. 232-237, 2010.

[45] JOHNSON DAVID, NTLATLAPA NTSIBANE and AICHEL CORINNA, «Simple pragmatic approach to mesh routing using BATMAN», in *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, CSIR, Pretoria, South Africa, 6-7 October 2008.

[46] ABOLHASAN M., HAGELSTEIN B. and WANG J. C.-P., «Real-world performance of current proactive multi-hop mesh protocols», in *Conference on Communications, 2009.APCC 2009.15th Asia-Pacific*, págs. 44 - 47, 8-10 Oct. 2009.

[47] MURRAY D., DIXON M. and KOZINIEC T., «An experimental comparison of routing protocols in multi hop ad hoc networks», in *Telecommunication Networks and Applications Conference (ATNAC), 2010 Australasian*, págs. 159-164, Oct. 31-Nov. 3-2010.

[48] ATHANASIOU G., KORAKIS T., ERCETIN O. and TASSIULAS L., «A Cross-Layer Framework for Association Control in Wireless Mesh Networks», in *IEEE Transactions on Mobile Computing*, Vol. 8, Issue 1, Págs. 65-80, Jan. 2009.

[49] Página de *Wireless Battle Mesh*, http://battlemesh.org/, [Novembro 15, 2012].

[50] NEUMANN AXEL, AICHELE CORINNA, and LINDNER MAREK, «B.A.T.M.A.N Status Report. Technical report, 2007», in http://downloads.open-mesh.org/batman/papers/batman-status.pdf, [Novembro 15, 2012].

[51] BOWITZ ANNE GABRIELLE, «Simulation of a Secure Ad Hoc Network Routing Protocol», Master of Science in Communication Technology, NTNU - NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY, DEPARTMENT OF TELEMATICS, Trondheim, June 2011.

[52] Página do *firmware OpenWRT*, https://openwrt.org/, [Novembro 15, 2012].

[53] Página do projeto de redes *mesh Freifunk*, http://start.freifunk.net/, [Novembro 15, 2012].

[54] Página do *firmware Tomato*, http://www.polarcloud.com/tomato, [Novembro 15, 2012].

[55] Página do *firmware Gargoyle*, http://www.gargoyle-router.com/index.php, [Novembro 15, 2012].

[56] Página do *firmware Robin-Mesh*, http://wiki-robin.meshroot.com/, [Novembro 15, 2012].

[57] Página do *firmware DD-WRT*, http://www.dd-wrt.com/site/index, [Novembro 15, 2012].

[58] Página do *routerMesh-Potato*, http://villagetelco.org/mesh-potato/, [Novembro 15, 2012].

[59] Página do *router Dragino MS-12*, http://www.dragino.com/, [Novembro 15, 2012].

[60] Página do *router Allnet 0305*, http://www.allnet.de/wlan-outdoor.html?&tx_mmallnetproductplugin_pi1[showUid]=490862&cHash=9607eea9657978accc937fe8b43d305f, [Novembro 15, 2012].

[61] Lista de *hardware* compatível com *OpenWRT*, http://wiki.openwrt.org/toh/start, [Novembro 15, 2012].

[62] Centro de desenvolvimento *OpenWTR*, https://dev.openwrt.org/, [Novembro 15, 2012].

[63] Instalação de ferramenta *OpenWRT Buildroot*, http://wiki.openwrt.org/doc/howto/buildroot.exigence, [Novembro 15, 2012].

[64] Página do projeto BATMAN-ADV, introdução ao conceito "bat0", http://www.open-mesh.org/projects/batman-adv/wiki/Quick-start-guide, [Novembro 15, 2012].

[65] Aplicação para visualização de gráficos BATMAN-ADV, http://www.graphviz.org/About.php, [Novembro 15, 2012].

[66] Página de *internet*, com manual de ferramenta de gestão "batctl", http://downloads.open-mesh.org/batman/manpages/batctl.8.html, [Novembro 15, 2012].

[67] JOHNSON DAVID, MATTHEE KAREL, SOKOYA DARE, MBOWENI LAWRENCE, MAKAN AJAY e KOTZE HENK, «Building a Rural Wireless Mesh Network - A do-it-yourself guide to planning and building a Freifunk based mesh network - Version: 0.8», *Wireless Africa*, *Meraka Institute*, South Africa, 30 October 2007.

[68] Configuração de clientes de correio *Gmail*, https://support.google.com/mail/bin/answer.py?hl=pt&answer=78799, [Novembro 15, 2012].

[69] OLIVEIRA DANIEL FILIPE MOURA, «Sistema Web de gestão e monitorização remota de elementos de rede baseado em SNMP», Dissertação de Mestrado em Engenharia Eletrotécnica e de Computadores – Ramo Telecomunicações, FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO, 2011.

Anexo A. Compilação de *firmware OpenWRT*

Neste anexo é descrito como compilar o *firmware OpenWRT*. Existem 2 possibilidades: a versão estável *Backfire* ou a versão experimental *Attitude Adjustment*. Em relação ao protocolo BATMAN-ADV, verificou-se que a versão estável ainda não fora corrigida quanto a algumas funcionalidades existentes nesse protocolo, apresentando alguns erros impossíveis de resolver. No entanto a versão experimental revelou que as funcionalidades do protocolo BATMAN-ADV funcionam corretamente.

Assim, e para compilação da imagem, procedeu-se da seguinte forma (instalação no sistema operativo *Linux – Ubuntu 11.04*):

1. É preciso instalar os pacotes:

   ```
   gcc-multilib bison autoconf screen gcc g++
   binutils patch bzip2 flex make gettext unzip
   libc6 git-core
   ```

   (conforme o sistema operativo, o comando a usar para a instalação está descrito em [63])

2. Executa-se o seguinte comando:

   ```
   svn co svn://svn.openwrt.org/openwrt/trunk/
   ```

   Este comando vai criar um diretório chamado de "*trunk*", para onde vai ser copiado todo o *download* proveniente do repositório *OpenWRT*, que é necessário para compilar e criar a imagem final que irá correr no *router*.

   Caso se pretenda a versão estável, deverá executar-se:

   ```
   svn co svn://svn.openwrt.org/openwrt/branches/backfire
   ```

3. Mudar para a diretoria criada:

   ```
   cd trunk
   ```

   Caso seja a versão estável, mudar para:

   ```
   cd backfire
   ```

4. É necessário editar o ficheiro, onde é determinado que pacotes se pretendem incluir na compilação. É preciso permissões de *root* para poder editá-lo:

```
sudo gedit feeds.conf.default
```

```
src-svn packages svn://svn.openwrt.org/openwrt/packages
#src-svn xwrt http://x-wrt.googlecode.com/svn/trunk/package
src-svn luci http://svn.luci.subsignal.org/luci/trunk/contrib/package
#src-svn phone svn://svn.openwrt.org/openwrt/feeds/phone
#src-svn efl svn://svn.openwrt.org/openwrt/feeds/efl
#src-svn xorg svn://svn.openwrt.org/openwrt/feeds/xorg
#src-svn desktop svn://svn.openwrt.org/openwrt/feeds/desktop
#src-svn xfce svn://svn.openwrt.org/openwrt/feeds/xfce
#src-svn lxde svn://svn.openwrt.org/openwrt/feeds/lxde
#src-link custom /usr/src/openwrt/custom-feed
```

Neste caso, só foram descomentados os pacotes base do *OpenWRT* na 1ª linha e o pacote para a *interface gráfica* na 3ª linha, todos as linhas que começarem por #, serão ignoradas.

