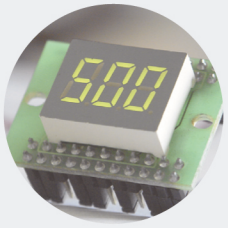
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






UAVProtocol: Sistema de encaminhamento entre UAVs

JOÃO PEDRO MESQUITA AZEVEDO
novembro de 2022

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

UAVProtocol: Sistema de encaminhamento entre UAVs

João Pedro Mesquita Azevedo

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Telecomunicações

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Novembro, 2022

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Telecomunicações.

Candidato: João Pedro Mesquita Azevedo, Nº 1111476,
1111476@isep.ipp.pt

Orientação Científica: Jorge Botelho Da Costa Mamede, jbm@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Novembro, 2022

À minha mãe.

Agradecimentos

Em primeiro lugar gostaria de deixar um agradecimento especial com carinho aos meus pais, irmãos e restante família, pela força que me deram durante todo este percurso.

Agradecer também aos amigos que estiveram presentes não só nas fases mais fáceis, mas também nas fases mais difíceis, sempre com um gesto animador, em especial ao João Gameiro, Tomás Cunha, Filipe Santos e Inês Pinto.

Deixar também um especial agradecimento à minha entidade patronal, a *Capgemini Engineering*, e a todos os colegas que me foram dando suporte ao longo de todo este processo como a equipa de *Communications* do projecto *Gen5* assim como aos colegas do departamento de V&V, em especial à Raquel Ribeiro, por toda a ajuda e compreensão.

A todos os profissionais do Instituto Superior de Engenharia do Porto, deixo os maiores agradecimentos pelos conhecimentos transmitidos durante todo o percurso não só na licenciatura bem como no Mestrado de Engenharia Eletrotécnica e de Computadores – ramo de Telecomunicações.

Quero também agradecer em especial ao meu orientador o engenheiro Jorge Mamede, pela disponibilidade, paciência e orientação durante a concretização deste projeto.

As recordações dos momentos passados no ISEP, serão para sempre lembradas como uma excelente parte da minha vida.

Resumo

A manutenção de veículos aéreos não tripulados, em inglês *Unmanned Aerial Vehicle* (UAV) e também conhecidos como *drones*, depende em grande parte do controlo remoto. O alcance da operação é limitado pelo sucesso das comunicações via rádio entre o controlador e o UAV. De forma a expandir esse alcance, podem ser utilizadas redes ou enxames de UAVs de modo a definir uma *mesh* com o objetivo de se poder encaminhar comandos de controlo para unidades mais distantes.

Devido aos obstáculos naturais e artificiais que existem no nosso meio ou então devido a interferências intencionais, a comunicação entre os *drones* é suscetível a interrupções e nesse sentido, é necessário criar soluções que consigam optimizar a comunicação evitando ao máximo as interrupções que possam ocorrer devido aos obstáculos de tal forma que é importante avaliar os diversos protocolos de encaminhamento sejam eles ao nível de *Internet Protocol* (IP) ou então mais baseados numa camada inferior, como o *Multiprotocol Label Switching* (MPLS), tirando mais partido das redes de operador de telecomunicações.

Ao longo desta tese é proposta uma especificação, desenvolvimento e teste de um protocolo de sobreposição para encaminhar instruções do controlador para comandar um UAV remoto numa base *multi-hop*, tendo em atenção as vantagens que oferecem os diferentes tipos de encaminhamento em cenários de conectividade de UAVs para trocas de volumes consideráveis de informação.

Palavras-Chave: UAV, Drones, Redes, Controlo, Alcance, Mesh, Multi-hop, Encaminhamento

Abstract

The maintenance of unmanned aerial vehicles, commonly called UAV, depends on the remote control. The scope of operation is limited by the success of radio communications between the controller and the UAV. In order to expand this range the UAVs networks or swarms can be used in order to define a mesh so that control commands can be routed to more distant units.

Due to the natural and artificial obstacles that exist in our environment or due to intentional interference, the communication between drones is susceptible to interruptions and in this sense, it is necessary to create solutions that can optimize the communication avoiding as much as possible the interruptions that may occur due to obstacles in such a way that it is important to evaluate the various routing protocols, be they at the IP level or more based on a lower layer, such as MPLS, taking more advantage of the telecommunications operator networks.

Throughout this thesis a specification, development and testing of an overlay protocol for routing controller instructions to command a remote UAV on a multi-hop basis is proposed, taking into account the advantages offered by different types of routing in UAV connectivity scenarios for exchanges of considerable volumes of information.

Keywords: Uav, Drones, Networks, Control, Extend, Mesh, Multi-Hop, Routing

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Lista de Acrónimos

AODV	*Ad hoc On-Demand Distance Vector*
AP	*Access Point*
API	*Application Programming Interface*
B2U	*Base-to-UAV*
BSD	*Bad Signal Detection*
CLI	*Command Line Interface*
d2d	*drone-to-drone*
DEM	*Digital Elevation Model*
DF	*Dronecode Foundation*
DHCP	*Dynamic Host Configuration Protocol*
DLL	*Data Link Layer*
DNS	*Domain Name System*
DV	*Distance Vector*
EP	*End Point*
FANETs	Flying Ad-Hoc Networks
FEC	*Forwarding Equivalence Class*
FFD	*Full Function Device*
GCS	*Ground Control Station*
GPR	*Ground Penetrating Radar*
GPS	*Global Positioning System*
GRE	*Generic Routing Encapsulation*
GUI	*Graphical User Interface*

ICMP *Internet Control Message Protocol*

IP *Internet Protocol*

IPv4 *Internet Protocol Version 4*

IPv6 *Internet Protocol Version 6*

ISM *Industrial, Scientific and Medical*

JSON *JavaScript Object Notation*

KML *Keyhole Markup Language*

LER *Label Edge Router*

LiDAR *Light Detection and Ranging*

LLC *Logical Link Control*

LS *Link State*

LSP *Label Switched Path*

LSR *Label Switching Router*

MAC *Medium Access Control*

MANET *Mobile Ad-Hoc Network*

MPLS *Multiprotocol Label Switching*

MPR *MultiPoint Relaying*

MSC *Message Sequence Chart*

OLSR *Optimized Link-State Routing*

OSI *Open System Interconnection*

P2P *Peer-to-peer*

PHY *Physical Layer*

ping *Packet InterNet Groper*

PTM *Point-to-multipoint*

PTP *Point-to-point*

QoS *Quality of Service*

RFC *Request for Comments*

RFD *Reduced Function Device*

RP *Relay Point*

RTT *Round Trip Time*

SOC *State of Charge*

SP *Start Point*

SSH *Secure Shell*

U2G *UAV-to-Ground*

U2U *UAV-to-UAV*

UAV *Unmanned Aerial Vehicle*

UAV-AP *UAV Access Point*

UAV-FAP *UAV with user Fairness-driven AP*

UDP *User Datagram Protocol*

VTOL *Vertical Take-Off and Landing*

WCT *Wireless Client Station*

WG *Working Group*

Wi-Fi *Wireless Fidelity*

WMN *Wireless Mesh Node*

WMN *Wireless Mesh Network*

XML-RPC *Extensible Markup Language - Remote Procedure Call*

Capítulo 1

Introdução

1.1 Contextualização

Os UAVs, também conhecidos como *drones*, são cada vez mais uma realidade nos dias que correm. A sua área de actividade é bastante extensa e tem vindo a emergir em diversos níveis, tanto a nível civil como a nível militar, incluindo escoltas militares, troca de mercadorias em algumas cidades, gestão do trânsito, fotografia aérea, segurança urbana e por aí adiante [1, 2, 3].

Devido aos obstáculos naturais que podem existir nos diferentes cenários descritos ou então, por interferências intencionais, a comunicação entre os UAVs é sensível a interrupções [4].

Nesse contexto, têm ocorrido nos últimos algumas catástrofes naturais que colocam em causa a segurança de pessoas e equipamentos. Um exemplo desses casos é o caso que ocorreu no ano de 2018 na gruta de Tham Luang, no norte da Tailândia, onde ficaram retidos 12 jovens e o seu treinador durante duas semanas a aguardarem resgate. No caso dessa situação, e em outras idênticas com condições de difícil acesso, é cada vez mais necessário garantir que toda a comunicação é feita sem falhas e também para realizar uma análise, através de fotografia ou *video streaming*, de como é o local de modo a facilitar a evacuação em caso de necessidade garantindo assim a segurança de todos os agentes envolvidos e nesse sentido, é necessário encontrar medidas para tornar a comunicação mais eficiente.

Também nos últimos anos os drones tem sido utilizados para eventos lúdicos tais como casamentos, concertos e eventos que reúnam uma grande quantidade de

pessoas.

Com a chegada das tecnologias robustas das redes *wireless*, os drones equipados com *transceivers* podem ser habilitados para comunicar com os nós terrestres assim como outros UAVs [5]. Deste modo é possível que um comando qualquer de uma estação possa chegar a unidades cada vez mais remotas.

1.2 Definição do Problema

Nos contextos apresentados, é claro que um dos problemas a resolver é do da cobertura de sinal de comunicações. Para esse efeito, existem redes/conjuntos de UAVs que alcançam grandes áreas de cobertura.

No entanto, as redes actualmente existentes baseiam-se essencialmente em métodos de encaminhamento IP, que a nível da *Quality of Service* (QoS) apresentada e tempo de processamento de rotas tem um custo elevado para certos casos. [6]. Nesse sentido, os protocolos de rede de operador, como é o caso do MPLS que apresentam uma solução baixo custo e com configuração *overhead* mínima [7], ajudam a alcançar uma melhor QoS e um melhor tempo de processamento de rotas, uma vez que o seu processamento é feito ao nível da camada 2 do modelo OSI [8].

Pretende-se com este trabalho ter uma visão geral de protocolos de encaminhamento IP e fazer uma análise comparativa com tecnologias de operador, como é o caso do MPLS, no serviço de *routing* outras tecnologias que possam fazer sentido na capacidade de aumentar a velocidade de transferência de dados entre os vários *hops* de uma rede multi-IP.

1.2.1 Objectivos

Este trabalho pretende ir ao encontro de todas as situações abordadas, na secção anterior, e tem como objetivos os seguintes pontos:

- Solucionar os problemas de falta de cobertura de sinal de telecomunicações em ambientes cujo o acesso é escasso, por exemplo montanhas, vales e grutas.
- Através de *video streaming* realizar recolhas de imagens de locais de díficíl acesso, auxiliando assim a evacuação dos mesmos em caso de necessidade.

1.3 Estrutura da Dissertação

No capítulo 1 é contextualizado o trabalho, bem como definido o problema que lhe deu origem e também definidos os objectivos para a sua concretização.

No capítulo 2 é descrito o estado da arte e são apresentados trabalhos relacionados, assim como a maneira como se organizam a nível de redes, explorar as diferentes áreas de aplicação dos UAVs, do *software* de controlo dos mesmos e também estudar

o seu comportamento nos diferentes protocolos de comunicação e a sua eficiência energética relacionada com o tempo de voo.

No capítulo 3 são apresentados os requisitos base do funcionamento do protocolo assim como a arquitectura geral dos equipamentos que o compõem. No capítulo 4 é apresentado o modelo geral do protocolo, definindo as várias fases de funcionamento e a sua implementação na solução final.

No capítulo 5 são apresentados os propósitos de teste do protocolo, assim como os equipamentos e ambiente utilizados para a sua execução, também são apresentadas as especificações dos testes de configuração do protocolo, conectividade e de performance. Por fim, são apresentados e analisados os resultados.

No capítulo 6 serão apresentadas as conclusões e discutidas as melhorias que poderão ser desenvolvidas e implementadas no futuro.

Capítulo 2

Estado da Arte

2.1 Redes UAV

Dependendo do contexto em que estão inseridos, os UAVs podem estar dispostos e ser controlados de diversas formas. Ao longo desta secção serão analisadas diferentes arquitecturas de redes UAV, o modo como funcionam e como os UAVs interagem entre si e em rede.

2.1.1 Controlo Directo

Neste tipo de arquitectura, cada UAV é controlado directamente pela sua estação controladora, também denominada de *Ground Control Station* (GCS) fazendo com que não exista qualquer tipo de interação inter-UAV. Sendo que cada comando é enviado directamente para cada UAV e sendo executado apenas pelo próprio UAV como se ilustra na figura 1.

Figura 1: A arquitectura de controlo directo

As GCSs podem ser implementadas de diferentes formas: podem ser implementadas através de uma antena Wi-Fi, que distribui o sinal de comando para os UAVs. Ou então, numa versão mais simples através de telecomando sendo que esta é opção a mais utilizada em captação de imagem e vídeo.

No entanto, como esta arquitectura é baseada num modelo de controlo mais centralizado possui algumas desvantagens tais como:

- Se existir um ponto de falha, o sistema fica comprometido;
- Nem todos os UAVs estão ligados à GCS simultaneamente [9].

2.1.2 Enxames de UAVs

Um enxame de UAVs, ou *swarm*, é capaz de formar redes expansíveis, para além das infraestruturas, que permitem o acesso dos nós terrestres. Ao beneficiar dos recursos de alta flexibilidade e de rápida provisão, um *swarm* de UAVs é uma solução viável para recuperar a comunicação de forma rápida e eficaz, especialmente em cenários onde os recursos de comunicação são escassos ou inexistentes, como nos ambientes pós-desastre [10].

Nesta arquitectura, as ligações *UAV-to-UAV* (U2U), também conhecidas como *drone-to-drone* (d2d), são necessárias para fazer a comunicação entre os UAVs existentes no enxame. Num enxame, existe uma hierarquia que é constituída por um líder e pelos seguidores. O líder é o UAV que recebe os comandos, e os seguidores são os que seguem a rota do líder e os comandos que lhe são passados por este.

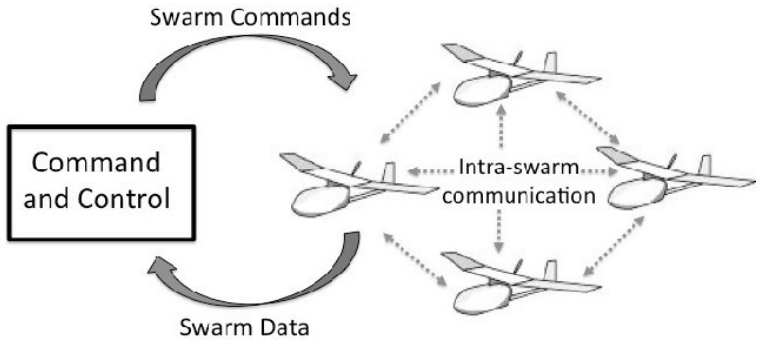


Figura 2: Diagrama de Controlo e Comando de um enxame UAV [11]

Esta topologia suporta não só a troca de mensagens de controlo entre os UAVs, de modo a prevenir colisões e a calcular rotas de voo, mas também a transmissão dos dados para serem acedidos por outros UAVs. Existem UAVs específicos dentro do enxame que estão equipados com interfaces para comunicar com as infraestruturas ou satélites, estes estabelecem as *gateways* entre o enxame e as outras redes.

Nos meios rurais ou nos meios de pós-desastre onde as infraestruturas de cobertura são escassas, os enxames UAV são formados como infraestruturas aéreas temporárias de acesso para os veículos de apoio a esses meios [10].

No exemplo citado no artigo [12], é estudada a possibilidade de um enxame de UAVs, através do algoritmo proposto, contar com vários líderes mas também, ao longo do tempo haver a possibilidade de aumentar o número de seguidores, a fim de reduzir o consumo de comunicação e melhorar a adaptabilidade do sistema e também a sua expansibilidade.

2.1.3 Redes *Mesh*

Uma rede *mesh* sem fio consiste em nós de rádio organizados numa topologia *mesh* ou malha. Estas redes são normalmente compostas por *routers* e *gateways mesh* que distribuem o sinal de *Wi-Fi* igualmente por todos os pontos da rede.

Nestes cenários de rede, os *routers* comunicam entre si e enviam sinal *Wi-Fi* reciprocamente, bem como, para a área envolvente. Por exemplo, se um nó A quiser comunicar com o nó C irá utilizar o nó B que fica entre ambos para fornecer melhor possibilidade de comunicação, conforme é ilustrado na figura 3.

À transmissão de informação entre os nós passando por meio de um nó intermediário (nó B), dá-se o nome de "salto"ou "*hop*"da rede mesh. Cada um destes *hops* introduz um nível de atraso na rede por isso, é necessário que o número de saltos seja minimizado encontrando por exemplo o caminho mais direto entre os dois nós. No entanto, quando um dos *routers* da rede *mesh* fica *offline* ou inutilizado o sinal consegue encontrar outros caminhos graças aos nós alternativos [13].

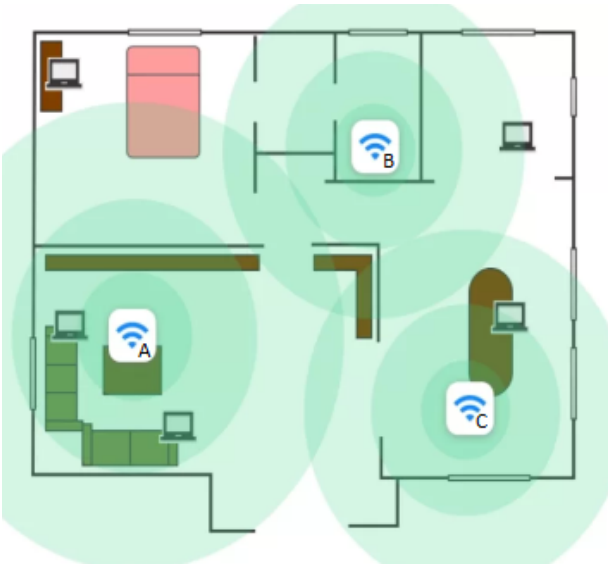


Figura 3: Exemplo rede *mesh* com 3 nós *wireless* [13]

Nesta configuração de rede mesh encontra-se o projeto UAVNet referenciado no artigo [5], que é uma *framework* baseada numa altamente adaptável *Wireless Mesh Network* (WMN), sendo que o principal objectivo da projecto é fornecer a possibilidade de implantar uma rede de comunicação completa em cenários de emergência e recuperação de desastres de uma forma fácil e rápida. O conceito é baseado em pequenos UAVs que são anexados a *Wireless Mesh Node* (WMN). Os WMNs estão directamente ligados à electrónica de voo dos UAVs e controlam a implementação autónoma da rede. Toda a rede pode ser configurada, implantada e monitorizada por um único utilizador, usando uma aplicação de controlo remoto de fácil utilização que funciona num *iPad* ou *iPhone*. Esta aplicação de controlo remoto monitoriza toda a UAVNet e mostra todos os participantes envolvidos, tais como UAVs e clientes, num mapa interactivo.

A figura 4 mostra um cenário típico do UAVNet. Dois computadores portáteis usam dois UAVs voadores para comunicar entre si. As setas tracejadas indicam ligações sem fios e as setas sólidas representam ligações em série.

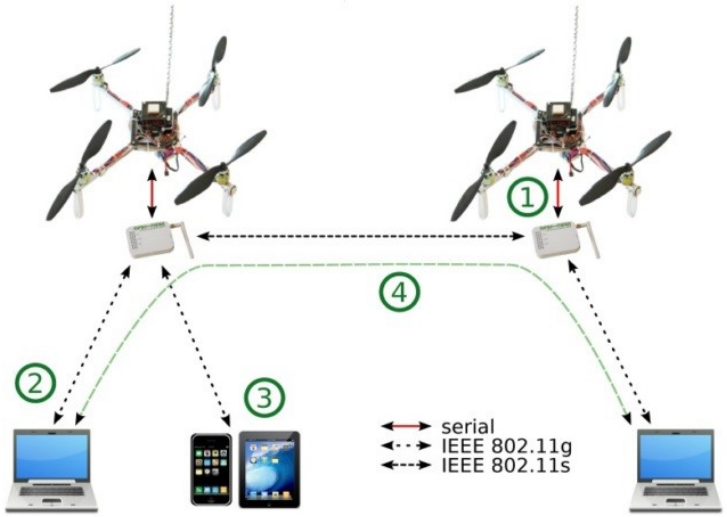


Figura 4: Exemplo de aplicação do *UAVNet*
[5]

2.1.4 Redes Ad-Hoc

A tecnologia *ad-hoc* permite a criação de redes de dispositivos móveis em áreas onde não existem infraestruturas de comunicações. Neste tipo de ligação, não existe um nó central para onde todas as informações são enviadas pelos outros nós, fazendo com que não seja necessária a existência de um *router* que faça a comunicação da rede com outros dispositivos externos. As ligações entre nós são independentes entre si, de maneira que, se uma falhar, seja por perda de conexão ou falha de algum dispositivo, todas as outras ligações continuam a funcionar [14].

Como as redes *ad-hoc* funcionam sem nenhuma infraestrutura é necessário que existam mecanismos de descoberta de rotas e encaminhamento de informação. Para isso, estas redes recorrem a protocolos de rotas de dois tipos: os reativos e os pró-ativos.

Dentro dos protocolos reactivos, que são protocolos que não tomam iniciativa de manter as rotas e apenas as determinam, quando necessário, fazendo *flooding* de pedidos na rede. Este protocolo tem como vantagem o facto de usar recursos apenas quando é necessário, no entanto, inundam a rede com pedidos e introduzem atraso no inicio de envio do tráfego pois tem de ser primeiramente determinada a rota. Neste tipo de protocolos inclui-se o *Ad hoc On-Demand Distance Vector* (AODV) definido pela *Request for Comments* (RFC) *3561*, onde um dos nós envia um *route request* a outro de forma a efectuar comunicação com este e recebe desse outro nó um *route reply* indicando a rota mais próxima.

No caso dos protocolos pró-activos, as rotas são construídas utilizando tráfego de controlo contínuo e mantidas todas as rotas. Este processo tem como vantagem ter as rotas sempre disponíveis e mantém o tráfego de controlo constante. Um exemplo deste tipo de protocolo é o *Optimized Link-State Routing* (OLSR) definido pela norma *RFC 3626*. Este protocolo implementa um sistema de deteção de conectividade a nós vizinhos, através de mensagens *HELLO* que efectuam os pedidos de encaminhamento de forma optimizada através do mecanismo de *MultiPoint Relaying* (MPR). Para além destes dois aspectos, este protocolo envia mensagens de estado das ligações e cálculo de rotas [15].

Em relação aos drones existem estudos sobre as Flying Ad-Hoc Networks (FANETs), que são redes *ad-hoc* constituídas por UAVs, como se referem os autores do artigo [16]. Um dos exemplos de rede ad-hoc com equipamentos UAV é o que está referenciado no artigo [17] que fala de dois cenários de funcionamento, no primeiro o UAV atua como um nó de rádio proeminente que efectua a conexão dos pontos de rádio terrestres desactivados. No segundo cenário, a rede permite que os grupos de UAVs comuniquem entre si para estender o alcance operacional dos UAVs mais pequenos.

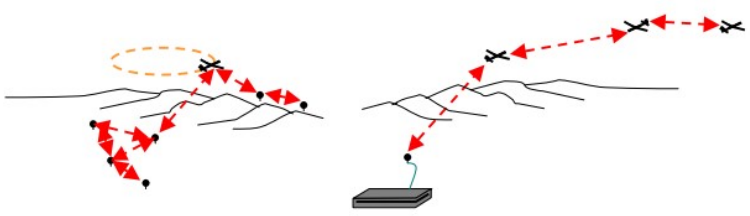


Figura 5: Figuras ilustrativas do cenário 1 (esquerda) e do cenário 2 (direita)
[15]

2.2 Áreas de aplicação das redes UAV

Tal como referido na secção 1.1, os UAV e as suas redes tem várias utilidades tais como: escoltas militares, troca de mercadorias em algumas cidades, gestão do trânsito, fotografia aérea e segurança urbana. No entanto existem outros serviços e aplicações em que estas podem operar.

2.2.1 Retransmissão de Comunicação

Os UAVs podem funcionar como nós de retransmissão que conectam *clusters* desconectados da *Mobile Ad-Hoc Network* (MANET). Nesse caso, os nós pertencentes a diferentes *clusters* podem comunicar entre si utilizando um UAV, que pode ser colocado numa posição estratégica entre os dois. Por essa questão, um ou mais UAVs podem fornecer essa função em grandes MANETs, proporcionando eficiência de comunicação e flexibilidade. Adoptando essa estratégia, pode estender-se uma MANET numa área geográfica muito grande, onde os nós podem ter que ser agrupados em diferentes regiões devido aos requisitos de topologia de terrenos, localização de nós e mobilidade impostos pelo aplicação [9].

