João Paulo Bodanese

INFRAESTRUTURA DE COMUNICAÇÃO SEM FIO PARA UM VEÍCULO AÉREO NÃO TRIPULADO DE CURTO ALCANCE

Florianópolis (SC)

2014

João Paulo Bodanese

INFRAESTRUTURA DE COMUNICAÇÃO SEM FIO PARA UM VEÍCULO AÉREO NÃO TRIPULADO DE CURTO ALCANCE

Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Automação e Sistemas.
Orientador: Leandro Buss Becker, Prof. Dr. - PGEAS - UFSC
Coorientador: Gustavo Medeiros de Araújo, Prof. Dr. - UFSC

Florianópolis (SC)

2014

Ficha de identificação da obra elaborada pelo autor através do
Programa de Geração Automática da Biblioteca Universitária da UFSC.

A ficha de identificação é elaborada pelo próprio autor

Maiores informações em:
http://portalbu.ufsc.br/ficha

João Paulo Bodanese

INFRAESTRUTURA DE COMUNICAÇÃO SEM FIO PARA UM VEÍCULO AÉREO NÃO TRIPULADO DE CURTO ALCANCE

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Automação e Sistemas", e aprovada em sua forma final pela Universidade Federal de Santa Catarina.

Florianópolis (SC), 25 de março 2014.

Prof. Jomi Fred Hübner, Dr.
PGEAS - UFSC
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas

Banca Examinadora:

Leandro Buss Becker, Prof. Dr. - PGEAS - UFSC
Orientador

Gustavo Medeiros de Araújo, Prof. Dr. - UFSC
Coorientador

Prof. Leandro Buss Becker, Dr.
PGEAS - UFSC

Prof. Marcelo Maia Sobral, Dr.
IFSC

Prof. Antônio Augusto Fröhlich, Dr.
PGEAS - UFSC

Prof. Marcelo Ricardo Stemmer, Dr.
PGEAS - UFSC

Dedico este trabalho a vocês que sempre me fizeram acreditar na realização dos meus sonhos e trabalharam muito para que eu pudesse realizá-los, meus pais, Rudi e Suzana.

AGRADECIMENTOS

Aos meus pais, Rudi Bodanese e Suzana Gazzoni Bodanese, pelo amor, carinho, apoio e imensa dedicação em minha criação tanto pessoal quanto profissional.

Ao meu irmão, Mateus Bodanese, pelo companheirismo e incentivo em todos os momentos.

À minha namorada, Mayara Moreira de Mattos, melhor amiga e companheira de todas as horas, pelo amor e compreensão. Por me apoiar e incentivar nos momentos difíceis, por estar sempre comigo e me compreender da melhor forma possível.

Aos meus amigos, pela paciência e colaboração em todas etapas de minha jornada.

Aos alunos Vinicius Woyakewicz e Guilherme Interlandi que me auxiliaram nos experimentos práticos.

Ao meu orientador, Leandro Buss Becker, e coorientador, Gustavo Medeiros de Araújo, por suas orientações à execução deste trabalho.

Aos professores Jörg Kaiser e Christoph Steup da Universidade Otto Von Guericke de Magdeburg que muito contribuíram com ideias e sugestões durante a minha estadia na Alemanha.

Enfim, a todos aqueles que, de uma maneira ou de outra, contribuíram para que este percurso pudesse ser concluído.

O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.

José de Alencar

RESUMO

Essa dissertação descreve uma infraestrutura de comunicação sem fio projetada para uso em um Veículo Aéreo Não Tripulado (VANT) de pequena escala. A mesma tem como objetivo fornecer os *links* de comunicação entre o VANT e uma estação base (para fins de recebimento de comandos de controle) e entre o VANT e uma Rede de Sensores Sem Fio (RSSF) (para fins de coleta de dados). O desafio é utilizar o mesmo hardware para comunicar-se com dispositivos potencialmente heterogêneos e compartilhar diferentes protocolos e tipos de tráfego. O trabalho detalha a camada de software projetada para prover uma comunicação confiável com a estação de base, além de priorizar a economia de recursos com os nós de uma RSSF. Para ambos os casos, é usado apenas um *transceiver* IEEE 802.15.4 de longo alcance. Com o intuito de demonstrar a infraestrutura proposta, dois diferentes cenários foram implementados e testados. O primeiro consiste na comunicação do VANT com sensores esparsamente distribuídos sobre o solo. Já o segundo consiste na entrega dos dados coletados para a estação base. Ambos os cenários de comunicação diferem-se em tempo de conexão e outros requisitos. Os resultados obtidos mostram que a solução proposta atende os desafios relacionados.

Palavras-chave: Veículo Aéreo Não Tripulado, Rede de Sensores sem Fio, Mobilidade, Arquitetura de Comunicação

ABSTRACT

This paper describes a wireless communication infrastructure designed for a small-scale Unmanned Aerial Vehicle (UAV). It aims at providing a communication link between the UAV and a base station and also additional links that may allow collecting data from a remote Wireless Sensor Network (WSN). The challenge here is using the same hardware to communicate with potential heterogeneous devices and using different protocols. The paper details the software layer designed to support a reliable communication with the base station, as well a highly efficient best-effort communication with nodes from a WSN. For both cases, a single IEEE 802.15.4 extended range transceiver is used. To demonstrate the proposed infrastructure, two different utilization scenarios were implemented and tested. The first consists of the UAV node communicating with spatially distributed sensors on the ground. In the second scenario, the UAV node delivers the collected data to the base station. Both communication scenarios are different in terms of connection time and other communication requirements. Obtained results show that the proposed solution meets the related challenges.

Keywords: Unmanned Aerial Vehicle, Wireless Sensor Network, Mobility, Communication Architecture

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACK - *Acknowledgment*

CSMA - *Carrier Sense Multiple Access*

GPS - *Global Position System*

LQI - *Link Quality Indicator*

MAC - *Medium Access Control*

MANET - *Mobile Ad Hoc Network*

QoS - *Quality of Service*

RSSF - *Redes de Sensores Sem Fio*

RSSI - *Received Signal Strentgh Indicator*

UAV - *Unmaned Aerial Vehicle*

VANT - *Veículo Aéreo Não Tripulado*

WLAN - *Wireless Local Area Network*

WSN - *Wireless Sensor Network*

RTT - *Round-Trip Time*

RTO - *Retransmission Time-Out*

6LowPan - *IPv6 over Low power Wireless Personal Area Networks*

SUMÁRIO

1 INTRODUÇÃO

Os Veículos Aéreos Não Tripulados (VANTs) tornaram-se uma tecnologia chave no domínio militar e oferecem uma tecnologia promissora em muitas aplicações comerciais e civis. Como exemplo dessas aplicações, podem-se citar monitoração de ambientes, segurança pública, busca e resgate, inspeção remota e agricultura de precisão. Embora os VANTs autônomos não dependam dos *links* de comunicação com a estação base durante todo o voo, a comunicação é essencial para a realização de tarefas complexas de missão, nas quais o operador em terra é uma parte indispensável da missão. Os requisitos de comunicação do VANT, como a taxa de dados e o alcance, são determinados de acordo com a natureza de aplicação do mesmo.

Em muitas dessas aplicações, é possível imaginar o uso de VANTs como agentes móveis para comunicação com sensores de apoio espalhados pelo ambiente, onde tais sensores formam uma Rede de Sensores Sem Fio (RSSF). Essas aplicações têm atraído significativos interesses nos últimos anos. Em uma RSSF, os VANTs aumentam a capacidade da rede e permitem uma adaptação dinâmica às mudanças do ambiente [Banatre 2008].

Dois cenários de comunicação são vislumbrados no presente projeto de pesquisa. O primeiro consiste em estabelecer a comunicação com a estação base, onde são transmitidos os dados coletados da rede de sensores, os dados da missão de voo e as mensagens de controle. Já o segundo cenário refere-se propriamente à coleta dos dados da RSSF pelo VANT.

As propriedades de comunicação e a disponibilidade de recursos diferem-se entre os cenários. Na comunicação do VANT com a estação base, é necessário um conjunto de protocolos que ofereçam mecanismos de retransmissão, priorização de mensagens e segurança. Tal necessidade dá-se pela exigência de que a comunicação com a estação base seja confiável a fim de evitar possíveis perdas de dados, além de oferecer prioridade às mensagens específicas, tais como de controle e alarme e, finalmente, para que permita o acesso remoto do operador de qualquer ponto da internet de forma segura.

Por outro lado, a comunicação do VANT com a RSSF demanda um conjunto de protocolos que minimizem o consumo de recursos da rede, uma vez que os sensores da rede, usualmente, tem baixo poder de processamento, memória limitada e disponibilidade restrita de energia.

Nesse contexto, considerando que não existe solução de prateleira para o problema em questão, há a necessidade de se projetar uma infraestrutura de hardware e software para permitir a comunicação entre um VANT e uma estação de apoio em solo, assim como entre o VANT e uma RSSF.

1.1 MOTIVAÇÃO

O desenvolvimento do projeto é constituído por duas motivações, as quais consistem em fornecer uma infraestrutura de comunicação entre o VANT e a estação base, ambos em desenvolvimento pelo projeto PROVANT[1], e também aplicá-lo na coleta de dados de uma RSSF.

O PROVANT é um projeto em desenvolvimento no Departamento de Automação e Sistemas (DAS) na Universidade Federal de Santa Catarina (UFSC) e tem o objetivo de desenvolver uma aeronave de pequeno porte e curto alcance do tipo *tiltrotor*, capaz de voar de forma autônoma em trajetórias predefinidas. A intenção do projeto é usar a aeronave para coletar dados de uma RSSF situada supostamente na ilha do Arvoredo, a qual está localizada ao norte da Ilha de Santa Catarina.

Os nós sensores em questão deverão ser instalados em pontos estratégicos da ilha do Arvoredo e terão uma distância entre eles que possivelmente deve exceder algumas dezenas de metros. Tipicamente, a longa distância entre os sensores e a presença de obstáculos naturais (como árvores e pedras) levam à partição da RSSF em um conjunto de áreas isoladas de comunicação, também chamados *clusters* [Dios et al. 2013]. Tais *clusters* impossibilitam o roteamento dos dados pela rede, ao menos que nós adicionais sejam instalados em posições estratégicas. Entretanto, o difícil acesso à ilha dificulta a inserção dos mesmos. Desta forma, é proposto que sejam utilizados VANTs para a coleta de dados da RSSF, sem que haja a necessidade de nós de roteamento adicionais.

1.2 OBJETIVOS

Conforme já destacado, este trabalho propõe o projeto de uma infraestrutura de comunicação sem fio, voltada para uso em um VANT de pequena escala. O objetivo desta infraestrutura é fornecer um *link* de comunicação que implemente uma classe de protocolo para dois cenários distintos: (1) permitir a comunicação confiável e segura do VANT com sua estação base; (2) permitir uma comunicação que priorize a economia de recursos dos nós da RSFF na coleta dos dados. A abordagem tem como proposta utilizar apenas um dispositivo transceptor (rádio) de longo alcance para comunicar-se com os dispositivos potencialmente heterogêneos e compartilhar diferentes protocolos e tipos de tráfego. Como benefício do uso de apenas uma rádio, tem-se a redução de interferência, menor consumo de bateria, além da diminuição de

[1] http://provant.das.ufsc.br

espaço em placa e custo.

Para atingir os objetivos listados acima, podem-se delinear os seguintes objetivos específicos:

- Especificar um protocolo de comunicação robusto para redes *wireless* com altas taxas de perda;

- Desenvolver uma API que forneça abstração à heterogeneidade dos protocolos e serviços desenvolvidos;

Os resultados desse projeto poderão servir de base para desenvolvimentos futuros de VANTs para a indústria e a academia. Além disso, aplicações em outros domínios também poderão se beneficiar da infraestrutura desenvolvida como, por exemplo, robótica, automotiva, sensores de monitoramento e etc.

1.3 ORGANIZAÇÃO

Esta dissertação está organizada da seguinte forma: o Capítulo 2 faz uma revisão das principais tecnologias relacionadas. O Capítulo 3 faz uma breve descrição do projeto PROVANT. A revisão da literatura sobre o problema a ser investigado é abordado no Capítulo 2.5. O Capítulo 4 discute a problemática envolvida na comunicação entre dispositivos 802.15.4 heterogêneos e o suporte do Linux aos protocolos utilizados. A apresentação da abordagem de implementação para comunicação com VANT com a estação base e a RSSF é discutida no Capítulo 5. O Capítulo 6 descreve os experimentos do protótipo desenvolvido, assim como a discussão dos resultados obtidos. As conclusões e perspectivas de trabalhos futuros estão apresentadas no Capítulo 7.

2 TECNOLOGIAS E TRABALHOS RELACIONADOS

O objetivo deste capítulo é apresentar um resumo das principais tecnologias relacionadas ao desenvolvimento proposto nesta dissertação, além de mostrar suas principais características e de que maneira elas estão sendo utilizadas na proposta.

Inicialmente, são abordados os principais conceitos presentes nas redes de sensores sem fio, as suas formas de utilização e as restrições que implicam em uma série de requisitos para os protocolos de comunicação. Por seguinte, faz-se uma descrição geral do IEEE 802.15.4, padrão de comunicação sem fio de baixo consumo que tem sido utilizado como solução adequada para redes de sensores em muitos ambientes e áreas de atuação. Também serão explorados algumas das suas principais funcionalidades, as quais são utilizadas no desenvolvimento desse projeto. A posteriori, detalha-se brevemente o 6LowPan, protocolo que utiliza o IEEE 802.15.4 e tem como objetivo prover conectividade IP às aplicações em dispositivos sem fio que exigem baixo consumo de energia. Finalmente, é apresentado uma visão geral dos principais componentes que compõem a infraestrutura de hardware utilizada. Dentre esses componentes, estão o rádio de comunicação, a placa de desenvolvimento baseada em ARM e a distribuição Linux embarcada.

2.1 REDE DE SENSORES SEM FIO (RSSF)

Os recentes avanços na área de microeletrônica e comunicação sem fio permitiram o desenvolvimento de sensores de baixo custo, baixo consumo e tamanho reduzido, os quais possibilitam que aplicações conectem o mundo físico ao mundo virtual. Uma rede de sensores sem fio (RSSF) consiste em um conjunto de nós-sensores com a finalidade de monitoramento de grandezas e fenômenos físicos, incluindo temperatura, umidade, vibrações, eventos sísmicos, entre outros. Essa rede de sensores tem grande aplicação em locais de difícil acesso ou áreas perigosas, como, por exemplo, na área militar, industrial, de aviação e ambiental [Yick, Mukherjee e Ghosal 2008].

Devido à dimensão reduzida de um nó sensor, a restrição de recursos de software e hardware é um dos grandes desafios das RSSF. Cada nó da rede, usualmente, tem baixo poder de processamento, memória limitada e quantidade de energia bastante restrita. A fonte de energia geralmente consiste em uma bateria com capacidade limitada, dimensionada para apresentar um tempo de vida suficiente para cumprir as necessidades da aplicação para que foi designada, sendo a comunicação sem fio um dos principais consumi-

dores de energia [Lu, Krishnamachari e Raghavendra 2004]. Em aplicação ambiental, motivação de estudo deste trabalho, geralmente, os sensores são colocados em áreas remotas, fato que não permite o fácil acesso a esses sensores para manutenção. Portanto, a conservação de energia é um dos aspectos mais importantes a serem considerados no projeto de uma RSSF [Zhao e Guibas 2004].

Uma rede de sensores sem fio pode ser definida sob diferentes enfoques. De acordo com uma definição clássica da literatura, é vista como um tipo especial de rede *ad hoc*. Nessa definição, a comunicação entre os nós é feita em um esquema de múltiplos saltos (*multi-hop*), no qual um nó transmite os dados para o outro até que seja encontrado o nó de destino. O destino pode ser implementado em um nó sensor chamado de sorvedouro (*sink*) ou em uma estação base.

O nó *sink*, geralmente, é um nó com maior poder computacional e com maiores recursos. A finalidade desse nó é coletar as informações dos demais nós da rede e, eventualmente, tomar decisões referentes aos determinados eventos. Na maioria dos casos, o *sink* é estático. Entretanto, dependendo da região de monitoramento, é possível que o roteamento dos demais nós sensores para o *sink* seja inviabilizado por diversos fatores como, por exemplo, a longa distância em regiões muito amplas. Uma solução frequentemente utilizada, é o uso de nós móveis. Uma forma de se implementar nós móveis é a instalação de sensores em VANTs. Os VANTs podem ser de grande utilidade a uma RSSF devido à capacidade de se moverem em três dimensões e de utilizarem sensores mais sofisticados e, também, de serem autônomos. Desta forma, a utilização de nós sensores móveis com o uso de VANTs possibilita aumentar, significativamente, a capacidade e precisão do monitoramento de regiões específicas.

A seguir, será dado destaque ao padrão de transmissão sem fio, IEEE 802.15.4, que tem como característica o baixo consumo de energia e é frequentemente utilizado nas redes de sensores.

2.2 O PADRÃO IEEE 802.15.4

O padrão IEEE 802.15.4 é um conjunto de especificações que define protocolos de interconexão de dispositivos via comunicação por rádio em uma rede pessoal - WPAN (Wireless Personal Area Network). O padrão especifica a camada física (PHY) e o controle de acesso ao meio (MAC), com o objetivo de estabelecer uma rede de baixo custo, baixa complexidade e baixo consumo de energia. Devido às suas características, o padrão vem sendo utilizado como uma solução adequada para rede de sensores sem fio [Zheng e Lee 2004].

De acordo com o padrão IEEE 802.15.4, a WPAN suporta dois tipos de dispositivos [Koubâa, Alves e Tovar 2005]:

- *Full Function Device* (FFD): dispositivos capazes de operar com um conjunto completo de funcionalidades do MAC. Eles podem suportar três tipos de operações:
  - Coordenador PAN: o principal controlador da rede. Esse dispositivo identifica a sua própria rede, na qual os outros dispositivos podem se associar. A rede pode ter apenas um coordenador PAN;
  - Coordenador: sincroniza os dispositivos através da transmissão de *beacons*. O coordenador deve ser associado a um coordenador PAN e não pode criar a sua própria rede;
  - Dispositivo Simples: dispositivos que não implementam as funcionalidades acima.
- *Reduced Function Device* (RFD): dispositivos com um conjunto limitado de funcionalidades do MAC. Geralmente, são associados a sensores com restrições de recursos, como, por exemplo, um sensor de temperatura.

Dois tipos básicos de topologias são definidos no 802.15.4. O primeiro é a topologia em estrela, e a segunda é a *peer-to-peer*. Um terceiro tipo de topologia, *cluster-tree*, pode ser considerada como um caso particular da topologia *peer-to-peer*.

Na topologia em estrela (Figura 1), apenas um nodo pode operar como coordenador PAN. Qualquer dispositivo FFD pode criar a sua própria rede e se tornar um coordenador PAN, desde que escolha uma identificação que não esteja em uso pelos demais coordenadores na área de influência.