5. Atualizar todos os pacotes copiados para a diretoria (executar):

   ```
   scripts/feeds update
   ```

   ou

   ```
   ./scripts/feeds update
   ```

   (conforme o sistema operativo)

6. Instalar os pacotes que possam estar em falta e neste caso poderá ser pertinente fazê-lo para o pacote BATMAN-ADV, que ficará disponível para a compilação:

   ```
   scripts/feeds install -a
   ```

   ou

   ```
   ./scripts/feeds install -a
   ```

   (conforme o sistema operativo)

7. Instalar especificamente o módulo de BATMAN-ADV:

   ```
   scripts/feeds install kmod-batman-adv
   ```

   ou

   ```
   ./scripts/feeds install kmod-batman-adv
   ```

   (conforme o sistema operativo)

8. Iniciar a aplicação que permite selecionar os pacotes a incluir na compilação:

   ```
   make menuconfig
   ```

9. Depois de iniciada a aplicação, primeiro é preciso selecionar o tipo de *chip* onde vai correr a imagem:

```
Target System --->


Atheros AR7xxx/AR9xxx
Atmel AT91
Atmel AVR32
Broadcom BCM2708/BCM2835
Broadcom BCM2708/BCM2835
Broadcom BCM947xx/953xx
…
…
…
etc…
```

   Como para este trabalho foram usados *routers Linksys WRT54GS*, foi escolhida a arquitetura *Broadcom BCM947xx/953xx*. De salientar que para o protocolo BATMAN-ADV, a ditribuição *Linux* escolhida tem de ser superior ou igual a 2.6.

10. Para seleção de *Target Profile*:

```
Target Profile --->

Broadcom BCM43xx WiFi (b43, default)
Broadcom BCM43xx WiFi (wl, proprietary)
Atheros WiFi (ath5k)
No WiFi
BCM4705/BCM4785, Broadcom BCM43xx WiFi (b43)
BCM4705/BCM4785, no WiFi
…
…
…
etc…
```

    Neste caso foi selecionado *Broadcom BCM43xx WiFi (b43, default)*, que se encontra na 1ª linha. Cada *router* terá de ser estudado e verificado quanto aos seus componentes e arquiteturas, de maneira a escolher as opções que se lhe adequam.

11. Devemos selecionar o tipo de imagem que queremos que seja criada. Para isso selecionamos *Target Images*:

```
ramdisk  --->
--- Root filesystem archives
cpio.gz
tar.gz
--- Root filesystem images
ext4
jffs2
squashfs
--- Image Options
```

Para o tipo de imagem a compilar, escolheu-se *squashfs*, que se trata de uma imagem comprimida para instalação no *router*.

12. Este passo é opcional. Trata-se da instalação da componente gráfica. Este *interface* não é de todo necessário, uma vez que não possui a possibilidade de administração da rede *mesh* com BATMAN-ADV. A inclusão desta opção permite ter acesso a imensos dados do *router*, no entanto vai fazer com que a imagem final fique maior, o que para *routers* com pouca memória *flash* poderá condicionar o seu desempenho ou inviabilizar mesmo a instalação.

```
LuCI  --->
Collections  --->
luci-ssl. Standard OpenWrt set with HTTPS support

Modules  --->
--- luci-mod-admin-core..... Web UI Core module
--- luci-mod-admin-full. LuCI Administration -
full-featured for full control
```

Foi selecionado a *interface* gráfica LUCI, com as coleções e módulos acima especificados.

13. Para selecionarmos o módulo de BATMAN-ADV, de maneira a fazer parte da imagem final a instalar no *router*, deve-se proceder da seguinte forma:

```
Kernel modules --->
Network Support --->
kmod-batman-adv
enable bridge loop avoidance
enable batctl
```

Depois de selecionado, carrega-se na tecla "*escape*" + "*escape*" (2 vezes), até que o sistema pergunte se se deseja gravar as configurações para a compilação. Seleciona-se "*yes*".

14. Compilar a imagem:

```
make j -5
```

15. Esperar uma boa meia hora, dependendo da velocidade de processamento do computador e esperar que chegue ao fim.

16. É criada uma pasta de nome "bin" dentro da diretoria *trunk* (ou *openwrt*, se for a versão estável), onde estarão as imagens e pacotes todos compilados. Escolhe-se a imagem para o "nosso" *router* e procede-se à instalação, esperando que funcione.

É possível instalar outros pacotes. Por exemplo, se se quiser incluir os pacotes *X-WRT*, ou SNMP, ou EGGDROP, etc., é uma questão de acrescentá-los. Existem centenas de variações e programas que podem ser desde logo incluídos.

Anexo B. Instalação de *firmware OpenWRT* (*firmware flash*)

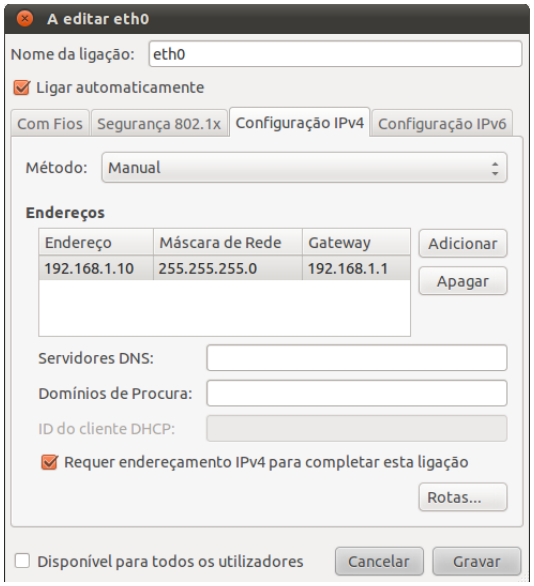
Neste anexo descrevem-se os procedimentos a seguir de forma a mudar o *firmware* de origem de um *router* para o *firmware OpenWRT*. Este procedimento também é válido para instalação de qualquer *firmware*.

Numa primeira fase deve-se consultar em [61], se consta o *router* (marca e modelo) onde se pretende instalar o *firmware OpenWRT*. Se não constar, não é seguro que funcione. Seguidamente explica-se o procedimento a adotar (o computador onde reside a imagem do *firmware* criado tem instalado o sistema operativo *Linux – Ubuntu 11.04*):

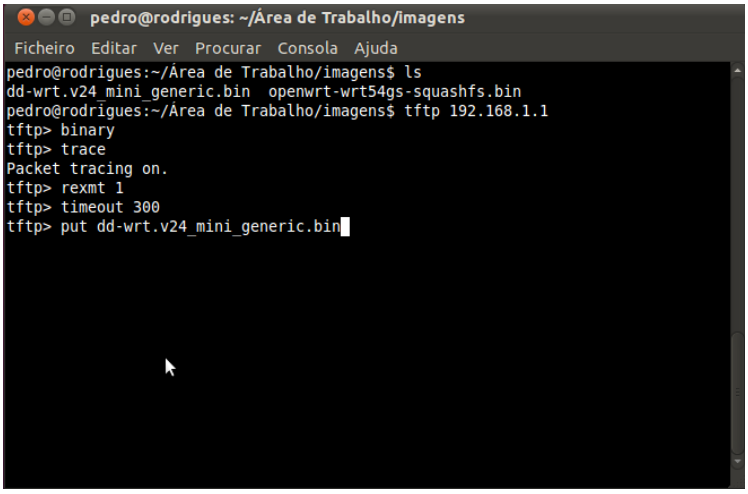
1. Editar as ligações com fios do computador, onde se encontra a imagem do *firmware* a instalar.