2.2.2 Gateways de Rede

Em áreas geográficas remotas ou áreas atingidas por desastres, um ou mais UAVs podem fornecer conectividade a redes de *backbone*, infraestrutura de comunicação ou acesso à Internet agindo como nós de *gateway*. Essa função pode desempenhar um papel essencial para restaurar a cobertura de telefone, *internet* ou satélite desesperadamente necessária nesses locais. Essa conectividade pode apoiar esforços de busca e salvamento de vidas e reconstrução. Os UAVs podem ser implantados de maneira rápida e eficiente para executar essa tarefa de maneira muito dinâmica e económica [9].

2.2.3 UAV-assisted Sensing

Várias aplicações requerem vários UAVs de colaboração para efectuar a detecção de uma área ou inspeccionar uma infraestrutura utilizando um ou vários tipos de sensores como câmaras fotográficas, sensores de temperatura, leitores de radiação e monitores de gás. Essas aplicações exigem uma eficiência grande de comunicação de modo a permitir uma melhor detecção entre os vários UAVs. Alguns dos UAVs podem individualmente lidar com algumas das tarefas de detecção. Uma solução eficiente para este problema passa por usar vários UAVs juntos na organização das operações e fazer uma colheita colectiva de informações mais precisas e confiáveis [9]

2.2.4 UAV-assisted Acting

Alguns casos, como fins agrícolas e militares, exigem dispositivos actuadores, como os UAVs. Nesse tipo de aplicações, vários UAVs podem colaborar entre si para realizar as tarefas necessárias. No caso da agricultura, vários UAVs trabalham juntos para realizar a pulverização de grandes campos com pesticidas ou fazer distribuição rápida de sementes em grandes áreas [9, 18, 19].

2.2.5 UAV-based data storage

Apesar de algumas aplicações de UAVs enviarem os dados recolhidos directamente para a estação base, outros podem necessitar que os UAVs armazenem a recolha de dados por diversos motivos. O primeiro é que os dados recolhidos necessitam de grande largura de banda de comunicação e os dados podem não estar sempre disponíveis para a transferência dos UAVs para a estação base. A segunda razão é que não existe a necessidade de enviar os dados recolhidos imediatamente para a estação logo após a recolha, pois a informação será usada e processada após a recolha.

Os UAVs podem ser homogéneos ou heterogéneos em termos de capacidade de armazenamento e capacidade de recolha de dados. Os UAVs podem recolher quantidades iguais ou diferentes de dados, dependendo claro do tipo de aplicação [9].

2.2.6 UAV-based data processing

Os UAVs podem ser equipados com unidades de processamento de última geração que podem ser utilizados por aplicações que precisem de processamento de alto desempenho, como processamento de imagem de alta resolução, processamento de vídeo, reconhecimento de padrões, fluxo de dados e planeamento de tarefas online. Uma tarefa de processamento de dados de alto desempenho pode ser obtida utilizando uma unidade ou várias unidades computorizadas num UAV ou em vários UAVs. Neste último caso, precisa de ser efetivamente utilizada a abordagem de processamento distribuído pelos processadores disponíveis. Isto é geralmente muito importante se os UAVs estiverem a operar em zonas distantes das estações base e quando os resultados forem necessários no momento para acionar uma opção adequada. Por exemplo, num campo de batalha, um UAV pode precisar de identificar uma unidade inimiga que esteja próxima de algumas das suas unidades. Nesse caso o processamento de imagens e o reconhecimento de padrões são necessários para localizar o inimigo e tentar destruí-lo [9].

2.3 Software de Controlo do UAV

Nesta secção pretende-se ter uma visão ampla relativa a *software* de controlo, que atualmente podem ser encontrados no mercado.

2.3.1 ArduPilot

O software de controlo *ArduPilot* é uma tecnologia que visa permitir a criação e o uso de sistemas de veículos não tripulados confiáveis e autónomos para o benefício pacífico de todos. Fornece um conjunto abrangente de ferramentas adequadas para quase todos os veículos e aplicações. Como um projecto de *open source*, está em constante evolução com base no rápido *feedback* de uma grande comunidade de utilizadores.

Embora o *ArduPilot* não fabrique qualquer *hardware*, o seu *firmware* funciona numa grande variedade de diferentes hardware para controlar UAVs de todos os tipos. Juntamente com o *software* de controlo terrestre, os UAVs que executam o *ArduPilot* podem ter funcionalidades avançadas, incluindo a comunicação em tempo real com os operadores.

O *Mission Planner*, ilustrado na figura 6, é um software para a GCS completo e suportado pelo *ArduPilot* que permite aos utilizadores configurar, configurar, testar, e afinar o UAV. Oferece interacção *point-and-click* com o seu *hardware*, scripts personalizados, e simulação [20].

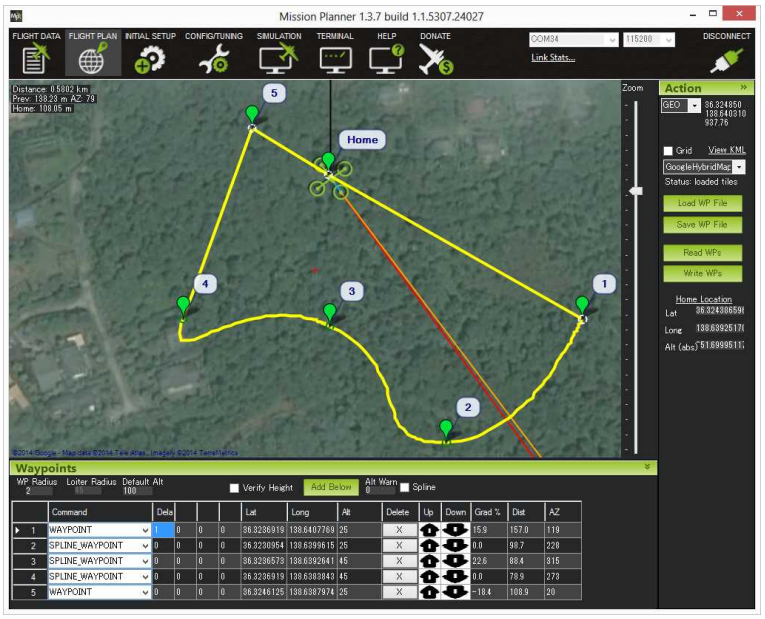


Figura 6: *Screenshot* do *Mission Planner* a controlar uma missão com o *ArduPilot* [20]

2.3.2 Dronecode

O Dronecode é uma *stack* de soluções de *hardware* e *software* desenvolvolvida pela *Dronecode Foundation* (DF). ADF acolhe projectos de *open-source* e de *open-standard* críticos para a industria dos *drones*. A DF é uma fundação neutra em termos de fornecedores para projectos *open drone* como parte da *Linux Foundation* [21].

Debaixo da alçada da DF estão diversos projectos, que fazem parte da *stack* completa de controlo de um *drone*, tais como:

- ***PX4 Autopilot*** - fornece orientação, navegação, e algoritmos de controlo para UAVs, sejam eles do tipo *fixed wing*, *multirotor* ou *Vertical Take-Off and Landing* (VTOL), juntamente com estimadores de atitude e posição [21].
- ***MAVLink*** - é um protocolo de mensagens *lightweight* para comunicar com *drones* mas também com componentes *onboard* do próprio *drone*. Segue um padrão híbrido *publish-subscribe* e um *design pattern point-to-point*: os fluxos de dados são enviados/publicados como tópicos enquanto os sub-protocolos de configuração como o protocolo de missão ou protocolo de parâmetros são ponto-a-ponto com retransmissão [22].
- ***MAVSDK*** - é uma colecção de bibliotecas para várias linguagens de programação para interagir com os sistemas *MAVLink*, tais como drones, câmaras ou sistemas terrestres. Esta coleção fornece uma API simples para gerir um ou mais veículos, fornecendo acesso programável à informação e telemetria de veículos, e controlo sobre missões, movimentos e outras operações [23].
- ***QGroundControl*** - fornece controlo total de voo e configuração do veículo para veículos com *PX4* ou *ArduPilot*, ilustrado na figura 7, permitindo a utilização fácil e directa para principiantes, ao mesmo tempo que oferece suporte a funcionalidades de topo de gama para utilizadores experientes entre as suas funcionalidades estão:
  - Planeamento de missão para voo autónomo.
  - Visualização do mapa de voo mostrando a posição do veículo, pista de voo, pontos de passagem e instrumentos do veículo.
  - *Streaming* de vídeo com sobreposições da exibição dos instrumentos.
  - Suporte para a gestão de múltiplos veículos.
  - funciona em plataformas *Windows*, *OS X*, *Linux*, *iOS* e dispositivos *Android* [24].

.

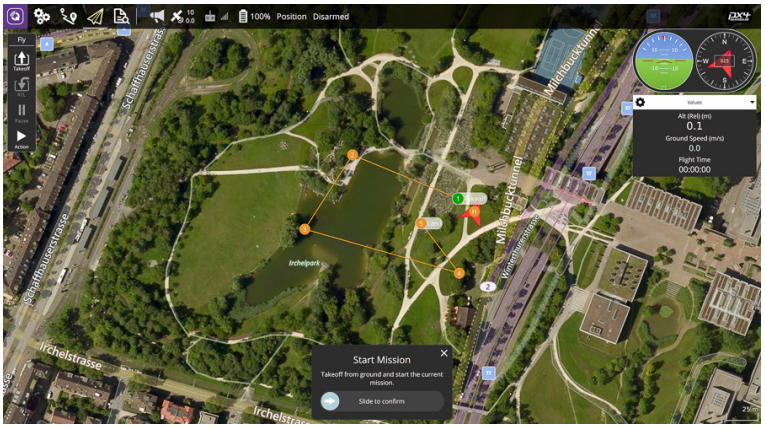


Figura 7: *Screenshot* do *QGroundControl* a controlar uma missão [24]

2.3.3 UgCS

O sofware de controlo de UAVs *UgCS* é uma solução *non-open source* apresenta uma solução para sistemas integrados baseados em drones com sensores para recolha de dados nas áreas da geofísica, hidrográfica, arqueológica e industrial, para além da monitorização ambiental [25]. Suporta as mais populares plataformas de UAV incluindo *DJI M300, M600, M2X0*, *Inspire*, *Phantom series*, *Mavic series* e também os *drone* compatíveis *MAVLink* como o *Pixhawk with ArduPilot/PX4* [26].

Tem como principais vantagens:

- **Poupança de tempo** - não há necessidade de usar múltiplos programas, importa e cria rotas a partir de dados *Keyhole Markup Language* (KML) fornecidos pelo cliente e poupa mais de 50% do tempo.
- **Aumentar a produtividade da recolha de dados** - planeia e executa missões mesmo sem ligação à *internet* mesmo numa àrea deserta com a capacidade de armazenamento em cache dos mapas *offline* e também tem a capacidade de executar rotas longas, retomando o voo a partir de um determinado ponto de passagem após troca da bateria.
- **Maior segurança de voo** - utiliza dados *Digital Elevation Model* (DEM) pré-instalados e para aumentar a precisão e a segurança para missões com o terreno a seguir e ajusta o alcance de voo permitido e zonas *No-Fly* para voar de acordo com os requisitos regulamentares.

A figura 8 ilustra o software no seu modo de *Telemetry Window* que apresenta dados de telemetria incluindo o nível de carga da bateria, a ligação de rádio e a qualidade do sinal *Global Positioning System* (GPS), a rota e o rumo actual, a velocidade, a altitude e muito mais.

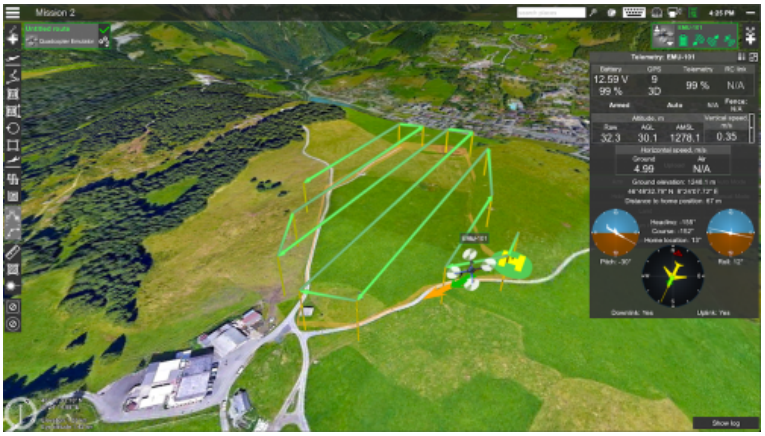


Figura 8: *Screenshot* da *Telemetry Window* do *UgCS*[26]

2.4 Protocolos de Comunicação

Nesta secção são descritos protocolos de comunicação estudados existentes e são encontrados em algumas aplicações com UAVs.

2.4.1 ZigBee

O protocolo *ZigBee* [27, 28, 29] é desenvolvido pela *ZigBee Alliance* composto por centenas de empresas, entras quais estão a *Ember*, *Freescale*, *Chipcon*, *Invensys* e *Mitsubishi*. O termo deriva da dança criada pelas abelhas em forma de *zig-zag* para permitir a troca de informação sobre a localização do pólen. É uma tecnologia de comunicação *wireless* com pequeno volume, baixo consumo energético e baixa taxa de transmissão.

O *ZigBee* é baseado na norma *IEEE802.15.4* no entanto as duas não são a mesma coisa. O protocolo *ZigBee* (*software*) é responsável pelas *layers* de aplicação, de rede e de segurança enquanto que o *IEEE802.15.4* (*hardware*) define as camadas de *media access control* e física.

O protocolo trabalha nas bandas de frequência *Industrial, Scientific and Medical* (ISM) de 868 MHz (Europa), 915 MHz (América do Norte e Austrália) e 2,4GHz (disponível mundialmente) com data rates até 20kbps, 40kbps e 250kbps respectivamente.

Uma rede de sensores *ZigBee* pode adoptar vários tipos de configuração de rede, mas cada um deles deve conter um nó coordenador (*gateway*) e um nó terminal. Numa rede *ZigBee* um dispositivo pode ser classificado em funções distintas: coordenador, nó terminal ou *router*. O dispositivo coordenador é um tipo especial de *Full Function Device* (FFD) que é utilizado para conseguir muitos dos serviços *ZigBee*. O nó terminal pode ser tanto um FFD como um *Reduced Function Device* (RFD). O *router* é um equipamento opcional que pode ser necessário em alguns casos de configurações de rede especiais. A figura 9 mostra um exemplo de uma configuração de rede *ZigBee*.

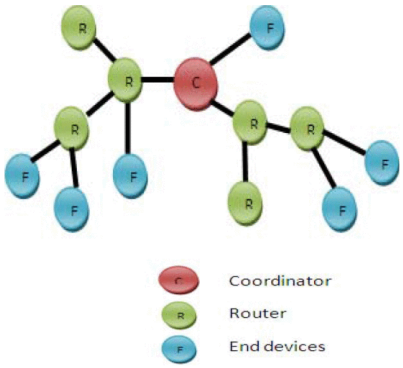


Figura 9: Um exemplo de uma rede *ZigBee* [27]

O *ZigBee* pode ser utilizado para uma série de aplicações e configurações de rede como foi abordado acima. É possível a sua utilização em contexto de redes de UAVs como poderemos ver a seguir.

No exemplo, apresentado no artigo [30], é proposto um sistema de gestão de UAV para uma *smart city* que são apresentados um conjunto de soluções técnicas e resultados de uma simulação, para suportar o conceito de gestão de um pequeno grupo de UAVs são apresentadas. O focos principais deste projecto são o *collision avoidance* e comunicação U2U e *UAV-to-Ground* (U2G). O conceito proposto implica um grupo de cinco UAVs a comunicarem uns entre si utilizando a tecnologia *ZigBee*.

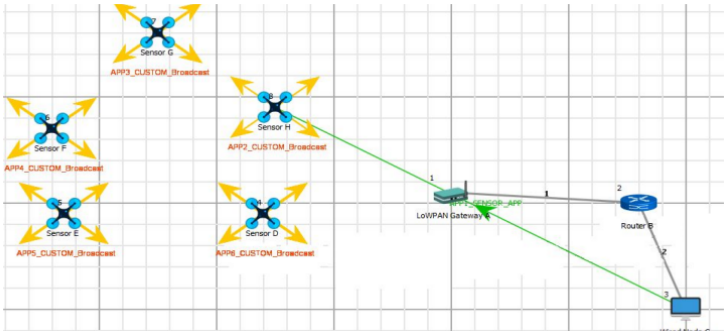


Figura 10: Conceito de utilização de drones e durante uma simulação da tecnologia *ZigBee*. [30]

No exemplo, apresentado no artigo [31], é proposto um método de patrulha de segurança usando o UAV para resolver os problemas causados pelos métodos tradicionais de patrulha, tais como: baixa frequência de patrulha, baixa eficiência de patrulha e alto custo de patrulha de segurança. Neste caso rede *ZigBee* foi utilizada para localizar e melhorar a precisão do posicionamento do UAV. A tecnologia de posicionamento *ZigBee* é complementada com base na tecnologia tradicional de posicionamento GPS e depois a informação de posicionamento é recolhida e enviada para um filtro *Kalman* para a operação de *denoising*.

2.4.2 Wi-Fi

A tecnologia Wi-Fi, que é o termo utilizado para designar a família de protocolos IEEE 802.11 [32], é a tecnologia dominante do tráfego de redes locais sem fios, nos nossos dias. Desde a sua primeira implementação em 1997 foram feitas revisões à norma original para optimizar a produção ou especificar elementos para garantir uma melhor segurança ou interoperabilidade. As revisões mais conhecidas são *802.11a, 802.11b, 802.11g, 802.11n, 802.11ac, 802.11ad* e *802.11ax* [33, 32].

A gama de frequência utilizada pelos dispositivos pode variar entre os 2,4GHz, 5GHz, 6GHz e 60GHz isto no caso do mais recente *802.11ad* [33, 32, 34, 35]. Ao nível de *data rate* os valores das diferentes estão representados na tabela 1.

Tabela 1: Frequências e *data rates* suportados pelas diferentes revisões [33, 36, 37, 38, 39]

Norma	**802.11a**	802.11b	802.11g	802.11n	**802.11ac**	**802.11ax**	**802.11ad**
Frequência (GHz)	5	2.4	2.4	2.4/5	5	2.4/5	60
Data Rate (Mbps)	54	11	54	150	433-6.933	9.6 Gbps	385-2502

A norma IEEE 802.11 define as duas camadas mais baixas da rede: *Physical Layer* (PHY) e *Data Link Layer* (DLL). Esta última é também subdividida em duas *sublayers*: a *Logical Link Control* (LLC) e a *Medium Access Control* (MAC). A figura 11 ilustra a arquitectura proposta pelo *Working Group* (WG) do 802.11 em relação ao modelo OSI [40].

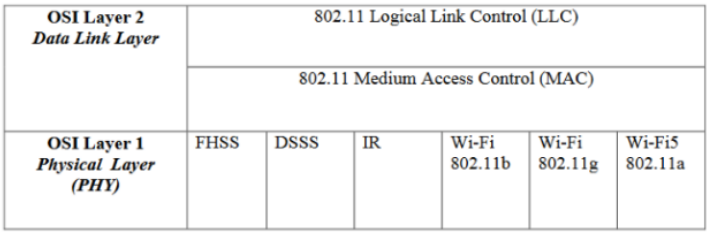


Figura 11: As *layers* definidas pelo Wi-Fi no modelo OSI [40]

A tecnologia Wi-Fi tem dois tipos de componentes: *Wireless Client Station* (WCT) e AP. A WCT é um *end device*, como por exemplo uma estação fixa ou móvel que tenha uma placa de rede *wireless*. O WCT tem também a possibilidade de funcionar no modo de *ad-hoc* permitindo ligar-se a outros WCTs de forma a constituir uma rede *Peer-to-peer* (P2P) ou *Point-to-point* (PTP) e ainda *Point-to-multipoint* (PTM) ou seja, cada máquina pode desempenhar, ao mesmo tempo, o papel de WCT e o papel de AP [40]. Exemplo de uma rede *ad-hoc* na figura 12.

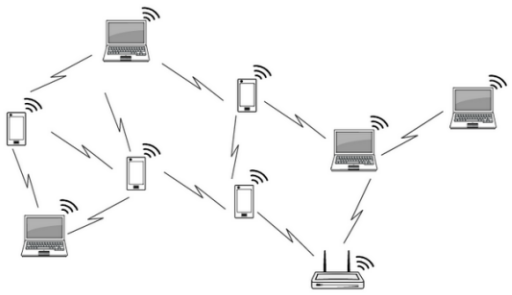


Figura 12: Exemplo de uma topologia *ad hoc* [40]

Um AP funciona como uma *bridge* entre a rede fixa e a rede *wireless*. Ela organiza e concede acesso a partir de várias WCTs para a rede fixa [41]. Este modo permite

às WCTs ligarem-se a uma rede fixa (normalmente *Ethernet*). Permite a ligação entre WCTs através de um AP comum. Um exemplo é dado na figura 13 [41, 40].

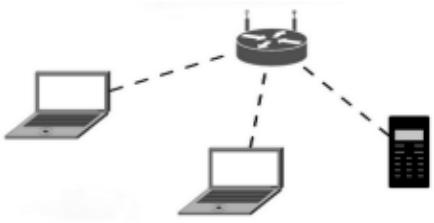


Figura 13: Exemplo de uma topologia de AP [40]

Assim como o protocolo *ZigBee* também o protocolo Wi-Fi pode ser aplicado nas mais diversas áreas e configurações de rede. Nos exemplos abaixo são mostrados algumas dessas aplicações.

No projecto apresentado no artigo [42] pretende-se implementar uma nova arquitectura de rede Wi-Fi Direct suportada por UAVs. A tecnologia Wi-Fi Direct introduz alguns melhoramentos como a poupança de energia e descoberta dinâmica de serviços que tem potencial para trazer vários benefícios. No projeto apresentado é utilizado um algoritmo simples mas eficiente para colocar os UAVs na rede de forma optimizada para melhorar a *performance* geral da rede. É também apresentado um caso interessante de colocação de UAVs em regiões de mobilidade proibida, onde o movimento do UAV é restrito a uma dimensão, seguindo em linha recta.

No exemplo [43] é proposta a solução designada por *UAV with user Fairness-driven AP* (UAV-FAP) para levar o *UAV Access Point* (UAV-AP) a locais que podem oferecer a melhor *performarce max-min* do valor de *throughput*. A solução UAV-FAP reduz hierarquicamente a área de cobertura/serviço que contém localização com *throughput* máximo e mínimo, movendo o UAV-AP para sondar e encontrar áreas de desinteresse a nível de serviço com base numa estimativa do tempo de execução.

2.5 Protocolos de Encaminhamento

O processo de encaminhamento e construção de rotas é importante numa rede de informação. É através deste processo que é transferida a informação que utilizamos no dia-a-dia. Nesse sentido, é importante perceber-mos como é distribuída e encaminhada essa informação como pode ser feita a um nível mais alto de rede, como é o caso do IP ou então, ao nível do operador como é o caso do MPLS.

2.5.1 IP

O encaminhamento de pacotes *Internet Protocol*, numa rede IP, é o conjunto de tarefas necessárias para mover um pacote IP de *router* para *router* até ao seu destino especificado no cabeçalho IP. O endereçamento IP existe em em duas versões: IPv4 e IPv6. A principal diferença entre IPv4 e IPv6 está nos seus formatos de endereçamento, ilustrado na figura 14. O IPv4 usa endereços de 32 *bits* (4 *bytes*) para identificar de forma única os nós dentro da Internet global. O IPv6 usa endereços de 128 *bits* (16 *bytes*) para identificar os nós de forma única dentro da *Internet* global. Com o IPv6 grande espaço de endereçamento, é claramente capaz de resolver o problema de esgotamento de endereços no IPv4 [44].

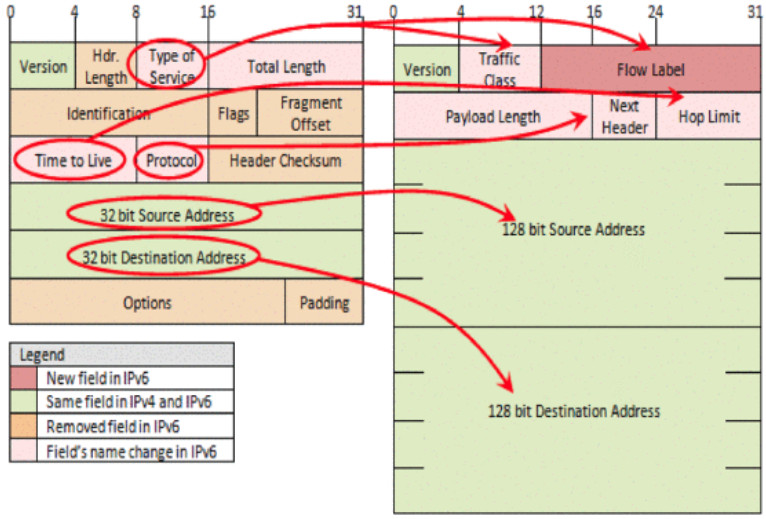


Figura 14: Similaridade, diferenças e relação entre os campos de cabeçalho IPv4 e IPv6 [44]

As rotas podem ser estáticas ou dinâmicas.