O modelo de comunicação da topologia estrela é centralizada. Cada dispositivo (FFD ou RFD) que pertençam à rede e desejem se comunicar com os demais dispositivos, devem transmitir os dados para o coordenador, o qual, posteriormente, é encaminhado para o destinatário final. Nessa topologia, o coordenador PAN deve permanecer ligado durante todo o ciclo de vida da rede.

A topologia *peer-to-peer* também possui um coordenador PAN. Entretanto, ele é nomeado apenas por convenção por ser o primeiro dispositivo a se comunicar em um canal específico. O modelo de comunicação é descentralizado, o que significa que qualquer dispositivo pode se comunicar diretamente com outro no seu raio de alcance. Em comparação à topologia em estrela, o consumo de recursos é menor, pois não há a dependência de um único dispositivo para comunicação.

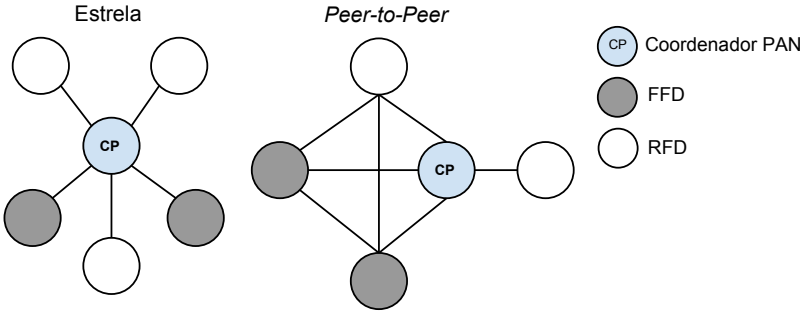

Figura 1 – Topologias em estrela e *peer-to-peer*.

A *peer-to-peer* também fornece maior flexibilidade, entretanto, é exigido uma complexidade adicional por fornecer a conectividade fim-a-fim entre todos os nodos. Basicamente, ela opera em modo *ad doc* por permitir que os dados sejam roteados por nós intermediários até o nó de destino. Porém, essa funcionalidade deve ser implementada na camada de rede, função não contemplada pelo IEEE 802.15.4.

A topologia *cluster-tree* (Figura 2) combina os benefícios das duas topologias mencionadas anteriormente, os quais apresentam melhor escalabilidade, manutenção e gerenciamento eficiente de energia. A topologia é do tipo em árvore, no qual os nodos são organizados em grupos lógicos, chamados de *clusters*. Nesse modelo, os nós sensores podem adotar dois papéis: Cluster-Head (CH) e Cluster-Member (CM). Os CH são responsáveis por coordenar a operação dos CM. Os CH podem também assumir o papel de roteadores. Em cada *cluster*, pode haver apenas um CH. Cada CM transmite periodicamente os dados coletados para o CH do seu *cluster*. Por seguinte, cada CH agrega os dados recebidos à partir dos nós CM e transmite para a estação base roteando os dados através dos demais CH. [Manjeshwar e Agrawal 2001].

A camada física é responsável pela transmissão e recepção de dados através do canal físico do rádio. São definidas três bandas de frequências operacionais: 2.4 GHz com 16 canais; 915 MHz com 10 canais e 868 MHz com apenas 1 canal [Lee et al. 2010].

As taxas de transmissão são 250 kpbs para 2.4 Ghz, 40 kpbs para 915 MHz e 20 kbps para 868 MHz. Enquanto a frequência de 2.4 GHz é mais adequada para maior *througput* e menor latência, as faixas de frequências menores são mais adequadas para transmissões de longa distância devido às baixas perdas de propagação e maior sensitividade pela baixa taxa de transmissão de dados. Todas as frequências suportam as técnicas de espelhamento espectral com sequenciamento direto (DSSS) para o modo de acesso ao ca-

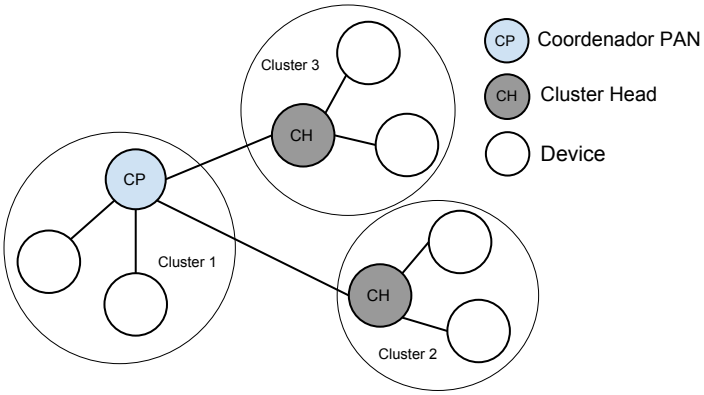

Figura 2 – Topologia *cluster-tree*.

Frequência (MHZ)	**Parâmetros de espalhamento**		**Parâmetro dos dados**	
-	Chip Rate	Modulação	Bit Rate (kbps)	Symbol
868	300	BPSK	20	Binary
915	600	BPSK	40	Binary
2400	2000	O-QPSK	250	16-ary

Tabela 1 – Caraterísticas da camada física.

nal de comunicação. Com relação à sinalização, a frequência de 2.4 GHz é baseada na técnica de OQPSK (*Orthogonal Quadrature Phase Shift Keying*), enquanto que as frequências de 868/915MHz utilizam a técnica BPSK (*Binary Phase Shift Keying*). As características de cada banda de frequência são resumidas na Tabela 1.

A camada física também é responsável pelos seguintes serviços: ativação e desativação do rádio transceptor para redução do consumo de energia em períodos ociosos, detecção do nível de energia (ED) do canal, indicação da qualidade do link (LQI) para pacotes recebidos, seleção da frequência do canal e verificação de ocupação do canal (CCA).

A camada MAC do IEEE 802.15.4 provê uma interface entre a camada física e os protocolos das camadas superiores do WPAN. Ela utiliza a técnica de acesso múltiplo com verificação de portadora com prevenção de colisão (CSMA/CA) e algoritmo de *backoff* para reduzir a probabilidade de colisão. Para redução do consumo de energia, é empregado o processo de *blind backoff*, ao invés do tradicional procedimento de *backoff* do IEEE 802.11. Para

transmitir os dados, a estação escuta o meio apenas em intervalos de tempos preestabelecidos, o que causa menor utilização do canal e, consequentemente, menor consumo [Adams 2006].

O protocolo MAC suporta dois modos operacionais que podem ser selecionados pelo coordenador, os quais serão explicados a seguir:

- *Beacon-enabled*: periodicamente, um conjunto de informações de sincronismo, chamado de *beacons*, é gerado pelo coordenador. Nesse modo de operação, é determinado um tempo chamado de superquadro, no qual tem-se a divisão em duas partes: o CAP (*Contention Access Period*), em que não há garantia de acesso ao meio; e o CFP (*Contention Free Period*), onde há a possibilidade de uso de compartimentos garantidos, chamado de GTS (*Guaranteed Time Slots*).

- *Non Beacon-enabled*: no modo sem *beacon*, os dispositivos podem transmitir os dados usando o *unslotted* CSMA/CA a qualquer instante. Ou seja, não há o uso de superquadro para sincronização dos dispositivos.

Neste trabalho é tratado, somente, o modo *Non Beacon-enabled*, visto que esse modo apresenta maiores taxas de transmissão e é o mais adequado para a aplicação desta pesquisa. As características desse modo tornam a rede mais suscetível a colisões e a maior disputa pela utilização do meio. Tais características fazem com que haja uma exigência na complexidade para as camadas superiores [Lu, Krishnamachari e Raghavendra 2004].

Três tipos de transferência de dados podem ocorrer. A primeira é a transferência do coordenador para o *device*; a segunda é a transferência do *device* para o coordenador e a terceira é a transferência entre *devices*. Na topologia em estrela, apenas os dois primeiros tipos de transferência podem ser usados, pois ela pode ocorrer somente entre o *device* e o coordenador. Na topologia *peer-to-peer*, a transferência pode ocorrer entre quaisquer *devices*, logo todos os três tipos podem ser usados nessa topologia [IEEE Standard 802.15.4 2011].

Quando um *device* deseja transferir os dados para o outro *device* em uma topologia de rede *peer-to-peer*, ele transmite o frame de dados utilizando o modo *unslotted* CSMA-CA. O *device* que recebeu os dados envia uma mensagem opcional de reconhecimento (*acknowlegment*), indicando que o frame foi recebido com sucesso. A sequência é resumida na Figura 3.

A estrutura dos frames do 802.15.4 foi desenvolvida para manter uma baixa complexidade com o compromisso de torná-la suficientemente robusta para a transmissão em um canal com ruídos. São definidos quatro tipos de formatos de frame:

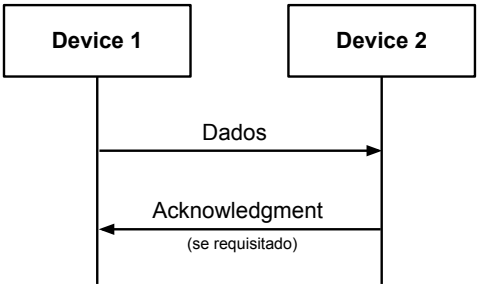

Figura 3 – Tipo de transferência entre *devices* em uma rede *peer-to-peer*.

- *Data*: usado para transferência de dados;
- *Beacons*: utilizado pelo coordenador para transmitir os *beacons* de sincronização;
- *Acknowledgment*: usado para confirmar o recebimento de um frame;
- *MAC*: utilizado para enviar comandos específicos do MAC.

Neste trabalho, será utilizado apenas os frames de dados e reconhecimento. Como o frame de reconhecimento é transmitido automaticamente pela maioria dos rádios, sem a atuação do *host* microcontrolador, o seu detalhamento será omitido.

O frame de dados é formado pelo *header* do MAC (MHR), *payload* do MAC e pelo MAC *footer* (MRF). O formato do frame deve ser conforme segue na Figura 4. O tamanho do *header* e do *payload* juntos não devem exceder o MTU do 802.15.4 de 127 bytes.

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>0/5/6/10/14</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

Figura 4 – Formato do frame de dados.

O *header* do MAC é formato pelos seguintes campos:

- *Frame Control*: campo de 16 bits que contém informações que define o tipo do frame e outras *flags* de controle (Figura 5). A funcionalidade de campo é descrito abaixo:

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

Figura 5 – Bits de controle do campo *Frame Control*.

  - *Frame Type*: especifica o tipo do frame: data, reconhecimento, *beacon* ou MAC;
  - *Security Enable*: define o suporte para implementações de segurança como o algoritmo AES-128;
  - *Frame Pending*: notifica o destinatário da existência de dados pendentes;
  - *Ack Request*: informa o destinatário que o originador da mensagem irá aguardar uma mensagem de reconhecimento;
  - *PAN ID Compression*: quando a origem e destinos estão no mesmo PAN, este bit pode ser setado para omitir o endereço do PAN de origem no *header*;
  - *Destination Addressing Mode*: especifica o tamanho do endereço do destinatário. Esse tamanho pode ser de 16 bits (*short address*) ou 64 bits (*extended address*);
  - *Frame Version*: define a versão do frame para compatibilidade;
  - *Source Addressing Mode*: especifica o tamanho do endereço de origem, semelhante ao campo *Destination Addressing Mode*.

- *Sequence Number*: número de sequência de 8 bits que identifica cada frame;

- *Destination PAN*: campo de 16 bits que identifica unicamente o PAN de destino. O valor 0xFFFF representa o valor de *broadcast*, que deve ser aceito por todos os destinatários que estão usando o mesmo canal;

- *Destination Address*: endereço de identificação do destinatário. Pode ser de 16 ou 64 bits, dependendo do campo *Destination Addressing Mode subfield* no campo de *Frame Control*;

- *Source PAN*: campo de 16 bits que identifica o número do PAN de origem do frame;

- *Source Address*: endereço de identificação do originador do frame. Pode ser de 16 ou 64 bits, dependendo da configuração do campo *Destination Addressing Mode* no campo de *Frame Control*.

O *MAC payload* é um campo de tamanho variável que contém os dados da aplicação. O menor *header* configurável para o frame de dados é de 11 bytes, sendo assim, o tamanho máximo do *payload* para essa configuração é de 116 bytes. Caso o pacote a ser transmitido for maior que o MTU, dá-se a necessidade da sua fragmentação, serviço não contemplado pelo padrão.

O *MAC footer* é um campo que contém a sequência de verificação do frame (FCS). O MRF é um campo de 16 bits que contém o código de redundância cíclica (CRC). O valor, geralmente, é calculado automaticamente pelo próprio hardware do rádio.

A seção seguinte discute sobre o protocolo 6LowPan, padrão utilizado para transmissão de pacotes IPv6 utilizando o IEEE 802.15.4.

2.3 IPV6 SOBRE IEEE 802.15.4 (6LOWPAN)

6LowPan, acrônimo de IPv6 *Over Low Power Wireless Personal Area Networks*, é o grupo de trabalho do Internet Engineering Task Force (IETF) que tem o objetivo de definir padrões de IPv6 sobre redes sem fio de área pessoal (WPAN) de baixo consumo de energia como o IEEE 802.15.4. O desenvolvimento do protocolo foi impulsionado pelo advento da Internet das Coisas (IdC), conceito que vem se popularizando com o paradigma de que qualquer dispositivo, mesmo que seja muito pequeno e simples, possa se comunicar pela Internet. Portanto, essa tecnologia permite a fácil integração entre sensores e computadores normais. No topo do protocolo 6LowPan, existem tecnologias como o protocolo TCP/IP. Entre os principais benefícios do 6LowPan, estão incluídos o uso de protocolos abertos, a confiabilidade, a transparência de integração entre redes e a escalabilidade global.

A capacidade do padrão IEEE 802.15.4 é mais limitada em comparação com outras tecnologias de transmissão sem fio, como o 802.11. O tamanho do frame, a banda e a potência de transmissão são significativamente menores. Para prover uma comunicação IP eficiente, o 6LowPan implementa uma camada de adaptação entre a de rede do IPv6 e a MAC do 802.15.4 (Figura 6) [Puccinelli e Haenggi 2005].

A camada de adaptação do 6LowPan tem três grandes objetivos, os

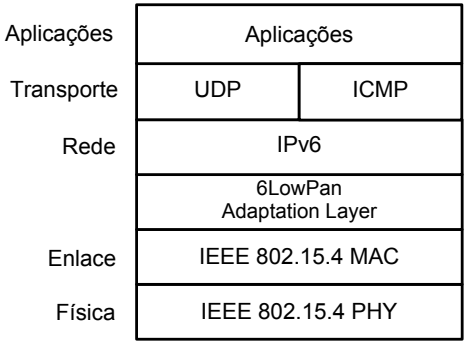

Figura 6 – Camada de adaptação do 6LowPan no modelo TCP/IP.

quais estão descritos a seguir:

- Compressão do *header*: o MTU do IEEE 802.15.4 é de 127 bytes, enquanto o MTU do 802.11 é de 1280 bytes. O tamanho máximo do *header* do MAC do 802.15.4 é de 25 bytes. Também são necessários 21 bytes adicionais para uso de encriptação dos dados na camada de enlace para segurança, resultando, dessa forma, apenas 81 bytes efetivos para o *payload*. O *header* do IPv6 é de 40 bytes e do UDP é de 8 bytes, restando, assim, apenas 33 bytes para os dados. Sem compressão, não é possível a transmissão efetiva de dados utilizando o IPv6 via 802.15.4.

- Fragmentação: os pacotes IPv6 devem ser fragmentados em múltiplos frames para se adequarem ao MTU do 802.15.4. Cada fragmento carrega informações, as quais permitem que ele seja remontado na sequência correta no destino, mesmo que os fragmentos estejam fora de ordem. Ao contrário dos fragmentos dos pacotes IPs tradicionais, os fragmentos do 6LowPan apenas incluem o *header* do IP no pacote inicial, o que disponibiliza mais espaço para o *payload* na camada de enlace. Entretanto, a performance de fragmentação para pacotes de tamanho grande é pobre pelas características de baixo consumo de energia da rede. A perda de um único fragmento faz os demais já recebidos pelo receptor serem descartados. Por essa razão, as aplicações que utilizam o 6LowPan devem evitar a fragmentação com o uso de pacotes do menor tamanho possível [Hummen et al. 2013].

- Roteamento: o protocolo de roteamento do 6LowPAN deve permitir o roteamento dentro da rede de sensores e, também, para a rede IP externa como a Internet. Dessa forma, os protocolos de roteamento são

divididos em duas categorias: o *mesh-under* e o *route-over*. Na categoria *mesh-under*, a decisão de roteamento é feita na camada de adaptação do 6LowPan e, no *route-over*, a decisão é realizada na camada de rede.

Similar à camada de transporte do modelo OSI, a camada de transporte do 6LowPan é responsável pela entrega fim-a-fim dos dados. A ISO define o protocolo de transporte para operar em dois modos: (i) orientado a conexão; (ii) não-orientado a conexão. Como exemplo de protocolo orientado à conexão, pode-se citar o TCP (*Transmission Control Protocol*), protocolo confiável desenvolvido para fornecer garantia na entrega de todos os pacotes entre um emissor e um receptor. Como exemplo de protocolo não-orientado à conexão, pode-se citar o UDP (*User Datagram Protocol*), protocolo não-confiável projetado para ser mais leve e simples e que, portanto, não oferece garantia de entrega.

A seção seguinte discute sobre o rádio de comunicação sem fio adotado para comunicação entre o VANT, a estação base e a RSSF.

2.4 RÁDIO MRF24J40MC

O rádio *transceiver* utilizado para comunicação sem fio é o MRF24J40MC da Microchip. Ele pertence ao padrão IEEE 802.15.4 com a camada física (PHY) e a camada de enlace (MAC), que são integradas em um único chip. O rádio também contém amplificador de potência de transmissão (PA), amplificador de baixo nível de ruído (LNA) e conector de antena externa, permitindo o alcance de até 1200 metros. As principais características são:

- Operação na frequência de 2.405 GHz - 2.475 GHz;
- Sensibilidade de -108 dBm;
- Potência máxima de transmissão de +19 dBm;
- Velocidade de 250 kbps (IEEE 802.15.4) e 600 kpbs (modo proprietário);
- Retransmissão automática de pacotes;
- Transmissão automática do frame de reconhecimento (ACK);
- Interfaceamento via SPI.

Dada a possibilidade da distância de comunicação do VANT com a estação base frequentemente ultrapassar algumas centenas de metros, houve a necessidade do uso de *transceivers* que atendesse tal exigência. Frente aos demais modelos avaliados, o MRF24J40MC foi a escolha mais adequada.

O interfaceamento com o *host* microprocessador (BeagleBone) é feito via SPI. O barramento consiste em três linhas de transmissão de informações

de 8 bits. Também são usados um pino de interrupção, um de reset e um de *wake-up*. A interrupção sinaliza o *host* do recebimento e envio de um frame. A interconexão entre o processador e o rádio está representado na Figura 7.