2. Alterar a ligação de cabo *Ethernet*, conhecida como *eth0*, relativamente ao IP.

3. Mudar de DHCP para manual e atribuir como endereço IP, por exemplo, 192.168.1.10, máscara de rede 255.255.255.0 e *gateway* 192.168.1.1, este último trata-se do IP do *router*, conforme se pode ver na figura seguinte:



4. Abrir o terminal, mudar para a diretoria onde se encontre a imagem (.bin), que tanto pode ser *DD-WTR*, *OpenWRT* ou a própria imagem original do *router*. Se for possível convém que não seja muito grande, pois numa primeira mudança deve-se escolher a imagem mais pequena possível, quer devido ao tempo disponível para a enviar, quer devido à capacidade de memória *flash*. Apresenta-se um exemplo na figura seguinte:



5. Executa-se o programa TFTP, conforme se demonstra na figura acima.

```
tftp 192.168.1.1
tftp> binary
tftp> trace
Packet tracing on.
tftp> rexmt 1
tftp> put dd-wrt.v24_mini_generic.bin
```

Cada linha é seguida de *enter*.

- A opção "*tftp 192.168.1.1*" indica para onde se vai enviar a imagem, neste caso é o IP do *router*.
- A opção "*binary*" indica que é um ficheiro no formato binário.
- A opção "*trace*" implica que seja efetuada a transferência no modo verboso (com informação do que está a acontecer durante a operação).
- A opção "*rexmt 1*" indica que vai ser tentado enviar o ficheiro de 1 em 1 segundo até conseguir receber sinal para o seu envio, que é dado pelo *router*.
- A opção "*timeout*" indica que durante 300 segundos vai ser tentado o envio do ficheiro e, no caso de não se ter conseguido enviar a imagem, termina a tentativa ao fim desse tempo.

- A opção "*putxxxxxx.bin*" envia a imagem de *firmware* para o *router* ("*xxxxxx.bin"* deve ser alterado pelo nome da imagem a enviar).

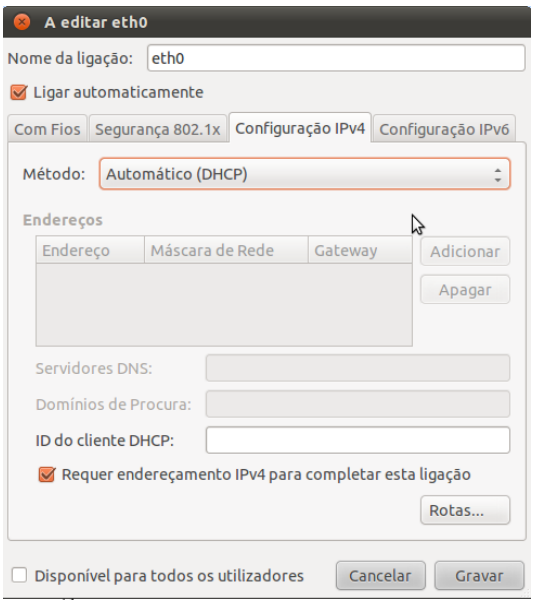
6. Liga-se uma extremidade do cabo *Ethernet* à porta do computador e outra extremidade a uma das portas LAN do *router*.

7. Este passo é o mais delicado, pois pressupõe que todas as instruções que estão descritas no número 5 estejam efetuadas menos a última: *"put xxxxxx.bin*". Uma vez que o *router* ao iniciar espera um tempo pelo recebimento de ficheiros, deve-se enviar a imagem nesse espaço de tempo. Assim, deve-se ligar o *router* à corrente, esperar que a ligação via cabo *Ethernet – eth0*, se concretize (costuma ser rápido, talvez uns 5 segundos), e então nessa altura é que se deve carregar na tecla *enter*, para executar o comando *"put xxxxxx.bin*", para se proceder ao envio do novo *firmware*.

8. Aguarda-se até que a imagem seja totalmente transferida. Isso acontece quando a transferência chega a 100%.

9. Aguarda-se que o *router* faça o *reboot* por sua iniciativa.

10. Após a transferência da imagem deve-se voltar a colocar a ligação *Ethernet* no modo DHCP automático, conforme mostra a seguinte figura:



Se tudo correu bem, a nova imagem foi gravada no *router* e vai arrancar com o novo *firmware*. Caso apresente algum erro é pode-se tentar com outro *firmware*, como por exemplo o *firmware Freifunk*, que também é *OpenWRT* e por vezes funciona bem.

Anexo C. Configuração e instalação de chaves de autenticação no *firmware OpenWRT*

Este anexo descreve os procedimentos a adotar para proceder à criação e instalação de chaves de autenticação no *firmware OpenWRT*. Este método foi experimentado nos *routers Linksys WRT54GS* (equipamentos usados na rede *mesh*).

Após a instalação do *firmware OpenWRT*, para se poder aceder ao sistema *OpenWRT*, deve-se ligar um cabo *Ethernet* a uma das portas LAN do *router*, e aceder ao sistema através do protocolo *Telnet*. O IP de acesso é 192.168.1.1 e deve ser acedido da seguinte forma, seguido de *enter*:

```
telnet 192.168.1.1
```

A página de entrada é semelhante à página ilustrada na Figura 25 [Fig 25]. Seguidamente deve-se mudar a *password*. Por um lado, para evitar o acesso de desconhecidos ao sistema, assim como para permitir a utilização do protocolo SSH, uma vez que para fazer a autenticação, será necessário uma *password*, usado este protocolo. Para definir a *password* deve-se executar o seguinte comado e repetir a *password* escolhida:

```
passwd XXXXXXX
```

Para evitar que seja sempre pedida a *password* foi usado um sistema de chaves de autenticação. Foi criado um par de chaves de autenticação, uma privada e outra pública. A chave privada fica em poder do administrador da rede, enquanto a chave pública é copiada para cada *router*. Para a criação das chaves, abre-se outra janela de terminal, e executa-se o seguinte comando, seguido de *enter* (esta ação foi realizada num computador pessoal, com um instalação *Linux Ubuntu 11.04*):

```
ssh-keygen -t dsa
```

Copia-se a chave pública para a RAM de cada *router* (aqui irá ser pedida a *password* definida anteriormente):

```
scp /home/pedro/.ssh//id_dsa.pub root@192.168.1.1:/tmp
```

O próximo conjunto de operações é feito no *router*, voltando ao terminal ligado através de *telnet*:

```
cd /etc/dropbear
```

```
cat /tmp/id_*.pub >> authorized_keys
chmod 0600 authorized_keys
```

(cada linha descrita corresponde a comando que deve ser executado com *enter*). Assim fica habilitado o acesso direto, ou seja, sempre que o administrador pretenda trabalhar algo dentro do *firmware*, já não necessita de fazer a autenticação. Caso exista mais do que um administrador poderá ser criada uma nova chave privada, que poderá ser habilitada com os mesmos procedimentos, podendo a administração dos *routers* ser feita por mais do que uma pessoa.

Este procedimento deve ser replicado para cada nó BATMAN-ADV.

Anexo D. Configuração do *firmware OpenWRT*

Este anexo descreve os procedimentos a adotar para proceder à configuração do *firmware OpenWRT*. Este método foi experimentado nos *routers Linksys WRT54GS* (equipamentos usados para a conceção da rede *mesh*).

Na rede criada existem dois tipos de nós: os servidores e os clientes. Os nós servidores serão os nós que desempenham a função de *gateways* e os nós clientes serão os nós que propagam a rede *mesh*, para além dos nós servidores.