2.5.1.1 Rotas estáticas

As rotas estáticas são definidas manualmente num determinado nó da rede ao adicionar o endereço de rede e o próximo *hop* para chegar lá.

A partir desse ponto os próximos *hops* são definidos pelos próximos equipamentos pertencentes à rede, ou seja, as rotas estáticas registadas nas tabelas de *routing* são removidas se o próximo *hop* na rede for desligado.

Como é óbvio as rotas estáticas não possuem uma grande escalabilidade. Assim que a rede começa a crescer a tarefa de manter as rotas estáticas torna-se cada vez mais e mais complicadas [45].

2.5.1.2 Rotas dinâmicas

Os protocolos de encaminhamento dinâmico permitem que cada *router* descubra um ou mais caminhos para cada destino dentro da rede. Quando a topologia de rede é alterada, como quando são adicionados novos caminhos ou quando esses caminhos ficam fora de serviço. Nesses casos, os protocolos de encaminhamento dinâmico ajustam automaticamente o conteúdo da tabela de *routing* de modo a reflectir a nova topologia de rede.

Os protocolos de *routing* usados hoje em dia são baseados num de dois algoritmos: *Distance Vector* (DV) ou *Link State* (LS). Os algoritmos DV fazem o *broadcast* da informação de *routing* para todos os *routers* vizinhos. Quando um *router* recebe uma rota de um vizinho que não consta da sua tabela, esta rota é adicionada se essa rota já existir o *router* manterá a rota que tenha um menor caminho associado. Os algoritmos de LS operam num paradigma diferente. Primeiro, cada *router* constrói o seu próprio mapa topológico da rede completa baseado nos *updates* dos vizinhos. Depois. cada *router* usa o algoritmo de *Dijkstra* para computação o caminho mais curo para cada destino do mapa topológico [45].

2.5.2 MPLS

As redes *Multiprotocol Label Switching* (MPLS) são redes de operador que operam um nível abaixo da camada de IP e que podem melhorar a utilização da rede uma vez que possui um menor *overhead*, minimizando a latência e melhorando a QoS. O MPLS é escalável, orientado à conexão, e independente de qualquer tecnologia de transporte de pacotes. O MPLS melhora o encaminhamento de pacotes numa rede e supera as desvantagens do encaminhamento IP [46].

O MPLS é uma boa solução para reduzir a sobrecarga de reencaminhamento do endereço IP correspondente. De acordo com a estratégia tradicional de reencaminhamento MPLS, todos os pacotes de dados com o mesmo endereço de destino são agrupados no mesmo FEC.

Quando FEC chega ao *Label Edge Router* (LER), uma nova *label* (20 *bits*) será atribuída. E o pacote MPLS, com o cabeçalho ilustrado na figura 15, será mapeado para um determinado *Label Switched Path* (LSP). Este LER de entrada na rede MPLS é denominado de *ingress*-LER [47].

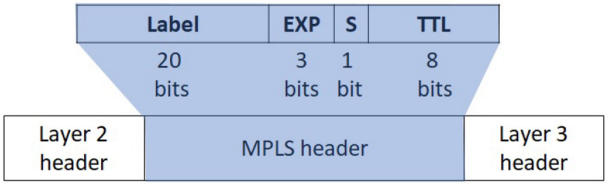


Figura 15: Cabeçalho MPLS[46]

Ao longo do LSP o pacote MPLS será encaminhado por *Label Switching Router*s (LSRs) que irão realizar as trocas de *labels* ao longo da rede MPLS. Posteriormente, o pacote será encaminhado para o LER do final da rede, denominado de *egress*-LER, que irá remover a *label* do pacote e encaminhá-lo para o nó de destino, todo o processo é ilustrado na figura 16 [47].

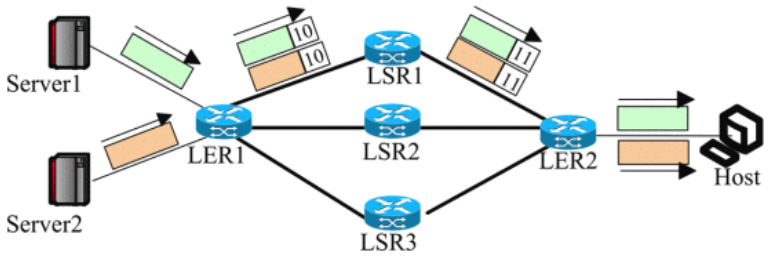


Figura 16: Encaminhamento da FEC através de uma rede MPLS [47]

2.6 Tempo de Voo e Consumo energético

Uma das funções principais dos UAVs é a sua capacidade de voo e o consumo energético que isso acarreta. Para se explicar o tempo de voo tem sempre que se ter também em consideração a energia que é consumida pelo drone nos diversos estados apresentados na tabela 2. De facto, o consumo de energia dominante de um UAV reside no sistema de controlo de propulsão que acelera o UAV e mantém a sua altura de voo [48].

Tabela 2: Estados de um UAV [49]

Modo	Descrição
Idle	*Drone* ligado mantido no solo e sem as hélices a rodar.
Armed	*Drone* ligado, mantido no chão com as hélices a rodar.
Take-off	O *drone* deixa o chão e começa a voar.
Horizontal	Executa um movimento paralelo ao chão.
Vertical	Move-se em ângulo recto em relação ao chão (para cima e para baixo)
Hover	Mantém a mesma posição no ar.

No artigo [49] são demonstrados os consumos energéticos e os tempos de voo estimados para o drone *Intel Aero Ready to Fly Drone* baseados em vários parâmetros tais como, o peso do drone, a sua capacidade energética e também tensão elétrica que a bateria é capaz de garantir: Nesse sentido foram analisados consumos energéticos com base nos seguintes testes:

- A potência e energia consumidas nos vários modos de funcionamento;
- O impacto energético dos sistemas de comunicação como GPS e Wi-Fi.

Em missões *indoor*, o *wifi* é principal modo de comunicação entre o UAV e a GCS. Assim, é importante quantificar o consumo de energia para a comunicação Wi-Fi.

Neste estudo, o UAV foi utilizado como *node* e a GCS foi usada como um AP. O UAV partilhava a sua informação de calibração, informação de estado, informação de bateria e parâmetros básicos de configuração, incluindo altitude de descolagem, velocidade, com a estação terrestre usando Wi-Fi.

O consumo total de energia do UAV enquanto mantém Wi-Fi com a estação terrestre não tem uma diferença significativa com a de quando não se comunica com a estação terrestre.

O mesmo estudo como acima foi realizado com o módulo GPS do UAV activado, e com o UAV tendo uma fechadura GPS 3D com 20 satélites. Pode-se notar que o consumo de energia para a comunicação GPS é negligenciável.

Embora mantendo sempre uma linha de visão clara entre o UAV e a estação terrestre, a distância entre a estação terrestre e o UAV foi gradualmente aumentada, até 50 metros, para quantificar o impacto da distância na comunicação. A potência instantânea em cada ponto foi calculada para um período de 20 minutos, e o consumo médio de energia em cada ponto foi considerado no gráfico na figura 17. A linha azul representa o consumo total de energia com comunicação Wi-Fi e a linha vermelha representa o consumo total de energia tanto com Wi-Fi como com GPS.

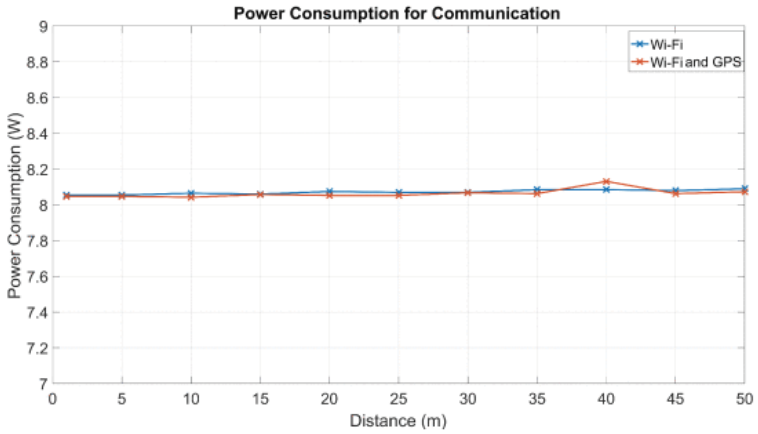


Figura 17: Impacto da distância na potência consumida pelo UAV para comunicação [49]

Todas as experiências foram realizadas no drone *Intel Aero Ready to Fly Drone* com baterias *Dualsky 4000 mAh 4S 25c, ECO-S LiPo*, em ambientes exteriores sem obstáculos, onde o vento era mínimo. As especificações do UAV e das baterias utilizadas estão listadas nas tabelas 3 e 4 respectivamente [49].

Tabela 3: Especificações do *Intel Aero Ready to Fly Drone* [49]

Parâmetro	**Medida**
Peso (sem bateria)	865 g
Dimensões (*hub-to-hub*)	360 mm
Altura	222 mm
Comprimento da hélice	230 mm

Tabela 4: Especificações da bateria *Dualsky LiPo Battery* [49]

Parâmetro	**Medida**
Capacidade	4000 mAh
Tensão	14.8 V
Peso	399 g
Dimensões (Comp.xLarg.xProf.)	141 x 44 x 31 mm

As medidas de potência apresentadas são a potência total consumida pelo UAV na realização das acções indicadas. Isto inclui a energia para o estado *armed*, a potência básica para as comunicações e a potência para os movimentos do UAV. O UAV foi feito para descolar e pairar a diferentes altitudes para compreender o comportamento do consumo de energia ao pairar e o impacto da altitude sobre ele.

A figura 18 mostra a voltagem, o desenho da corrente e o padrão de consumo de energia para um ciclo completo de *take-off, hovering* e *landing*. Os consumos de energia razoavelmente estáveis nos estados *Idle*, *Armed* e *Hover* e os picos repentinos no consumo de energia devido aos motores que começam a rodar as hélices e quando ocorre a descolagem pode ser visto.

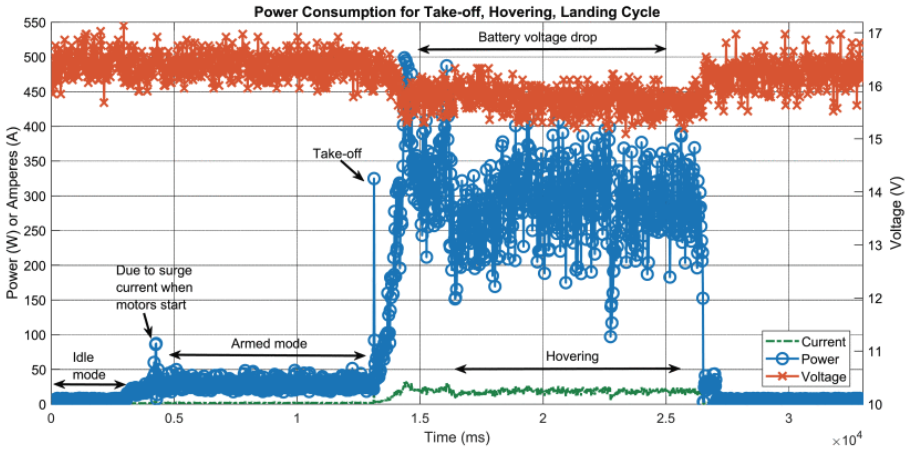


Figura 18: Tensão, corrente e potência consumida do uav para *take-off, hovering* e *landing*. [49]

No artigo [50] é referido mediante perfil de *State of Charge* (SOC) estudado pode se estimar o tempo de voo com base na fórmula abaixo.

$$t_{flight} = \frac{100}{r_{dis,hover}} \tag{1}$$

onde t_{flight} é o tempo de voo do(s) UAV(s) e o $r_{dis,hover}$ é a taxa de descarregamento da bateria (%/s).

2.7 Sumário

Este capítulo 2 pretende apresentar de uma forma geral o que existe a nível tecnológico e científico sobre os UAVs e sobre as tecnologias e protocolos de comunicação utilizados por eles assim como estudar o sua *performance* a nível energético.

Na secção 2.1 é abordado o tópico das arquitecturas de redes, com um UAV e com vários UAVs, dando a conhecer algumas das formações e redes mais utilizadas atualmente. Na secção 2.2 é dada uma visão sobre as várias aplicações e áreas em que os UAVs já estão presentes e exemplos de como podem ser utilizados nesses ambientes. Na secção 2.3 são apresentadas algumas soluções de controlo de UAVs por *software*, seja ele licenciado/pago ou *open source*. Na secção 2.4 são descritos dois dos protocolos mais utilizados para comunicações *UAV-to-Ground* e *UAV-to-UAV* e as suas aplicações. Na secção 2.5 são apresentados protocolos de encaminhamento de dados para sistemas de rede. Por último, a secção 2.6 é apresentado um estudo do consumo energético não só com os protocolos de comunicação mas também com todos os estados de funcionamento de um UAV desde a descolagem até à aterragem.

Capítulo 3

Especificação

Hoje em dia vivemos numa era digital em que quase não se vive sem estar ligado em rede e nesse sentido os UAVs e as suas ligações em rede vieram trazer diversas possibilidades de aplicação às mais diversas áreas de atividade.

Nesse sentido ao longo deste capítulo será especificado o protocolo *UAVProtocol* desenvolvido neste trabalho e que vai de encontro aos objectivos apresentados e que serão divididos em vários requisitos definidos na secção 3.1 para ajudar na definição de arquitectura do protocolo apresentado nesta tese.

3.1 Requisitos

Nesta secção são abordados os requisitos previstos para a especificação do protocolo de comunicação. O protocolo *UAVProtocol*, assenta em dois principais agentes de funcionamento a estação de controlo, também denominada de *base station* e as unidades móveis (UAVs). Nesse sentido, as subsecções seguintes são divididas nos requisitos individuais de cada um dos agentes e nos requisitos gerais comuns a ambos.

3.1.1 Requisitos gerais

De uma forma geral o sistema deve garantir que:

Req. 1 - Todas as unidades, sejam elas *base station* ou UAVs, devem poder ser endereçadas individualmente.

Req. 2 - As comunicações deverão ser asseguradas também em cenários com falta de cobertura de sinal do operador de telecomunicações.

Req. 3 - Permite que um *software* de controlo dos UAVs seja executado por cima do protocolo *UAVProtocol*.

3.1.2 Requisitos da estação de controlo

Para um correcto funcionamento da *base station* o protocolo deve garantir:

Req. 4 - A instalação de um ponto de acesso *wireless* para garantir a ligação das unidades móveis.

Req. 5 - Que seja atribuído automaticamente um endereço às unidades móveis e que estes sejam facilmente identificáveis.

Req. 6 - Uma base de dados que contenha todas as informações das unidades moveis e *base station*, como endereços IP e rotas a serem atribuídas.

3.1.3 Requisitos das unidades móveis

Para um correcto funcionamento das unidades móveis o protocolo deve garantir:

Req. 7 - As unidades móveis sendo alimentadas por baterias, deverão poder ser substituídas por outras carregadas de modo a assegurar a estabilidade da ligação.

Req. 8 - As unidades móveis envolvidas numa ligação terão que comunicar entre si de modo a garantir que a informação chega a todas.

Req. 9 - Ao detectar má qualidade de sinal de comunicações, as unidades móveis devem iniciar o processo automático de configuração de uma nova ligação.

Req. 10 - De modo a recolherem as informações necessárias, todas as unidades móveis devem estar equipadas com uma câmara de vídeo e sensores para os diversos tipos de atividades.

3.2 Arquitectura

Nesta secção é apresentado o modelo base de funcionamento do protocolo proposto nesta tese. A arquitectura deste sistema assenta num modelo de interconexão dos agentes referidos na secção 3.1, a *base station* e as unidades móveis, podendo ser uma ou várias, dependendo da extensão que pretendemos cobrir. Para esse efeito a seguir serão especificadas todas as funcionalidades específicas de cada elemento e o seu papel no protocolo apresentado.

3.2.1 Base Station

A *base station* como em qualquer sistema de grupos de UAVs assume o papel de estação de controlo de cada um dos equipamentos. Neste sistema manterá esse papel mas também funcionará como controladora de todo o protocolo de rede e onde são guardadas todas as configurações de rede não só de si mesma como de cada elemento de rede ou UAV adicionado à mesma.

Na rede a *base station* tem o papel de *Start Point* (SP). O termo SP deriva do facto de ser não só o primeiro elemento na rede mas também de ser o elemento que controla todas as definições da mesma e também o controlo dos UAVs. Na figura 19 é ilustrado o sistema interno proposto neste projecto para a *base station* em que cada módulo será especificado nas seguintes sub-subsecções.

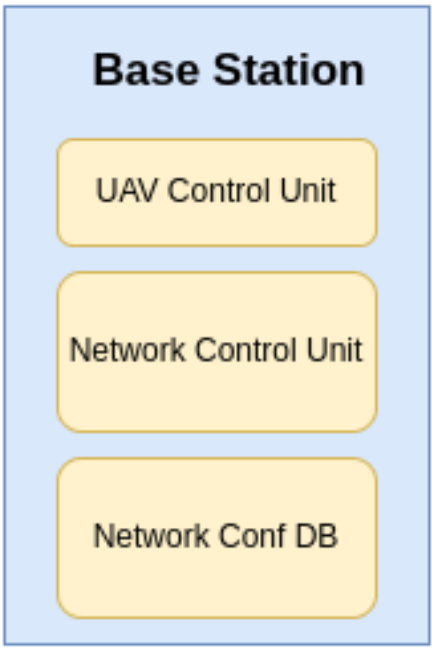


Figura 19: Constituição do sistema base da *Base Station*

3.2.1.1 Unidade de Controlo dos UAVs - *UAV Control Unit*

Como ilustrado na figura 20 a unidade onde se encontra o *software* de controlo dos UAVs através da *base station*, módulo de *Control Software*. Inclui também a unidade responsável pela visualização e análise dos dados recolhidos pelos mesmos, módulo *Analysis Unit*.

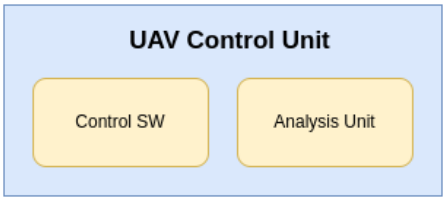


Figura 20: Arquitetura base da unidade de controlo de UAVs na *base station*

3.2.1.2 Unidade de Controlo de Rede - *Network Control Unit*

A unidade de controlo de rede na *base station* é responsável por todas configurações do *setup* da rede. Como se ilustra na figura 21 este módulo encontra-se dividido em três partes: a configuração do AP, as configurações do *Dynamic Host Configuration Protocol* (DHCP) e *Domain Name System* (DNS) e por último o a parte responsável pela tabela de *routing* da *base station*.

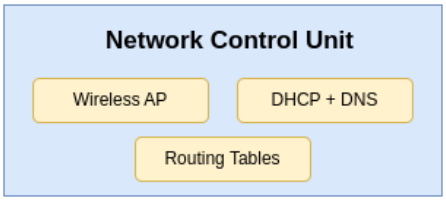


Figura 21: Arquitetura base da unidade de controlo de rede na *base station*

O módulo do ponto de acesso é responsável pela criação da estrutura física de acesso dos UAVs à *base station*, é responsável pela criação da ligação *wireless* sendo definido os parâmetros conforme o *hardware* disponível na *base station*.

Ligado também a esse AP estará ligado um servidor DHCP que atribuirá os endereços IP assim que ocorrer a ligação dos UAVs. Para além do servidor DHCP, está também configurado um servidor DNS para que tanto a *base station* como os UAVs sejam facilmente identificáveis dentro da rede.

O módulo de tabelas de *routing* do *UAVProtocol* é responsável por adicionar as novas rotas que vão sendo configuradas pelos UAVs. As novas rotas podem ser de dois tipos IP ou MPLS.

3.2.1.3 Base de Dados de Configurações de Rede - *Network Conf DB*

A base de dados de configuração da rede é onde se encontra toda a informação de configuração da *base station* e dos UAVs. Contém toda a informação necessária para a configuração do protocolo como se ilustra na figura 22.

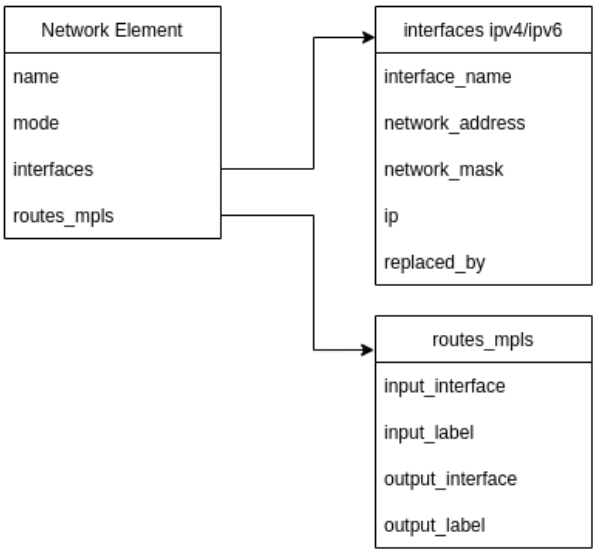


Figura 22: Diagrama representativo da estrutura da base de dados de configurações

Como ilustrado na figura 22 cada elemento tem o seu ficheiro *JavaScript Object Notation* (JSON) associado estando ele estruturado da seguinte forma:

- ***name*** - Indica o nome do elemento
- ***mode*** - Indica o modo de funcionamento do protocolo, real ou simulado.
- ***interface*** - Lista de todas as *interfaces* de rede utilizadas no protocolo.
- ***routes*** - Tabela de *routing* MPLS do elemento seleccionado.

3.2.2 UAVs

Os UAVs são parte integrante e essencial na arquitetura deste protocolo. De facto, são eles os responsáveis por levar uma melhor cobertura de sinal a zonas em que ele não é tão abundante. Na figura 23 é ilustrada a arquitectura base do sistema instalado nos UAVs. Tal como na *base station* existe um módulo de controlo de rede, o módulo que faz a leitura dos dados dos sensores e captura de imagens e por fim uma secção de *interface* API para poder aceder a aplicações do UAV através da *base station*.

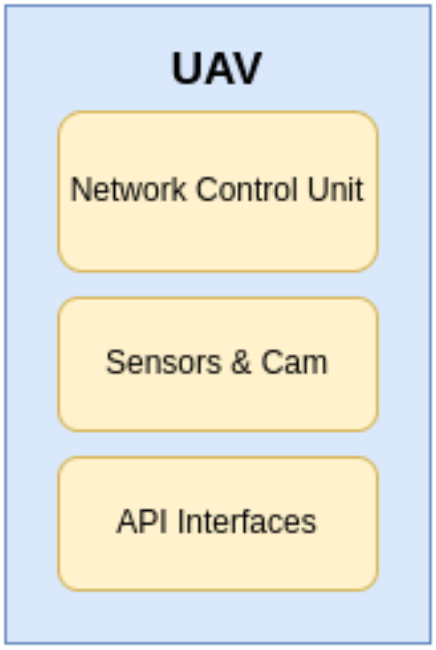


Figura 23: Constituição do sistema base dos UAVs

3.2.2.1 Unidade de Controlo de Rede - *Network Control Unit*

O módulo de configuração de rede é, tal como na *base station*, responsável por todas as configurações de rede e *routing* no UAV, podendo o encaminhamento ser realizado por IP ou MPLS. No caso do UAV ele pode ter dois modos de funcionamento: os modos *Relay Point* (RP) e de *End Point* (EP), que variam consoante a sua posição na rede e mediante o destino da informação que é enviada.

Cada elemento encaminha o tráfego para o EP passando por um caminho definido na tabela de encaminhamento, como num *router*. Se fosse aplicado o modo *switch* apenas teríamos acesso ao endereço MAC do equipamento mais próximo, dificultando assim a chegada a um elemento mais distante. Assim, como se fosse utilizados os UAVs como *Access Point*s (APs) Wi-Fi iria implicar, não só o aumento de número de redes mas também complicar o processo de ligação a essas redes fazendo com que fosse perdida conexão durante o processo.

Relay Point UAV

Um UAV com a modo *Relay Point* tem a capacidade de funcionar como um *hop* na rede reencaminhando toda a informação vinda de qualquer ponto da rede para o seu destino final.

A figura 24 ilustra o processo de reencaminhamento de um pacote com início na *base station* e final no EP.