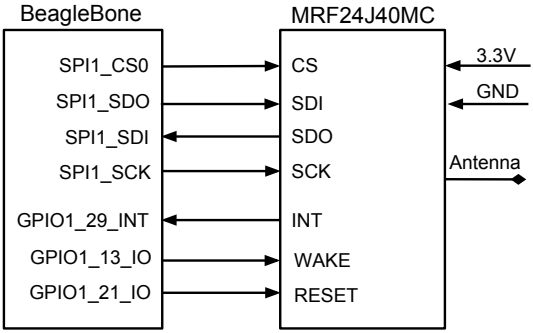

Figura 7 – Interfaceamento SPI entre a BeagleBone e o rádio MRF24J40MC.

2.5 ESTADO DA ARTE

Recentemente, o uso de mobilidade em uma RSSF tem sido largamente explorado por aumentar a capacidade da rede e permitir uma adaptação dinâmica às mudanças do ambiente. Diversos trabalhos científicos consideram o uso de VANTs como agente móvel para coleta de dados em tais redes. Exemplos dessa utilização são descritos em [Ye, Gong e Wang 2009], [Teh et al. 2008], [Yick, Mukherjee e Ghosal 2008] e [Barrenetxea et al. 2008]. Esses trabalhos observam que a transmissão solo-ar permite melhor qualidade de sinal do que a transmissão solo-solo e que, assim, oferece maior confiabilidade na comunicação.

Os métodos de coleta de dados podem ser basicamente classificados em duas categorias: mobilidade aleatória e mobilidade controlada [Kansal et al. 2004]. O trabalho em [Grossglauser e Tse 2002] conclui que a mobilidade aleatória de todos os nós em uma rede *ad hoc* aumenta significativamente a capacidade de coleta de dados em um sistema. Um algoritmo, para roteamento de dados entre agentes móveis, é sugerido em [Araujo e Becker 2011], no qual os dados são roteados entre os nós intermediários até encontrarem o nó de destino. Em [Shah R.C. e Brunette 2003] nós coletores, chamados de *data mules*, movem-se de forma aleatória pelo ambiente, coletando dados de forma oportunística dos nós sensores quando em alcance de comunicação. Esses nós são supostamente estáticos e aguardam a passagem do sistema de

MULE para iniciarem a comunicação. Eventualmente, o MULE passa pelas estações base ou demais nós de interesse e transmite os dados armazenados para então serem disponíveis aos usuários finais.

O uso de mobilidade controlada é considerado em [Chakrabarti, Sabharwal e Aazhang 2003]. A trajetória do coletor (ou *data-sink*) é determinada de forma a passar próximo dos nós sensores para que ocorra a troca de dados. Em [Freitas et al. 2010], a mobilidade é utilizada para conectar diversas áreas de comunicações isoladas uma das outras. A solução utiliza um ou mais VANTs como nós *sink*, que movimentam-se de forma a cobrir toda a área que abrange a RSSF. Com isso, é permitido que os nós da rede sempre encontrem um *sink* para entrega dos dados e que, assim, seja possível que os dados cheguem até a estação base. A abordagem mantém a conectividade proposta, a qual está baseada no mecanismo de propagação de mensagens, *beacons*, que são transmitidas periodicamente pela estação base para o(s) VANT(s). Aqueles que estiverem próximo da estação base, recebem o *beacon* e reencaminham para seus vizinhos. O procedimento é realizado até que o *beacon* alcance todos os VANTs da rede.

Os métodos de coleta também podem ser classificados de acordo com o tipo de detecção dos nós para com os coletores. Além da abordagem tradicional, que emprega transmissões periódicas de *beacons*, outras formas de descoberta também são encontradas na literatura. Em [Zhang et al. 2004], é proposto um sistema chamado de ZebraNet, em que é implementado um mecanismo de rastreamento de migração de animais. A implementação faz o uso de um *middleware*, em que a comunicação do coletor com os demais nós são previamente agendadas para um determinado horário. Em outros trabalhos, também são encontrados o uso de rádios *wake up*. O canal de *wake up* é continuamente monitorado e, quando há a detecção de energia, é iniciado um processo para acordar o processador e o transmissor.

Experimentos práticos, semelhantes ao cenário de utilização desse projeto, são descritos em [Teh et al. 2008]. O trabalho considera o uso de uma simples abordagem para a realização de *data mule*, no qual o VANT sobrevoa os nós sensores, transmitindo mensagens *beacons* e aguardando pelas respostas dos mesmos. O experimento constitui-se na utilização de um aeromodelo TF4050 Boomerang que transporta um sensor CSIRO Fleck 3 e uma estação base para monitoramento com o mesmo sensor. Esse é configurado para transmitir periodicamente *beacons payloads* que contém o número de sequência e o nível de potência em que a mensagem foi transmitida. Os *beacons* são transmitidos em diferentes potências, na sequência: -10, -2, +6, + 10, -10, -2 dBm, etc. O propósito da variação da potência de transmissão é obter uma melhoria na estimação do *range* de distância entre o aeromodelo e os sensores em solo. As mensagens de resposta dos sensores contém o número de

sequência, a potência do sinal copiado do *beacon*, o endereço do sensor, a temperatura e o nível de bateria.

A análise dos dados do experimento de *data mulling* é focada em duas métricas: taxa de transferência e taxa de entrega. Após a coleta, os dados podem ser transmitidos diretamente para a estação base ou serem transportados através de manobras para a estação base. Entretanto, a abordagem carece da discussão da necessidade dos parâmetros de comunicação do VANT com a estação base, importantes para garantia de entrega dos dados de forma confiável.

Uma proposta de cooperação dinâmica entre uma RSSF e um VANT para coleta de dados em uma vasta região é apresentada em [Dios et al. 2013]. O método explora a cooperação nas seguintes direções: (1) os resultados das operações da RSSF são usadas para alterar a trajetória do plano de voo do VANT; (2) a trajetória do VANT é considerada na configuração da RSSF para otimizar a performance da coleta dos dados. Os métodos apresentados nesse trabalho melhoram a performance de coleta e, consequentemente, aumentam a duração das baterias dos sensores e prolongam a vida útil da rede. Tais características são desejáveis em uma RSSF que opera em um ambiente de difícil acesso e/ou muito grande.

O trabalho em [Freitas et al. 2009] apresenta uma arquitetura de *middleware* para a integração de RSSFs e veículos aéreos em um sistema de monitoramento para segurança eletrônica. As propriedades de comunicação, requeridas pelo sistema, são analisadas de acordo com os tipos de mensagens da rede. Três principais classes de mensagens são consideradas de acordo com o tipo do nó *peer*: 1) VANT - VANT: controle de dados críticos como padrões de formação, negociação para atribuição de tarefas e troca dos dados dos sensores (ex. fusão de sensores); 2) VANT - Estação Base: mensagens de comandos, atribuição da missão de voo e dados de *payload*; 3) VANT - RSSF: alarmes de alerta de ocorrência de fenômenos de interesse, requisição e resposta de dados sensoreados pelos nós *low-end*. Porém, este trabalho provê apenas uma arquitetura muito geral e não apresenta em detalhes uma solução viável que disponibilize as propriedades de comunicações necessárias. Além da arquitetura utilizar o 802.15.4 para a comunicação com a RSSF, é também usado o 802.11b MAC/PHY para a comunicação com a estação base, o que requer um segundo *transceiver* o qual pode causar interferência. Por exemplo, a aquitetura mencionada poderia tirar proveito da abordagem proposta neste trabalho, os quais usa apenas um *transceiver* para comunicação com a estação base e a RSSF.

3 PROJETO PROVANT

O objetivo deste capítulo é apresentar em linhas gerais o Projeto de um Veículo Aéreo Não Tripulado, intitulado de PROVANT, em desenvolvimento no departamento de Automação e Sistemas (DAS) da Universidade Federal de Santa Catarina. O projeto tem como objetivo desenvolver uma aeronave *tiltrotor* de pequeno porte e de curto alcance, capaz de voar de forma autônoma de acordo com os pontos da trajetória [Goncalves et al. 2013].

3.1 INTRODUÇÃO

O desenvolvimento dos veículos aéreos não tripulados tem atraído enorme interesse da academia e da indústria nos últimos anos. Pesquisas nas áreas de robótica e de teoria de controle têm sido exploradas para melhoria de performance dos subsistemas que compõem um VANT. Como exemplo desses subsistemas, podem-se citar a fusão de sensores, a visão computacional, os estimadores de estado, a metodologia de controle, o projeto do sistema embarcado, os sistemas de comunicação e outros.

Embora existam diversos VANTs desenvolvidos pela academia e indústria, o uso de seus hardware e software proprietários e confidenciais, além da falta de documentação, tornam difícil a reprodução de tais projetos. Mesmo a aquisição de VANTs prontos para o uso, a sua manutenção e personalização tornam-se tarefas muito difíceis de serem realizadas. Por essas razões, o projeto PROVANT tem como intenção o desenvolvimento de um projeto aberto, que possa ser utilizado como referência de desenvolvimento de futuros VANTs.

3.2 PROPOSTA DO VANT

A proposta do projeto PROVANT é de desenvolver um VANT de pequena escala, birotor VTOL (*Vertical Takeoff and Landing*), com a configuração *tiltrotor*. A proposta visa a concepção do projeto do início, desde o desenvolvimento dos algoritmos de controle até a infraestrutura de comunicação sem fio. A ilustração do VANT pode ser visualizada na Figura 8.

O principal objetivo do VANT é o voo de forma autônoma sobre uma trajetória pré-definida pela missão. Tal missão é estabelecida pelo operador do sistema com o suporte de um sistema supervisório em uma estação base em solo que se comunica com o VANT por meio da utilização de tecnologias de comunicação sem fio.

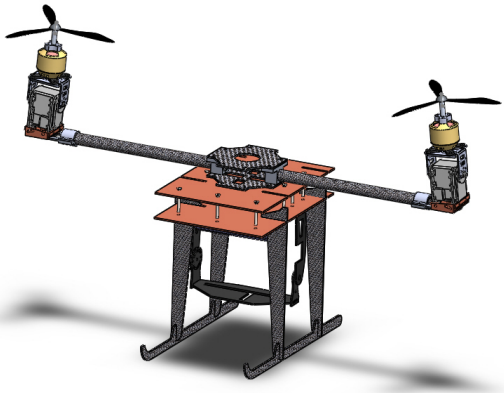

Figura 8 – Configuração do protótipo do VANT do projeto PROVANT.

Através da estação base, o operador do sistema pode definir os objetivos da missão e estabelecer os pontos da trajetória que o VANT deve seguir.

Outras possíveis ações são:

- Iniciar o voo autônomo para a missão definida;
- Abortar a missão por meio de duas funções distintas: retornar ao ponto de partida ou efetuar o pouso de emergência;
- Monitorar informações de voo;
- Realizar testes de verificação dos equipamentos antes da missão.

Cada uma das ações listadas deve desencadear uma sequência diferente de operações na plataforma embarcada que compõe o VANT. Por exemplo, quando o modo de voo autônomo é iniciado, tem-se a necessidade de carregar a missão para o VANT e confirmar o início da mesma. Para que o voo autônomo seja realizado, é necessário a execução de alguns subsistemas como o de controle de estabilidade, controle de trajetória, monitoramento da bateria, procedimento de pouso e decolagem e detecção e prevenção de obstáculos.

O procedimento de abortar a missão pode ser visto como uma ação extraordinária e excepcional. Uma vez que essa necessidade possa ocorrer por diversos fatores, os quais vão desde pelas más condições climáticas até alguma falha do sistema, duas possíveis ações podem ser disparadas. A primeira é conhecida como *return-to-home* (retornar para casa), no qual

o VANT deve retornar para uma específica localização geográfica, normalmente próximo da estação base. A segunda possível ação é o pouso de emergência no ponto seguro mais próximo da localização do VANT.

A monitoração das informações de voo é uma das possíveis ações que podem ser requisitadas pelo operador em solo. Dessa forma, é possível fazer decisões imediatas, caso seja observado algum evento extraordinário. E, finalmente, o teste de verificação dos equipamentos do VANT, o qual pode ser feito como ação preventiva e é similar aos procedimentos feitos por pilotos de avião antes do voo. Essas ações iniciam um teste de rotina no equipamento para garantir a integridade do sistema, bem como a verificação da comunicação com a estação base.

3.3 ARQUITETURA DO SISTEMA

A arquitetura proposta é composta pela estação base e a plataforma embarcada do VANT. A estação base é, como citado anteriormente, responsável por fornecer o suporte básico para as atividades do VANT. Ela é composta por um computador e uma infraestrutura de comunicação sem fio (detalhada no capítulo 5). A seguir, será apresentada a plataforma embarcada adotada.

O dispositivo de mais alto nível na estrutura do VANT é a plataforma embarcada Beaglebone. Ela é responsável pela execução dos algoritmos de controle, os quais tem por objetivo manter a estabilidade e garantir a trajetória com base na missão especificada. As características da plataforma são: baixo custo, alta performance e tamanho reduzido. A placa é equipada com o processador da Texas Instruments AM3358, cuja base é o *core* ARM Cortex-A8, que utiliza a arquitetura ARMv7-A. Suas características gerais são:

- 720 MHz superscalar ARM Cortex-A8 AM3358/9;
- 256 MB DDR2 RAM;
- 10/100 Ethernet RJ45;
- Slot para MicroSD card para o sistema operacional e arquivos;
- Diversos pinos de I/O como: 2 I2C, 5 UART, SPI, CAN, 66 GPIO, 8 PWM, 8 ADC e USB;
- Consumo de 300-500 mA a 5V;
- Dimensões de 86.4 mm x 53.3 mm.

O sistema operacional Linux foi adotado devido às diversas vantagens que ele fornece sobre outros sistemas operacionais embarcados. Dentre as

vantagens, podem-se citar o suporte a diversos hardwares e softwares, o fato de ele ser livre de *royalties* ou taxas de licença e do *kernel* ser estável, além da possibilidade de ler, modificar e redistribuir o código fonte.

Frente às diversas distribuições de Linux, a escolhida foi o Ångström, uma distribuição com foco para sistemas embarcados. A distribuição é resultado da unificação dos desenvolvedores dos projetos OpenZaurus, OpenEmbedded e OpenSIMpad. A comunidade responsável conta com um número de colaboradores bastante ativo ao redor do mundo. A distribuição atual conta com a versão 3.8 do *kernel* do Linux.

Em paralelo à BeagleBone, a estrutura do VANT também é composta pela plataforma embarcada STM32-H407. Essa é uma plataforma de baixo custo, a qual possui um processador ARM Cortex-M4 da ST. Para garantir que as restrições de tempo real da aplicação sejam atendidas, escolheu-se o sistema operacional embarcado FreeRTOS, que é de tempo real, além do seu código fonte ser pequeno, simples, aberto e de fácil uso. O objetivo da placa STM32-H407 é a comunicação dedicada com os sensores e atuadores listados abaixo:

- 1 Sensor infravermelho
- 1 Sonar
- 1 GPS
- 1 Unidade de medição inercial (IMU)
- 2 Controladores *brushless* (ESCs)
- 2 Servo-motores

A adição desse segundo dispositivo de gerenciamento realizou-se com base em análises nos requisitos de tempos de resposta que esses sensores e atuadores exigem.

3.4 MODELAGEM DO SISTEMA

A fase de modelagem tem como objetivo principal o desenvolvimento de um modelo funcional e comportamental do sistema. Nessa fase, pretende-se realizar o aperfeiçoamento de cada módulo e, assim, tornar a análise mais detalhada. Desenvolveu-se o modelo proposto com o uso da ferramenta *Simulink*. O modelo é dividido em três grandes subsistemas, os quais são a lógica de evento discreto, o processamento de dados e o controle contínuo.

O papel da lógica de evento discreto é inicializar e monitorar as funcionalidades gerais do sistema. Ela consiste em dois diagramas de estados, que

são um para a estação base e outro para o VANT. A Figura 9 representa o diagrama da estação base, no qual é conectado o bloco, que representa o VANT, detalhado na Figura 10. Esse modelo mostra como as mensagens enviadas pela estação base são interpretadas pela máquina de estado. Essa abstração é importante, pois explicita todas as mensagens de comunicação requeridas entre a estação base e o VANT.

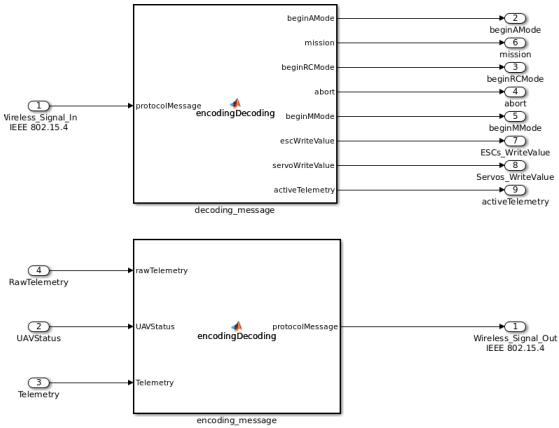

Figura 9 – Modelo de nível mais alto do Simulink.

Para iniciar o funcionamento do VANT, o operador da estação base deve primeiro escolher o modo de operação. Se for escolhido o modo de voo autônomo, a missão deve ser configurada de forma a selecionar os pontos que compõem a trajetória desejada. Após esse procedimento, é feito o *upload* da missão para o VANT através dos protocolos de comunicação confiáveis. Nesse ponto, todos os dados necessários para realizar a missão estão prontos e o VANT aguarda apenas por uma mensagem de confirmação da estação base para iniciar o voo. A partir desse momento, o voo autônomo é gerenciado pelo sistema de controle contínuo e a estação base entra em modo de monitoração da missão. A qualquer momento, o operador pode abortar a missão. Se for detectado algum problema técnico, o modo de segurança é ativado. Caso isso ocorra, uma mensagem de alerta é enviada para a estação base e o VANT efetua um pouso de emergência.

Caso seja selecionado o modo teste, o operador tem acesso a todos os dados dos sensores e atuadores. Além disso, comandos podem ser enviados diretamente para os atuadores. Essa operação é detalhada na Figura 11.