1. Configuração BATMAN-ADV

A *interface* virtual *mesh*, na realidade, é o sítio para onde se vão enviar os pacotes de dados, de maneira a que estes sejam entregues no destino correto. A *interface* chama-se "*bat0*". É criada pelo protocolo BATMAN-ADV e é atribuído automaticamente um endereço do tipo *Ethernet* a esta *interface*. Qualquer pacote que entre nesta *interface* será entregue noutra *interface* "*bat0*", no endereço de destino [64].

A configuração do ficheiro localizado em */etc/config/batman-adv*, neste caso referente ao nó servidor/*gateway*, utilizada foi:

```
config 'mesh' 'bat0'
        option 'interfaces' 'wifi'
        option 'aggregated_ogms'
        option 'ap_isolation'
        option 'bonding'
        option 'fragmentation'
        option 'gw_bandwidth'
        option 'gw_mode' 'server'
        option 'gw_sel_class'
        option 'log_level'
        option 'orig_interval'
        option 'vis_mode' 'server'
        option 'bridge_loop_avoidance'
```

Conforme se pode observar, esta configuração diz respeito à *interface mesh*, denominada "*bat0"*. As várias opções que são apresentadas no ficheiro são configuráveis.

- *option 'interfaces'* – Nesta opção, define-se que a *interface* que o protocolo BATMAN-ADV vai usar, chama-se "*wifi*". Esta *interface* será criada no ficheiro

"*network*", localizado em */etc/config/network*, conforme será demonstrado mais adiante. É possível habilitar mais do que uma *interface* BATMAN-ADV para usar o protocolo. Nesse caso será necessário configurar cada *interface* criada. Pode-se acrescentar as *interfaces* que se pretender, imediatamente a seguir à *interface* inicial.

- *option 'aggregated_ogms'* – De maneira a reduzir o cabeçalho do protocolo criado por todos os participantes na rede, BATMAN-ADV tem a capacidade de recolher e agregar essas mensagens (OGMs), e envia-as num único pacote em vez de pequenos pacotes. Esta função está habilitada por omissão. Se se pretender usar o BATMAN-ADV num ambiente altamente móvel, como por exemplo num cenário de carros em movimento, poderá ser preferível desabilitar esta opção devido ao atraso provocado na rede (normalmente insignificante). Para desabilitar esta opção configura-se "*option 'aggregated_ogms' '0'*". Para voltar a habilitar deverá ser configurada como "*option 'aggregated_ogms' '1'*". Nesta configuração, esta opção encontra-se habilitada por omissão.

- *option 'ap_isolation'* – Os *access points* tradicionais suportam uma função conhecida como "*AP isolation*" que previne que dois clientes sem fios ligados ao mesmo nó possam "conversar" entre si. Na maioria das situações isto é considerado uma função de segurança. Se um AP Wi-Fi está ligado a uma rede *mesh*, poderá ser desejável manter o isolamento dos seus utilizadores em toda a rede *mesh*. BATMAN-ADV tem a capacidade para o fazer através desta opção, que está desabilitada por omissão. Para habilitar esta opção configura-se "*option 'ap_isolation' '1'*", para voltar a desabilitar configura-se "*option 'ap_isolation' '0'*". Nesta configuração, esta opção, encontra-se desabilitada por omissão.

- *option 'bonding'* – Quando os nós da rede *mesh* têm múltiplas *interfaces*, o BATMAN-ADV é capaz de otimizar o fluxo de tráfego de maneira a melhorar o desempenho. Por omissão o protocolo opera no modo "*interface alternating*" (que é o mais adequado para a maioria das situações). Este modo faz com que a *interface* seja comutada em cada salto de modo a evitar a acumulação e reencaminhamento de informação. Em alternativa, o BATMAN-ADV pode ser comutado para o "*bonding mode*", onde são usadas todas as *interfaces* para receber e enviar dados ao mesmo tempo. Contudo, este modo só é recomendado em casos especiais de um só

salto. Para habilitar esta opção, configura-se "*option 'bonding' '1'*", para voltar a desabilitar esta opção configura-se "*option 'bonding' '0'*". Nesta configuração, esta opção encontra-se desabilitada por omissão.

- *option 'fragmentation'* – BATMAN-ADV implementa, na camada 2 OSI, fragmentação de pacotes *unicast* que atravessam a rede *mesh*, de maneira a funcionarem em *intefaces*/conexões que não permitam o aumento do valor padrão do MTU que é 1500 *bytes*. Quando a fragmentação está habilitada, BATMAN-ADV automaticamente fragmenta os pacotes que excedem o tamanho e volta a compactá-los quando atinge o fim dessa *inteface*/conexão. Por omissão a fragmentação está habilitada e inativa (se o pacote não excede o tamanho permitido), mas é possível desativar a fragmentação por completo. Para habilitar configura-se "*option 'fragmentation' '1'*", para voltar a desabilitar a opção configura-se "*option 'fragmentation' '0'*". Nesta configuração, esta opção encontra-se habilitada por omissão.

- *option 'bridge_loop_avoidance'* – Esta função é pertinente em redes onde existem nós ligados por mais do que uma *interface*. Imagine-se que dois nós pertencentes ao *backbone* estão ligados via LAN um ao outro, mas também estão ligados via *wireless*. Se por exemplo for efetuado um pedido ARP, esse pedido ao chegar à LAN difunde-se pelos dois nós referidos, que por sua vez difundem novamente via *wireless*. Ao chegar o pedido ao nó, este vai propagá-lo via *wireless*, e por sua vez, vai voltar a ser recebido pelo outro nó, que vai enviar o pedido outra vez de volta à LAN. Tem-se um problema de repetição de pacotes (*bridge loop*). Esta opção trata este problema e evita-o. Para habilitar esta opção, configura-se "*option 'bridge_loop_avoidance' '1'*", para voltar a desabilitar esta opção "*option 'bridge_loop_avoidance' '0'*". Nesta configuração, esta opção encontra-se desabilitada por omissão.

- *option 'gw_mode'* – Esta opção permite definir se o nó é servidor ou cliente. Se for servidor partilha a sua conexão de *internet* com a rede *mesh*, se for cliente procura a melhor ligação à *internet*. No caso da rede implementada neste trabalho existem nós de ambos os tipos. Para habilitar um nó servidor configura-se "*option 'gw_mode' 'server'*", se for cliente configura-se "*option 'gw_mode' 'client'*". No exemplo acima descrito mostra-se a configuração para nó servidor.

- *option 'gw_bandwidth'* – Esta opção permite cada servidor/*gateway* anunciar a largura de banda disponível para a rede *mesh*. Contudo, esse valor tem de ser configurado manualmente, pois ainda não se encontra disponível uma função que calcule automaticamente a largura de banda e a divulgue pela rede *mesh*. No momento em que este trabalho está a ser escrito, encontra-se em desenvolvimento uma função que visa alcançar esse propósito. No decorrer do desenvolvimento da ferramenta de gestão da rede, ponderou-se a hipótese de calcular essa mesma largura de banda, através de amostragens em intervalos de tempo. No entanto, para termos um valor fidedigno teríamos que proceder à implementação de um método estatístico onde fosse reduzida a margem de erro, de forma a obter-se um nível de confiança elevado. Optou-se por declinar essa implementação, uma vez que seria necessário fazer um estudo prévio do uso de uma rede real, onde se pudesse aferir quais os tempos de amostragem ótimos, para se poder efetuar o cálculo da largura de banda o mais próxima possível da realidade. A função que está a ser desenvolvida, tem como princípio, o método utilizado por várias soluções existentes *online* conhecidas como "*speedmeter*". Esse método consiste no envio de um determinado número de pacotes para um servidor situado algures na *internet* que posteriormente envia resposta do seu recebimento, obtendo-se uma janela temporal que servirá para o cálculo da velocidade de transmisão. Essa velocidade é calculada em função da quantidade de dados enviada/recebida num determinado intervalo de tempo. Esta opção "*option 'gw_bandwidth'*" pode ser configurada nos nós servidores. Caso não seja configurada, anunciará larguras de banda por omissão de *2048KBit/512KBit*, referentes a *download* e *upload*, respetivamente. Por exemplo, para ser configurada a velocidade de 5 *MB* de *donwload* e 1 *MB* de *upload*, configurar-se-ia "*option 'gw_bandwidth' '5mbit/1024'*".