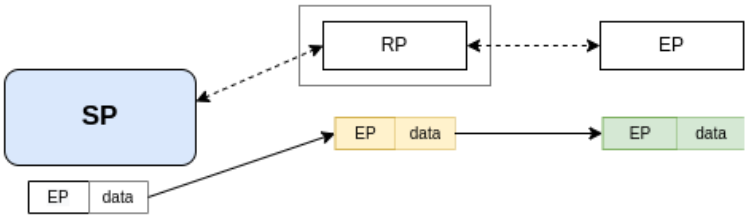


Figura 24: Modo de funcionamento de um RP

End Point UAV

O UAV com a função de *End Point* é o dispositivo final da rede ou para o qual o comando é destinado. No protocolo cada UAV, mediante o seu posicionamento ou então, se for o destino final do pacote IP, pode desempenhar tanto o modo EP como o modo RP. A figura 25 ilustra um sistema com uma *base station* e dois UAVs em que o último dada a sua posição na rede actua como EP da rede.

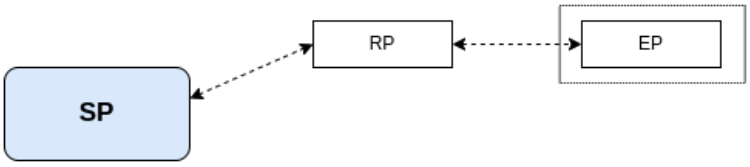


Figura 25: Modo de funcionamento de um EP

3.2.2.2 Unidade de leitura de dados

Representada na figura 23 como *Sensors & Cam*, a unidade de leitura de dados tem como principal objetivo a recolha e envio de dados tanto de vários sensores, como da câmara utilizada para *video streaming* e captura de imagem. Na figura 26 estão representados alguns dos sensores que podem fazer parte do módulo

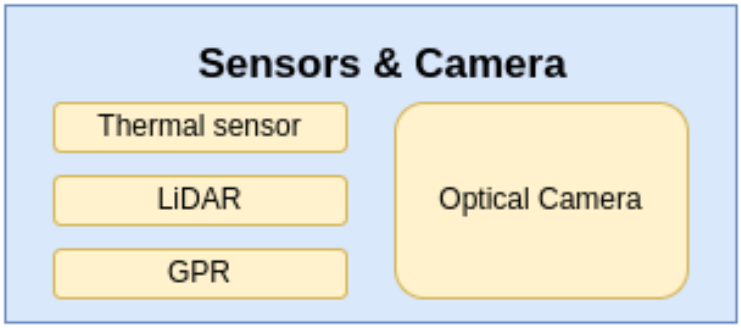


Figura 26: Constituição do módulo de leitura de dados

Os sensores podem ter a seguinte gama de aplicações:

- **Câmara** - para retirar uma melhor fotogrametria de um determinado terreno a ser avaliado.
- **Sensor de temperatura** - para criação de mapas térmicos que permitam a identificar a presença de uma pessoa ou animal num determinado ambiente.
- **Sensor *Light Detection and Ranging* (LiDAR)** - para ler dados das condições geológicas do terreno e criação de mapas 3D de alta definição de um terreno.
- **Sensor *Ground Penetrating Radar* (GPR)** - para a deteção de objectos soterrados [51].

3.2.2.3 APIs

Neste módulo estão definidas interfaces para testes automáticos. Entre as APIs disponíveis para utilização deverão estar analisadores de rede, como o *tshark* e interfaces para interação com o *hardware* em caso de necessidade, como ilustra a figura 27.

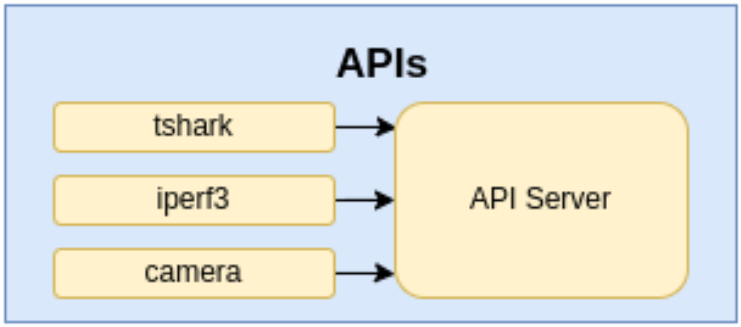


Figura 27: Constituição do módulo de APIs

3.3 Sumário

Neste capítulo pretende-se especificar e apresentar a arquitectura do protocolo *UAVProtocol*.

Na secção 3.1 são apresentados os requisitos do sistema. Os requisitos foram divididos em três partes, a parte referente à *base station*, a parte referente aos UAVs e por fim requisitos gerais do sistema.

A *base station* tem como requisitos, a instalação de um AP *wireless* para a ligação dos UAVs, que sejam atribuídos automaticamente endereços para facilitar a identificação dos UAVs e uma base de dados que contenha todas as informações correspondentes à rede do protocolo.

Os UAVs tem como requisitos a substituição de baterias assim que esta esteja descarregada afim de assegurar a estabilidade das ligações do protocolo assim como ao detetar que o sinal de comunicação não reúne as condições necessárias iniciar o processo de configuração automática de uma nova ligação. Os UAVs envolvidos numa ligação têm que comunicar entre si garantindo que a ligação chega ao destino e para além disso estarem equipados com dispositivos de recolha de imagem e sensores para diversos tipos de atividade.

O protocolo *UAVProtocol* deve garantir que seja possível a todas as unidades, *base station* ou UAV, ser endereçadas individualmente assim como, as comunicações sejam asseguradas em cenários adversos e por fim, permitir que sejam executados *softwares* de controlo de UAVs por cima do *UAVProtocol*.

Com base nos requisitos é apresentada a arquitectura do sistema, na secção 3.2 incidindo nos agentes principais do protocolo proposto nesta tese: a *base station* e os UAVs.

A *base station* é composta por três módulos: o módulo *UAV Control Unit*, o *Network Control Unit* e o módulo *Network Conf DB*. O primeiro módulo é responsável pelo comandos de controlo dos UAVs assim como pela análise dos dados captados por eles. O módulo o *Network Control Unit* é responsável por todas configurações de rede na *base station* entre elas as rotas IP ou MPLS. Por fim, o módulo *Network Conf DB* que contém a base de dados para ser utilizada nas configurações do protocolo *UAVProtocol* tanto na base como nos UAVs.

Cada UAV é também composto por três módulos: o módulo *Network Control Unit*, o módulo *Sensors & Cam* e o módulo *API Interfaces*. Tal como na *base station*, o módulo *Network Control Unit* é responsável pela configuração e gestão da rede, o módulo de *Sensors & Cam* é reponsável pela parte de leitura de dados por parte de sensores e câmaras instalados nos UAVs e por último, o módulo *API Interfaces* que disponibiliza APIs para se fazer análises ao protocolo em contexto de testes sempre que necessário.

Capítulo 4

UAVProtocol

Neste capítulo será apresentado modelo de funcionamento do protocolo *UAVProtocol* desenvolvido neste trabalho e que vai ao encontro dos requisitos apresentados na secção 3.1. Ainda neste capítulo será apresentada a forma como o protocolo proposto foi implementado em projecto real.

4.1 Modelo proposto

O modelo base deste sistema assenta numa ideia de transmissão de comandos, ou pacotes IP, numa base *multi-hop* desde a *base station* até ao UAV mais distante. Partindo de uma constituição de um protocolo de comunicação sem fios em que os UAVs estão dispostos sequencialmente em linha.

Como já foi referido, a *base station* actua como *Start Point* sendo o ponto de partida dos comandos na rede. Ao longo da rede, os comandos vão sendo transmitidos até ao seu destino final passando por UAVs denominado *Relay Point* até chegarem ao *End Point*-UAV. Todo este processo tem como objectivo final estender o alcance da ligação de rede entre os pontos mencionados, desde a *base station* até ao *End Point*-UAV, tendo em vista a máxima distância possível.

Para esse efeito, o modelo é baseado numa rede Wi-Fi *ad-hoc*, com possibilidade encaminhamento por IP ou MPLS, com endereçamento de IPv4 e IPv6.

O protocolo proposto tem duas fases de funcionamento distintas: a fase inicial de ligação dos drones diretamente à base (*Base-to-UAV*) utilizando uma ligação de rede Wi-Fi entre ambos. Na segunda fase e à medida que cada UAV se vai deslocando

para locais mais distantes da *base station* é configurada uma ligação entre o UAV mais próximo da base e o UAV que se encontra em movimento, através de uma ligação *UAV-to-UAV*.

4.1.1 Ligação Base-to-UAV

Nesta fase tal como explicado é feita directamente entre a *base station* e o UAV envolvido. Nesta fase é primeiro configurada toda a estrutura do protocolo na base e logo a seguir são enviados os comandos específicos do UAV, que serão referidos na secção de especificação do protocolo. Note-se que nesta fase os comandos podem ser enviados com o UAV em terra ou em modo de voo. Na figura 28 é ilustrado o estado inicial das diversas unidades num esquema exemplo desta ligação.

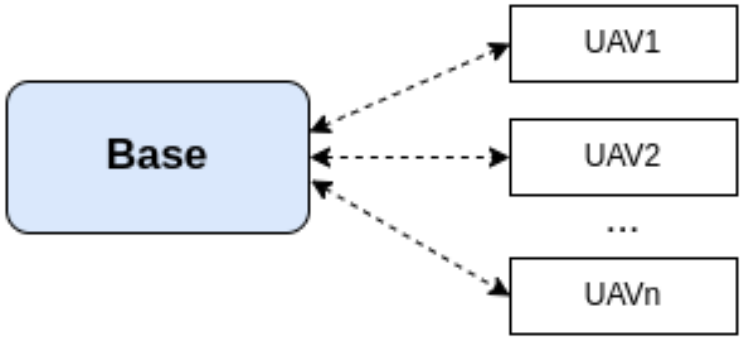


Figura 28: Esquema exemplo da fase B2U

Ainda nesta fase é medida em cada UAV a capacidade de detecção do sinal transmitido pela base ao UAV de modo a que este consiga manter uma conexão de rede estável e permitindo que este possa interligar-se com outros equipamentos semelhantes noutras fases do sistema proposto.

4.1.2 Ligação UAV-to-UAV

Partindo do cenário B2U e a partir do momento em que se tem toda a configuração necessária com um UAV no ar podemos iniciar de seguida a configuração de uma nova unidade móvel.

A nova unidade irá ser ligada à base tal como no exemplo da figura 28 e irá prosseguir o seu movimento até detectar que a qualidade do sinal Wi-Fi não é conveniente à realização de comunicação, processo que será explicado ainda neste capítulo.

A partir desse momento é desencadeado um pedido à *base station* para indicação do UAV mais próximo a essa unidade. Quando essa informação é recebida pela unidade, é enviado um pedido a esse equipamento para que a nova ligação à rede seja configurada. Ficando a rede com a *base station* como *Start Point* (SP), a

unidade mais próxima da *base station* como *Relay Point* (RP) e a unidade mais afastada como *End Point* (EP), como ilustrado na figura 29.

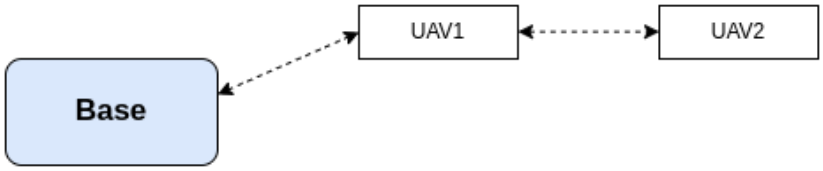


Figura 29: Esquema exemplo da fase U2U

4.2 Implementação do UAVProtocol

O *UAVProtocol* é o protocolo que é proposto, especificado e desenvolvido ao longo desta dissertação com vista aos requisitos. Na secção 4.1 foi proposto o modelo de funcionamento deste protocolo tendo em vista combinação das fases apresentadas e com a integração da *base station* de n equipamentos móveis.

O protocolo, desenvolvido na linguagem *python* [52], será ao longo das próximas secções abordado num sentido mais técnico e de implementação de funcionalidades do mesmo.

4.2.1 Esquema Geral

O esquema geral deste protocolo, ilustrado na figura 30, é assente num esquema de cliente-servidor através do protocolo *User Datagram Protocol* (UDP) para envio e recepção de comados que permitem o *request/response* remoto para qualquer uma das unidades do protocolo. O protocolo é composto por vários componentes que irão ser apresentados nas próximas secções.

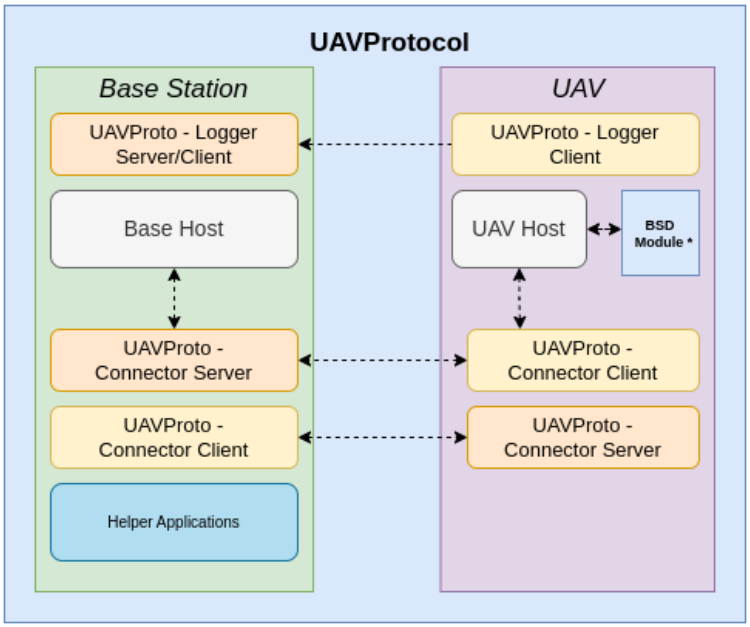


Figura 30: Esquema Geral do protocolo *UAVProtocol*

4.2.1.1 UAVProtocol Connector

O *UAVProtocol Connector*, na figura 30 é utilizado o diminutivo *UAVProto*, é o módulo responsável pela configuração *client-server* UDP no contexto do projecto. Este módulo está implementado tanto na *base station* como nos UAVs sendo adaptado de acordo com os requisitos do equipamento. Este módulo tem três modos de funcionamento.

- ***Server*** - *Interface* que fica à escuta por comandos enviados pelos clientes sejam eles de um UAV ou da *base station*.
- ***Host*** - é uma parte do *server* quando este necessita de executar um comando no equipamento a nível local sem necessidade de envio de outro comando.
- ***Client*** - É o modo responsável pelo envio da informação pela rede através de um *socket* UDP. Este modo pode enviar pedidos tanto a um servidor externo como o local.

4.2.1.2 BSD Module

É o módulo responsável pela leitura e análise da qualidade do sinal de Wi-Fi. Este módulo é implementado por uma *thread* que fica permanentemente a ler e registar os valores de qualidade do sinal.

Quando a leitura chega perto do limiar em que não é possível manter uma ligação é enviado um sinal ao módulo do servidor do UAV para iniciar a configuração automática de uma nova ligação através de outro UAV mais próximo.

4.2.1.3 UAVProto Logger

O módulo *logger*, implementado através do módulo *logging* [53] do *python*, é utilizado para ajudar a perceber o estado da configuração do protocolo nas suas várias etapas. No caso da imagem 31 é apresentado o final da configuração do túnel *Base-to-UAV* entre a *base station* e um dos UAVs.

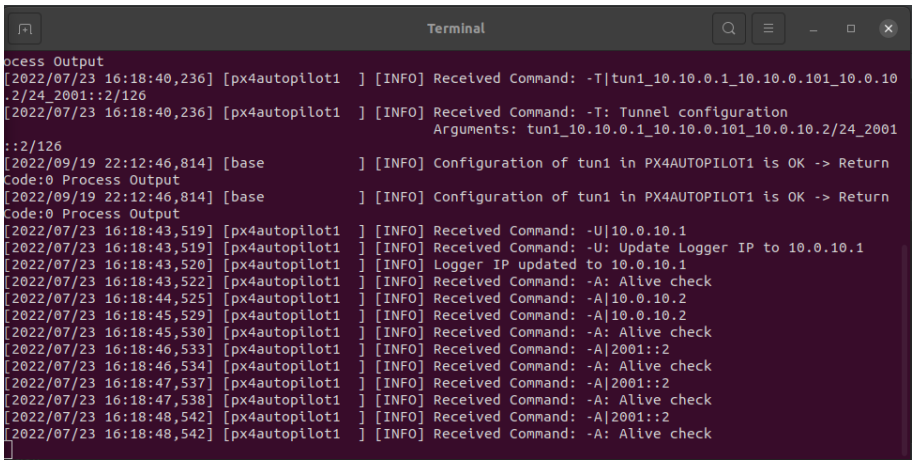


Figura 31: Módulo *Logger* implementado no *UAVProtocol*

No caso do protocolo é utilizado em dois modos, definidos na classe *DatagramHadler* do módulo *python*, o servidor e *client*. Na listagem 1 encontra-se um exemplo do código de inicialização de um *logger* como *client*.

- ***Server*** - Instalado na *base station*, o servidor, recebe todos pedidos de *log* feitos pelo cliente quer na *base station* ou em qualquer UAV.
- ***Client*** - O modo cliente, configurado tanto na *base station* como nos UAVs, é por onde podem ser enviados os registos de forma remota para serem agregados no conjunto final de *logs*.

```
1  import logging
2  import logging.handlers
3
4  handleLogs(logger_name, ip, port):
5      # create logger
6      logger = logging.getLogger(<logger_name>)
7      logger.setLevel(logging.DEBUG)
8
9      # create formatter
10     formatLogMessage = '[%(asctime)s,%(msecs)03d] [%(name)-13s]
           [%(levelname)-2s] %(message)s'
11     formatter = logging.Formatter(fmt=formatLogMessage, datefmt='%
           Y/%m/%d %H:%M:%S')
```

```
12
13      # define type of handler
14      socketHandler = logging.handlers.DatagramHandler(ip, port)
15
16      # add formatter
17      socketHandler.setFormatter(formatter)
18
19      # add handlers to logger
20      logger.addHandler(socketHandler)
```

Listagem 1: Exemplo de inicialização de um *logger* no modo cliente.

4.2.1.4 Aplicações de apoio

Neste módulo estão incluídas as *tools* que foram criadas ao longo do processo de desenvolvimento para ajudar ao suporte do teste e *debug* do protocolo. Nesse sentido surgiu a *UAVApp*, é baseada num *Graphical User Interface* (GUI) desenvolvido com base na biblioteca *tk* ou *tkinter* e que é representado na figura 32.

Figura 32: *UAVApp* - Painel de operações completo

A sua criação teve por base a automatizaçao de todo o processo de configuração do protocolo, que será abordado na secção 4.2.2, com a finalidade de realizar testes manuais de análise de dados, fazendo-o de uma forma simples e visualmente mais apelativa.

Para o processo de testes automáticos desencadeados pela *tool Jenkins* [54] e desenhados em *RobotFramework* [55, 56] foi desenvolvida uma API, chamada *uavapp_api* que tem a mesma base de funcionamento da *UAVApp*.

A API, representada na listagem 2, é implementada tendo por base uma classe com vários métodos, estando nesse exemplo apenas representado parte de um desses métodos.

```
1  import os
2  import subprocess
3  from elements.uav import Uav
4
5  class UAVAppApi():
6      # Constructor method
7      def __init__(self, params):
8          # UAV Logger Module Initialization
9          self.__init_uavprotocol_logger()
10         ...
11
12     # Public Methods
13     def uav_start(self, name, tunnel_name, route):
14             uav = Uav(name, self.uav_logger)
15             uav.start()
16     ....
```

Listagem 2: Parte da implementação da API da *UAVApp*.

A *uavapp_api* é disponibilizada através de um servidor XML-RPC, que é implementado na biblioteca *RobotRemoteServer* do *python* e que pode ser visto no exemplo 3.

```
1  from robotremoteserver      import RobotRemoteServer
2  from uavapp_api          import UAVAppApi
3
4  def main(params):
5      # Getting Parameters for Api
6      api=UAVAppApi(params)
7
8      #Start XML-RPC server
9      RobotRemoteServer(api, host=params["ip"], port=params["port"])
10
11 ...
```

Listagem 3: Parte da implementação server XML-RPC para a API da *UAVApp*.

Por fim, os métodos da API disponibilizados no servidor podem ser chamados no *RobotFramework*, como uma *keyword*/função normal como está ilustrado na figura 33.

```
*** Settings ***
Documentation  Resource file for UAV Command Forward Tests
Library        Remote      ${uavapp_server_ip}              WITH NAME       uavapp


*** Keywords ***
Start Drone
    [Documentation]        Start drone by uavapp api
    [Arguments]                    ${name}  ${tunnel_name}  ${route}
    ${route}=  convert to integer  ${route}
    uavapp.uav_start    ${name}  ${tunnel_name}  ${route}
```

Figura 33: Exemplo de *import* de uma *Remote Library* e a sua utilização numa *keyword*.

4.2.2 Configuração do Protocolo

Nesta subsecção será apresentado todo o processo do configuração do protocolo *UAVProtovol* sendo descritos todos os passos necessários para esse processo assim como esquematizados todos os comandos utilizados no protocolo através de MSCs. Este protocolo foi baseado na sua concepção na implementação de túneis GRE entre nas ligações B2U e U2U.

O protocolo GRE é um protocolo de túnel desenvolvido originalmente pela Cisco, a fim de permitir o encapsulamento de uma grande variedade de protocolos da rede através de ligações ponto-a-ponto.

Um túnel GRE é também usado quando os pacotes precisam de ser transmitidos de uma rede para outra enquanto atravessam um público ou inseguro como a *Internet*. O princípio do GRE é criar um túnel virtual entre dois *routers* em que os pacotes são então enviados através do túnel criado [57].

4.2.2.1 Configuração dos túneis B2U

A ligação da *base station* aos UAVs é feita em duas fases fundamentais e uma fase de validação de configuração: a inicialização do protocolo *UAVProtocol* no UAV, seguido da indicação de disponibilidade do protocolo (*ready signal*) e a fase de configuração do túnel GRE entre a base e o UAV.

Ready Signal

Neste fase é inicializado o servidor UDP que irá atender todos os pedidos de configuração feitos pela *base station*. Este processo é esquematizado pelo MSC na figura 34. O sinal *ready sinal* indica que o UAV está pronto para receber as configurações do túnel.

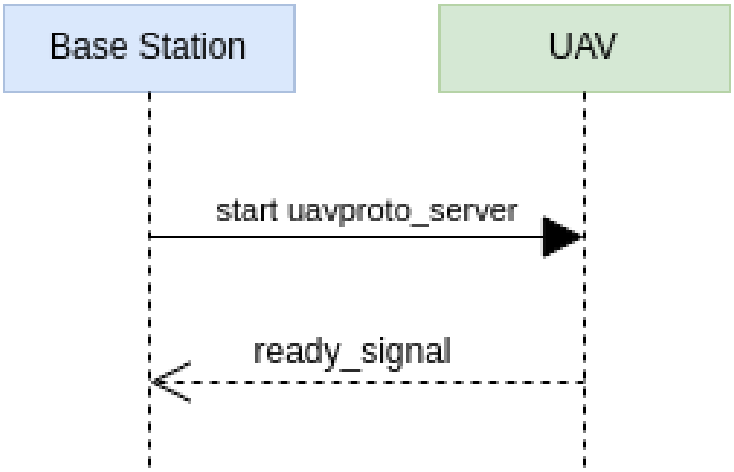


Figura 34: MSC representativo o procedimento inicial e a resposta do *ready signal*

Numa primeira fase é ativado o servidor UDP do *uavprotocol* na *base station* e de seguida é enviado um pedido por SSH, ainda sem o túnel GRE configurado, para o UAV de modo a ser ser ativado o servidor do *UAVProtocol*, processo desencadeado pelo comando na listagem 4. Durante este processo, na *base station* é iniciado também um servidor *udp* que ficará aguardar o comando *ready* enviado pelo UAV. Após toda esta configuração se encontrar completa a unidade móvel, ou seja a inicialização do *server* do *UAVProtocol*, está preparada para receber os comandos enviados pela *base station* ou outros UAVs e iniciar a configuração dos túneis GRE.

```
1 ssh <username>@<uav_host> "cd <file_location>; python3
    uavprotocolconnector.py --hostname <uav_host> --ip :: --port <
    port> --option server > <repo_location>/logs/output.txt &"
```

Listagem 4: Envio do commando *start_uavproto_server* via SSH.

Alive Check

Esta etapa de *alive check* é uma etapa complementar apenas de validação de comunicação com o servidor *UAVProtocol*. Numa primeira fase, é executado este processo como confirmação de funcionamento do protocolo e posteriormente para confirmação da correcta configuração dos túneis GRE que será abordada a seguir.