O subsistema de processamento de dados é composto pelos sensores, pelos atuadores e pela transformação dos dados brutos para a unidade

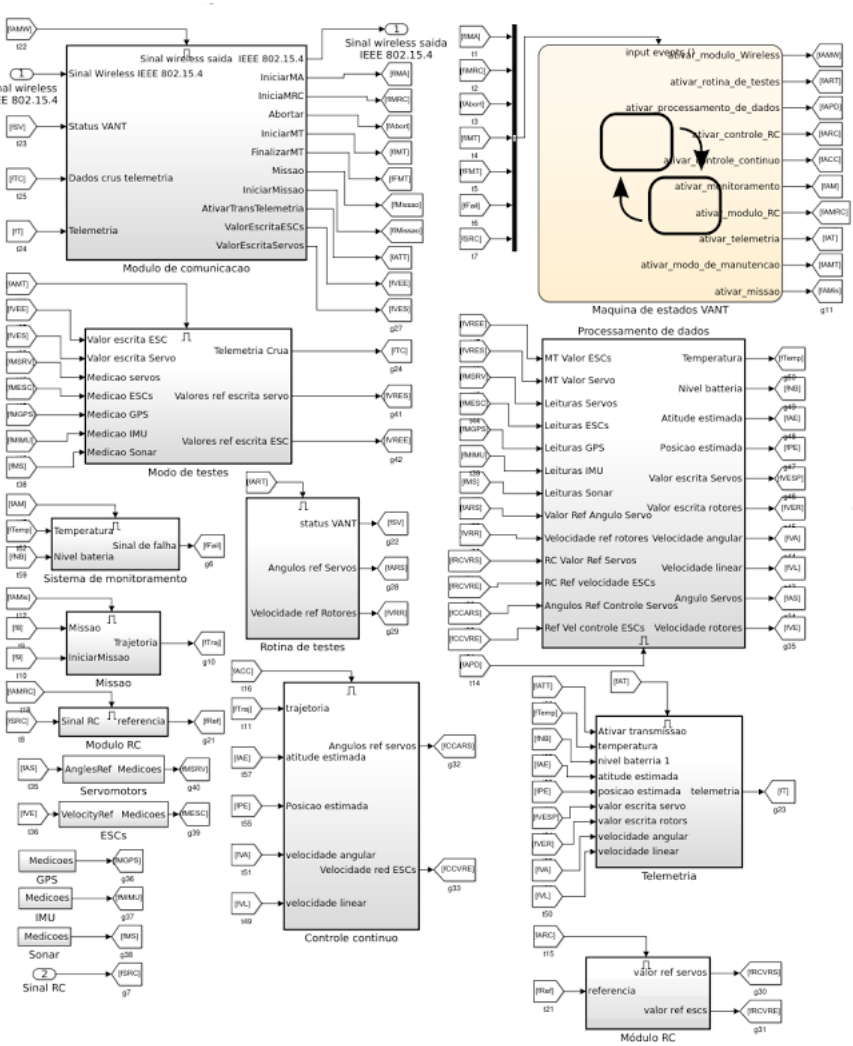

Figura 10 – Subsistemas do VANT.

de medição desejada para cada componente. As medidas necessárias para controlar o sistema podem ser vistas nas saídas do subsistema de processamento dos dados. A posição e velocidade linear são obtidas através do GPS; a velocidade angular, a aceleração linear e a pressão do ar, são dadas por um disposivo de unidade de medida inercial (IMU); e uma altitude mais precisa,

para procedimentos de aterrissagem, é obtida por um sonar.

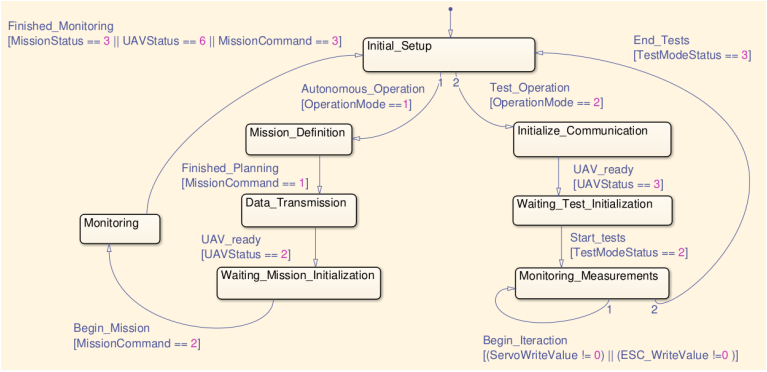

Figura 11 – Fluxo de estado da estação base.

Como atuadores, o veículo possui dois motores *brushless*, cada um com um conjunto de hélices em uma combinação chamada de "*propeller group*". Esses são controlados por um controlador eletrônico de velocidade (ESC), cuja entrada é a velocidade de rotação desejada. Além disso, cada motor é combinado com um servo motor, que rotaciona o motor para o ângulo de inclinação desejado.

Por último, tem-se o subsistema de controle contínuo, o qual é composto por dois componentes. O primeiro é o algoritmo para estimar a configuração vetorial, baseado nos dados recebidos dos sensores. E o segundo, é o sistema de controle contínuo para seguir a trajetória de referência e, também, a transformação dos dados de saída do algoritmo de controle (força) para os sinais requeridos pelos atuadores (ângulo de referência para os servo motores e velocidade angular para referência dos motores *brushless*).

Para disponibilizar uma estimativa mais precisa da posição do VANT, o bloco estimador utiliza um filtro *kalman* para fazer a fusão das medições da IMU com os dados do GPS.

O controle de trajetória é responsável por controlar a aeronave de acordo com a trajetória recebida pela estação base, de forma a rejeitar perturbações, erros de modelagem e outros eventos imprevistos. As saídas são a força de sustentação e os torques do UAV, os quais devem reduzir, e eventualmente negar, o erro entre a referência e a posição estimada.

4 VISÃO GERAL DA INFRAESTRUTURA PROPOSTA

Esse capítulo descreve, de modo geral, a problemática envolvida na comunicação entre dispositivos 802.15.4 heterogêneos, juntamente com a adoção do 6LowPan. Também faz-se a descrição do suporte do Linux aos protocolos utilizados na proposta.

4.1 INTRODUÇÃO

As propriedades de comunicação e a disponibilidade de recursos diferem-se entre os dois cenários vislumbrados. A comunicação do VANT com a estação base requer um protocolo de comunicação confiável que ofereça mecanismos de retransmissão, fragmentação, priorização de mensagens, segurança, conectividade IP e compatibilidade com a estrutura de comunicação via *sockets* do Linux. Neste cenário, ambos os dispositivos são equipados com recursos razoáveis de hardware.

Por outro lado, na comunicação com a RSSF, o consumo de energia é um ponto central. Dessa forma, demanda-se um conjunto de protocolos que minimizem o consumo da rede, uma vez que os sensores, usualmente, tem baixo poder de processamento, memória limitada e quantidade de energia bastante restrita. Assim, não basta apenas a aquisição de dispositivos com os mesmo tipos de rádios, mas também é necessário um protocolo de comunicação específico para cada tipo de dispositivo com o objetivo de lidar com as suas particularidades. Deseja-se, ainda, que a complexidade dos protocolos seja abstraída por uma API para facilitar o uso pelas aplicações.

4.2 COMUNICAÇÃO COM A ESTAÇÃO BASE

Pelo fato da comunicação com a estação base exigir um conjunto complexo de protocolos, optou-se por utilizar a pilha de protocolo IP do 6LowPan, cuja implementação contempla alguns dos mecanismos necessários, como por exemplo, fragmentação, segurança e conectividade IP.

Entretanto, o protocolo confiável TCP foi desenvolvido para redes cabeadas e parte do princípio de que a perda de pacotes causada por erro é muito inferior a 1%. Dessa forma, o TCP possui baixa eficiência e performance quando utilizado em uma rede com alta taxa de perda devido ao seu mecanismo de controle de congestionamento. Esse fato ocorre porque o TCP não é capaz de distinguir se as perdas foram relacionadas ao congestionamento

da rede ou a erros de transmissão. A perda de pacotes pela falsa detecção de congestionamento faz o algoritmo entrar na fase de *slow start*, o qual diminui a janela de envio para 1 o que, consequentemente, leva à redução da taxa de envio [Shelby e Bormann 2010].

Além disso, o cabeçalho do TCP é mais pesado e difícil de comprimir quando comparado ao cabeçalho do UDP, o que leva ao aumento do *overhead* do pacote e, consequentemente, causa diminuição da vazão de transmissão. Assim, uma estratégia recomendada é de a aplicação implementar um mecanismo de entrega confiável sobre o UDP.

Nesse contexto, foi desenvolvido um protocolo de comunicação confiável sobre o UDP, que contorna as desvantagens de uso do TCP quando utilizado em conjunto com o IEEE 802.15.4 em um ambiente com alta taxa de erro.

4.3 COMUNICAÇÃO COM A RSSF

Na comunicação com a Rede de Sensores Sem Fio, necessita-se de um protocolo simples que minimize o consumo de recursos. Dessa forma, é desejável que o protocolo não utilize o 6LowPan, pois o *overhead* do cabeçalho, na comunicação, exige maior consumo de energia, maior recurso de hardware para armazenamento da pilha de protocolo, além da diminuição da quantidade de dados úteis coletados pelo VANT, uma vez que esse sobrevoará a RSSF por um curto período de tempo. Vale ressaltar também, que a implementação da pilha do 6LowPan não está disponível para todos os modelos de sensores comercializados.

No cenário de comunicação com a rede de sensores, os nós sensores serão instalados em pontos estratégicos da ilha e terão uma distância entre eles que pode frequentemente exceder algumas centenas de metros. A longa distância entre os sensores leva à partição da RSSF em várias ilhas de comunicação, denominadas de (*clusters*). A partir da ideia de que os *clusters* estejam previamente organizados na topologia de *cluster-tree*, surge a necessidade de um mecanismo de coleta dos dados do *cluster-head*.

Assim, foi desenvolvido um protocolo que visa a redução do consumo dos recursos da RSSF e também um mecanismo de coleta de dados de uma RSSF.

A seção a seguir descreve sobre o suporte do Linux aos protocolos utilizados.

4.4 SUPORTE DO LINUX AOS PROTOCOLOS UTILIZADOS

A atual versão do *kernel* do Linux já disponibiliza suporte para o IEEE 802.15.4 e 6LowPan. O projeto responsável pela implementação é chamado de linux-zigbee[1] e iniciou-se em 2012 pela Siemens com o objetivo de implementar o protocolo Zigbee. Entretanto, devido à incompatibilidade de tal protocolo com a licença GPL, o objetivo do projeto mudou para a implementação do 6LowPan.

A transferência dos dados da aplicação é realizada através da API de *sockets* do Linux. Também é usado um protocolo especial sobre o *genetlink* para gerência e configuração de parâmetros como, por exemplo, o número do canal do rádio, os endereços (*short* e *extended*), o PAN ID e etc. A integração nativa com os subsistemas do Linux permite o uso de diversas aplicações como SSH, telnet, ping, nc, analisadores de rede (Wireshark) e etc. A estrutura de comunicação está representada na Figura 12.

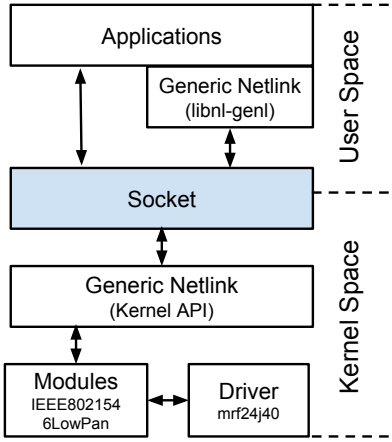

Figura 12 – Estrutura de comunicação da aplicação com os módulos do *kernel*.

Neste trabalho, diversas modificações de *drivers* e módulos do *kernel* foram realizadas a modo de compatibilizar com as infraestruturas de software e hardware necessárias para os requisitos do projeto. A primeira modificação necessária foi realizada no *driver* de controle do *transceiver* MRF24J40. A versão disponibilizada no *mainline kernel* é compatível apenas para o modelo mais simples do rádio, chamado de MA. As alterações contemplaram

[1] http://sourceforge.net/apps/trac/linux-zigbee/wiki

as necessidades específicas do modelo MC, as quais são: os parâmetros de inicialização, o controle do amplificador externo (PA), controle de potência em tempo real e outras que serão detalhadas no próximo capítulo.

5 DETALHAMENTO DA PROPOSTA

Esse capítulo descreve o projeto da infraestrutura de software para a comunicação do VANT nos dois cenários vislumbrados, os quais são a comunicação com a estação base e com a RSSF. Ambos os cenários diferenciam-se nas propriedades de comunicação, logo, há a necessidade de análise e projeto de um conjunto de protocolos específicos para cada cenário. Em ambos, é usado apenas um *transceiver* IEEE 802.15.4 de longo alcance. Dessa forma, é proposta uma arquitetura *dual-stack*, que é ilustrada na Figura 13.

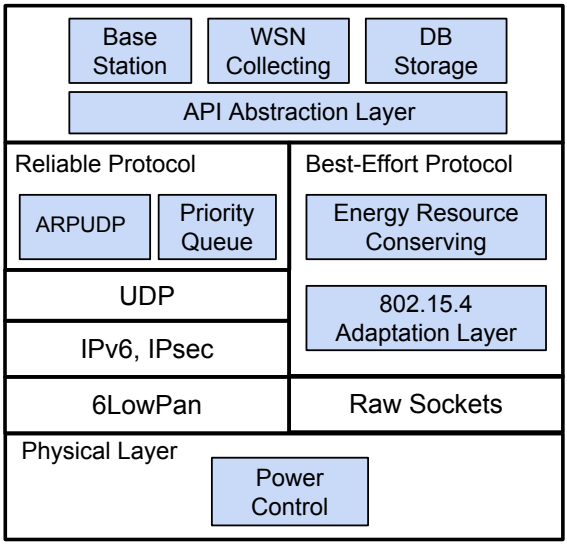

Figura 13 – Arquitetura de comunicação *dual-stack* do VANT com a RSSF e a estação base.

Na seção a seguir, são detalhados os dois protocolos que fazem parte da abordagem. Primeiro, será abordada a comunicação do VANT com a estação base, seguido da comunicação com a RSSF.

5.1 COMUNICAÇÃO DO VANT COM A ESTAÇÃO BASE

Essa seção apresenta a primeira classe de protocolos da abordagem adotada, a qual considera um protocolo de comunicação confiável que forneça

mecanismos de retransmissão, fragmentação, priorização de mensagens, segurança, conectividade IP e compatibilidade com a estrutura de comunicação via *sockets* do Linux.

5.1.1 Protocolo de Comunicação Confiável

Conforme explicado na seção 2.3, o uso do TCP possui baixa eficiência e performance quando utilizado em conjunto com o 6LowPan em uma rede IEEE 802.15.4. Dessa forma, a proposta implementa um protocolo de comunicação confiável sobre o UDP, o qual é designado de ARPUDP-6LowPan (*Aplication Reliable Protocol Over 6LowPan UDP in IEEE 802.15.4 links*) e está representado na parte esquerda da Figura 13.

O protocolo proposto fornece a integridade e a garantia de entrega em sequência de um fluxo de bytes. Ele oferece um serviço orientado à conexão, isto é, um dispositivo estabelece uma conexão com um outro de forma que ambos possam trocar dados em modo *full duplex*. Além de fornecer a confiabilidade entre os dispositivos, o protocolo incorpora técnicas para o aumento de desempenho em *links* com alta taxa de perda.

As operações do protocolo são semelhantes ao TCP, portanto são divididas em três fases. A primeira fase é o estabelecimento de conexão através de um processo conhecido como *three-way handshake*. Nessa fase, são trocados alguns parâmetros como o número de sequência (ACK) para garantir a entrega ordenada e robustez durante a transferência. A segunda fase é a transferência dos dados, detalhada a seguir. E a terceira é o encerramento da conexão após a transferência dos dados, a qual libera os recursos alocados. O fluxo dos estados do protocolo está resumido na Figura 14.

Para manter a confiabilidade da comunicação, o emissor transmite a mensagem com o seu próprio número de sequência e o receptor confirma a recepção dessa mensagem emitindo uma mensagem de ACK para o emissor com o número de sequência recebido. Quando o emissor envia os dados, ele inicia um mecanismo de *timeout* para receber a confirmação de recebimento. Caso a confirmação não seja recebida no tempo determinado, ele retransmite a mensagem. Esse mecanismo de controle de erros é conhecido por ARQ (*Automatic Repeat Request*) [Tanenbaum 2002].

5.1.1.1 Mecanismo de atribuição dinâmica de *timeout*

A escolha do valor *timeout* é muito importante para garantir a eficiência de transmissão dos dados. Se o valor for muito baixo, acarretará em retrans-

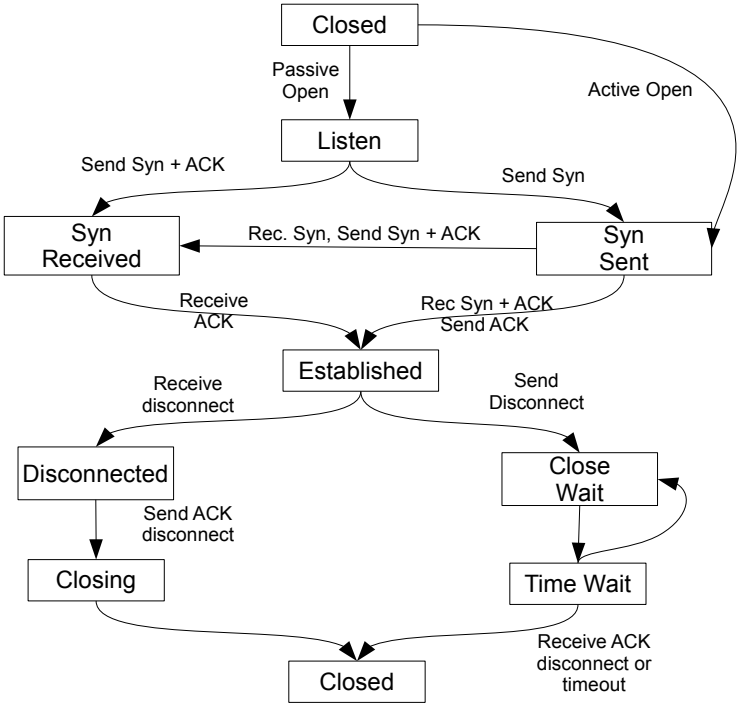

Figura 14 – Fluxo dos estados do protocolo ARPUDP.

missões desnecessárias, o que levará a consumir os recursos da rede. Caso o valor seja muito alto, o protocolo será lento nas retransmissões e diminuirá o *throughput* [Mori 2005].

A solução é utilizar um algoritmo dinâmico que ajuste constantemente o valor de *timeout* com base no monitoramento do atraso de envio das mensagens. Frente às técnicas disponíveis na literatura, foi escolhido o algoritmo *round trip time variance estimation*, introduzido por [Jacobson 1988] e frequentemente aplicado no controle de congestionamento do TCP .

O algoritmo funciona de forma que é mantido uma variável (*SmoothedRTT*) que representa a melhor estimativa no momento para o tempo de percurso de ida e volta. Quando uma mensagem é enviada, um timer é disparado para verificar quanto tempo a confirmação leva para chegar, e o mesmo armazena esse valor em *MeasuredRTT*. Em seguida, o algoritmo atualiza a variável *SmoothedRTT* de acordo com a Equação 5.1, sendo α o fator de suavização ($0 < \alpha < 1$) com o valor recomendado por Jacobson de 0.9.

$$SmoothedRTT = \alpha.SmoothedRTT + (1-\alpha).MeasuredRTT \quad (5.1)$$

A variável *SmoothedRTT* é atualizada a cada momento que uma medição é feita, onde 90% de cada estimativa é derivada da medição anterior e 10% da nova medição.