- *option 'gw_sel_class'* – Se existirem múltiplas *gateways* disponíveis, o nó BATMAN-ADV cliente selecionará a melhor *gateway*, com base em critérios como a qualidade de ligação, a largura de banda anunciada, etc. Por omissão é configurado como melhor ligação, aquela que apresentar um certo valor de qualidade de transmissão (valor por omissão 20), ou seja, quando uma ligação superar em 20 pontos o valor da qualidade de transmissão em vigor, será alterada a ligação para essa nova ligação. A configuração para este caso seria "*option 'gw_sel_class' '20'*". Também é possível definir como preferência a ligação mais

rápida, que considera a largura de banda anunciada, como também é possível definir a ligação com base na qualidade de transmissão. Para este caso a configuração seria "*option 'gw_sel_class' '1'* ". Outra hipótese pode ser a estabilidade da ligação, ignorando a largura de banda e tendo em consideração a qualidade de transmissão – "*option 'gw_sel_class' '2'*". Outra alternativa consiste na escolha da ligação com base na transmissão de qualidade, ou seja, quando encontrar outra ligação melhor, muda imediatamente – "*option 'gw_sel_class' '3'*". Por fim, pode-se definir quantos pontos de qualidade de transmissão serão necessários para se mudar a ligação; neste caso a configuração seria "*option 'gw_sel_class' 'XX'*", onde "*XX*" será o valor de pontos a considerar para a mudança. Como é óbvio não é possível utilizar os valores "*1*", "*2*" nem "*3*" nesta última opção. Por omissão esta opção foi configurada tanto nos nós servidores como nos nós clientes, conforme se apresenta na configuração apresentada no início deste anexo.

- *option 'log_level'* – Esta opção não foi compilada no *kernel*. Serve para criação de *logs* relacionados com *routing*, envio de mensagens, rotas, tabelas de translação, etc. Esta opção é bastante útil quando se está a configurar a rede *mesh*, de forma a perceber o que acontece a cada momento. Esta opção não é configurada em nenhum nó, uma vez que isso implicaria que a imagem final do *firmware OpenWRT* ficasse maior, o que não é de todo desejável por razões já previamente explicadas.

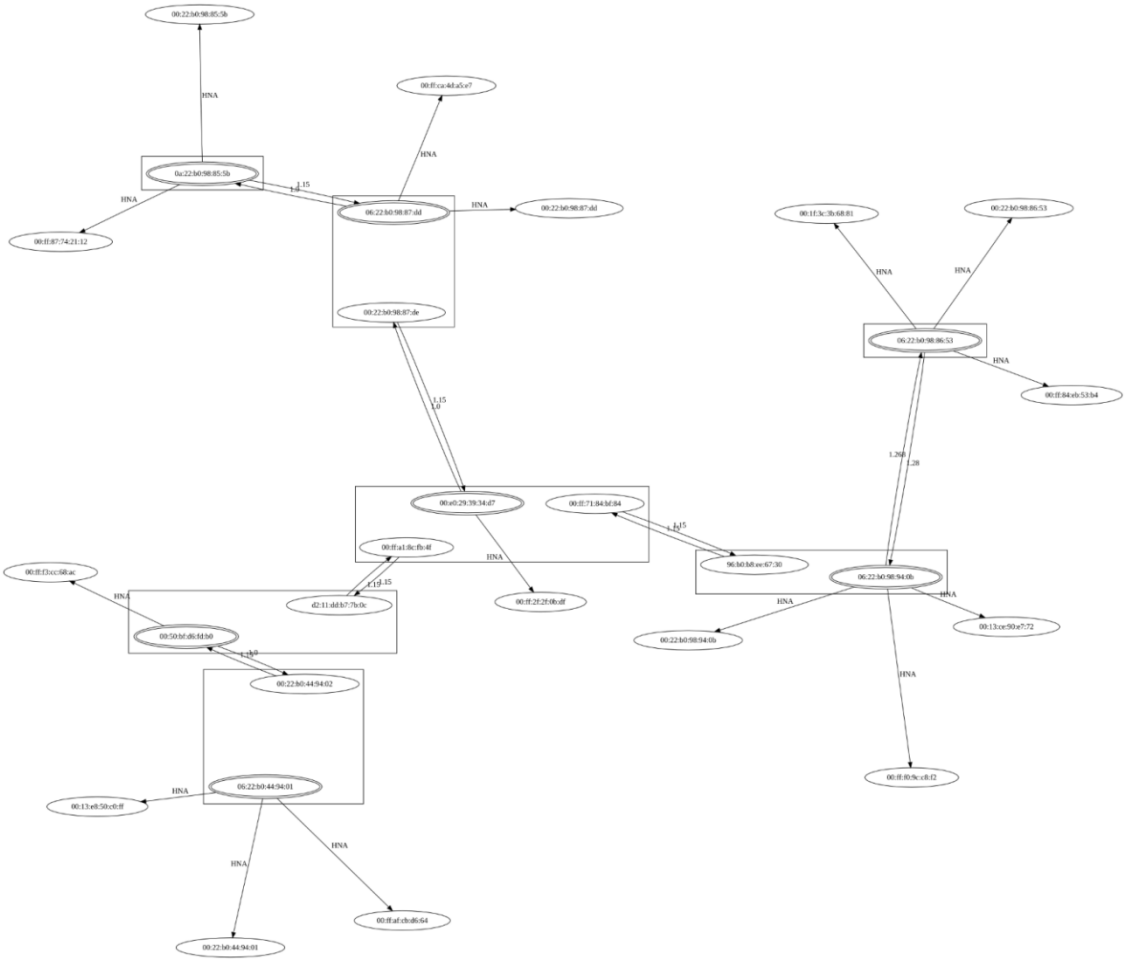
- *option 'orig_interval'* – Esta opção especifica o intervalo em milisegundos, com que o BATMAN-ADV envia OGMs para a rede. O valor de uma mensagem por segundo permite o protocolo reconhecer imediatamente a alteração de uma rota na sua vizinhança. Após um minuto reconhece a alteração de uma rota completa (a maior parte da vezes muito mais cedo). Num ambiente estático (nós que não se movem e raramente são ligados/desligados) pode-se aumentar o valor de "*option 'orig_interval'*", de modo a poupar largura de banda. Por outro lado, poderá ser benéfico decrementar o valor em ambientes altamente móveis, mas como consequência o volume de tráfego será aumentado. O valor por omissão é de 1000 *ms*. Na configuração dos nós foi deixado o valor por omissão. Para incrementar a

configuração configurar-se-ia por exemplo "*option 'orig_interval' '2000'*". Para decrementar configurar-se-ia por exemplo "*option 'orig_interval' '500'*".