A figura 35 representa o modo de funcionamento de *request* e *answer* do comando de *alive check*.

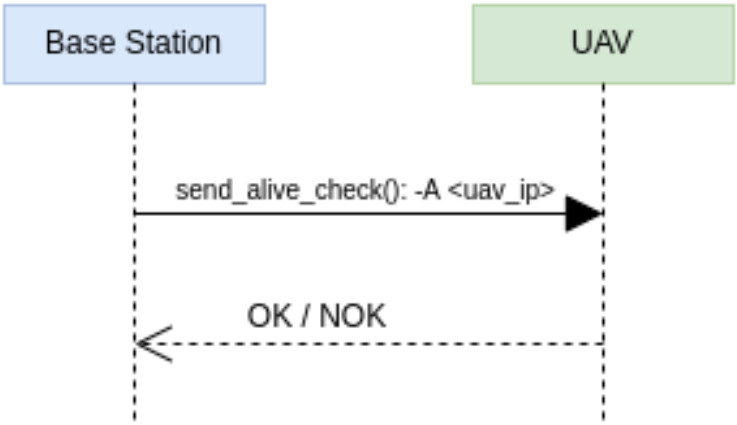


Figura 35: MSC da etapa de *Alive Check*

Nesta fase, apenas de validação, representada na figura 35 é enviado para o UAV um commando *alive command*: (-A) para o endereço IP do UAV naquele instante e este responde com sucesso (OK) se estiverem reunidas todas as condições ou com um erro (NOK) no caso de ocorrência de erro.

Configuração do túnel GRE

Esta última etapa é a etapa em que se realiza a configuração da ligação via túnel entre a *base station* e o UAV. É uma fase um pouco mais complexa do que as anteriores pois é necessário correr as configurações necessárias não apenas no UAV mas também na *base station*. Esta é a ultima etapa da configuração B2U uma vez que é nela que é feita a principal configuração do protocolo. A figura 36 ilustra todo o processo de configuração do túnel entre a base e o UAV.

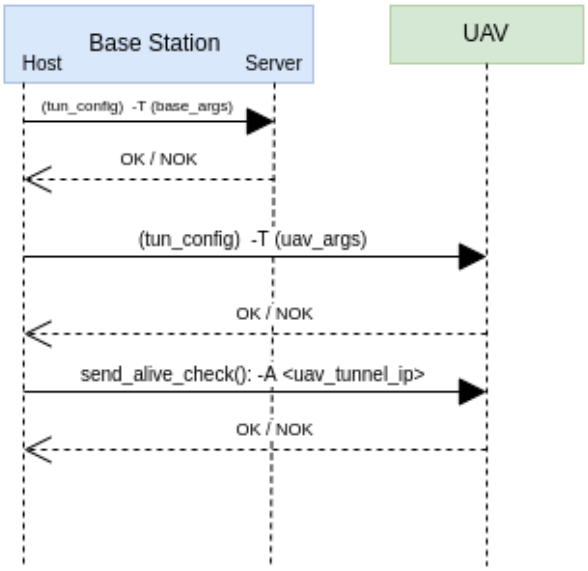


Figura 36: MSC da etapa de configuração da B2U

Nesta fase e depois da indicação de sucesso do comando *alive check* a *base station* inicia o processo de configuração do túnel GRE. Nesta fase entram em funcionamento os dois modos explicados na subsecção 4.2.1.1, o modo *host* e o modo servidor do *UAVProtocol*.

Numa primeira parte o *host* carrega todas configurações do túnel diretamente da base de dados, gera os argumentos para passar no comando de *start_simple_tunnel* e envia o pedido para o *server*. O *server* processa todos os comandos para a criação do túnel do lado da *base station*.

Ainda durante este processo, existem diferenças no procedimento comando de *start_simple_tunnel* devido ao facto do método de *routing* que seja selecionado.

Para o IP é utilizado o *script* para configuração do túnel apresentado no exemplo 5.

```
1 ip tunnel add <tunnel_name> mode gre remote <remote_ip> local <
      local_ip> ttl 255;
2 ip link set <tunnel_name> up;
3 ip addr add <ip> dev <tunnel_name>;
4 ip addr add <network_ipv6> dev <tunnel_name>;
```

Listagem 5: *Script* de configuração do túnel GRE.

No caso do MPLS é executado também o *script* do exemplo 5 para além da configuração da rota MPLS, representada no exemplo 6.

```
1 sysctl -w net.mpls.conf.<gre_tunnel_name>.input=1;
2 sysctl -w net.mpls.conf.lo.input=1;
```

```
3 sysctl -w net.mpls.platform_labels=1048575;
4 sysctl -w net.ipv4.conf.all.rp_filter=2;
5 ip6tables -t raw -I PREROUTING -m rpfilter --invert -j DROP;
6
7 ip route change <network_address> encap mpls <out_tag> via <
      ip_next_hop> dev tun1;
8 ip -6 route del <network_address_v6>
9 ip -6 route add <network_address_v6> encap mpls 100 dev tun1;
10 ip -f mpls route add <in_tag> dev lo;
```

Listagem 6: *Script* de configuração do túnel GRE.

Após a configuração dos comandos é retornada uma *string* com a informação de sucesso da operação (OK) ou com o código de erro e a informação de (NOK) em caso de insucesso.

De seguida é iniciado o mesmo processo mas para o UAV correspondente na cadeia sendo retornados exatamente os mesmos valores em caso de sucesso ou erro. Depois de toda essa configuração é feito novamente o passo intermédio do *alive check* para verificar a existência de comunicação com a nova configuração de IP através do túnel GRE.

4.2.2.2 Configuração dos túneis U2U

Nesta subsecção serão abordadas as configurações para quando o UAV se começa a deslocar e a começa a ficar mais distante da *base station* começando a perder sinal iniciando assim a configuração da ligação à *base station* através de outro UAV mais próximo.

Esta configuração tem quatro fases: a fase da detecção de má conexão de Wi-Fi, seguida da configuração do túnel entre dois UAVs, a inserção das rotas de encaminhamento entre os túneis e por fim a remoção de túneis e rotas que não são necessários.

Detecção de má qualidade de sinal

Numa primeira fase a ideia passa por, ao longo da trajetória de voo de um determinado UAV, ir sendo monitorizada a qualidade do sinal Wi-Fi e quando a mesma chegar a um nível abaixo de um limite expectável em que quase não seja possível a conexão seja enviado um sinal (*-B*) ao servidor *UAVProtocol* desse mesmo UAV que nesse instante responde com um pedido de dados de configuração à *base station*.

Para esta fase da configuração existem dois modos de funcionamento que são:

- Detecção automática de má qualidade do sinal - que basicamente segue o procedimento de uma forma automática, feita por medição através do módulo

Bad Signal Detection (BSD) no UAV. Este procedimento apenas foi idealizado não tendo no entanto sido implementado na solução atualmente.

- Simulação - O sinal é enviado através da base para efeitos de teste da configuração. Esta solução foi implementada como um *workaround* dos primeiros testes através de *virtual machine* e que é actualmente utilizado nos testes apresentados no próximo capítulo.

Como se pode ver na figura 37 no final do processamento do sinal de *bad signal* é enviado um pedido de configurações (*-C*) à *base station* de modo a poder ser iniciada a próxima fase.

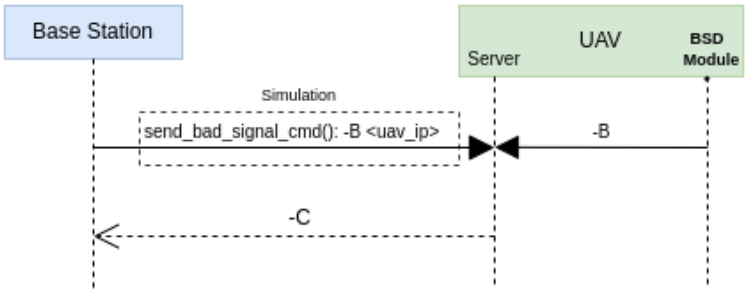


Figura 37: MSC da etapa de envio e processamento do *bad signal*

Configuração do túnel GRE entre UAVs

Esta é a etapa de configuração dos túneis U2U. O seu processo de configuração é semelhante ao processo explicado na etapa da configuração B2U mas neste caso será entre UAVs.

Na figura 38, partindo do pedido de configuração feito na etapa de *bad signal* à *base station*, é enviado ao UAV mais distante, no caso o UAV2, onde ocorreu a deteção de má qualidade de sinal, os dados para a configuração do novo túnel entre essa unidade móvel e a unidade mais próxima da *base station*, o UAV1.

Antes de enviar *feedback* à *base station*, utilizando os nomes do exemplo acima, o UAV2 processa a informação recebida da *base station* e envia o comando do *tunnel configuration* ao UAV1 com os dados do túnel relativos a essa unidade móvel que depois de ser implementado na unidade retorna tal como no processo de configuração B2U. O mesmo processo ocorre também no UAV2 no final destes dois processos é enviado *feedback* ao comando da *base station* em caso de sucesso (OK) e em caso de ocorrência de erro (NOK) e *error code*.

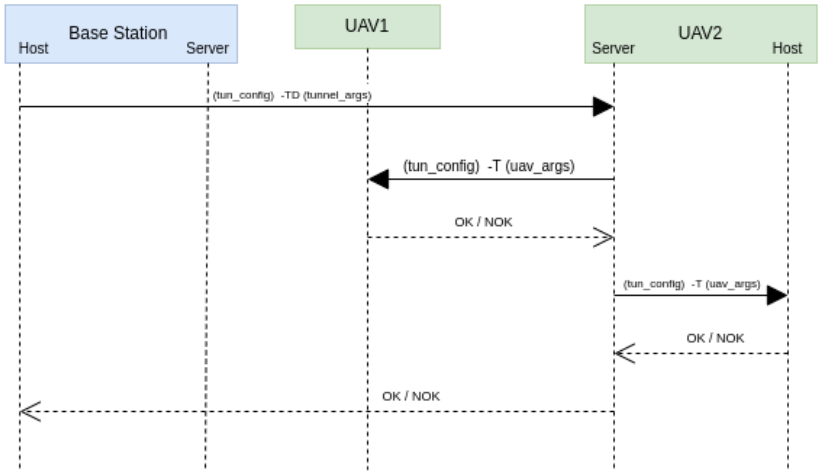


Figura 38: Exemplo do MSC da etapa de configuração do túnel U2U entre UAV1 e UAV2

Configuração de encaminhamento

No final da fase anterior apenas os UAVs que foram utilizados na configuração ficam com ligação entre eles. Para enviarmos dados para o último UAV é necessário que sejam configuradas as rotas de encaminhamento de uma extremidade (*base station*) até ao final da cadeia de UAVs.

A partir do momento em que é recebido na *base station* o sinal de confirmação de sucesso da configuração do túnel entre os UAVs, é processado o pedido de configuração de uma nova rota (*chain*) desde a *base station* até ao UAV mais distante. Depois dessa recolha de dados da base de dados e do respectivo processamento, a base manda para cada UAV da cadeia o comando de *route configuration*, mediante o seu posicionamento na rede e mediante o tipo de encaminhamento pretendido, e dos quais recebe o seguinte *feedback*: (OK) em caso de sucesso e (NOK) em caso de ocorrência de erro.

Tendo a *base station* como *Start Point* da rede é configurado para esta uma *simple route*, ou seja, seja a uma rota que permite a inserção de pacotes IP na rede de destino. O mesmo procedimento é seguido para o UAV com o funcionamento de *End Point*, nos *scripts* do exemplo 7.

```
#IP
ip addr add <network_ipv6> dev <tunnel_name>;
ip -6 addr add <network_ipv6> dev <tunnel_name>;

#MPLS
ip route add <network_address> encap mpls <out_tag> via <
    ip_next_hop> dev tun1;
ip -6 route add <network_address_v6> encap mpls 100 dev tun1;
```

Listagem 7: *Script* de configuração de uma rota simples em IP e MPLS

No caso do UAV que funciona como *Relay Point* é enviado pela *base station* o comando de *route forward* de modo a que os pacotes com destino a outros UAVs sejam reencaminhados para o destino final.

```
#IP
echo 1 > /proc/sys/net/ipv4/ip_forward;
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding;
iptables -F; iptables -X;
iptables -t nat -A POSTROUTING -o <iface_in> -j MASQUERADE;
iptables -A FORWARD -i <iface_in> -o <iface_out> -m state --state
    RELATED,ESTABLISHED -j ACCEPT;
iptables -A FORWARD -i <iface_out> -o <iface_in> -j ACCEPT;
iptables -t nat -A POSTROUTING -o <iface_out> -j MASQUERADE;
iptables -A FORWARD -i <iface_out> -o <iface_in> -m state --state
    RELATED,ESTABLISHED -j ACCEPT;
iptables -A FORWARD -i <iface_in> -o <iface_out> -j ACCEPT;
ip6tables -F; ip6tables -X;
ip6tables -t nat -A POSTROUTING -o <iface_in> -j MASQUERADE;
ip6tables -A FORWARD -i <iface_in> -o <iface_out> -m state --state
     RELATED,ESTABLISHED -j ACCEPT;
ip6tables -A FORWARD -i <iface_out> -o <iface_in> -j ACCEPT;
ip6tables -t nat -A POSTROUTING -o <iface_out> -j MASQUERADE;
ip6tables -A FORWARD -i <iface_out> -o <iface_in> -m state --state
     RELATED,ESTABLISHED -j ACCEPT;
ip6tables -A FORWARD -i <iface_in> -o <iface_out> -j ACCEPT;

#MPLS

ip -f mpls route del <tag_in>;
ip -f mpls route add <tag_in> as <tag_out> via inet <ipv4_out>;
ip -6 -f mpls route del <tag_in>;
ip -6 -f mpls route add <tag_in> as <tag_out> dev <out_if>;
```

Listagem 8: *Script* de configuração de uma rota de reencaminhamento em IP e MPLS

Na imagem 39 é representado todo o processo descrito acima para cada um dos UAVs envolvidos na cadeia.

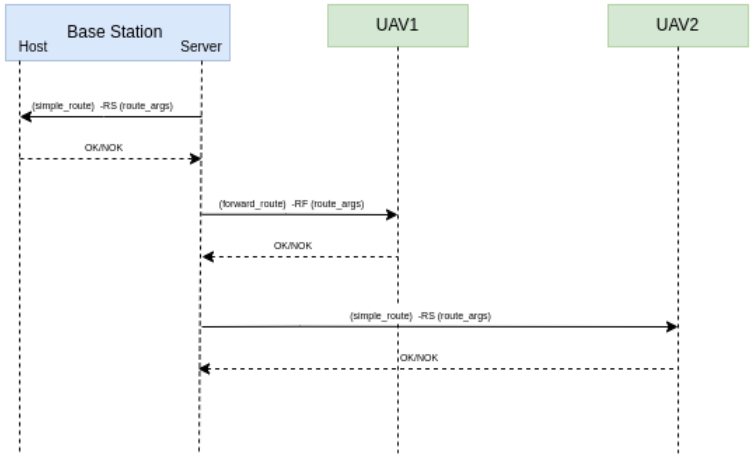


Figura 39: Exemplo do MSC da etapa de configuração da rota desde a *base station* até ao UAV2

Remoção de túneis não utilizados

Logo após a conclusão da configuração das rotas, a *base station* reúne os dados do túnel que irá ser substituído, ou no caso desactivado e removido. Depois de ter esses dados a *base station* envia para cada um os elementos da rede o comando de *tunnel replacement* de forma a desactivar e remover os mesmo.

Importa referir que nesta fase podem ser desativadas as configurações realizadas tanto no processo de configuração B2U como na configuração U2U dependendo da fase do processo em que se encontra o UAV nesse instante.

Na figura 40 é representado todo o processo descrito acima para os túneis e respectivos UAVs envolvidos na configuração de desativação de um túnel seguindo o exemplo acima com *base station* mais dois UAVs.

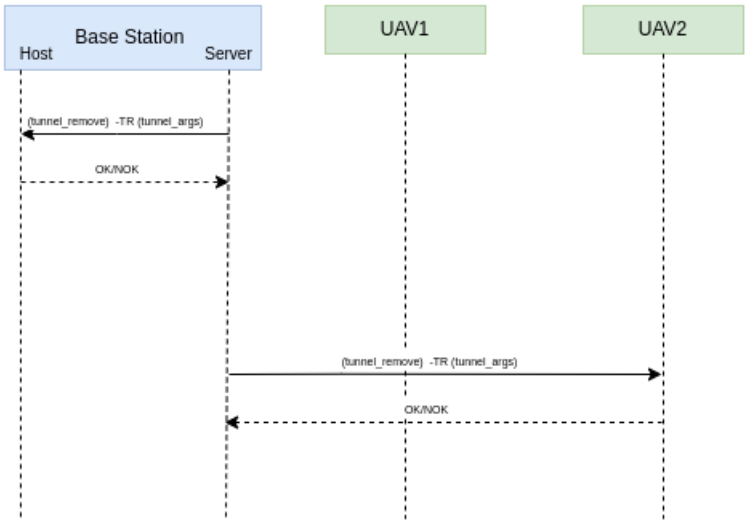


Figura 40: Exemplo do MSC da etapa de desactivação do túnel tun2

4.3 Sumário

Ao longo deste capítulo são apresentadas todas as fases da implementação e desenvolvimento do protocolo *UAVProtocol*.

Na secção 4.1 é apresentado o modelo base do protocolo e as suas fases de configuração: a fase de ligação directa da *base station* ao UAV, denominada de ligação *Base-to-UAV* (B2U), e a fase da ligação entre UAVs, conhecida como ligação *UAV-to-UAV* (U2U).

Na ligação B2U, é configurada uma ligação directa entre a *base station* e o UAV envolvido. Primeiro é iniciado o processo de configuração na *base station* e logo de seguida no respectivo UAV.

A ligação U2U é iniciada assim que, um UAV se começa a deslocar e detecta que a qualidade do sinal Wi-Fi da ligação à base não é conveniente à comunicação. A partir desse ponto é feito um pedido à *base station*, sobre informações de UAVs mais próximos, pelo UAV onde é detectada má qualidade de sinal e em seguida iniciando o processo de configuração da ligação tal como na ligação B2U.

Na secção 4.2 são apresentados de uma forma geral os módulos do protocolo que se encontram em cada um dos agentes e as suas aplicações.

Na subsecção 4.2.1 são apresentados os módulos que fazem parte do *UAVProtocol* tanto na *base station* como nos UAVs. Na *base station*, o esquema geram do protocolo é constituído pelo *UAVProto Logger*, que funciona como colector dos *logs* da própria *base station*, no modo *client*, e dos UAVs no modo *server*. Ainda na *base station* existe o módulo do *UAVProto Connector*, nos modos de *client*, *host*, *server* para efectuar a troca de comandos UDP com os UAVs. Por fim na *base station* encontra-se o módulo das *Helper Application*, no qual se encontram aplicações, *UAVApp* e

uavapp_api, que foram criadas para fins de teste e *debug* durante o desenvolvimento do protocolo *UAVProtocol*. Nos UAVs, tal como na *base station*, existem os módulos *UAVProto Logger*, que nos UAVs apenas funciona como *client*, e também o módulo *UAVProto Connector* com os mesmos modos de funcionamento da *base station*. Para além desses módulos, encontra-se também o módulo *BSD Module*, que é composto por uma *thread* responsável pela monitorização da qualidade do sinal Wi-Fi e quando esta atinge um nível crítico comunica de imediato com o *UAVProto Connector* para iniciar uma nova ligação.

Por fim, na subsecção 4.2.2 foram abordados todos as fases e possíveis de serem utilizados, nas fases de configuração dos túneis B2U e U2U, até atingir o modelo final. Na configuração do túnel B2U, entre a *base station* e um UAV são executadas duas operações. A operação de *ready signal* indica que depois de ser iniciado o protocolo na *base station* e UAVs o *UAVProtocol* está pronto a fazer troca de comandos entre *base station* e UAVs. Logo após essa fase, é iniciado o processo de validação de troca de comandos através da operação de *alive check*. Por fim é iniciada a configuração do túnel GRE entre *base station* e UAVs. Depois de concluída esta fase é apresentada a fase de configuração do túneis U2U, entre UAVs. Nesta fase é necessária a ocorrência de má qualidade do sinal, através do módulo *BSD Module* que apenas é idealizado está implementado. Para fazer a simulação dessa situação foi criado um comando (-B) que simula precisamente essa situação. Depois de ocorrência de má qualidade de sinal, é enviado um pedido de dados para a *base station* e iniciado o processo de configuração do túnel GRE entre o UAV onde ocorreu a detecção e o UAV mais próximo da *base station*. Após este processo, apenas ficam ligados os dois UAVs sendo necessário definir rotas de encaminhamento de dados, por IP ou MPLS, desde a *base station* até ao UAV mais distante, o que acontece na operação de configuração de encaminhamento. Logo depois dessa operação estar concluida, são removidas os túneis e rotas que já não são necessários na ligação final, através da operação de remoção de túneis não utilizados.

Capítulo 5

Teste e Análise do Sistema

Ao longo deste capítulo são apresentados o *setup*, os testes realizados a este protocolo e analisados os resultados obtidos. Numa primeira secção são apresentados os propósitos de teste para o protocolo *UAVProtocol*, bem como as condições em que foram executados os testes. Numa fase mais adiantada serão apresentados os resultados dos testes realizados. E por fim, serão analisados de forma crítica os resultados obtidos nos testes.

5.1 Propósitos do Teste

De acordo com a requisitos, apresentados na secção 3.1, e pela especificação apresentada na secção 3.2 optou-se por um cenário de teste constituído por uma *base station* seguida por três unidades móveis. Como o âmbito do projecto é apenas validar o funcionamento da *stack* de rede não seria necessária a aquisição de três UAVs reais, pelo que se optou por utilizar soluções mais em conta. Numa primeira fase com *virtual machines* para validar o correcto funcionamento do protocolo em fase de desenvolvimento e por fim em ambiente real com *Raspberry Pis* em vez de UAVs reais. Para esse efeito, foi criado um *job* no *Jenkins*[54] que coordena a execução automática dos testes através da *base station*.

Sendo assim nestes testes pretende-se ter uma comparação de protocolos de encaminhamento entre as tecnologias IP e MPLS de modo a perceber qual é o mais vantajoso a nível de processamento e rapidez na transferência de dados.

5.2 Setup do Teste

Na impossibilidade de ter os *Raspberries* sem baterias acopladas, ou seja, impeditivo de não termos grandes distâncias entre *Raspberries* e também como apenas foram utilizado o sistema de simulação de má qualidade de sinal, optou-se por um ambiente que permitisse algumas distâncias mínimas entre eles, no caso testado estimou-se uma distância mínima de 2 metros, entre a *base station* e o primeiro *Raspberry*, e de 6 metros com os 2 seguintes *Raspberries* com algumas barreiras como paredes ou armários entre eles para introduzir alguma distorção que possa ocorrer, como ilustrado na figura 41.

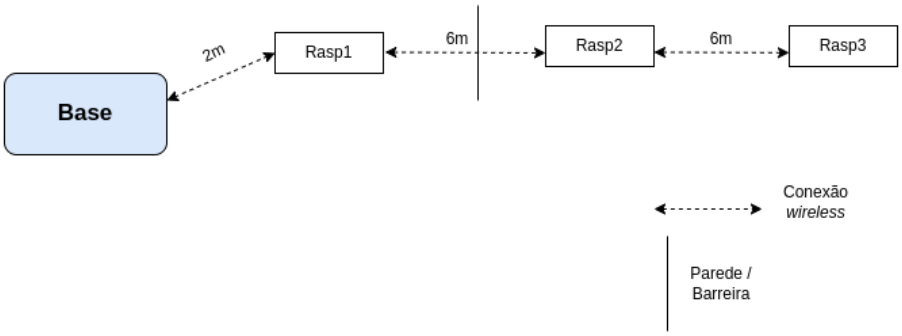


Figura 41: Esquema ilustrativo da disposição da *base station* e dos UAVs no ambiente de teste.

Nas subsecções seguintes serão analisados os componentes utilizados para o *setup* de realização deste teste.

5.2.1 Hardware & Software utilizados

5.2.1.1 Hardware

Para a *base station* tendo em conta a capacidade de processamento e a possibilidade de correr tarefas paralelamente em *threads* optou-se pela seguinte configuração:

- **CPU**: Intel Core i7-8665 @ 1.8GHz com 8 *cores*
- **RAM**: 16 GB
- **Capacidade de Disco**: 512 GB

Tal como na *base station* teve-se em conta a capacidade de processamento e a possibilidade de correr tarefas paralelamente em *threads* para além das características de rede e nesse âmbito optou-se pela seguinte configuração:

- **CPU**: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- **RAM**: 4 GB

- **SD-Card**: 32 GB
- **Placa de rede *wireless***: 2.4 GHz e 5.0 GHz IEEE 802.11ac *wireless on-board*
- **Camâra**: Módulo de câmara *OV5647* com sensor 1080p de 5 megapixels

5.2.1.2 Sistema Operativo

Para a *base station* o sistema operativo escolhido foi o *Ubuntu 20.04.5 LTS 64bit*, primeiro por ser *open source*, por ter uma grande comunidade de apoio ao desenvolvimento, e por último por causa da compatibilidade com o sistema de controlo *qGroundControl* e o protocolo *MAVLink* usados para efeitos de simulação.