Para responder às flutuações dos tempos de ida e volta na rede, a RFC793 recomenda que o valor do *timeout* de retransmissão (*RTO*) seja conforme a Equação 5.2.

$$RTO = min[UBOUND, max[LBOUND, \beta.SmoothedRTT]] \quad (5.2)$$

A constante β é o fator de variação do atraso e tem o valor de 1.3. A constante UBOUND representa o máximo *round-trip time* que RTO pode assumir, cujo valor mínimo recomendado é de 100 ms. A constante LBOUND representa o RTT mínimo com valor de 30 ms. As constantes de β, LBOUND e UBOUND foram obtidas através da realização de testes empíricos de transmissão entre o VANT a a estação base. Tais testes foram necessários para adequar o valor das constantes às características da rede IEEE 802.15.4 e serão discutidos no próximo capítulo.

Quando uma mensagem é retransmitida pela falta de reconhecimento do envio anterior, não é possível distinguir se o reconhecimento recebido é da mensagem original ou da retransmitida. Esse comportamento pode ocorrer, pois há a possibilidade de a primeira transmissão ter apenas sofrido um atraso, mas não descartada. Esse problema é conhecido como ambiguidade da retransmissão. Ele é resolvido pelo algoritmo de Karn [Tanenbaum 2002], o qual evita a ambiguidade impedindo que o estimador de RTT seja atualizado quando receber o reconhecimento para essa mensagem. O procedimento de cálculo do *RTO* só é normalizado quando houver o reconhecimento de uma mensagem que não foi retransmitida.

A maioria das implementações do TCP usa uma estratégia de recuo exponencial, (*exponential back-off*), na qual o valor de RTO é duplicado a cada retransmissão. Sabe-se da existência do conhecimento do tempo máximo de atraso, logo a técnica pode ser estendida para a Equação 5.3, sendo γ o fator multiplicativo com valor 2.

$$RTO = min[UBOUND, \gamma.RTO] \quad (5.3)$$

5.1.1.2 Mecanismo de janela deslizante

O mecanismo descrito anteriormente pode levar a um baixo desempenho, pois o emissor não transmitiria nenhuma mensagem até que o reconhecimento da mensagem anterior fosse recebida. Para minimizar esse problema, foi incorporado, também, o protocolo ARQ denominado de *Selective Repeat*, o qual é baseado no princípio de "janela deslizante". A janela consiste no número de mensagens que podem ser enviadas sem que seja recebido o reconhecimento. A cada reconhecimento recebido na sequência correta, o emissor avança uma posição da janela para a direita e envia a próxima mensagem. Caso o receptor receba uma mensagem com numeração fora do intervalo da janela, a mesma é descartada. Entretanto, se a mensagem estiver no intervalo, porém fora de ordem, ela é armazenada. Para cada mensagem enviada, o emissor inicia um *timeout*, conforme descrito anteriormente.

Quando ocorrem perdas de transmissão, o algoritmo proposto não diminui o tamanho da janela, como é feito pelo algoritmo de controle de congestionamento do TCP. A justificativa é que a perda das mensagens está relacionada aos erros de transmissão, e não ao congestionamento da rede. A redução da janela faz com que a taxa de envio seja diminuída, quando que, nessa situação, o tamanho da janela deveria ser mantida ou, até mesmo, aumentada. Tal técnica é abordada em diferentes pesquisas, como em [Balakrishnan et al. 1997], no esforço de adequar o TCP para redes sem fio ou em outras com alta taxa de erro.

A perda de múltiplos segmentos da janela tem efeito degradante na vazão. As implementações mais tradicionais do TCP utilizam reconhecimento acumulativo. Ou seja, um determinado reconhecimento indica que todos os segmentos de numeração inferior a dele já foram recebidos com sucesso. Esse mecanismo força o transmissor a aguardar o *timeout* para descobrir quais foram perdidos, o que causa diminuição na vazão. A proposta do TCP SACK (*Selective Acknowledgment*) [Mathis Jamshid Mahdavi 1996] aumenta a performance do TCP em redes com alta taxa de perda. Ele tem como objetivo recuperar múltiplos segmentos perdidos no intervalo de um RTT. Para isso, o receptor fornece informação suficiente sobre os segmentos perdidos em uma janela de congestionamento. Com isso, o transmissor sabe exatamente quais segmentos foram perdidos e os retransmite no intervalo de um RTT. No algoritmo de reconhecimento da proposta, faz-se uso da técnica do reconhecimento seletivo devido aos benefícios apresentados.

O diagrama da Figura 15 ilustra a troca de mensagens entre a estação base e o VANT. A estação base envia alguma requisição para o VANT, o qual responde com quatro mensagens. A mensagem *Msg 1* é recebida corretamente. Já a mensagem *Msg 2* é perdida por interferência na transmissão.

No momento em que a estação base recebe a mensagem *Msg 3*, representada no passo 2, ela percebe a falta da mensagem entre *Msg 1* e *Msg 3* e envia um ACK duplicado da mensagem *Msg 1*, além de anexar, no mesmo ACK, a informação do recebimento da mensagem *Msg 3*. O passo 3 é semelhante e também anexa o recebimento da mensagem #3. No passo 4, o VANT recebe o ACK duplicado da mensagem *Msg 1* e o SACK da mensagem *Msg 3* e *Msg 4*. Desta forma, ele deduz que a mensagem *Msg 2* não foi recebida e retransmite antes do seu *timeout* expirar.

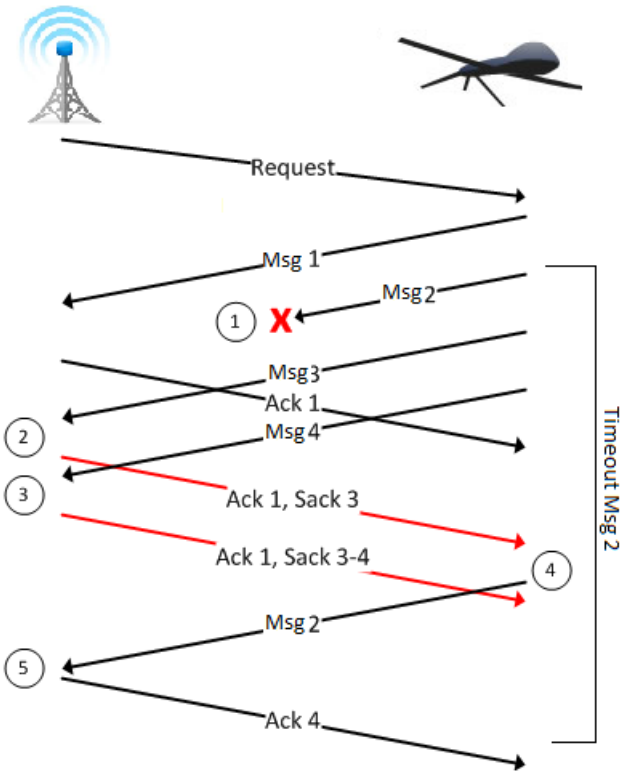

Figura 15 – Ilustração da comunicação entre o estação base e o VANT o qual utiliza a mesma técnica do TCP SACK.

Um comparativo entre o ARPUDP e o TCP pode ser visualizado na Tabela 2. O tamanho dos cabeçalhos descritos já estão comprimidos com técnica de compressão IPHC (IP Header Compression) [Ayadi D. Ros 2010]. No TCP é considerado a compressão do tipo *mostly compressed header TCP segment*.

-	**ARPUDP**	**TCP**
Conexão	Orientado à conexão	Orientado à conexão
Camada no modelo TCP/IP	Aplicação	Transporte
Estabelecimento de conexão	3-way handshake	3-way handshake
Controle de congestionamento	Janela fixa	Janela dinâmica
Tamanho do cabeçalho	6 bytes	15 bytes
Campos	1. Seq number, 2. Data type + UDP: 1. Length, 2. Source port, 3. Dest port, 4. Check Sum	1. Seq Number, 2. AcK number, 3. Data offset, 4. Reserved, 5. Control bit, 6. Window, 7. Urgent Pointer 8. Options, 9. Padding, 10. Check Sum, 11. Source port, 12. Dest port
Fluxo dos dados	Mensagens enviadas individualmente	Bytes como *stream* de dados (segmentos)
Priorização por mensagens	Sim	Não
Nr. retransmissão por classe de prioridade	Sim	Não
Complexidade	Baixa	Alta
Controle de fluxo	Não	Sim

Tabela 2 – Comparação entre o ARPUDP e o TCP.

5.1.2 Estrutura dos Dados de Comunicação

A abordagem permite que as aplicações de software do VANT se comuniquem com a estação base de maneira confiável e eficiente. Sempre que o VANT estiver ao alcance da estação base, ambos podem iniciar a comunicação. Estabelecida a comunicação, mensagens de controle, alarme e dados coletados dos sensores podem ser trocados. Para verificar se a estação base está no raio de alcance (ou vice-versa), o VANT pode enviar periodicamente mensagens de ICMP *echo request* (*ping*) e esperar pela resposta.

Para aumentar a eficiência de transmissão dos dados, eles podem ser montados através do uso de *frame aggregation*. O conceito de *frame aggregation* permite que múltiplas mensagens sejam enviadas no mesmo frame, reduzindo o tempo médio de transmissão e reduzindo dessa forma a utilização do canal sem fio. Esta abordagem é bem útil para este cenário, pois grande parte dos dados coletados da rede de sensores são de tamanho pequeno, o qual será descrito na seção 5.2.2.

O *frame aggregation* (Figura 16) é composto de uma sequência de campos, o qual permite a transmissão de múltiplas mensagens no mesmo frame. O campo *Flags* é um bit que identifica a utilização opcional do *frame aggregation*. O *Traffic Class* indica a prioridade da mensagem. O *Message ID* identifica o conteúdo da mensagem. O *Size* representa o tamanho do *payload* da mensagem. Conforme mencionado anteriormente, para razões de performance, o tamanho do *frame aggregation* deve preferencialmente ser pequeno o suficiente para caber em um único frame do 802.15.4. Além do tamanho do *frame aggregation*, o tamanho do *header* do 6LowPan deve ser levado em conta na formação do mesmo.

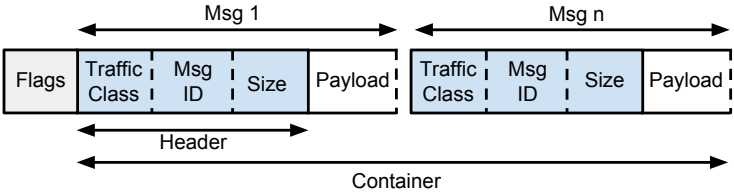

Figura 16 – Estrutura do contâiner na comunicação do VANT com a estação base.

O campo *Traffic Class* permite a distinção de mensagens entre as três diferentes classes de prioridades. Quando a *thread* de transmissão prepara o *frame aggregation* a ser enviado, as mensagens são retiradas da fila de acordo com as prioridades (começando da maior para a menor). O procedimento é

semelhante para a *thread* de recepção, as mensagens recebidas são entregues para a aplicação de acordo com a ordem de prioridade.

5.1.3 Mecanismo de Priorização das Mensagens

O protocolo confiável também permite a distinção de mensagens entre três diferentes classes de prioridades: (i) baixa, (ii) normal, e (iii) alta. As mensagens são armazenadas em uma fila de prioridades implementada por um tipo especial de árvore, a qual é chamada de *heap*. Quando a *thread* de transmissão é escalonada, as mensagens são retiradas da fila segundo a prioridade. Dessa forma, as mensagens com maior prioridade recebem o imediato número da sequência e são alocadas na janela de transmissão. A componente de prioridade é nomeada de *Priority Queue* e também é visualizada na figura da arquitetura de comunicação.

Para cada classe de prioridade, existe um número máximo de tentativa de retransmissão associado, o qual pode ser configurado via software. Por exemplo, pode ser definido que mensagens de alta prioridade possuam um número ilimitado de tentativas de retransmissão. Enquanto mensagens com média prioridade possuem cinco tentativas e mensagens com baixa prioridade não possuem retransmissões. Com essa estratégia, pretende-se evitar que mensagens de baixa prioridade aumentem o atraso médio das mensagens de alta prioridade, especialmente em situações com baixa taxa de entrega em função de interferência ou sinal fraco. Ainda é possível configurar a ação tomada quando a mensagem excede o número máximo de retransmissão. Ela pode ser retirada da janela de transmissão e recolocada na fila de prioridades, ou simplesmente descartada. A possibilidade de descartar a mensagem pode ser útil para evitar que, por exemplo, centenas ou milhares de mensagens de telemetria sobrecarreguem o *link* de comunicação.

A seção a seguir decreve a segunda classe de protocolos da abordagem proposta, que possibilita a comunicação entre o VANT e a rede de sensores sem fio.

5.2 COMUNICAÇÃO DO VANT COM A RSSF

O objetivo dessa comunicação é prover um protocolo de transporte simples que minimize o consumo de recursos da RSSF. Dessa forma, é desejável que o protocolo não faça uso do IPv6, até porquê a implementação da *stack* do 6LowPan não está disponível para todos modelos de sensores disponíveis no mercado.

Para possibilitar a comunicação *dual-stack* do VANT com os sensores, assim como com a estação base, é necessário um mecanismo para efetuar o contorno da pilha de protocolo TCP/IP do *kernel*. Esse mecanismo é disponibilizado pelo *kernel* do Linux com o uso dos conceitos de *raw sockets*.

5.2.1 Estrutura de Comunicação *Dual-stack*

O mecanismo de *raw sockets* do Linux permite que protocolos sejam implementados pela aplicação no espaço de usuário. Esse mecanismo fornece a flexibilidade para a aplicação de transmitir e receber os frames, incluindo o *header* da camada MAC. A visualização da abordagem adotada comparada ao modelo OSI pode ser vista na Figura 17. Uma vez que os *raw sockets* contornam a pilha do protocolo do 6LowPan, para cada transmissão, a aplicação deve montar o *frame* de dados do 802.15.4, conforme mostrado na seção 2.2. Similarmente, cada frame recebido deve ser decodificado pela aplicação, incluindo o *header* do 802.15.4. A aplicação responsável por essa decodificação é chamada de *802.15.4 Adaptation Layer* (ver Figura 13).

Entretanto, o mecanismo de *raw sockets* faz com que todas as mensagens recebidas pelo *driver* sejam encaminhadas para as aplicações que o implementem, o que ocasionaria a recepção indesejada dos pacotes IPv6 provenientes da comunicação com a estação base. Assim, esses pacotes devem ser filtrados e descartados. A estratégia aferida foi a implementação de um validador de pacotes no módulo *ieee802154.ko*, de modo a filtrar os pacotes que são IPv6 e não enviá-los para a aplicação de coleta dos dados dos sensores.

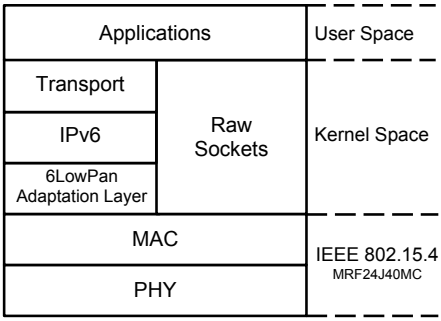

Figura 17 – Modelo da comunicação paralela. Protocolos confiáveis no topo do IPv6 em paralelo aos protocolos de baixo nível da RSSF.

5.2.2 Protocolo de Comunicação Com Economia de Recursos

A proposta de um protocolo que prioriza a economia de recursos para o recebimento dos dados dos sensores é baseada no formato do frame, o qual é mostrado na Figura 18. O campo *Sensor ID* é o número de identificação único do sensor. O *Data Type* especifica o tipo do fenômeno físico que o sensor está observando. Já o *Time Stamp* é usado para identificar o momento que o sensor coletou os dados. O *Battery Level* informa a quantidade de energia remanescente no sensor, importante para determinar o seu ciclo de vida. E, finalmente, o *payload* é o conjunto dos dados físicos de acordo com o *Data Type*. O *header* do sensor requer 6 bytes do *payload* do 802.15.4 e utiliza endereçamentos curtos, restando 110 bytes para o *payload*.

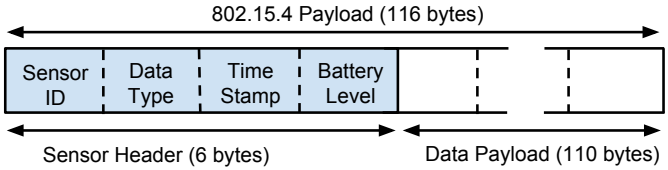

Figura 18 – Estrutura da mensagem do sensor na comunicação do VANT com a RSSF.

A aplicação responsável pela coleta dos dados é chamada de *Data Collecting*. Para iniciar a coleta, transmite-se uma mensagem de *broadcast* à rede de sensores com o intuito de indicar a solicitação dos dados. A partir desse momento, a aplicação mantém uma variável de *timeout* que é iniciada em 150 ms. Caso não seja recebido nenhuma mensagem no tempo de *timeout* programado, a mensagem de requisição é enviada novamente, e o procedimento se repete. Caso alguma mensagem seja recebida, a variável *timeout* é atualizada para o valor inicial de 150 ms independente do valor atual. A mensagem de requisição contém um número de sequência, o tipo de dados que o VANT está interessado em coletar. Esse método de coleta foi optado por apresentar uma solução mais genérica quando comparado, por exemplo, ao procedimento de agendar a transmissão dos dados do sensor para um determinado horário.

Sempre que um nó sensor (*cluster-head*) receber a mensagem de requisição e desejar transmitir os dados, ele o faz através do *unslotted* CSMA-CA no formato do frame descrito. Quando o VANT receber a mensagem, ele transmite um frame de *acknowledgment*, em nível de MAC, confirmando o recebimento. Se houver mais dados no *buffer* do sensor, eles podem ser transmitidos

em rajada até que o *buffer* se esvazie ou exceda o número de 3 tentativas de retransmissão. Caso exceda o número de retransmissão, o sensor deve aguardar até que ocorra o *timeout* da aplicação do VANT e ele envie novamente a mensagem de requisição. A ilustração básica dos passos da comunicação pode ser visto na Figura 19. O fluxo de estados completo do procedimento de coleta está representado na Figura 20.

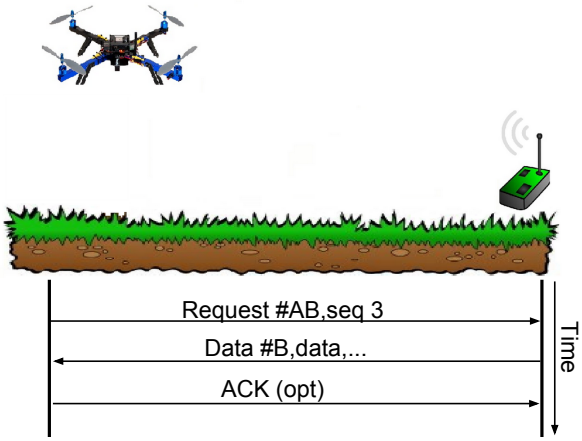

Figura 19 – Passos do UAV para a coleta dos dados da RSSF.