- *option 'vis_mode'* – Esta opção é usada para obtenção de um sistema gráfico de visualização da rede BATMAN-ADV. Para a visualização deste gráfico é necessária uma ferramenta de interpretação de grafos. O protocolo cria um ficheiro "*.dot*", que terá de ser interpretado de maneira a obter-se um gráfico idêntico ao gráfico aqui apresentado. Na conceção deste trabalho foi usada a aplicação "Visualizador interativo para ficheiros de pontos *Graphviz*", existente no centro de programas *UBUNTU* 11.04, do sistema operativo *Linux*. Outra opção de aplicação para visualização de gráficos poderá ser *Graphviz - Graph Visualization Software* [65]. É necessário definir se o nó é um nó servidor de visualização. Após essa definição, automaticamente todos os outros nós se transformam em clientes de visualização, enviando informação para o nó servidor, que criará o ficheiro "*.dot*". Na rede implementada foi definido que os nós servidores/*gateways* da rede *mesh* também são os nós servidores do modo de visualização,.

  Na figura seguinte, é ilustrada uma representação gráfica de uma rede BATMAN-ADV, utilizando para o efeito a ferramenta de visualização disponível na sua própria implementação, onde:

  - Elipses representam nós BATMAN-ADV e hóspedes da rede, que são identificados pelos seus MAC *addresses*.
  - Caixas representam *interfaces* pertencentes a um nó BATMAN-ADV.
  - Círculos duplos representam a *interface* primária de um nó BATMAN-ADV, que é conhecida pelos outros nós BATMAN-ADV, exceto os vizinhos diretos.
  - Elipses com uma seta HNA representam os clientes da rede *mesh*, que pode ser também um nó BATMAN-ADV de outra rede *mesh*.
  - Setas com números representam a qualidade de transmissão na forma 1/TQ, desde uma *interface* de um nó BATMAN-ADV para outra *interface* de outro nó BATMAN-ADV.



Para melhor utilização e depuração de todas estas opções foi criada a ferramenta "*batctl*" – (*batman control*), que permite configurar e apresentar informação, ferramenta fundamental e de extrema utilidade [66]. Para solicitar ajuda, executa-se "*batctl –h*", no terminal de ligação ao *router*.

2. Configuração do ficheiro *network*

É no ficheiro localizado em */etc/config/network/*, que será criada a *interface mesh*, que conforme já foi referido, será denominada de "*wifi*". A configuração é a seguinte:

```
#### VLAN configuration
config switch eth0
        option enable   1

config switch_vlan eth0_0
        option device   "eth0"
        option vlan     0
        option ports    "1 2 3 4 5"

config switch_vlan eth0_1
        option device   "eth0"
        option vlan     1
```

```
        option ports    "0 5"

#### Loopback configuration
config interface loopback
        option ifname   "lo"
        option proto    static
        option ipaddr   127.0.0.1
        option netmask  255.0.0.0


#### LAN configuration
config interface lan
        option type     bridge
        option ifname   "eth0.0 bat0 wlan0"
        option proto    static
        option ipaddr   10.0.0.1
        option netmask  255.255.255.0
        option mtu      1500

#### WLAN configuration
config interface        wifi
        option proto    none
        option ifname   "wlan0"
        option mtu      1528

#### WAN configuration
config interface        wan
        option ifname   "eth0.1"
        option proto    dhcp
```

O *firmware OpenWRT* permite a configuração de VLANs, permitindo a gestão das portas disponíveis no *router*. Neste caso foram criadas duas VLANs: a "*vlan eth0*_1", à qual pertence a porta WAN, e a "*vlan eth0_0*", à qual pertencem as outras quatro portas *switch Ethernet* do *router*.

É criada a *interface mesh*, que foi definida no ficheiro localizado em *etc/config/batman-adv*, como sendo a *interface* que está à disposição do protocolo BATMAN-ADV, de nome *interface* "*wifi*". O protocolo aplicado a esta *interface* é "none", uma vez que o BATMAN-ADV é que se encarregará de fazer a sua gestão. O nome atribuído é "*wlan0*". E o MTU definido é de 1528, devido à necessidade de encapsulamento do cabeçalho BATMAN-ADV.

Na secção "*LAN configuration*" é criada uma *bridge*, que integra a *interface mesh* "*bat0*". Assim ficarão ligadas por uma ponte a "*vlan eth0_0*", a *interface bat0* e a *interface* "*wifi*". A *interface* "*bat0*" é que fará a gestão da entrega dos pacotes no destino correto. Se o destino do pacote for o próprio nó, a *interface* "*bat0*" entrega o pacote a "*vlan eth0_0*", se

o destino do pacote não for o próprio nó, a *interface* "*bat0*" entrega o pacote a "*wifi*". Por sua vez a *interface* "*wifi*" transmite o pacote para o *router* escolhido (*best next hop*), que o recebe na sua própria *interface* "*wifi*" e volta a entregar o pacote à *interface* "*bat0*". Também aqui nesta secção é determinado o IP do *router* que servirá para o uso de ferramentas da camada 3 de ISO, como por exemplo o acesso ao *router* através de SSH.

A porta WAN faz sentido nos nós servidores/*gateways*. Nos nós clientes esta porta deixa de ser necessária e pode ser configurada como sendo uma porta *Ethernet*, aumentando assim a possibilidade de aceitar mais um cliente através de cabo. Para isso basta eliminar uma das VLANs e atribuir mais uma porta à VLAN remanescente, neste caso "vlan eth0_0". Cada *router* poderá ter cinco utilizadores de cabo ou até poderá ser ligado a um *switch* e ter um número de utilizadores maior. O número de utilizadores poderá ainda ser maior se o nó BATMAN-ADV tiver um AP agregado. É claro que quantos mais clientes ligados ao *router*, menor será a largura de banda disponível e menor o desempenho global, consideração a ter em conta na atribuição de IPs.

3. Configuração do ficheiro DHCP

A configuração do ficheiro localizado em */etc/config/dhcp* é a seguinte, neste caso para o nó servidor:

```
config dnsmasq
        option domainneeded     1
        option boguspriv        1
        option filterwin2k      0  # enable for dial on demand
        option localise_queries 1
        option rebind_protection 1  # disable if upstream must serve
RFC1918 addresses
        option rebind_localhost 1  # enable for RBL checking and
similar services
        option local    '/lan/'
        option domain   'lan'
        option expandhosts      1
        option nonegcache       0
        option authoritative    1
        option readethers       1
        option leasefile        '/tmp/dhcp.leases'
        option resolvfile       '/tmp/resolv.conf.auto'

config dhcp lan
        option interface        lan
        option start    100
        option limit    40
        option leasetime        5m
```

```
config dhcp wan
        option interface        wan
        option ignore   1
```

A atribuição de IPs aos clientes é determinada pelos nós servidores. O nó, do qual se apresenta o ficheiro *dhcp*, pode atribuir 40 IPs, conforme se pode observar na secção "*config dhcp lan*", ou seja, permite a ligação de 40 clientes no máximo. A atribuição começa no valor 100, o qual vai sendo incrementado conforme a distribuição de IPs. O tempo de cada atribuição de IP é de 5 minutos, definido em função da necessidade de obtenção do número de clientes de cada nó em tempo real, quando solicitado pela ferramenta de gestão de redes *mesh*, desenvolvida. Verificou-se que para tempos de atribuição maiores, o número de clientes de um nó demorava a ser atualizado, sempre que um cliente se ligava ou desligava da rede. O cliente não percebe que o seu tempo de atribuição terminou, pois sempre que solicita um pedido à rede, é renovado por mais 5 minutos a sua atribuição de IP (se eventualmente entretanto a atribuição terminou).