Para os UAVs o sistema operativo escolhido foi o *Raspberry Pi OS Lite 64bit* uma vez que é o sistema mais utilizado na plataforma e é suportado por uma grande comunidade *open source*. Outro dos pontos para a escolha foi o facto de ter disponível apenas a versão *Command Line Interface* (CLI) o que nos permite ter um sistema mais leve e que liberta espaço para outros dados e registos.

5.3 Testes funcionais

Numa primeira fase de configuração foram desenhados testes que validassem todo o processo de configuração apresentado na sub-secção 4.2.2 de configuração do protocolo *UAVProtocol*. Por fim, com o correcto funcionamento do encaminhamento dos pacotes IP desde a *base station* até ao último UAV, quer por encaminhamento IP ou MPLS são executados testes de conectividade *end-to-end*.

5.3.1 Configuração do setup

Tal como referido estes testes realizam a configuração do protocolo na *base station* e UAV. O conjunto de testes de configuração estão na lista apresentada no relatório obtido da execução do *RobotFramenwork*, ilustrado na figura 42. Estes testes utilizam a *uavapp_api* para realizar o envio dos comandos definidos nas configurações de B2U e U2U.

Figura 42: Lista de testes utilizados na configuração do *UAVProtocol* através da *uavapp_api*.

Cada um destes testes está definido para as duas vertentes, IP e MPLS como iremos ver a seguir no desenho de cada um dos testes e das *keywords*. As *keywords* no *RobotFramework* tem o mesmo objectivo do que uma função na linguagem *Python* ou *C*, ou seja, são um conjunto de instruções ou comandos que são executados para garantir o funcionamento de um determinado comportamento esperado. Neste caso *keywords* são utilizadas para definir os *steps* de cada teste e também ajudarem na validação do resultado final da configuração, ilustrada na figura 43.

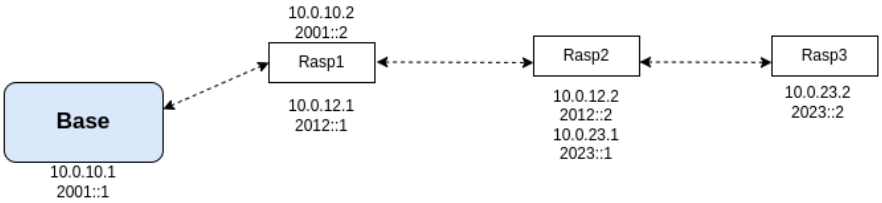


Figura 43: Esquema ilustrativo da disposição da *base station* e dos UAVs no ambiente de teste já com os IPs finais configurados.

5.3.1.1 Configuração Base-to-UAV

Os teste *Setup UAV1 and UAV2* e *Test Setup UAV3* são testes de configuração B2U. Durante estes testes é executada a *keyword Start Drone*, ilustrada na figura 44, que interage com a *uavapp_api* para garantir a execução das fases do *Ready Signal*, *Alive Check* e por fim configuração do túnel B2U.

```
Start Drone
    [Documentation]        Start drone by uavapp api
    [Arguments]                        ${name}  ${tunnel_name}  ${route}
    ${route}=  convert to integer  ${route}
    uavapp.uav_start    ${name}  ${tunnel_name}  ${route}
```

Figura 44: *Keyword* utilizada para a configuração de um túnel B2U.

Na figura 45 são ilustradas as fases do *Ready Signal* e *Alive Check*, o processo é independente do protocolo de encaminhamento seleccionado.

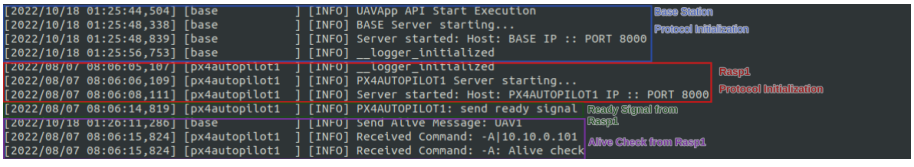
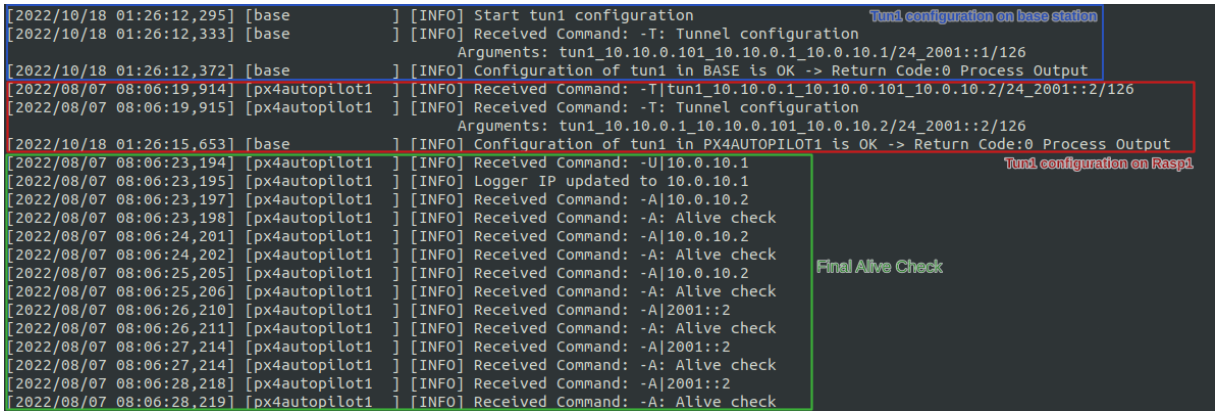
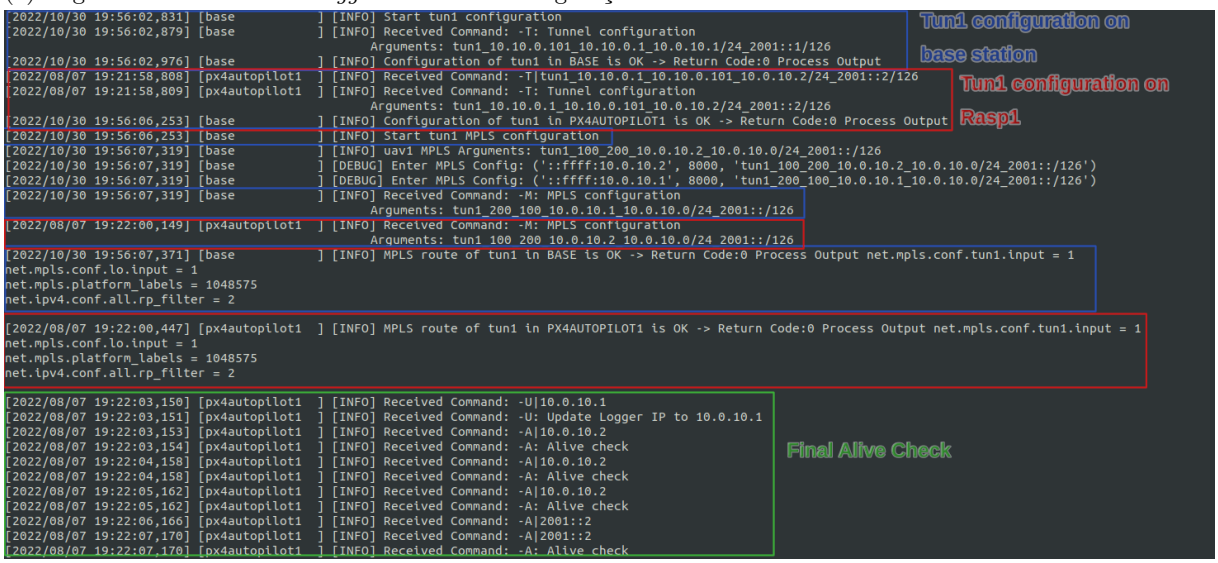


Figura 45: Registos do *UAVProto - Logger* das fases de *Ready Signal* e *Alive Check*

Logo depois das primeiras fases é executada a fase de configuração do túnel B2U ilustrado na figura 46, nas vertentes de encaminhamento IP e MPLS.



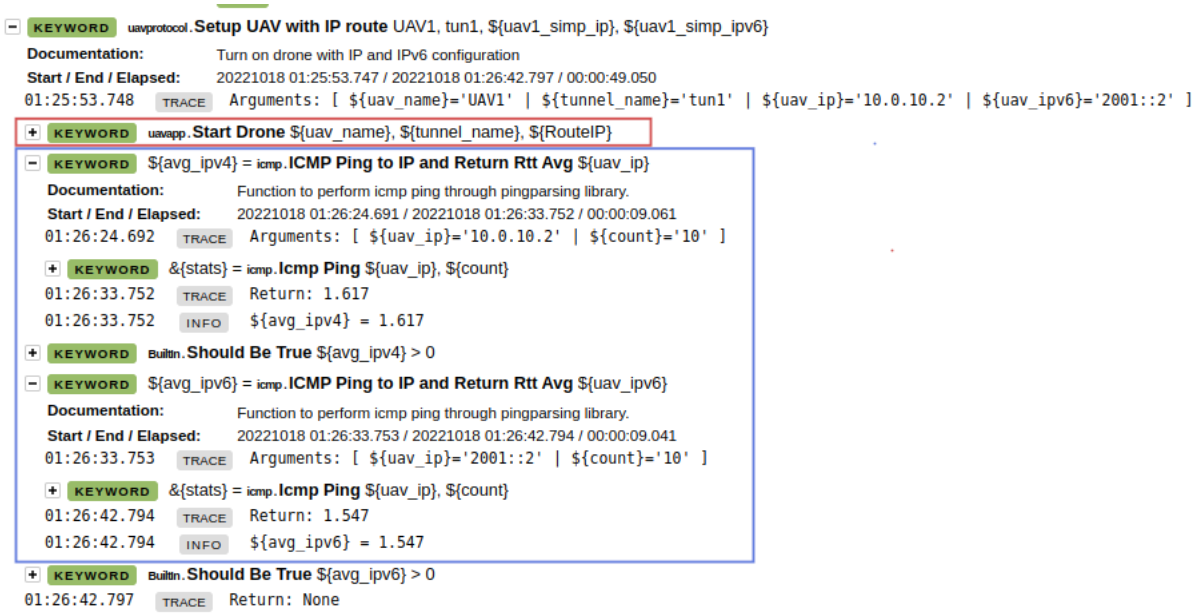
(a) Registos do *UAVProto - Logger* da fase de configuração do túnel B2U com encaminhamento IP



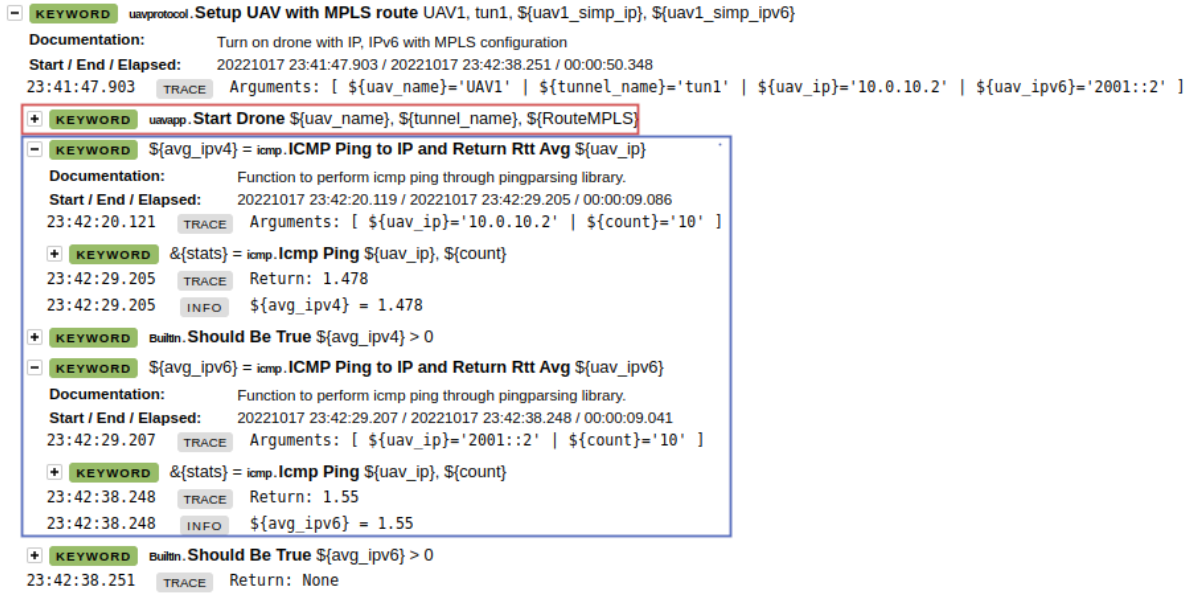
(b) Registos do *UAVProto - Logger* da fase de configuração do túnel B2U com encaminhamento MPLS

Figura 46: Registos do *UAVProto - Logger* da fase de configuração do túnel B2U

No final desta configuração cada teste executa uma *keyword* que executa um breve ping para validar a correcta configuração do túnel B2U. Os resultados desse ping estão ilustrados na figura 47 sendo que, a região a vermelho corresponde à *keyword* apresentada acima e a azul os resultados de confirmação da ligação entre a *base station* e o *Rasp1*.



(a) Confirmação da conexão do túnel B2U entre a *base station* e o *Rasp1* por IP



(b) Confirmação da conexão do túnel B2U entre a *base station* e o *Rasp1* por MPLS

Figura 47: Confirmação da conexão do túnel B2U entre a *base station* e o *Rasp1*

5.3.1.2 Configuração UAV-to-UAV

Os teste *Configure tunnel with UAV1 and UAV2*, *Configure tunnel with UAV1 and UAV3* e *Configure tunnel with UAV2 and UAV3* são testes de configuração do túnel U2U. Todos eles executam a mesma *keyword* de configuração mudando apenas os parâmetros de entrada da função que correspondem aos nomes dos UAVs,

por exemplo UAV1 e UAV2. A *keyword Create UAVs Tunnel*, que é executada na configuração dos túneis U2U é ilustrada na figura 48.

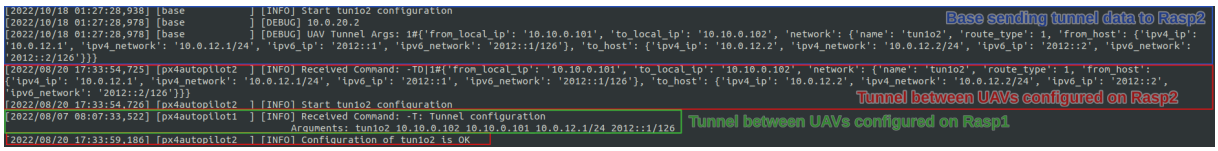
```
Create UAVs Tunnel
    [Documentation]         Create ip tunnel between drones.
    [Arguments]                 ${name}  ${from_uav}  ${to_uav}  ${route}
    ${route}=  convert to integer  ${route}
    uavapp.start_uavs_tunnel    ${name}  ${from_uav}  ${to_uav}  ${route}
```

Figura 48: *Keyword* utilizada para a configuração de um túnel U2U.

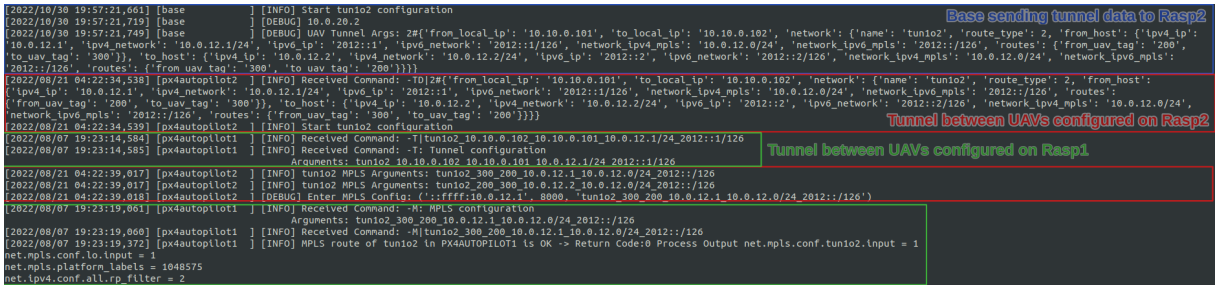
Na execução da *keyword Create UAVs Tunnel*, ilustrada na figura 48, que interage com a *uavapp_api* para garantir a execução das fases de detecção de má qualidade de sinal, ilustrada na figura 49, a criação do do túnel GRE entre os UAVs, ilustrada na figura 50, a configuração de novas rotas de encaminhamento, ilustrada na figura 51, e por fim, a remoção de túneis e rotas que não são mais necessários, ilustrada na figura 52. O processo ilustrado nas imagens é referente à configuração do túnel entre o UAV1 (Rasp1) e UAV2 (Rasp2).

```
[2022/10/18 01:27:28,934] [base           ] [INFO] Send bad signal connection to uav2
[2022/08/20 17:33:54,680] [px4autopilot2  ] [INFO] Received Command: -B|uav2
[2022/10/18 01:27:28,937] [base           ] [INFO] Got bad connection in uav2 ask configurations for new tunnel: -C
```

Figura 49: Registos do *UAVProto - Logger* da fase de configuração do túnel U2U - deteção de má qualidade do sinal.

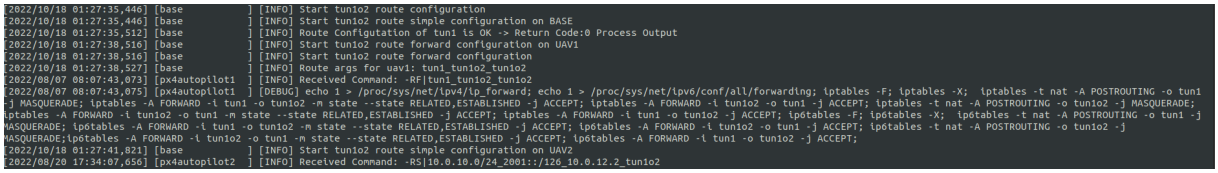


(a) Registos do *UAVProto - Logger* da fase de configuração do túnel U2U com encaminhamento IP

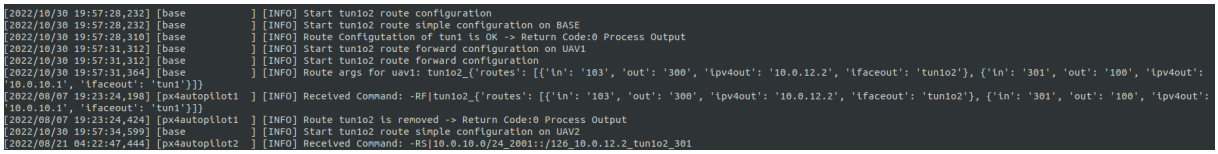


(b) Registos do *UAVProto - Logger* da fase de configuração do túnel U2U com encaminhamento MPLS

Figura 50: Registos do *UAVProto - Logger* da fase de configuração do túnel U2U



(a) Registos do *UAVProto - Logger* da fase de configuração de rotas IP do túnel U2U - tun1o2



(b) Registos do *UAVProto - Logger* da fase de configuração de rotas MPLS do túnel U2U - tun1o2

Figura 51: Registos do *UAVProto - Logger* da fase de configuração de rotas do túnel U2U - tun1o2

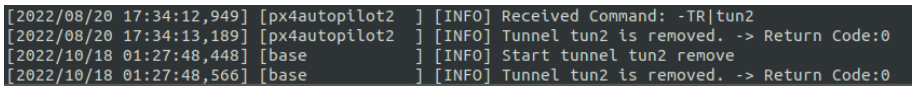
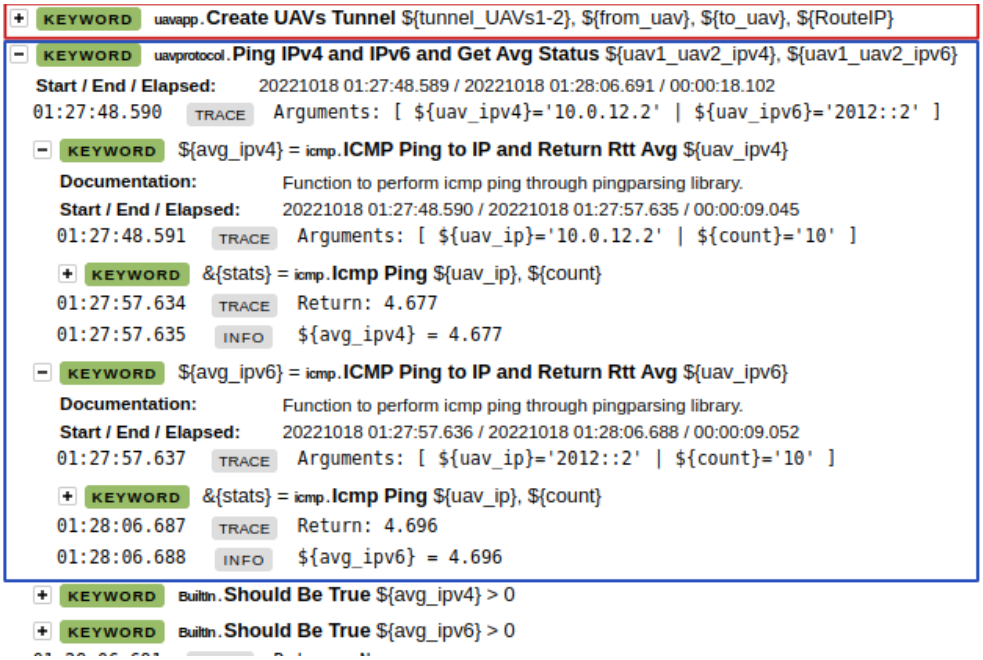
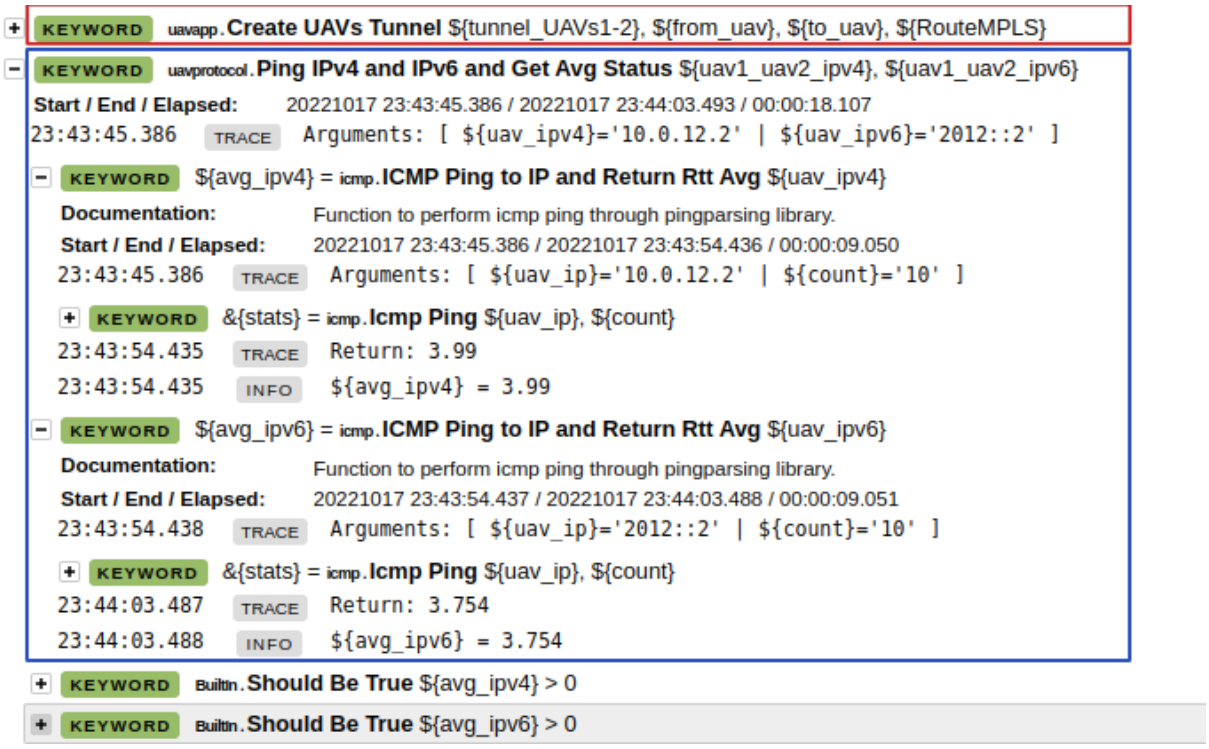


Figura 52: Registos do *UAVProto - Logger* da fase de remoção do túnel tun2 que será substutuído pelo tun1o2.