Como mencionado anteriormente, é ideal que o hardware de um nó sensor em uma RSSF seja de baixo consumo, tamanho reduzido e baixo custo. Dessa forma, a potência de transmissão dificilmente excede 0 dBm na maioria dos modelos comerciais. Como a potência de transmissão do modelo utilizado no VANT é de +19 dBm, o nó sensor, dependendo da sensibilidade, captaria a transmissão a uma distância que ultrapassaria centenas de metros. Nesse cenário, o nó sensor iria transmitir os seus dados, porém, eles não iriam ser recebidos pelo VANT devido ao curto alcance de transmissão. Além dessas transmissões desnecessárias causarem um consumo extra para os sensores, a comunicação de longo alcance causa uma alta interferência no canal sem fio. Para contornar esse problema, foi implementado no *driver mrf24j40.ko* um sistema para alteração da potência em tempo real. Esse sistema reduz a potência para um valor configurável na comunicação com os sensores e aumenta a potência para +19 dBm na comunicação com a estação base. O bloco responsável pelo procedimento é chamado de *Power Control*, o qual pode ser visualizado na ilustração da arquitetura de comunicação da Figura 13.

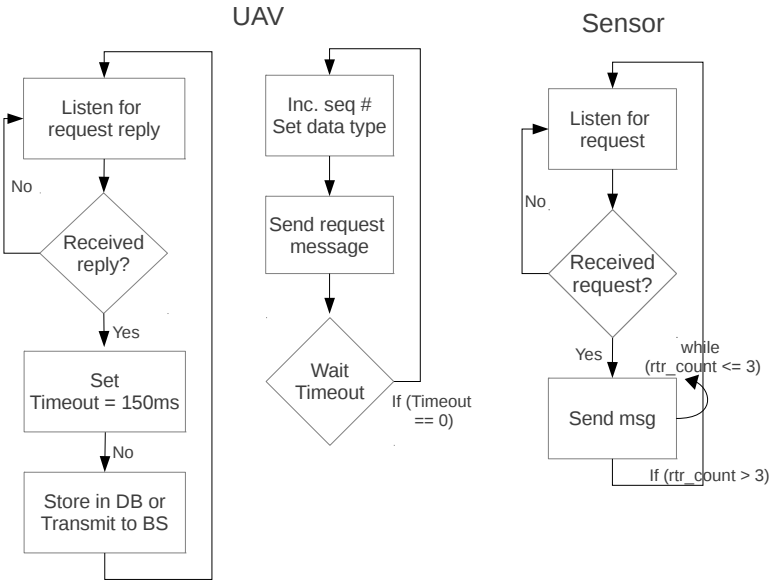

Figura 20 – Sequência de operações do VANT e do sensor para coleta dos dados.

Os dados coletados pelo VANT podem ser armazenados no banco de dados (SQLite) ou serem diretamente transmitidos para a estação base, se ela estiver ao alcance. A tabela do banco de dados, representada na Figura 21, contém os seguintes campos: *Sensor ID* juntamente com o *Time Stamp* identificam unicamente cada dado; o *GPS Coordinate* representa as coordenadas geográficas do nó sensor que originou os dados; o *Coordinated Universal Time* (UTC) representa o tempo de observação do dado, enquanto o campo *Link Quality Indication* (LQI) é a indicação de qualidade do enlace. Também tem-se o campo *Received Signal Strengh Indicador* (RSSI), o qual é a medição do nível de potência do sinal do dado recebido. Finalmente, *Battery Level* representa a energia restante do sensor, conforme mencionado anteriormente.

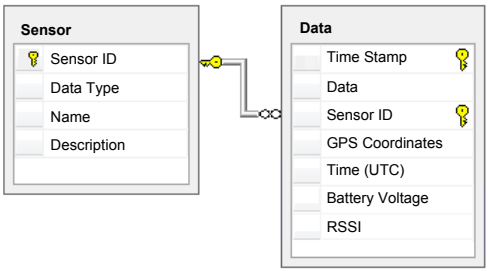

Figura 21 – Tabelas do banco de dados relacional para armazenamento dos dados coletados.

5.3 API DE COMUNICAÇÃO

Para prover uma abstração e omitir a complexidade dos protocolos de comunicação, foi desenvolvida uma API chamada de *API Abstraction Layer* para ser utilizada nas aplicações de mais alto nível. Dessa forma, as aplicações não necessitam saber como os protocolos trabalham internamente, somente como se comunicar através da sua interface. O diagrama de classes com as principais funcionalidades pode ser visualizado na Figura 22. Para facilitar a compreensão, são ilustradas apenas as principais classes envolvidas na arquitetura da API. O argumento da função também é omitido de forma a não saturar o diagrama.

A API foi desenvolvida em C++ e utiliza a biblioteca STL (*Standard Template Library*). Apesar de ter sido desenvolvida para plataforma Linux, ela pode ser facilmente portada para outros sistemas operacionais. Integrou-se, também, um modo de teste, o qual faz com que haja a possibilidade de que dois computadores ou dois processos no mesmo computador simulem a comunicação confiável através da pilha UDP, a fim de testes de desenvolvimento.

Duas principais classes abstratas fazem parte da API, as quais são a *BaseStation* e a *WSN*. Essas classes representam as entidades e os conceitos abstratos e são herdadas e implementadas pelas classes concretas *ReliableCommunication* e *EnergyConserving* respectivamente. Outra classe abstrata, também de importante na implementação, é a *RTO*, a qual fornece a assinatura básica dos métodos relacionados ao cálculo dinâmico do valor de *timeout* de envio das mensagens. Tal abstração torna fácil a implementação de novas técnicas para o cálculo do *timeout*. As classes *6LowPanInterface* e *RawSocketInterface* não são a implementação do 6LowPan e do *raw sockets*, mas

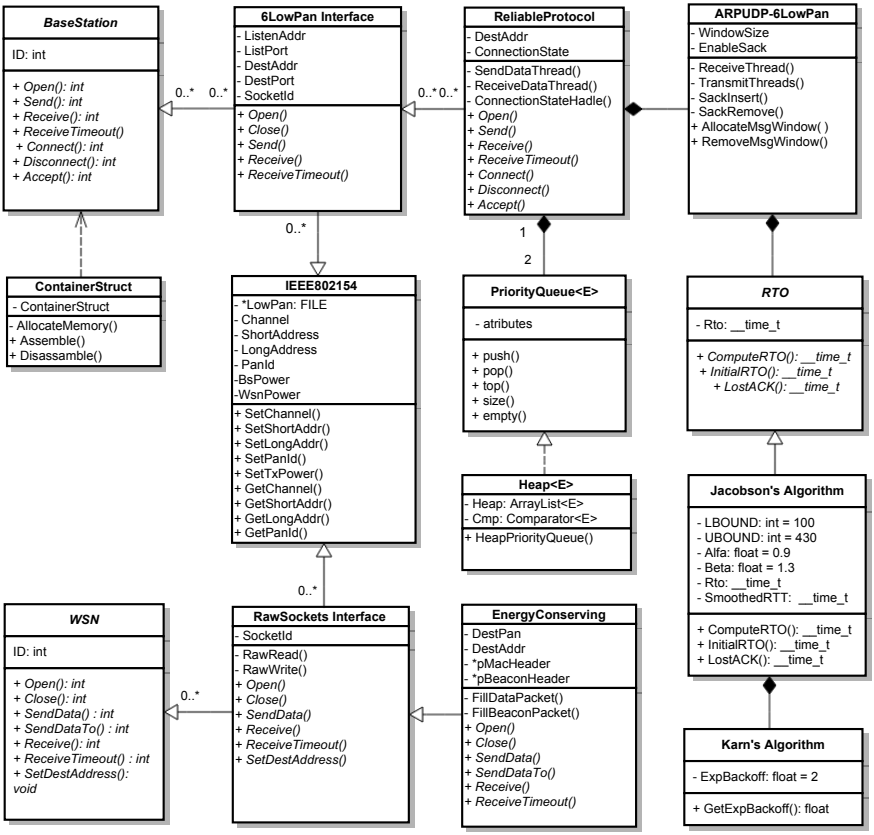

Figura 22 – Diagrama de classes da API.

métodos que fornecem abstração dos detalhes para as classes derivadas. Diferentes implementações de prioridades podem ser usadas apenas alterando as classes *PriorityQueue* e *Heap*.

Os códigos exemplos 5.1 e 5.2 demonstram como acessar explicitamente os membros da interface da comunicação confiável e os membros da comunicação com a RSFF. Os mesmos destinam-se apenas a ilustrar o conceito e mostram somente as funções relevantes. A aplicação pode instanciar a comunicação confiável com a estação base e/ou a comunicação com economia de recursos com a RSSF, de acordo com os requisitos.

O respectivo código que instancia os dois tipos de comunicação é representado pela primeira linha dos dois códigos exemplos. O procedimento

seguinte é a configuração dos parâmetros do IEEE 802.15.4, a qual está representada, no código, pelo ponteiro **ieee*. Os parâmetros incluem o canal, os endereços curtos e longos, o PAN ID e a potência de transmissão para comunicação com a estação base e a RSSF. Tais parâmetros são guardados em um arquivo e são únicos no sistema. Portanto, os valores devem ser configurados apenas uma vez e não há importância no número de aplicações que faz uso dos protocolos. A configuração dos parâmetros está representada nas linhas 7 a 11 do código exemplo 5.1.

A seguir, é necessário abrir a conexão, a qual é realizada através do procedimento *Open()*. Os argumentos necessários na comunicação com a estação base são a porta e o endereço IP do *host* que será aceito na conexão, ou *ANYADDR* para aceitar todos. É interessante restringir a conexão a um específico *host* como medida adicional para evitar a conexão de intrusos. Os argumentos aceitos na conexão com a RSFF são parâmetros do *cluster head* da RSSF, como os endereços curtos e longos e o PAN ID.

Em razão da comunicação com a estação base ser orientada à conexão, é estabelecido que a estação base tem o papel de cliente e o VANT de servidor. O cliente deve chamar a função *Connect()* a fim de estabelecer a conexão com o servidor, o qual deve estar aguardando-a através da chamada função *Accept()*. O parâmetro da função *Connect()* é o endereço IP do servidor e o argumento da função *Accept()* retorna o número IP do cliente que efetuou a conexão. O procedimento é ilustrado nas linhas 17 e 18 do exemplo 5.1.

Após os passos anteriores, é possível efetuar o envio e recebimento das mensagens, os quais são realizados pelas funções *Send()* e *Receive()* respectivamente. Na comunicação com a estação base, os argumentos da função *Send()* são o ponteiro para a região de memória onde está a mensagem a ser transmitida, o tamanho e a sua prioridade. A função *Receive()*, a qual é do tipo bloqueante, tem como argumento o ponteiro para a região de memória, onde a mensagem será copiada. Na comunicação com a RSSF, a função de envio é semelhante, porém sem a possibilidade do uso de prioridades. Logo, a função de recebimento da RSSF tem, como argumento adicional, um ponteiro que indica o tipo de mensagem recebida, o qual pode ser *beacon*, dados ou comandos do MAC. As mensagens de *beacons* e comandos do MAC não são utilizados pelas aplicações nesse projeto, porém, já são contempladas no desenvolvimento da API. Em uma implementação real, é usual a aplicação utilizar uma *thread* separada apenas para recebimento das mensagens. Uma outra função útil, porém não ilustrada nos exemplos, é a possibilidade da recepção de mensagens com *timeout*, chamada de *ReceiveWithTimeout()*. Essa função tem como argumento o tempo que ela ficará bloqueada até que não seja recebida nenhuma mensagem.

E, finalmente, quando a conexão não é mais requerida, os recursos

devem ser desalocados através da função *Close()*. No caso de uso dos protocolos confiáveis, é disponibilizado a função *Disconnect()* a fim de informar à outra ponta da desconexão.

```
BaseStation *baseStation = new ReliableProtocol();
IEEE802154 *ieee;

ieee = dynamic_cast<IEEE802154 *> baseStation;

// Configura os parâmetros do IEEE 802.15.4
ieee->SetChannel(11);
ieee->SetPanID(0x777);
ieee->SetShortAddress(0x3);
ieee->SetLongAddress(0xABCABCAD);
ieee->SetTxPower(BASE_STATION,WSN);

// Abre a conexão
baseStation->Open(ANY_ADDR,PORT);

// Conexão - Aguarda conexão
baseStation->Connect(REMOTE_IP); // Apenas Cliente
baseStation->Accept(&addr_ip);   // Apenas Servidor

// Envia e recebe mensagem
baseStation->Send(buffer,size,HIGH_PRIORITY);
baseStation->Receive(buf,MAX_SIZE);

// Pedido de desconexão
baseStation->Disconnect();
baseStation->Close();
```

Código 5.1 – Exemplo de implementação da comunicação confiável.

```
WSN *wsn = new EnergyConserving();

// Assume-se que os parâmetros do IEEE 802.15.4
  foram previamente configurados no exemplo acima

// Abre a conexão
wsn->Open(DEST_PAN,DEST_SHORT_ADDR,DEST_LONG_ADDR);

// Envia e recebe dados da WSN
wsn->SendData(buf,size);
wsn->Receive(buffer,MAX_SIZE,&type);

// Fecha conexão
wsn->Close();
```

Código 5.2 – Exemplo de implementação de comunicação com a RSSF.

5.4 INFRAESTRUTURA DA ESTAÇÃO BASE

A estação base é formada por uma estação transceptora e um sistema supervisório. A estação transceptora é composta por uma BeagleBone e o *transceiver* MRF24J40MC. O sistema supervisório é composto por uma interface gráfica em Java (Figura 23), o qual foi desenvolvida por um dos integrantes da equipe do PROVANT. Um computador poderia ser utilizado no local da BeagleBone da estação transceptora, entretanto seria necessário o desenvolvimento de um novo *device driver*, o qual estava fora do escopo do trabalho.

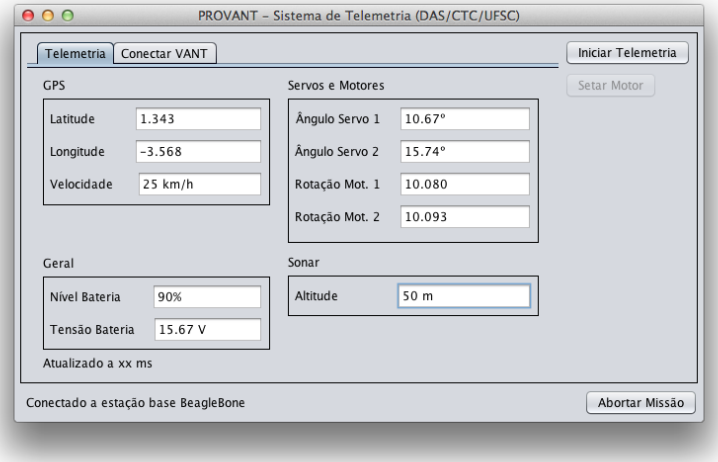

Figura 23 – Interface do sistema supervisório.

A comunicação do sistema supervisório e a estação transceptora é realizada através do protocolo TCP/IP, o qual permite o controle em tempo real através de uma máquina localizada em qualquer ponto da Internet. A estação transceptora funciona como um *gateway*, cuja responsabilidade é a interconexão entre a rede *ethernet* e a 802.15.4. Os pacotes IPv6 provenientes do VANT são reencaminhados para o sistema supervisório através de regras de roteamento (ou vice-versa). É possível também que o sistema supervisório esteja em uma rede IPv4 através de técnicas de tunelamento.

É de vital importância que os *links* de comunicação com o VANT ofereçam segurança, tanto para o controle do VANT com base em objetivos da missão quanto na entrega dos dados coletados para a estação base.

O *transceiver* MRF24J40MC disponibiliza um bloco de segurança em hardware que implementa o algoritmo AES (*Advanced Encryption Standard*) de acordo com o padrão IEEE 802.15.4-2003. Entretanto, a criptografia a nível de enlace assegura a segurança apenas *hop-by-hop* em nós 802.15.4. Dessa forma, as mensagens que deixam a estação base e continuam a trafegar pela Internet para um operador remoto não são protegidas por esse mecanismo.

A segurança fim-a-fim pode ser garantida com o uso do IPsec. O IPsec é um protocolo que opera na camada de rede, e permite que dois ou mais *hosts* se comuniquem de maneira segura por meio de autenticação e criptografia para cada pacote IP na sessão de comunicação. O modelo de criptografia adotado é o de chave simétrica, pois ele tende a ser mais rápido e necessita de menor poder computacional por parte dos dispositivos quando comparado ao de chave assimétrica. Além disso, o uso da mesma chave tanto para criptografar como para descriptografar os dados, fornece maior simplicidade na implantação do projeto. Entretanto, o suporte do IPsec para redes de baixo consumo ainda é uma questão em aberto, pois ainda não há uma RFC que defina um padrão. A primeira proposta para integrar uma extensão comprimida do IPsec para o 6LowPan foi introduzida por [Raza et al. 2011]. Dessa forma, por não haver a implementação da compressão no *kernel* do Linux, a arquitetura proposta não contempla a implementação, mas prevê o seu uso futuro.

6 AVALIAÇÕES E RESULTADOS

Neste capítulo, descrevem-se o escopo detalhado dos experimentos e a avaliação dos resultados, referentes aos testes das classes de protocolo da arquitetura *dual-stack*. Essas classes, como mencionadas no capítulo 5, são da comunicação do VANT com a estação base e com a RSSF.

Os objetivos dos experimentos são separados de acordo com a classe de protocolo, a qual é utilizada para a comunicação do VANT. Na classe de comunicação com a estação base, os objetivos são a avaliação de desempenho do protocolo confiável, desenvolvido para garantir que as mensagens sejam completamente transferidas. Na classe de comunicação com a RSSF, o objetivo é avaliar o protocolo implementado para a coleta dos dados da rede de sensores. A fim de obter os resultados, foram avaliados parâmetros de Qualidade de Serviço *QoS* como atraso, *jitter*, vazão e perda de pacotes.

6.1 CONFIGURAÇÃO DO AMBIENTE DE TESTE

Realizou-se a avaliação da proposta através de experimentos práticos. Como o VANT em desenvolvimento no projeto não ficou pronto, utilizou-se uma bicicleta como agente móvel[1]. Neste cenário, a bicicleta movimentou-se enquanto transportava o protótipo de teste por uma determinada região.

Para a realização dos testes, foram necessárias duas placas de desenvolvimento BeagleBone, dois rádios de comunicação, um sensor, um GPS e uma bicicleta. No teste de comunicação do agente móvel com a estação base, ambos os rádios utilizados foram o MRF24J40MC de longo alcance, configurados na potência máxima de +19 dBm. Para os testes de comunicação com a RSSF, o sensor escolhido foi o MICAz da empresa Crossbow, configurado a uma potência de transmissão de -1 dBm. Vale lembrar que a abordagem, a qual foi implementada, reduz automaticamente a potência de transmissão do agente móvel para -2 dBm na comunicação com os sensores. O GPS foi instalado na BeagleBone embarcada do agente móvel. As configurações da camada física e outros aspectos dos experimentos são mostrados na Tabela 3.