Os nós clientes não atribuem IPs. A sua configuração será idêntica à apresentada, exceto em "config dhcp lan":

```
config dhcp lan
        option ignore 1
```

Assim, a atribuição dos IPs aos utilizadores da rede será feito pelos nós servidores, independentemente de os utilizadores estarem ligados aos nós servidores ou aos nós clientes. Os nós clientes que compõem a rede *mesh* escolhem a rota para a *gateway* com base em critérios de qualidade e desempenho, conforme referido anteriormente. Assim, a rota escolhida para a *gateway*, pelo nó cliente, irá ser determinante quanto ao nó servidor que vai fazer a atribuição dos IPs. Se o valor de utilizadores num determinado nó servidor for mais elevado do que o número de atribuições definido, será caso para pensar no reforço dessa área com outro nó servidor, de maneira a satisfazer todos os pedidos, sem quebra da qualidade desejável.

4. Configuração do ficheiro *wireless*

O ficheiro localizado em */etc/config/wireless* tem a seguinte configuração:

```
config wifi-device radio0
        option type mac80211
        option channel 6
        option phy phy0
        option hwmode 11g
        option country PT
```

```
        option txpower 20
        option disabled 0

config wifi-iface
        option device radio0
        option network wifi
        option ssid PR
        option mode adhoc
        option bssid 00:13:10:7c:da:71
        option encryption wep
        option key 3252732954797e433b4e342f23
```

Este ficheiro tem por função configurar o dispositivo rádio e a rede *wireless*. Como foram usados *routers Linksys WRT54GS* para fazer a rede *mesh*, esta configuração é adequada a esse tipo de equipamentos. Por exemplo, a "*option type*" é autodetetada. Neste caso é "*mac80211*" para o dispositivo rádio que incorpora o *router*. Poderia ser "*broadcom*" para"*brcm-2.4*" ou "*atheros*" para "*madwifi*". O protocolo BATMAN-ADV só funciona nas versões de *kernel Linux* 2.6 ou superiores. O *framework* "*mac80211"* é a nova pilha *wireless* que está incluída no *kernel* desde 2.6.22, embora os *drivers* estejam só incluídos desde 2.6.24. No caso do *firmware OpenWRT* compilado e usado na rede *mesh* deste trabalho, a versão do *kernel* é 3.3.38.

O canal escolhido para a rede *mesh* foi o canal 6 (2, 437 *GHz*), conforme se pode observar em "*option channel 6*", canal que não interferirá com o canal 1 ou 11, que poderão ser usados para os nós AP, que servem os utilizadores sem fios.

Em "*option phy phy0*", é definida a camada física do rádio. Normalmente o *OpenWRT* usa a opção "*macaddr*", referente ao MAC *address* para identificar o rádio, mas com o *framework* "*mac80211"* é possível identificar o rádio como "*phy0*". Esta opção é autodetetada e não deve ser mudada. Foi escolhido a norma IEEE 802.11g para a rede *mesh*, definida em "*option hwmode 11g*". O país onde vai operar é Portugal "PT", definido em "*option country PT*" e a potência máxima de transmissão é de 20 dbm (100 *mW*), definido em "*option txpower 20*". A *interface* que opera neste rádio é a *interface* "*wifi*" (*interface* BATMAN-ADV), definida previamente no ficheiro *network*. O modo de rede criado é Ad-Hoc, típico de redes *mesh*, definido em "*option mode adhoc*". O nome pelo qual a rede se faz anunciar, ou seja, o *Service Set IDentifier* (SSID) poderá ser um qualquer à escolha, neste caso foi escolhido "PR", definido em "*option ssid PR*". Quanto ao *Basic Service Set IDentification* (BSSID), foi selecionado um dos MAC *addresses* de um dos *routers* existentes na rede *mesh*. Por fim foi escolhido o sistema de encriptação *Wired*

Equivalent Privacy (WEP), cuja chave mais complexa de autenticação aceite terá de ser no formato hexadecimal de 128 bits.

Esta configuração do ficheiro *wireless* terá de ser igual para todos os nós, sejam servidores ou clientes. Como a rede funciona em modo Ad-Hoc e os nós comunicam no formato *peer to peer* (P2P), é criada a rede em função dos *Independent Basic Service Sets* (IBSSs), que por sua vez usam o MAC *address* selecionado (BSSID) pertencente a um dos nós existentes na rede, para que seja formada a rede *mesh*. Uma vez que o protocolo BATMAN-ADV é um protocolo de camada 2 OSI, apenas o MAC *address* é requerido.

Em matéria de segurança, é possível através de um programa normalmente chamado de *sniffer* (por exemplo *Wireshark* ou *tcpdump*), conseguir intercetar uma trama. Assim consegue-se facilmente obter o MAC *address* que forma a rede BATMAN-ADV, podendo-se configurar um *router* com estas mesmas configurações e passar a pertencer clandestinamente a uma rede *mesh*. A encriptação disponível no *firmware OpenWRT* para redes Ad-Hoc é a encriptação WEP. E a chave máxima de encriptação aceite é do comprimento de 128 bits no formato hexadecimal. Tendo em conta que ”quebrar” a chave poderá ser uma tarefa bastante rápida, usando uma rotina computacional de geração de chaves, e que por cada chave criada bastará reiniciar a rede através de um *restart network* aplicado ao *router*, o acesso à rede não poderá ser evitado, embora possa ser retardado. Uma solução é a alteração da chave de encriptação várias vezes por dia.

5. Configuração do ficheiro *system*

O ficheiro *system* encontra-se localizado em */etc/config/system* e tem a seguinte configuração:

```
config system
        option hostname OpenWrt
        option zonename 'Europe/Lisbon'
        option timezone 'WET0WEST,M3.5.0/1,M10.5.0'

config timeserver ntp
        list server     0.openwrt.pool.ntp.org
        list server     1.openwrt.pool.ntp.org
        list server     2.openwrt.pool.ntp.org
        list server     3.openwrt.pool.ntp.org
        option enable_server 0
```

Esta configuração apresentada é a configuração dos nós servidores. A importância deste ficheiro de configuração prende-se com a atualização da data e hora dos *routers* na rede

mesh. Os *routers* servidores atualizam automaticamente a sua data e hora quando têm acesso à *internet*, procurando essa atualização nos servidores de *Network Time Protocol* (NTP). No caso dos nós clientes é necessário indicar os endereços dos nós servidores/*gateways* para que também esses nós atualizem a data e hora. É feito da seguinte forma:

```
list server     10.0.0.1
list server     XX.XX:XX:XX
```

É preciso acrescentar os IPs dos nós servidores/*gateways* que os servem. Poderão ser acrescentadas tantas linhas quantos os nós servidores existentes na rede, sendo que se algum falhar a atualização, essa atualização será tentada noutro nó servidor. Esta função é importante, na medida em que para se fazer a gestão da rede é necessário que os nós estejam sincronizados e que forneçam a data e hora quando solicitado.