O mesmo processo é executado para as ligações U2U entre UAV1 e UAV3 (Rasp3) e para a versão final entre UAV2 e UAV3. No final desta configuração, tal como na configuração B2U, cada teste executa um comando ping para o UAV do final da cadeia, indicado no título do teste, para validar a correcta configuração do túnel U2U. Nas figuras 53 e 54 são apresentados os resultados do ping comprovativo da ligações U2U sendo que, a região a vermelho corresponde às *keywords* apresentadas e a azul os resultados de confirmação da ligação entre os diversos UAVs.

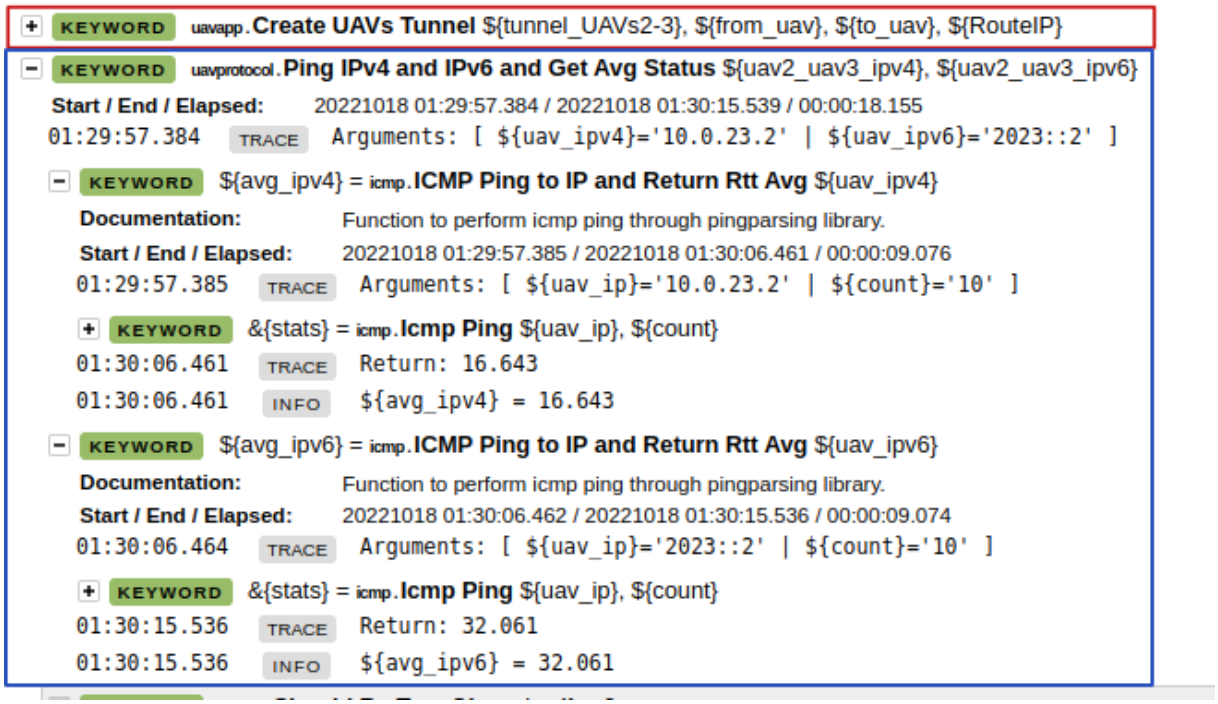


(a) Confirmação da conexão do túnel U2U entre o*Rasp1* e o *Rasp2* por IP

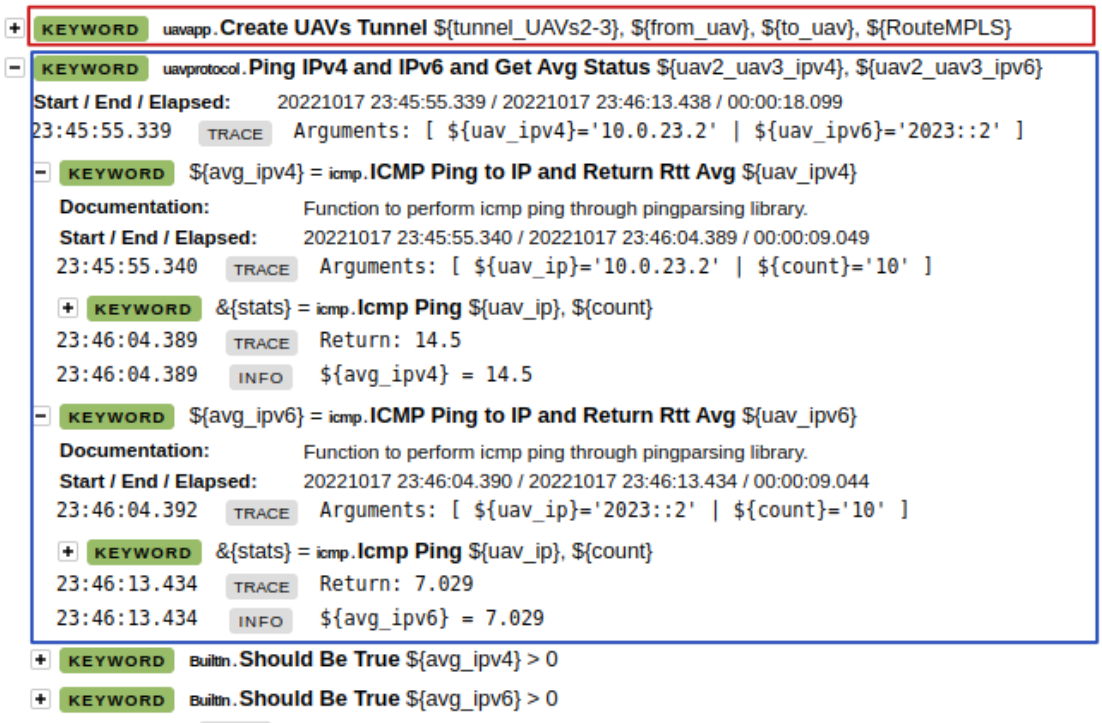


(b) Confirmação da conexão do túnel U2U entre o*Rasp1* e o *Rasp2* por MPLS

Figura 53: Confirmação da conexão do túnel U2U entre o*Rasp1* e o *Rasp2*



(a) Confirmação da conexão do túnel U2U entre o*Rasp2* e o *Rasp3* por IP



(b) Confirmação da conexão do túnel U2U entre o*Rasp2* e o *Rasp3* por MPLS

Figura 54: Confirmação da conexão do túnel U2U entre o*Rasp2* e o *Rasp3*

5.3.2 Conectividade end-to-end

Com este teste pretende-se validar a conectividade ao longo da cadeia, desde a base até ao UAV mais distante. Uma das ferramentas capazes de fazer esse teste de conectividade é o comando ping que é uma aplicação de rede utilizada para verificar se um nó solicitado é alcançado através de uma rede IP. O comando calcula o *Round Trip Time* do pedido enviado desde o *node* de origem até ao *node* de destino. O ping é parte do protocolo *Internet Control Message Protocol* (ICMP) na camada de rede. O ICMP gera um pacote de pedido de eco (ping) com IP e cabeçalho ICMP e espera pela resposta de eco do nó de destino [58].

Os testes automáticos que foram especificados para obterem os valores médios do RTT, e gerados os gráficos baseados nesses mesmos valores. Nesse sentido os testes foram desenhados tendo em conta o parâmetro, ***Count*** - representado pela opção *-c* representa o número de pacotes IP que são enviados pelo comando ping.

Uma vez que o protocolo *UAVProtocol* é implementado em duas versões, IP e também MPLS, os testes de conectividade foram divididos nesses modos com endereços IPv4 e IPv6.

Todos os testes foram executados em conta os parâmetros apresentados na figura 55.

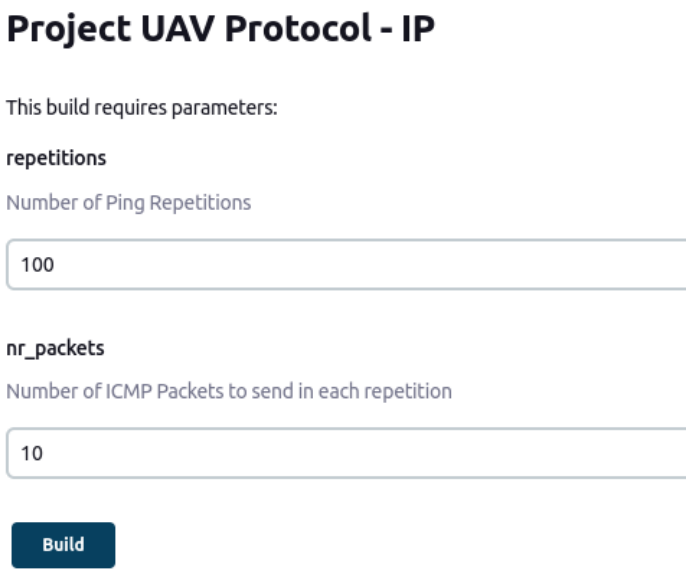


Figura 55: Parâmetros utilizados nos testes automáticos.

Na figura 56 é representado o teste base da categoria indicada na mesma figura no entanto, para todas as outras vertentes o teste é desenhado da mesma forma.

```
Performance Test - ICMP Ping - IPv4
    [Template]    Simple Ping To ${host} ${ip} - ${variant} with Load ${load}
    [Tags]        IP  IPv4
    UAV3      ${UAV3_IP}  ${Test Tags}[1]   0
```

Figura 56: *Test Case* base para a vertenteIP com endereço IPv4

5.3.2.1 IP

Nesta secção serão apresentados os testes automáticos feitos no modo de encaminhamento IP com endereços IPv4 e IPv6 e apresentados os respetivos resultados obtidos através da repetição da operação do envio de comando ping que envia 10 pacotes IP e desse comando são obtidos os valores médios de cada execução. Os resultados do teste são repetido com IPv4 e é representado na figura 57, e com IPv6 é apresentado na figura 58.

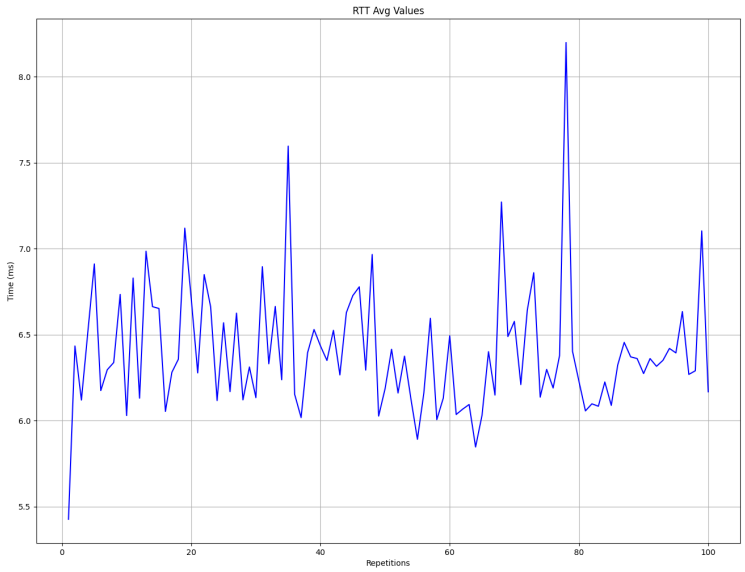


Figura 57: Variação do valor médio do RTT com o número de repetições do comando ping na vertente de IP com endereço IPv4

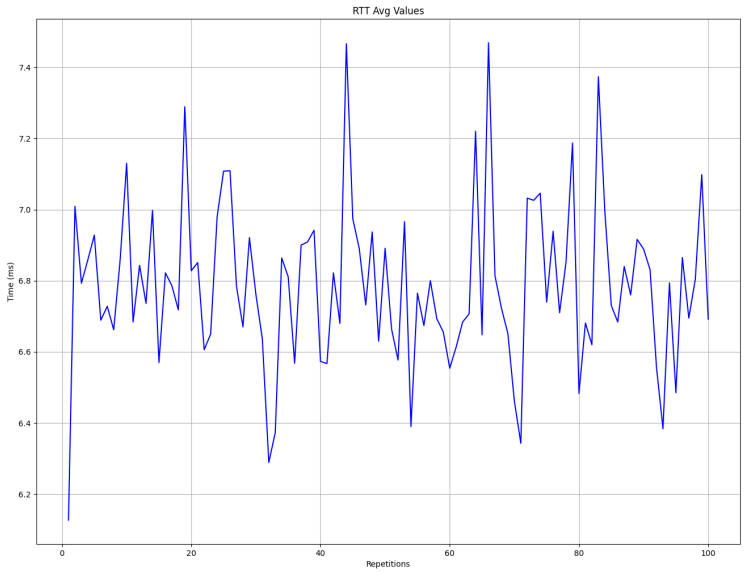


Figura 58: Variação do valor médio do RTT com o número de repetições do comando ping na vertente de IP com endereço IPv6

5.3.2.2 MPLS

Nesta secção serão apresentados os testes automáticos feitos no modo de encaminhamento MPLS com endereços IPv4 e IPv6 e apresentados os respetivos resultados obtidos. Tal como no encaminhamento IP pretende-se validar que todos os a conexão *end-to-end* e o encaminhamento MPLS através do mapeamento dos endereços IP, IPv4 ou IPv6. Desse ponto pretende-se avaliar se os pacotes IP criados pelo comando ping e contidos num FEC, atravessam todo o LSP configurado na secção 5.3.1.2. A partir desse ponto são executados as várias repetições do comando ping como descrito anteriormente. Os resultados do teste são repetidos com o endereço IPv4 e é representado na figura 59, e com endereço IPv6 é apresentado na figura 60.

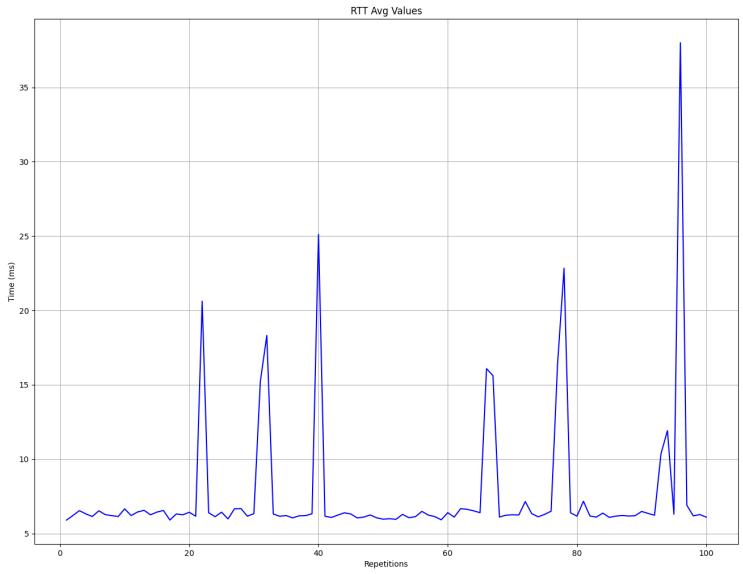


Figura 59: Variação do valor médio do RTT com o número de repetições do comando ping na vertente de MPLS com endereço IPv4

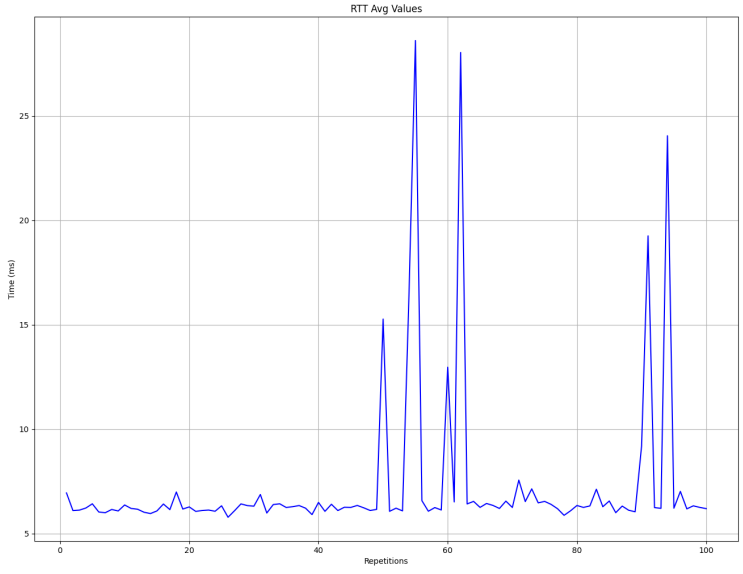


Figura 60: Variação do valor médio do RTT com o número de repetições do comando ping na vertente de MPLS com endereço IPv6

5.3.3 Análise

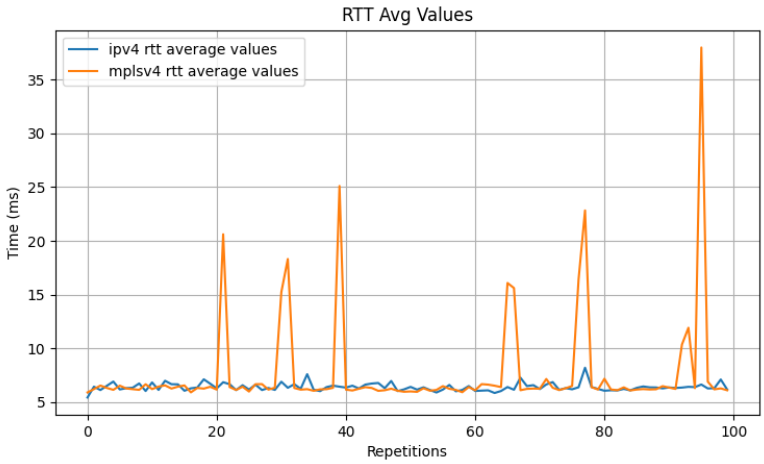


Figura 61: Comparação dos valores médios valor médio do RTT entre testes MPLSv4 e IPv4

Apesar de os valores estarem numa gama dentro do expectável não é possível retirar uma conclusão positiva da sua execução em contexto da comparação dos valores de IPv4 com MPLS com endereçamento IPv4, como se pode verificar na figura 61 pelos valores de RTT semelhantes, sempre dentro da mesma gama, salvo algumas janelas de erro, em que os valores MPLS com endereço saíram fora do normal.

Era expectável que, por se tratar de uma tecnologia de operador, de nível mais baixo ao nível das camadas protocolares, e também com um menor *overhead*, o MPLS apresentasse uma diferença mais significativa na ordem de 1 ou 2 ms. Tal efeito, pode estar relacionado com a dimensão do cenário de testes considerado, tanto em número de *hops* como de distâncias entre os UAVs nestes testes em específico.

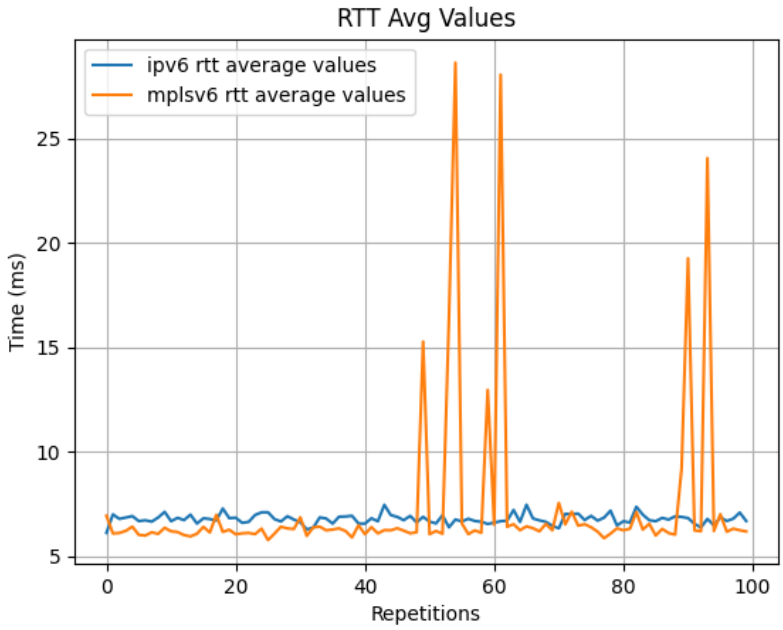


Figura 62: Comparação dos valores médios valor médio do RTT entre testes MPLSv6 e IPv6

Já quanto ao comparativo de valores de IPv6 com MPLS com endereçamento IPv6, é possível visualizar, no gráfico da figura 62, valores de acordo com o que é expectável.

5.3.4 Testes de stress

Para além de testar a conectividade é necessário também testar como protocolo reage a situações de *stress* do sistema, verificando se a conectividade se mantém estável e sem falhas mesmo nas situações mais críticas. Tal como, nos testes de conectividade, uma das ferramentas capazes de fazer esse teste de *stress* é o comando ping, neste caso pode-se fazer que sejam enviados pacotes IP, com mais carga em cada um de modo a conseguir-mos gerar maior tráfego na rede. No comando ping conseguimos fazer alteração, variando o parâmetro: ***Size*** - representado pela opção *-s* indica o número de *data bytes* que irá em cada pacote enviado para a rede.

No teste indicado na figura 63, é desenhado o teste com a capacidade máxima de tamanho do pacote que suportada pelo comando ping, 65$KBytes$.

```
Performance Test - ICMP Ping - IPv4
    [Template]    Simple Ping To ${host} ${ip} - ${variant} with Load ${load}
    [Tags]        IP  IPv4
    UAV3      ${UAV3_IP}  ${Test Tags}[1]   ${packet_size_65k}
```

Figura 63: *Test Case* de stress base com endereço IPv4

5.3.4.1 IP

O teste da vertente IP é idêntico ao teste desenhado secção IP dos testes de conectividade. Neste teste apenas é alterado o valor do tamanho de cada pacote IP enviado pelo comando ping, neste caso de 65$KBytes$.

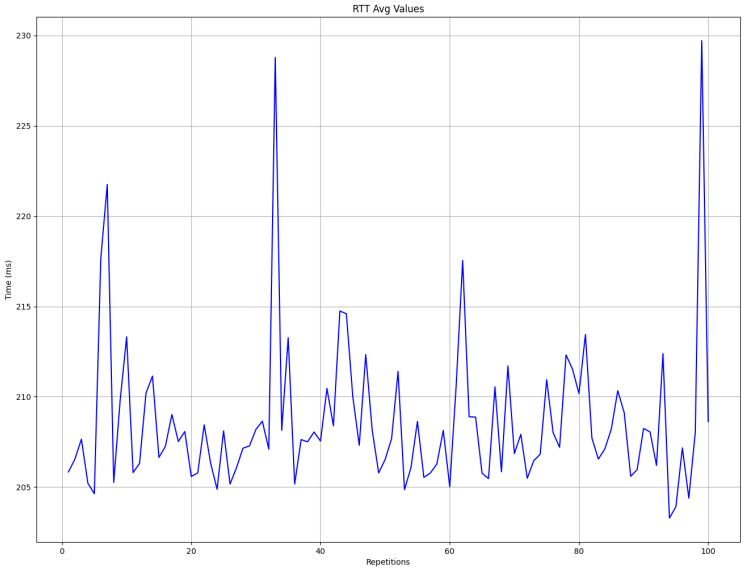


Figura 64: Variação do valor médio do RTT com o número de repetições do comando ping com *packet size* de 65$KBytes$ com endereço IPv4

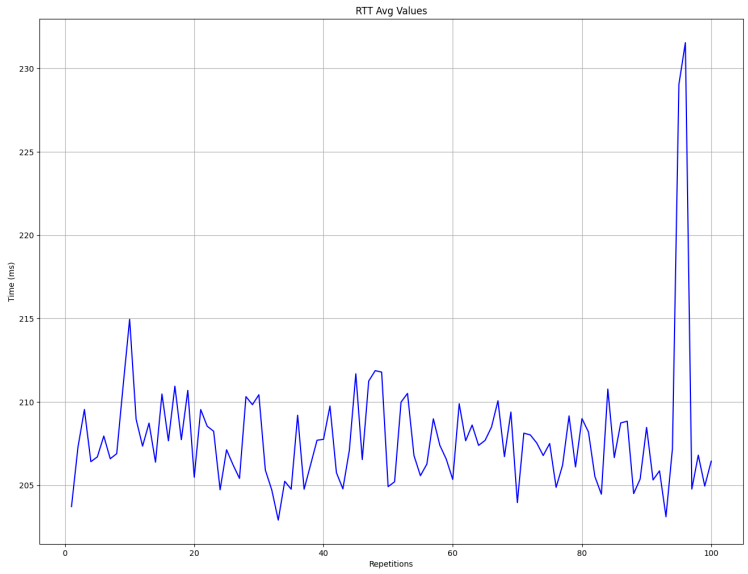


Figura 65: Variação do valor médio do RTT com o número de repetições do comando ping com *packet size* de 65*KBytes* com endereço IPv6

5.3.4.2 MPLS

O teste da vertente MPLS é idêntico ao teste desenhado secção MPLS dos testes de conectividade. Tal como nos testes IP, neste teste apenas é alterado o valor do tamanho de cada pacote IP enviado pelo comando ping, neste caso de 65*KBytes*.