Os experimentos foram divididos em duas fases, as quais realizaram-se em dois cenários. A primeira consiste na comunicação do agente móvel com a estação base e o cenário designado foi a Avenida Beira Mar Norte. Já a segunda fase baseia-se na comunicação com os sensores e foi realizada em um campo aberto. A Figura 24 ilustra a configuração do sistema para cada

[1] Entende-se por agente móvel como sendo um dispositivo de comunicação com capacidade de mobilidade.

Dados	Estação Base (EB) / Agente Móvel	Sensor
Modelo do Rádio	MRF24J40MC	MICAz MPR2400CA
Alcance Estimado	1200 metros	100 metros
Potência (TX)	+19 dBm com a EB e -2 dBm com sensores	0 dBm
Antena	Dipolo de onda completa	Dipolo de 1/2 onda
Sensibilidade	-108 dBm	-94 dBm
Canal	11	11

Tabela 3 – Dados dos experimentos

cenário.

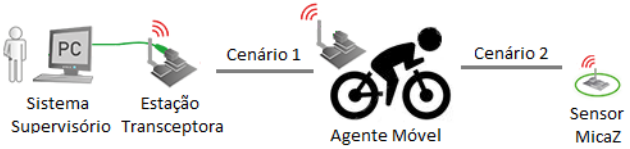

Figura 24 – Configuração do sistema para cada cenário: (1) comunicação com a estação base e (2) comunicação com os sensores.

A região escolhida para os testes de comunicação com a estação base foi a avenida Beira Mar Norte em Florianópolis, como dito anteriormente. Para a escolha, levou-se em consideração a longa extensão da avenida, aproximadamente 3 km, e a presença de obstáculos, como árvores, postes e pessoas, além da interferência de outras tecnologias de transmissão sem fio. Um ambiente sem obstáculos não seria adequado, pois a comunicação seria uniforme, o que ocasionaria poucas perdas de pacotes e dificultaria uma avaliação mais precisa da abordagem de implementação da camada confiável.

Para os testes com a RSSF, houve a necessidade de um lugar que dispusesse de pelo menos 250 metros em linha reta e sem obstáculos. Assim, escolheu-se um campo aberto.

As seções a seguir tratarão da descrição da avaliação dos experimentos para cada um dos cenários de maneira independente.

6.2 COMUNICAÇÃO DO AGENTE MÓVEL COM A ESTAÇÃO BASE

O experimento de comunicação do agente móvel com a estação base foi dividido em duas etapas. A primeira, chamada de etapa de manutenção, consistia no envio e recebimento de uma sequência de comandos, incluindo a simulação de *download* de um arquivo de missão. Nesse etapa, o agente móvel estava localizado estaticamente a três metros da estação base.

A segunda etapa consistia na mobilidade do agente móvel pela avenida Beira Mar. A sua trajetória dava-se por percorrer o caminho das imediações da praça Sesquicentenário até a proximidade do Restaurante Koxixos, cuja distância é de, aproximadamente, 1400 metros em linha reta. No percurso, incluiu-se a ida e a volta. A estação base foi instalada na praça, na altura mais elevada do terreno próximo da água.

A razão pela qual o experimento foi separado em duas etapas é de que cada uma fornece um cenário mais propício para aplicar um determinado conjunto de métricas para a avaliação do desempenho da proposta. Na primeira etapa, são analisadas a taxa de transferência (vazão), latência e a variação da latência (*jitter*). Na segunda etapa, são analisadas a taxa de entrega e o atraso fim-a-fim em um ambiente real com mobilidade.

A subseção a seguir trata da primeira etapa de manutenção, seguida da segunda etapa de mobilidade do agente móvel.

6.2.1 Etapa de Manutenção

Como mencionado anteriormente, a etapa de manutenção é realizada no pré voo e executa um determinado conjunto de comandos para validar a comunicação entre a estação base e o VANT. Inicialmente, é enviado um conjunto de mensagens com tamanhos diferentes a fim de medir a latência, cuja ilustração é feita no gráfico da Figura 25. A medição da latência é iniciada no momento em que a aplicação envia a mensagem para a API e termina no instante em que o *device driver* recebe o MAC-ACK do último *frame*. Os valores obtidos foram utilizados para ajustes das variáveis β, LBOUND e UBOUND conforme mencionado na seção 5.1.1.

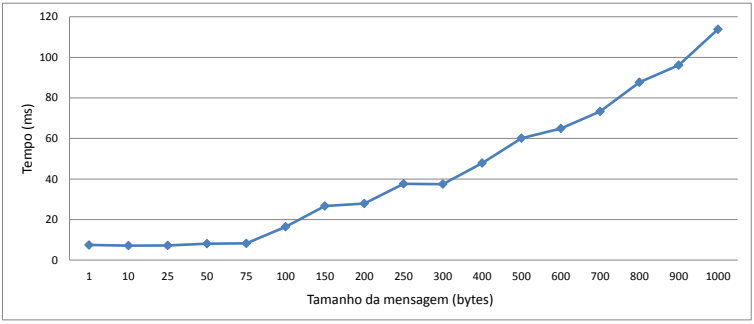

Figura 25 – Latência de transmissão entre a estação base e o VANT.

Para a medição do *jitter*, foi fixado o tamanho da mensagem em 10 bytes e variado o período de envio entre 1 ms e 1000 ms (Figura 26). É possível visualizar que conforme diminui o período de envio, maior é o *jitter* médio. Esse comportamento pode ser argumentado pela característica do Linux não ser um sistema operacional de tempo real.

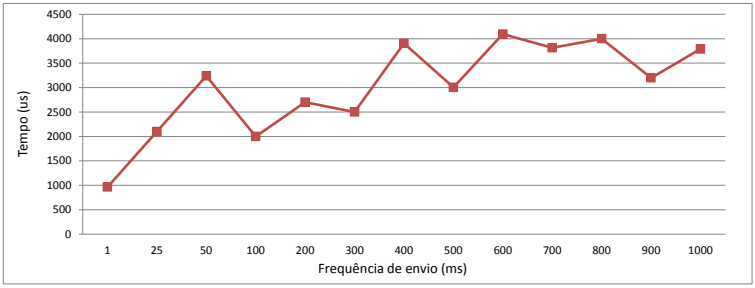

Figura 26 – *Jitter* de transmissão entre a estação base e o VANT.

Para medir a vazão, utilizou-se um arquivo de missão fictício com tamanho de 10 MB. Realizaram-se diversas transferências desse arquivo com a variação do tamanho da mensagem de dados que a aplicação transmitia ao 6LowPan. Para que houvesse essa variação, ficava a cargo da aplicação fragmentar o arquivo no tamanho determinado antes de enviá-lo ao 6LowPan e, também, de desfragmentar na recepção. Como mencionado anteriormente, sempre que a aplicação transmitir uma mensagem para o 6LowPan, ele também o fragmentará em múltiplos *frames* (se necessário) para adequar-se ao MTU de 127 bytes do 802.15.4. Cada fragmento do 6LowPan é enviado sem a confirmação da outra ponta, com exceção do MAC-ACK. Caso algum

único fragmento seja perdido, ele não será retransmitido e todos os demais recebidos pela outra ponta serão descartados pelo 6LowPan.

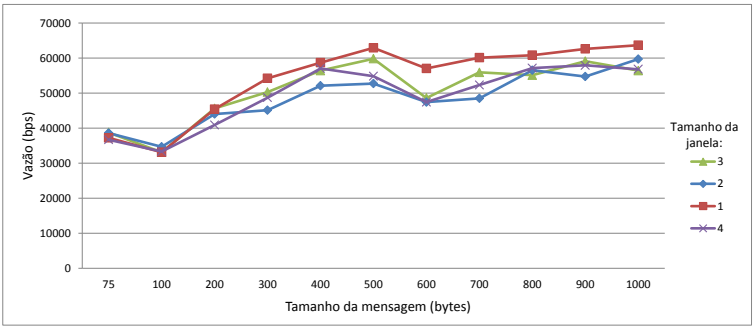

Figura 27 – Vazão na comunicação entre a estação base e o VANT.

Pode-se observar no gráfico da Figura 27 que, conforme é aumentado o tamanho da mensagem até o valor de 500 bytes, maior é a vazão apresentada. Após esse valor, a vazão tende a estabilizar-se no mesmo patamar. Tal comportamento ocorre pelo fato de que são necessárias um número menor de mensagens de confirmação a nível de aplicação, o que ocasiona menor *overhead*. Também é possível observar que o uso do tamanho da janela maior que um tem efeito negativo na vazão. Este fato ocorre, pois no momento em que o VANT tenta transmitir a mensagem de reconhecimento da aplicação, a estação base tenta transmitir a próxima mensagem que está alocada na janela de transmissão. A ocorrência múltipla desses eventos aumenta a probabilidade de colisões, no qual ocasionalmente, ocorrem perdas de transmissão.

Em seguida, repetiu-se o experimento, porém, em um *link* com a taxa de perda de 40%. Para induzir a probabilidade de perda, implementou-se no *device driver* do rádio, logo antes da escrita do *frame* na SPI, uma condição de que quando gerado um número aleatório e satisfeita a probabilidade de erro, alterava-se o campo PAN ID do *mac header* para um valor diferente do correto. Com essa estratégia, o canal era ocupado, porém a outra ponta rejeitava o *frame* por não pertencer ao mesmo PAN ID.

Observa-se, na Figura 28, que a vazão tem o comportamento contrário do experimento anterior. Ou seja, conforme há o aumento do tamanho da mensagem enviada ao 6LowPan, menor é a vazão. O motivo pelo qual ocorre esse comportamento é que quanto maior o tamanho da mensagem, maior é a probabilidade da perda de um único *frame* da mesma, o que acarreta no descarte dos demais *frames* já recebidos pelo 6LowPan.

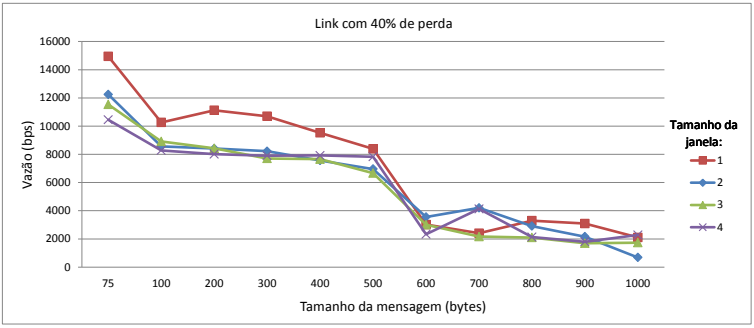

Figura 28 – Vazão na comunicação entre a estação base e o VANT, em um link com 40% de perda.

Fica claro observar que quando não há perdas na rede, é vantajoso a aplicação enviar ao 6LowPan blocos com tamanho maiores, pois um número menor de mensagens de reconhecimento da aplicação serão necessários. Entretanto, quando há perdas na rede, é mais vantajoso a aplicação enviar mensagens de tamanho menores, pois na perda de um único frame, não faz com que todos os demais já recebidos sejam descartados.

No seguinte experimento (Figura 29), foi fixado o tamanho da mensagem em 75 bytes e variado o tamanho da janela deslizante em um *link* com 20% de probabilidade de perda, porém agora, com um atraso inserido na rede de 50 ms.

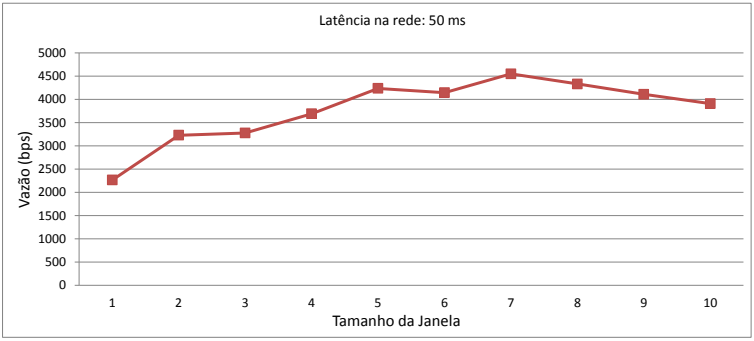

Figura 29 – Vazão na comunicação entre a estação base e o VANT, em um link com 20% de probabilidade de perda e 50 ms de atraso.

Constata-se que conforme aumenta-se o tamanho da janela, até o li-

Métrica	Consumo Médio (%)
Aplicação (1)	1
6LowPan (2)	0
Kernel Thread (3)	2
SPI (4)	1
Total*:	**4**

Tabela 4 – Consumo médio do teste de telemetria e transferência de dados. *Valores arredondados com precisão de 1%.

mite de 7, maior é a vazão apresentada. Assim, conclui-se que só há vantagem em utilizar o tamanho da janela maior que 1 quando há atraso na rede, caso contrário o número de colisões faz com que a vazão diminui.

Para o mesmo experimento da vazão, analisou-se também o consumo de CPU. A métrica, aferida para medição leva em consideração a soma individual de quatro critérios: (1) a aplicação em *user-space* que implementa a API de comunicação; (2) o módulo em *kernel-space* que implementa o 6LowPan; (3) a *thread* de *kernel* (*kworker*) que é criada para o tratamento de interrupção, *timers* e I/Os, e, finalmente, (4) ao *driver* de SPI utilizado para transmissão e recepção entre o *host* e o rádio. O uso individual de cada métrica pode ser visualizado na tabela 4. Conforme observado, o consumo total foi de aproximadamente 4%. Os valores foram obtidos de acordo com as ferramentas do próprio Linux com escala de precisão de 1%.

6.2.2 Etapa de Mobilidade do Agente Móvel

Estabeleceu-se duas rodadas de teste nessa segunda etapa, onde, em ambos, o agente móvel percorria os dois pontos a uma velocidade entre 15 e 30 km/h. O agente móvel foi configurado para transmitir a cada 1 segundo, sem o uso do *frame aggregation*, uma mensagem de telemetria com *payload* de 32 bytes: 128 bits para as coordenadas geográficas e 128 bits para a velocidade norte e oeste. O agente móvel e a estação base registravam em um arquivo de *log*, na memória *flash*, as operações, as quais incluíam a telemetria, os instantes de tempo, a indicação de qualidade do sinal (LQI), a potência do sinal (RSSI) e demais variáveis relacionadas a transmissão. A estação base, além de receber os dados, transmitia, periodicamente, uma mensagem de *ping* para monitoramento do enlace. Cada rodada foi realizada da seguinte maneira:

Primeira Rodada

As transmissões eram realizadas de modo *best-effort*, ou seja, sem utilizar os mecanismos de retransmissão a nível de aplicação. Para efeito comparativo, a rodada foi realiza duas vezes, uma com o MAC-ACK[1] habilitado e outra sem. O objetivo principal desta rodada, inicialmente, era o de tomar conhecimento do cenário de teste escolhido, fator importante para posterior comparação com os resultados da rodada descrita a seguir;

Segunda Rodada

As transmissões eram realizadas de modo confiável, ou seja, com os mecanismos desenvolvidos para retransmissões a nível de aplicação. O registro de *log* era feito da mesma forma que o teste acima, porém o agente móvel armazenava informações adicionais, como o *round-trip time*, o valor de *timeout* de retransmissão (RTO) e o número de retransmissões de cada mensagem.

Na primeira rodada do experimento, sem a utilização dos mecanismos de retransmissão a nível de aplicação, o agente móvel levou cerca de 20 minutos para percorrer duas vezes o trajeto de ida e volta. O primeiro trajeto foi realizado com o MAC-ACK habilitado, já o segundo o teve desabilitado. Quando não havia sucesso na transmissão da mensagem, ela era simplesmente descartada. A Figura 30 mostra a taxa média de entrega das mensagens, classificada a cada 100 metros. Os indicadores de RSSI e LQI são denotados na Figura 31. A partir dos dados obtidos, comprova-se que o uso apenas do mecanismo de retransmissão a nível de enlace não é suficiente para garantir a entrega de 100% dos dados, e que dessa forma necessita-se de um mecanismo de retransmissão na camada superior.

Após a conclusão da primeira rodada do experimento, o mesmo foi repetido, porém, usou-se o mecanismo de retransmissão a nível de aplicação. Nessa segunda etapa, quando não havia sucesso na transmissão da mensagem, ela era armazenada no *buffer* para posterior retransmissão. Em todo o percurso, foram transmitidas 489 mensagens. Como esperado, ao final da trajetória, 100% das mensagens foram recebidas integralmente pela estação base, independente da distância que elas foram geradas. O número de retransmissões computado foi de 219, aproximadamente 44,8%. Ou seja, quase metade das mensagens tiveram que ser retransmitidas. Logo, sem o mecanismo de retransmissão, o protocolo só alcançaria a taxa de 100% de entrega a uma

[1] É importante ressaltar que as retransmissões MAC-ACK são realizadas a nível enlace e de forma automática pelo hardware do MRF24J40MC. Para cada retransmissão, o rádio aguarda por 912 μs pela chegada no ACK. O MAC do rádio foi configurado para 3 tentativas de retransmissão. Se todas tentativas falharem, o frame é então descartado.

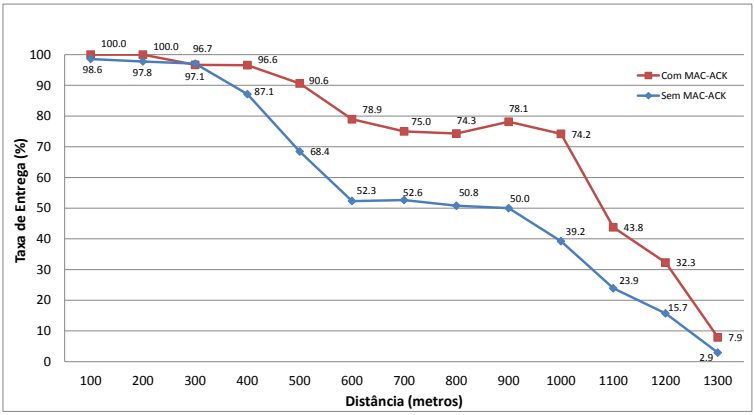

Figura 30 – Taxa de entrega das mensagens na primeira etapa sem a utilização de retransmissão a nível de aplicação.

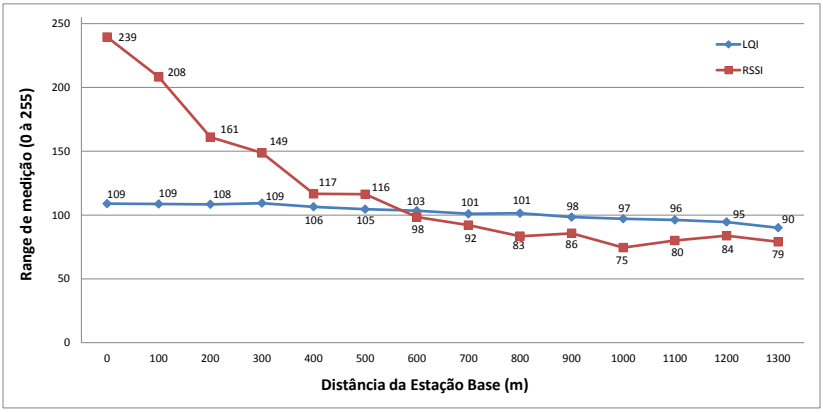

Figura 31 – Indicadores de RSSI e LQI da primeira etapa.

distância entre 0 a 200 metros.