Por último é necessário editar o ficheiro dos nós servidores na linha 12, localizado em */etc/init.d/sysntpd*, e acrescentar a *flag* "-l":

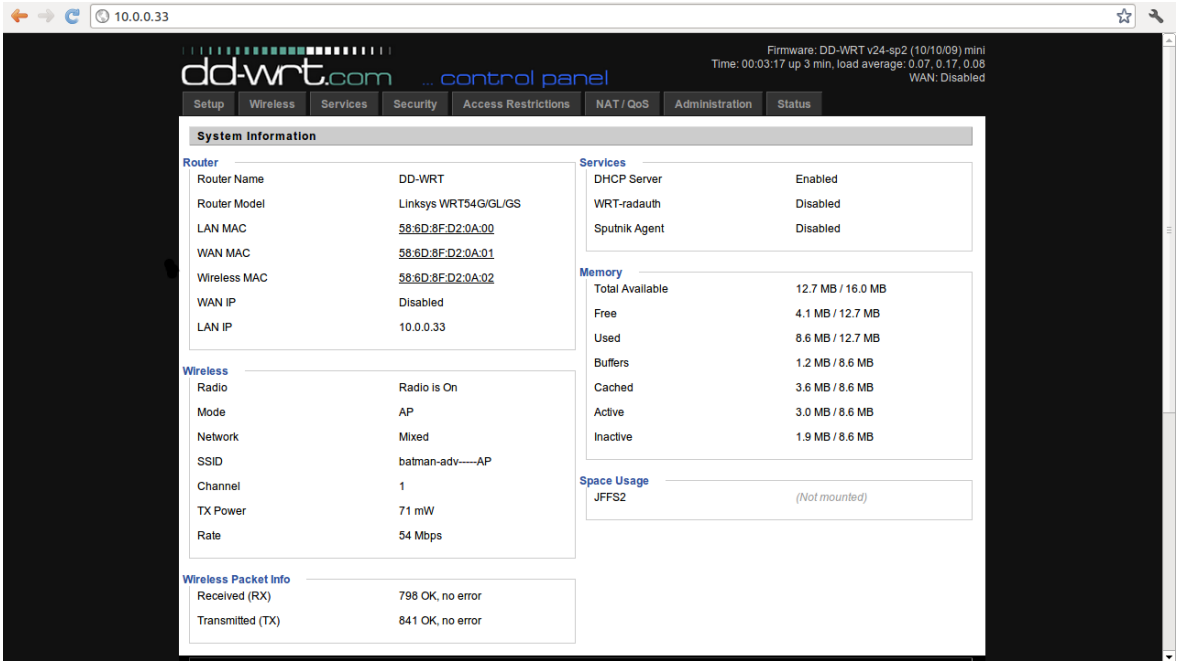
```
local args="-n -l"
```

Isto fará com que nos nós servidores seja ativada a função de fornecimento de hora e data, permitindo assim a atualização por parte dos nós clientes.

Anexo E. Configuração de APs pertencentes à rede BATMAN-ADV

Este anexo descreve os procedimentos a adotar para proceder à configuração dos APs, que se encontram agragados aos nós da rede BATMAN-ADV. Este método foi experimentado num *router Linksys WRT54GL* (equipamento usado como AP da rede *mesh*).

Para acesso sem fios, o nó BATMAN-ADV terá de ter acoplado um AP. Para o efeito foi usado um *router Linksys WRT54GL*, com 4 *MB* de memória *flash*, e com 16 *MB* de memória RAM, com um CPU de 200 *MHz*. O *firmware* utilizado foi o *firmware DD-WRT*, conforme se pode observar na figura seguinte.



É necessário fazer a ligação física do nó BATMAN-ADV ao AP, através de um cabo *Ethernet*, usando para isso as portas LAN de ambos os *routers*.

A configuração do AP faz-se da seguinte forma:

1. No menu *Wireless*, acede-se ao separador *Basic Settings* e altera-se:

Wireless Mode ---------- *AP*

Wireless Network Mode---------- *Mixed*

Wireless Network Name (SSID) ---------- *batman-adv-----AP*

Wireless Channel ---------- 1-2.412 *GHz*

Wireless SSID *Broadcast* ---------- *Enable*

O restante deixa-se ficar conforme está e faz-se "*Save*".

2. No menu *Setup*, acede-se ao separador *Basic Setup* e altera-se:

Connection Type ---------- *Disabled*

DHCP *Type* ---------- DHCP *Forwarder*

DHCP *Server* ---------- 10.0.0.3 (IP do nó BATMAN-ADV agregado)

Local IP *Address* ---------- 10.0.0.33 (IP escolhido para o AP)

Subnet Mask ---------- 255.255.255.0

Seleciona-se a *time zone* apropriada em *Time Settings* e faz-se "*Save*". O *router* reiniciará e arranca na sua nova função de AP, reencaminhando o sinal de e para a rede *mesh*, enviado pelos seus clientes via *wireless* [67].

O canal escolhido para os APs foi o canal 1 (2.412 *GHz*), de modo a não interferir com o canal 6 da rede *mesh*. Outro canal que não interfere com estes dois e pode ser usado é o canal 11 (2.462*GHz*). A encriptação usada para a *password* de acesso ao nó AP é WPA2 *Personal Mixed*, com o algoritmo *Temporal Key Integrity Protocol* (TKIP) em conjunção com *Advanced Encryption Standard* (AES).

Anexo F. Instalação e configuração de *SSMTP*

Este anexo descreve os procedimentos a adotar para a instalação e configuração do programa de envio de *e-mails* para o envio de notificações de alerta na utilização da ferramenta de gestão *batman-adv-admin*.

Em primeiro lugar é necessário criar uma conta no *Gmail*. Para este trabalho foi criado o *e-mail* batman.adv.mesh.alert@gmail.com.

Para se proceder à instalação do *SSMTP*, no servidor com sistema operativo *Linux – Ubuntu 11.04*, executou-se:

```
sudo apt-get install ssmtp
```

Após a instalação edita-se o ficheiro localizado em */etc/ssmtp/revaliases* da seguinte forma:

```
# sSMTP aliases
#
# Format:       local_account:outgoing_address:mailhub
#
# Example: root:your_login@your.domain:mailhub.your.domain[:port]
# where [:port] is an optional port number that defaults to 25.
root:batman.adv.mesh.alert@gmail.com:smtp.gmail.com:465
```

As linhas que começam com "#" não produzem efeitos. A última linha indica que vai ser utilizada a porta 465, para o envio do *e-mail*. Também poderia ser a porta 587, conforme explicado em [68]:

De seguida configura-se o ficheiro localizado em */etc/ssmtp/ssmtp.conf:*

```
# Config file for sSMTP sendmail

# The full hostname
hostname=localhost

# Where will the mail seem to come from?
rewriteDomain=gmail.com

# Username/Password
AuthUser=batman.adv.mesh.alert@gmail.com
AuthPass=XXXXXXXX
AuthMetod=plain

# Are users allowed to set their own From: address?
```

```
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
FromLineOverride=YES

# The place where the mail goes.
Mailhub=smtp.gmail.com:465

# Use SSL/TLS before starting negotiation
UseTLS=YES
```

Note-se que em "*AuthUser*=" é definido o *e-mail* criado no *Gmail* e em "*AuthPass*=" deve-se alterar "*XXXXXXXX*", pela *password* de acesso à conta *Gmail*.

Para que não existam problemas de permissões de acesso pelo *Apache* é necessário permitir acesso de leitura na diretoria */etc/ssmtp*:

```
sudo chmod 640 /etc/ssmtp
```

É necessário proceder à seguinte modificação no ficheiro de configuração do PHP, localizado em */etc/php/apache2/php.ini* (Para se configurar este ficheiro é necessária permissão de *root*):

```
; For Unix only.  You may supply arguments as well (default:
"sendmail -t -i").

; http://php.net/sendmail-path

sendmail_path = /usr/sbin/ssmtp -t
```

As linhas começadas por ";" são linhas comentadas que não produzem efeitos.

Após a execução na íntegra deste anexo é necessário reiniciar o servidor *Apache*:

```
sudo /etc/init.d/apache2 restart
```

Exemplo para envio de *e-mail*:

```
sudo ssmtp 1040891@isep.ipp.pt < meu-email.txt
```

Após a execução do comando será enviado para 1040891@isep.ipp.pt o conteúdo do ficheiro *meu-email.txt*.