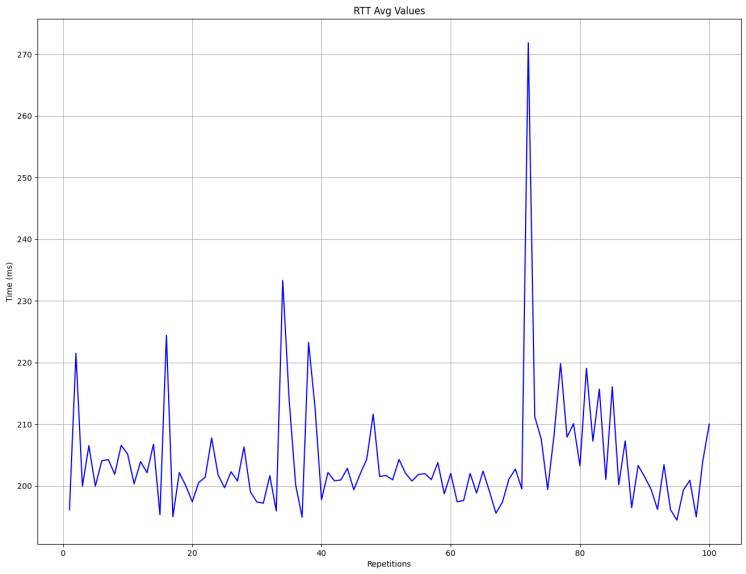


Figura 66: Variação do valor médio do RTT com o número de repetições do comando ping com *packet size* de 65*KBytes* na vertente de MPLS com endereço IPv4

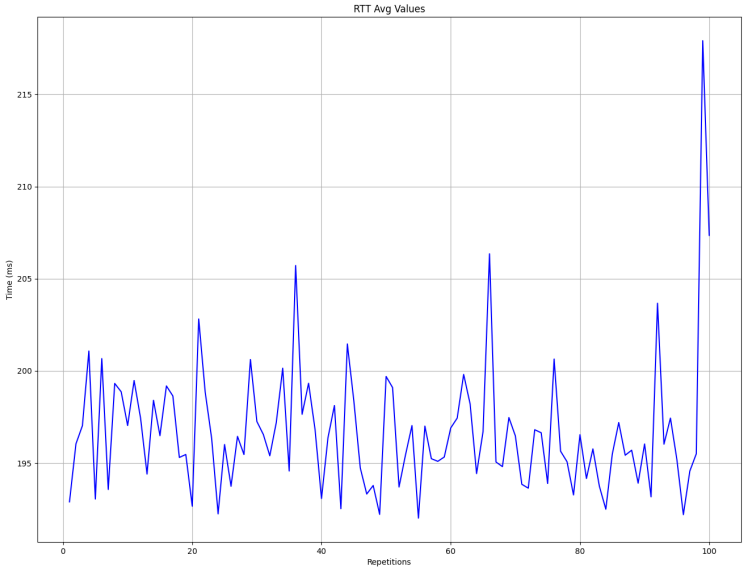


Figura 67: Variação do valor médio do RTT com o número de repetições do comando ping com *packet size* de 65*KBytes* na vertente de MPLS com endereço IPv6

5.3.5 Análise

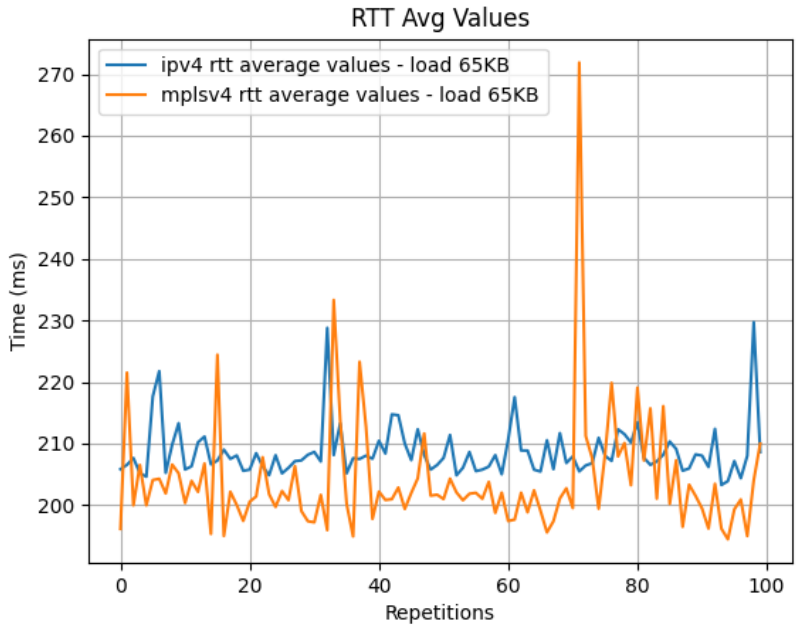


Figura 68: Comparação dos valores médios valor médio do RTT entre testes MPLSv4 e IPv4 com *packet size* de 65*KBytes*

Analisando o gráfico comparativo de valores IPv4 e MPLS com endereçamento IPv4 ilustrado no gráfico da figura 68, nota-se que há alguma evolução relativamente aos testes de conectividade. Ou seja, percebe-se que o MPLS é mais vantajoso quando se aumenta o tráfego na rede.

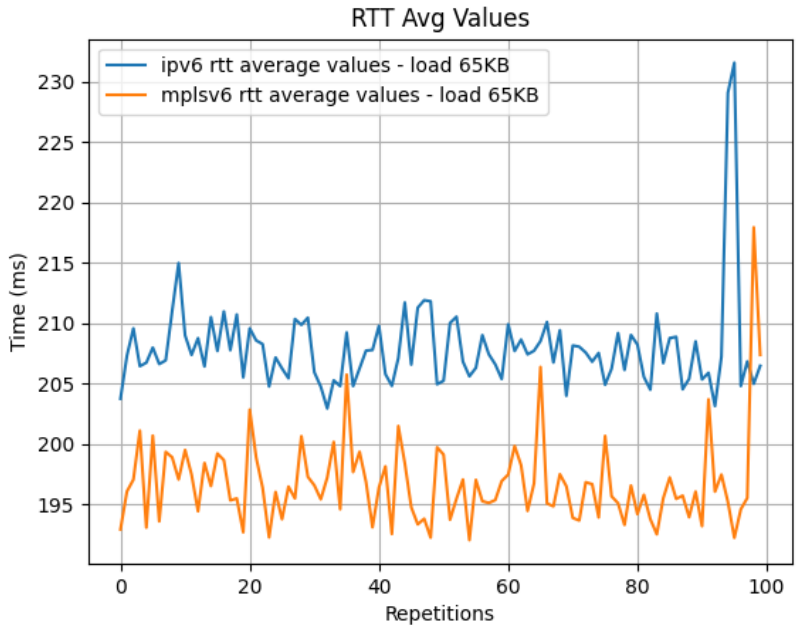


Figura 69: Comparação dos valores médios valor médio do RTT entre testes MPLSv6 e IPv6 com *packet size* de 65*KBytes*

Mais ainda se tivermos em conta o gráfico comparativo de valores IPv6 e MPLS com endereçamento IPv6 ilustrado no gráfico da figura 69, em que se prova que a diferença entre ambos os protocolos é bastante significativa. Consegue-se validar também, que mesmo em condições de uma maior carga de rede garante-se que não há falhas de comunicação.

5.4 Sumário

Este capítulo 5 aborda os testes de configuração feitos ao longo da configuração do protocolo *UAVProtocol*.

Na secção 5.1 são apresentados os propósitos dos testes realizados nas secções seguintes. Na secção 5.2 é apresentado o *setup*, ou ambiente, em que estes testes foram executados.

Na secção 5.3 são especificados os testes de configuração do protocolo e também os testes de conectividade do mesmo e os testes de stress feitos à rede gerada pelo protocolo *UAVProtocol*. São também apresentados os gráficos dos valores resultantes dos testes automáticos feitos ao sistema e feita uma análise crítica desses valores.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusões Finais

O protocolo *UAVProtocol* provou ser uma alternativa válida aos protocolos de comunicação entre UAVs e *base station*. De acordo com os resultados apresentados o *UAVProtocol* é de fácil e rápida configuração como podemos verificar nos testes de configuração contudo, há ainda um longo caminho a percorrer para a implementação de forma mais sustentada uma vez que nem todos os processos estão automatizados, nomeadamente o processo de detecção de ausência de sinal. No entanto, as aplicações de apoio desenvolvidas, como a *UAVApp* e a sua API, revelaram ser de grande utilidade no processo de teste facilitando muitas vezes o processo manual de *copy-paste* de *scripts* que foram descobertos ao longo da pesquisa para este trabalho.

Os resultados finais apesar na situação do IPv4 sem carga não apresentar resultados expectaveis, que era uma maior velocidade de encaminhamento do protocolo MPLS em relação ao IP. Conseguiu nas situações com maior carga da ligação, que o MPLS, seja com endereçamento IPv4 ou IPv6, garantir uma maior eficiência de encaminhamento nessa situação revelando-se uma vantagem em cenários que assim o exijam.

Por fim, é também a conclusão de muitos meses de pesquisa e trabalho que culminam na aprendizagem de novas metodologias e protocolos de rede existentes.

6.2 Trabalho Futuro

No futuro podem ainda ser desenvolvidas algumas melhorias tanto a nível de *hardware* como *software*. No *hardware* uma das melhorias será implementar toda esta solução em ambiente real com UAVs reais e testar o seu comportamento nos mais diversos cenários e distâncias de comunicação e estudo de energia gasta com o protocolo.

No *software* para além de melhorar a gestão de controlo da qualidade do sinal de Wi-Fi, fazer também a gestão energética das baterias dos UAVs implementando mecanismos de substituição de UAVs em voo sem perdermos todas as configurações de rede existentes nos outros equipamentos da rede.

Referências

[1] J. Jiang and G. Han, "Routing Protocols for Unmanned Aerial Vehicles," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 58–63, 2018. [Citado na página 1]

[2] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019. [Citado na página 1]

[3] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019. [Citado na página 1]

[4] G. Secinti, P. B. Darian, B. Canberk, and K. R. Chowdhury, "SDNs in the Sky: Robust End-to-End Connectivity for Aerial Vehicular Networks," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 16–21, 2018. [Citado na página 1]

[5] S. Morgenthaler, T. Braun, Z. Zhao, T. Staub, and M. Anwander, "UAVNet: A mobile wireless mesh network using Unmanned Aerial Vehicles," in *2012 IEEE Globecom Workshops*, pp. 1603–1608, Dec. 2012. [Citado nas páginas 2 e 8]

[6] T. D. Nadeau, ed., *MPLS network management: MIBs, tools, and techniques*. Atlanta, GA, USA: Elsevier, 2003. [Citado na página 2]

[7] S. Alouneh, S. Abed, M. Kharbutli, and B. J. Mohd, "MPLS technology in wireless networks," *Wireless Networks*, vol. 20, no. 5, pp. 1037–1051, 2014. [Citado na página 2]

[8] M. Brunner and J. Quittek, "MPLS management using policies," in *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings. Integrated Network Management VII. Integrated Management Strategies for the New Millennium (Cat. No.01EX470)*, pp. 515–528, May 2001. [Citado na página 2]

[9] I. Jawhar, N. Mohamed, J. Al-Jaroodi, D. P. Agrawal, and S. Zhang, "Communication and networking of UAV-based systems: Classification and associated architectures," *Journal of Network and Computer Applications*, vol. 84, pp. 93–108, 2017. [Citado nas páginas 6, 10 e 11]

[10] W. Shi, H. Zhou, J. Li, W. Xu, N. Zhang, and X. Shen, "Drone Assisted Vehicular Networks: Architecture, Challenges and Opportunities," *IEEE Network*, vol. 32, no. 3, pp. 1–8, 2018. [Citado nas páginas 6 e 7]

[11] A. G. Madey and G. R. Madey, "Design and evaluation of UAV swarm command and control strategies," in *SpringSim*, p. 1–8, Apr. 2013. [Citado na página 6]

[12] L. He, P. Bai, X. Liang, J. Zhang, and W. Wang, "Feedback formation control of UAV swarm with multiple implicit leaders," *Aerospace Science and Technology*, vol. 72, pp. 327–334, 2018. [Citado na página 7]

[13] D. Morelo, "NetSpot Pro - Rede WiFi Mesh e como ela pode melhorar a sua conectividade." Available at `https://www.netspotapp.com/pt/what-is-mesh-networking.html`, Nov. 2021. (Last accessed in 12/12/2021). [Citado nas páginas ix e 7]

[14] A. L. Dallora Moraes, A. F. dos Santos Xaud, and M. F. dos Santos Xaud, "Redes Ad Hoc - Protocolos." Available at `https://www.gta.ufrj.br/grad/09_1/versao-final/adhoc/redesadhoc.html`, 2007. (Last accessed in 08/07/2018). [Citado na página 8]

[15] M. Ricardo, "Redes Ad-Hoc." Available at `https://web.fe.up.pt/~mricardo/05_06/cm/acetatos/adhocv3.pdf`, 2005. (Last accessed in 11/07/2018). [Citado na página 9]

[16] İlker Bekmezci, O. K. Sahingoz, and Şamil Temel, "Flying Ad-Hoc Networks (FANETs): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013. [Citado na página 9]

[17] T. X. Brown, B. Argrow, C. Dixon, S. Doshi, R.-G. Thekkekunel, and D. Henkel, "Ad Hoc UAV Ground Network (AUGNet)," in *AIAA 3rd "Unmanned Unlimited"Technical Conference, Workshop and Exhibit*, (Chicago, Illinois), Sept. 2004. [Citado na página 9]

[18] V. P. Subba Rao and G. S. Rao, "Design and Modelling of anAffordable UAV Based Pesticide Sprayer in Agriculture Applications," in *2019 Fifth International Conference on Electrical Energy Systems (ICEES)*, (Chennai, India), pp. 1–4, Feb. 2019. [Citado na página 11]

[19] X. Huang, S. Zhang, C. Luo, W. Li, and Y. Liao, "Design and Experimentation of an Aerial Seeding System for Rapeseed Based on an Air-Assisted Centralized Metering Device and a Multi-Rotor Crop Protection UAV," *Applied Sciences*, vol. 10, no. 24, p. 14, 2020. [Citado na página 11]

[20] ArduPilot Dev Team, "ArduPilot - ArduPilot Documentation." Available at `http://ardupilot.org/ardupilot/index.html`. (Last accessed in 16/01/2019). [Citado nas páginas ix e 12]

[21] Dronecode Project, Inc., a Linux Foundation Collaborative Project, "Dronecode Foundation - Leading open-source components for UAVs." Available at `https://www.dronecode.org/projects/`. (Last accessed in 12/10/2022). [Citado na página 13]

[22] Dronecode Project, Inc., a Linux Foundation Collaborative Project, "MAVLink Developer Guide." Available at `https://mavlink.io/en/`. (Last accessed in 12/10/2022). [Citado na página 13]

[23] Dronecode Project, Inc., a Linux Foundation Collaborative Project, "Introduction - MAVSDK Guide." Available at `https://mavsdk.mavlink.io/main/en/index.html`. (Last accessed in 12/10/2022). [Citado na página 13]

[24] Dronecode Project, Inc., a Linux Foundation Collaborative Project, "Overview - QGroundControl User Guide." Available at `https://docs.qgroundcontrol.com/master/en/`. (Last accessed in 12/10/2022). [Citado nas páginas ix, 13 e 14]

[25] SPH Engineering, "Shop - SPH Engineering." Available at `https://shop.ugcs.com/`. (Last accessed in 13/10/2022). [Citado na página 15]

[26] SPH Engineering, "Ground Station Software - UgCS PC Mission Planning." Available at `https://www.ugcs.com/`. (Last accessed in 13/10/2022). [Citado nas páginas ix e 15]

[27] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, "Study on ZigBee technology," *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, vol. 6, pp. 297–301, 2011. [Citado nas páginas ix e 16]

[28] L. Liang, L. Huang, X. Jiang, and Y. Yao, "Design and implementation of wireless Smart-home sensor network based on ZigBee protocol," *2008 International Conference on Communications, Circuits and Systems Proceedings, ICCCAS 2008*, pp. 434–438, 2008. [Citado na página 16]

[29] Y. Wang, C. Chen, and Q. Jiang, "Security algorithm of Internet of Things based on ZigBee protocol," *Cluster Computing*, vol. 22, pp. 14759–14766, 2019. [Citado na página 16]

[30] P. Šuľaj, R. Haluška, Ľuboš Ovseník, S. Marchevský, P. Pulli, and V. Kramar, "UAV Management System for the Smart City," Tech. Rep. 062TUKE-4/2017 and 023TUKE-4/2017, Technical University of Košice, Faculty of Information Technology and Electrical Engineering, University of Oulu and Oulu University

of Applied Sciences, Košice, Slovak Republic and Oulu, Finland, Sept. 2018. [Citado nas páginas ix e 17]

[31] H. Li, F. Chen, H. Shabani, M. M. Ahmed, S. Khan, and al, "Unmanned Patrol System Based on Kalman Filter and ZigBee Positioning Technology," *Journal of Physics: Conference Series*, vol. 1168, p. 32063, 2019. [Citado na página 17]

[32] R. Costa, J. Lau, P. Portugal, F. Vasques, and R. Moraes, "Handling real-time communication in infrastructured IEEE 802.11 wireless networks: The RT-WiFi approach," *Journal of Communications and Networks*, vol. 21, pp. 319–334, 2019. [Citado na página 17]

[33] A. Z. Yonis, "Performance analysis of IEEE 802.11ac based WLAN in wireless communication systems," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, pp. 1131–1136, 2019. [Citado nas páginas xiii, 17 e 18]

[34] E. Grossi, M. Lops, and L. Venturino, "Adaptive Detection and Localization Exploiting the IEEE 802.11ad Standard," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 4394–4407, 2020. [Citado na página 17]

[35] A. Garcia-Rodriguez, D. Lopez-Perez, L. Galati-Giordano, and G. Geraci, "IEEE 802.11be: Wi-Fi 7 Strikes Back," *IEEE Communications Magazine*, vol. 59, pp. 102–108, 2021. [Citado na página 17]

[36] Y. Gao, X. Sun, and L. Dai, "Sum Rate Optimization of Multi-Standard IEEE 802.11 WLANs," *IEEE Transactions on Communications*, vol. 67, pp. 3055–3068, 2019. [Citado nas páginas xiii e 18]

[37] A. F. Rochim, B. Harijadi, Y. P. Purbanugraha, S. Fuad, and K. A. Nugroho, "Performance comparison of wireless protocol IEEE 802.11ax vs 802.11ac," in *2020 International Conference on Smart Technology and Applications (ICoSTA)*, (Surabaya, Indonesia), pp. 1–5, Feb. 2020. [Citado nas páginas xiii e 18]

[38] K. Nguyen, M. G. Kibria, K. Ishizu, and F. Kojima, "Performance Evaluation of IEEE 802.11ad in Evolving Wi-Fi Networks," *Wireless Communications and Mobile Computing*, vol. 2019, p. 11, 2019. [Citado nas páginas xiii e 18]

[39] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A Tutorial on IEEE 802.11ax High Efficiency WLANs," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 197–216, 2019. [Citado nas páginas xiii e 18]

[40] H. Zemrane, Y. Baddi, and A. Hasbi, "Comparison between IOT protocols: ZigBee and WiFi using the OPNET simulator," in *Proceedings of the 12th International Conference on Intelligent Systems: Theories and Applications*, (New York, NY, USA), pp. 1–6, Oct. 2018. [Citado nas páginas ix, 18 e 19]

[41] M. Hendaoui and K. Kahoul, "Comparative Study Between The IEEE 802.16 and The IEEE 802.11 Using OPNET," *SSRN Electronic Journal*, p. 6, 19. [Citado nas páginas 18 e 19]

[42] M. A. Khan, R. Hamila, M. S. Kiranyaz, and M. Gabbouj, "A Novel UAV-Aided Network Architecture Using Wi-Fi Direct," *IEEE Access*, vol. 7, pp. 67305–67318, 2019. [Citado na página 19]

[43] Y.-C. Wu, H.-R. Chang, M.-S. Wu, C.-Y. Li, and K. Wang, "UAV-FAP: User Fairness-Driven Access Point on UAV for Wi-Fi Networks," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, (Las Vegas, NV, USA), pp. 473–476, Jan. 2022. [Citado na página 19]

[44] M. Sailan, R. Hassan, and A. Patel, "A comparative review of IPv4 and IPv6 for research test bed," in *2009 International Conference on Electrical Engineering and Informatics*, (Bangi, Malaysia), pp. 427–433, Aug. 2009. [Citado nas páginas ix e 20]

[45] R. Malhotra, ed., *IP routing*. California: O'Reilly Media, Inc., 2002. [Citado nas páginas 20 e 21]

[46] M. A. Ridwan, N. A. M. Radzi, W. S. H. M. Wan Ahmad, F. Abdullah, M. Jamaludin, and M. N. Zakaria, "Recent trends in MPLS networks: technologies, applications and challenges," *IET Communications*, vol. 14, no. 2, pp. 177–185, 2020. [Citado nas páginas ix e 21]

[47] L. Han, J. Wang, C. Wang, and L. Cai, "A Variable Forwarding Equivalence Class for MPLS Networks," in *2009 International Conference on Multimedia Information Networking and Security*, (Wuhan, China), pp. 273–276, Nov. 2009. [Citado nas páginas ix, 21 e 22]

[48] J. Gong, T. H. Chang, C. Shen, and X. Chen, "Flight time minimization of UAV for data collection over wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 1942–1954, 2018. [Citado na página 22]

[49] H. V. Abeywickrama, B. A. Jayawickrama, Y. He, and E. Dutkiewicz, "Comprehensive energy consumption model for unmanned aerial vehicles, based on empirical studies of battery performance," *IEEE Access*, vol. 6, pp. 58383–58394, 2018. [Citado nas páginas ix, xiii, 22, 23, 24 e 25]

[50] S. Jung, Y. Jo, and Y.-J. Kim, "Flight Time Estimation for Continuous Surveillance Missions Using a Multirotor UAV," *Energies*, vol. 12, no. 5, 2019. [Citado na página 25]

[51] M. T. R. Institute, "UAV Deployed Sensors | Michigan Tech Research Institute (MTRI)." Available at https://www.mtu.edu/mtri/research/project-areas/transportation/sensors-platforms/uav-deployed-sensors/, 2022. [Citado na página 34]

[52] P. S. Foundation, "Python.org - About Python™." Available at https://www.python.org/about/, 2001. (Last accessed in 20/09/2022). [Citado na página 39]

[53] P. S. Foundation, "logging — Logging facility for Python — Python 3.10.7 documentation." Available at https://docs.python.org/3/library/logging.html, 2001. (Last accessed in 20/09/2022). [Citado na página 41]

[54] jenkins.io, "Jenkins User Documentation." Available at https://www.jenkins.io/doc/. (Last accessed in 27/09/2022). [Citado nas páginas 43 e 55]

[55] R. F. Foundation, "Robot Framework User Guide - Version 5.0." Available at https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html, 2016. (Last accessed in 22/09/2022). [Citado na página 43]

[56] R. F. Foundation, "Robot Framework - Introduction." Available at https://robotframework.org/, 2016. (Last accessed in 22/09/2022). [Citado na página 43]

[57] D. E. Idrissi, N. Elkamoun, and R. Hilal, "Study of the impact of failure on GRE Tunnel," in *2019 7th Mediterranean Congress of Telecommunications (CMT)*, (Fez, Morocco), pp. 1–4, Oct. 2019. [Citado na página 44]

[58] P. Rajankumar, P. Nimisha, and P. Kamboj, "A comparative study and simulation of AODV MANET routing protocol in NS2 & NS3," in *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*, (New Delhi, India), pp. 889–894, Mar. 2014. [Citado na página 65]