O gráfico da Figura 32 relaciona três variáveis importantes, as quais são associadas ao uso do algoritmo de retransmissão. A variável *Transmissões* denota o número total de mensagens que foram transmitidas, incluindo as entregues e as tentativas de retransmissões. A variável *Entregues* é o número de mensagens entregues com sucesso e *Retransmissões* é o número de tentativas de retransmissões de mensagens que não receberam o ACK. O gráfico mostra a média das variáveis classificadas a cada 200 metros de forma

contínua no tempo (trajeto de ida e volta). Já o gráfico da Figura 33 relaciona a taxa de entrega pela distância e RSSI.

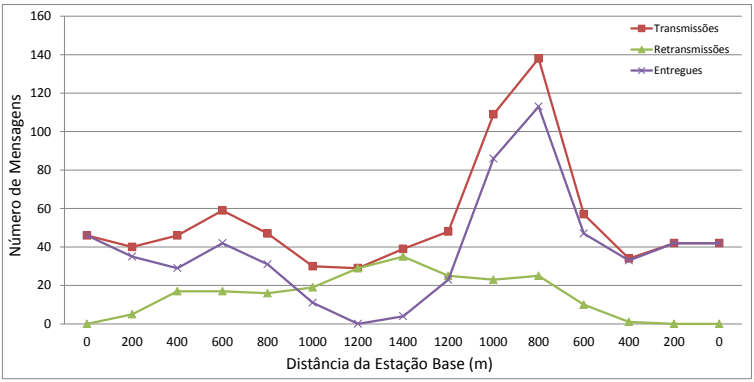

Figura 32 – Número de transmissões, retransmissões e mensagens entregues da etapa 2. Com o mecanismo de retransmissão, a taxa de entrega foi de 100% ao fim do percurso.

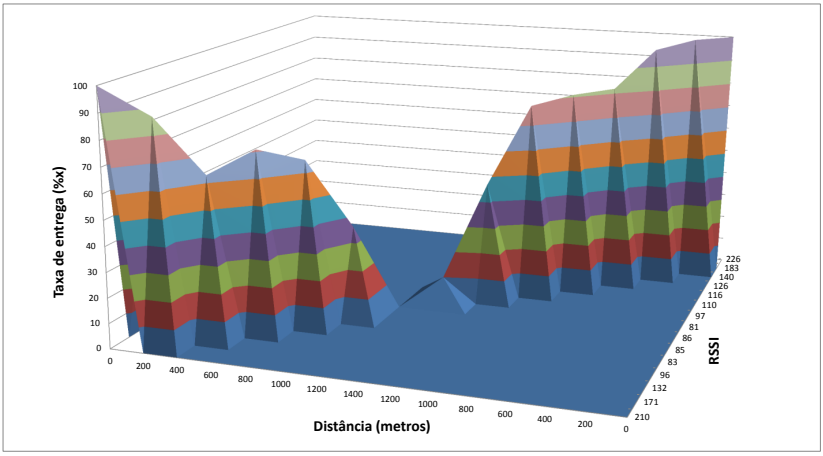

Figura 33 – Taxa de entrega relacionada pela distância e RSSI no trajeto de ida e volta. Ao final do percurso, a taxa de entrega foi de 100%.

Observa-se que, nos primeiros 900 metros do gráfico, o número de mensagens entregues é considerável. A partir dos primeiros 1000 metros do trajeto ida, aos seguintes 1200 metros da volta, a curva de entrega ficou em um

patamar reduzido e, como esperado, a curva de tentativas de retransmissões em um patamar elevado. Depois, próximo aos 800 metros da volta, constata-se um pico no número de mensagens totais e entregues. Esse elevado número de mensagens são as que estavam armazenadas no *buffer* geradas no período anterior, em que as mensagens não conseguiram ser entregues devido à longa distância da estação base. Ao se aproximar da estação base, próximo aos 400 metros, praticamente todas mensagens acumuladas no *buffer* foram entregues. A partir desse período até a chegada em 0 metros, não houve perdas de transmissão. Um histograma completo dos atrasos fim-a-fim inseridos no sistema, devido à perda e retransmissões das mensagens, pode ser visualizado na Figura 34. O atraso médio total das mensagens foi de 5 segundos.

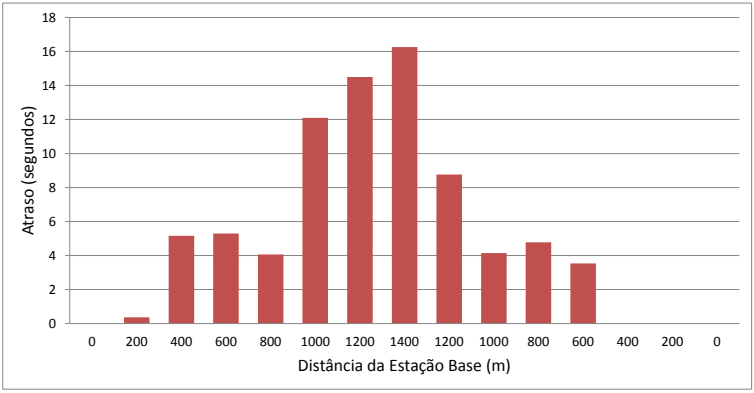

Figura 34 – Histograma dos atrasos inseridos no sistema devido a perda de mensagens e retransmissões.

6.3 COMUNICAÇÃO DO AGENTE MÓVEL COM A RSSF

Para que fosse realizado o experimento, o sensor MICAz foi posicionado no ponto de passagem do agente móvel, em uma disposição no terreno onde fosse possível um movimento retilíneo uniforme a uma velocidade de 25 km/h, passando a uma distância de 5 metros do sensor. Devido ao curto alcance do sensor, a trajetória completa do agente móvel pode ser reduzida para 250 metros. O ponto de partida foi estabelecido a 125 metros do sensor e o ponto de chegada a 125 metros após o sensor.

O sensor foi programado para transmitir 27 bytes de *payload* por mensagem e com uma quantidade ilimitada de mensagens no *buffer*. Além do

payload, cada mensagem era composta pela estrutura apresentada na seção 5.2.2. No *payload*, constavam os seguintes dados: 1) o número de tentativas de retransmissões; 2) a latência computada no envio anterior e 3) o número de mensagens perdidas.

No momento em que o sensor recebia o *broadcast* de requisição, iniciava a transmissão das mensagens em *unicast*, até que alguma mensagem excedesse o número de tentativas de retransmissão. Os dados recebidos eram armazenados no banco de dados conforme especificado na seção da proposta.

O gráfico da Figura 35 mostra o número de mensagens coletadas a cada 10 metros de forma contínua no tempo. Ao total do percurso, foram transmitidos 42 *broadcasts* e coletadas 2252 mensagens, o que totaliza a taxa de entrega em 98.17%. Vale ressaltar que o *broadcast* é enviado após o VANT permanecer 150ms sem o recebimento de nenhum *frame* do sensor, o que indica que o sensor excedeu o número de 3 retransmissões sem sucesso.

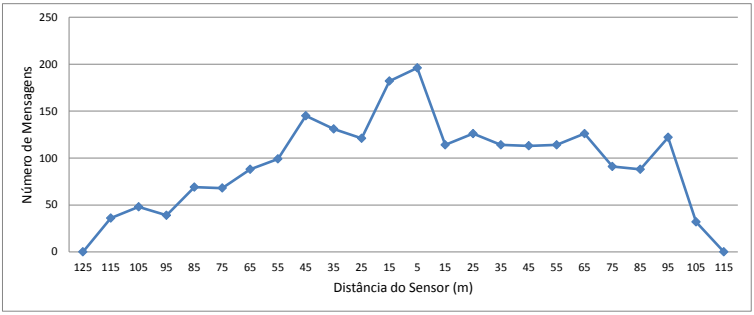

Figura 35 – Número de mensagens coletadas.

O gráfico da Figura 36 apresenta o indicador de intensidade do sinal recebido (RSSI). Como esperado, a intensidade mostrou-se maior conforme a aproximação do sensor até o pico máximo ao ponto mais próximo do mesmo.

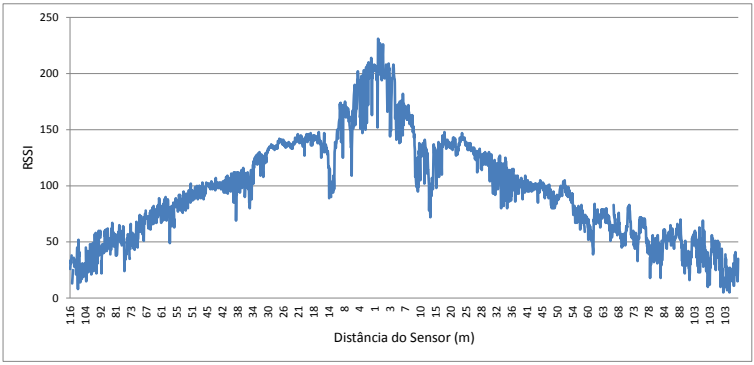

Figura 36 – Indicador de intensidade do sinal recebido (RSSI).

O gráfico da Figura 37 apresenta o indicador de qualidade do enlace (LQI). É possível observar que a a intensidade do sinal (RSSI) tem pouca relação sobre o LQI.

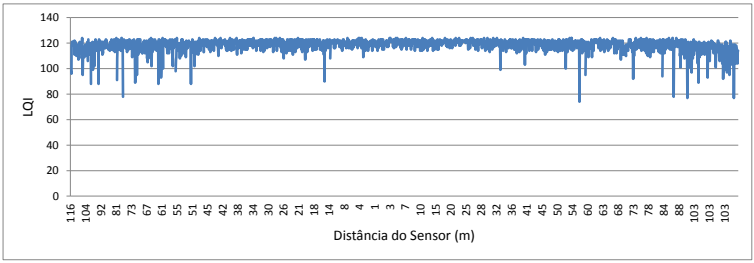

Figura 37 – Indicador de qualidade do enlace (LQI).

O gráfico da Figura 38 mostra a latência das mensagens transmitidas pelo sensor. A medição compreende-se na diferença de tempo entre o pacote ser enviado para o transceptor, até o recebimento do ACK a nível de MAC. A linha na cor preta do gráfico denota a curva de tendência polinomial. Como esperado, média da latência é maior conforme a distância que o agente móvel se encontra do sensor.

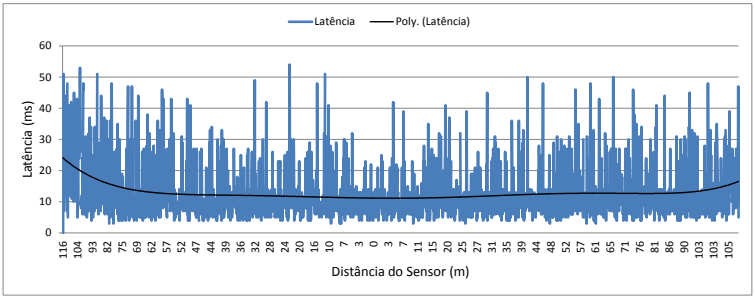

Figura 38 – Latência de transmissão do sensor.

O gráfico da Figura 39 mostra a *jitter* das transmissões do sensor. Conforme observado no caso da latência, a média do *jitter* é maior de acordo com a distância em que o agente móvel se encontra do sensor.

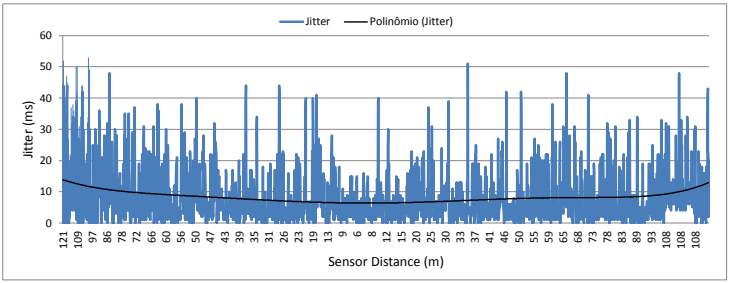

Figura 39 – *Jitter* de recepção dos pacotes.

7 CONCLUSÃO

O uso de VANTs tem atraído diversas pesquisas devido a seus benefícios e sinergias tanto na indústria quanto na academia. Em muitas aplicações, é possível imaginar o uso de VANTs como agentes móveis para coleta de dados em uma RSSF. Nesse contexto, dois cenários são vislumbrados, os quais são a comunicação do VANT com a estação base e a comunicação com a RSSF.

Essa dissertação propõe uma infraestrutura de comunicação sem fio designada aos dois cenários vislumbrados. A infraestrutura consiste no uso de uma única interface de comunicação IEEE 802.15.4 capaz de operar na plataforma embarcada do VANT. Além disso, ela oferece um conjunto de protocolos específico que opera de modo distinto de acordo com as propriedades de comunicação de cada cenário.

Na comunicação do VANT com a estação base, são disponibilizados um conjunto de protocolos que oferece confiabilidade e segurança na entrega das mensagens em uma rede IP. Por outro lado, na comunicação com a RSSF é ofertado um conjunto de protocolos que minimiza o consumo de recursos da rede. Para possibilitar a comunicação com tais dispositivos potencialmente heterogêneos com o uso do mesmo hardware, é proposto uma pilha de protocolo *dual-stack*. A abordagem faz o uso do 6LowPan como base de desenvolvimento para a comunicação com a estação base e, o uso do mecanismo de *raw sockets* do Linux para comunicação com a RSSF.

A proposta introduz um protocolo de comunicação confiável sobre o UDP para ser utilizado em uma rede IEEE 802.15.4 com alta taxa de perda. O protocolo é baseado na técnica de janela deslizante com reconhecimento seletivo para transmissão da mensagens e no algoritmo de Jacobson e Karn para cálculo de *timeout*. Ele pode ser utilizado como uma alternativa ao TCP, o qual possui baixa eficiência e performance em tais redes, devido ao seu mecanismo de controle de congestionamento e tamanho do cabeçalho.

O desenvolvimento e estudo do protocolo de comunicação confiável aponta para outros trabalhos de investigação futuros. Diferentes técnicas para o controle do tamanho da janela deslizante podem ser aplicadas com base em diversos estudos na área de transmissão sem fio em *links* com alta taxa de perda. Com isso, nota-se que há espaço para pesquisa de um mecanismo que possa realizar o ajuste dinâmico do tamanho da janela em tempo de execução, na qual, por exemplo, o tamanho da janela poderia ser aumentado em decorrência da taxa de perda e do atraso da rede. Além disso, constatou-se a necessidade de um mecanismo de medição da qualidade do enlace com o objetivo de regular dinamicamente o tamanho da mensagem enviada ao 6Low-

Pan a fim de manter a maior taxa de vazão possível.

Na comunicação do VANT com a RSSF, idealizou-se um protocolo que visa a economia de recursos da rede de sensores que, em conjunto com o mecanismo de coleta desenvolvido, mostrou-se eficiente na coleta dos dados do *cluster-head* de uma RSSF. A partir da infraestrutura de coleta desenvolvida, aponta-se espaço para pesquisa de um mecanismo que possa realizar o ajuste dinâmico do *timeout* de coleta de acordo com as condições da rede. Também há campo para implementação de diferentes métodos de coleta.

Uma outra contribuição deste trabalho é o desenvolvimento de uma API de abstração que omite a complexidade para lidar com ambos protocolos. Dessa forma, a aplicação pode utilizar a comunicação confiável e/ou com economia de recursos de acordo com os requisitos.

Ao final dos experimentos práticos, verificou-se que o modelo da arquitetura *dual-stack* é viável de ser utilizado de maneira eficiente para a comunicação de um VANT de curto alcance em aplicações que operam com de baixa largura de banda. Além disso, aplicações em outros domínios também poderão beneficiar-se da infraestrutura de comunicação sem fio desenvolvida como, por exemplo, robótica, automotiva, sensores de monitoramento e etc.

REFERÊNCIAS

ADAMS, J. An introduction to ieee std 802.15.4. In: **Aerospace Conference, 2006 IEEE**. [S.l.: s.n.], 2006. p. 8 pp.–.

ARAUJO, G. de; BECKER, L. A network conditions aware geographical forwarding protocol for real-time applications in mobile wireless sensor networks. In: **Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on**. [S.l.: s.n.], 2011. p. 38–45. ISSN 1550-445X.

AYADI D. ROS, L. T. A. **TCP header compression for 6LoWPAN**. [S.l.], July 2010. 1-28 p. Disponível em: <http://tools.ietf.org/html/draft-aayadi-6lowpan-tcphc-00>.

BALAKRISHNAN, H. et al. A comparison of mechanisms for improving tcp performance over wireless links. **Networking, IEEE/ACM Transactions on**, v. 5, n. 6, p. 756–769, 1997. ISSN 1063-6692.

BANATRE, M. **Cooperating Embedded Systems and Wireless Sensor Networks**. [S.l.]: Wiley, 2008. (Iste Series). ISBN 9781848210004.

BARRENETXEA, G. et al. Wireless sensor networks for environmental monitoring: The sensorscope experience. In: **Communications, 2008 IEEE International Zurich Seminar on**. [S.l.: s.n.], 2008. p. 98–101.

CHAKRABARTI, A.; SABHARWAL, A.; AAZHANG, B. Using predictable observer mobility for power efficient design of sensor networks. In: **in The second International Workshop on Information Processing in Sensor Networks (IPSN**. [S.l.: s.n.], 2003. p. 129–145.

DIOS, J. R. M. de et al. Cooperation between uas and wireless sensor networks for efficient data collection in large environments. **Journal of Intelligent and Robotic Systems**, p. 491–508, 2013.

FREITAS, E. P. de et al. Middleware support in unmanned aerial vehicles and wireless sensor networks for surveillance applications. In: **Intelligent Distributed Computing III**. [S.l.]: Springer, 2009. p. 289–296.

FREITAS, E. Pignaton de et al. Uav relay network to support wsn connectivity. In: **Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on**. [S.l.: s.n.], 2010. p. 309–314. ISSN 2157-0221.

GONCALVES, F. et al. Small scale uav with birotor configuration. In: **Unmanned Aircraft Systems (ICUAS), 2013 International Conference on**. [S.l.: s.n.], 2013. p. 761–768.

GROSSGLAUSER, M.; TSE, D. Mobility increases the capacity of ad hoc wireless networks. **Networking, IEEE/ACM Transactions on**, v. 10, n. 4, p. 477–486, 2002. ISSN 1063-6692.

HUMMEN, R. et al. 6lowpan fragmentation attacks and mitigation mechanisms. In: **Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks**. New York, NY, USA: ACM, 2013. (WiSec '13), p. 55–66. ISBN 978-1-4503-1998-0.

IEEE Standard 802.15.4. 2011. `http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf`.

JACOBSON, V. Congestion avoidance and control. In: **In Proceedings of SIGCOMM '88**. New York, NY, USA: ACM, 1988. p. 314–329.

KANSAL, A. et al. Intelligent fluid infrastructure for embedded networks. In: **In Proc. ACM MobiSys'04**. [S.l.: s.n.], 2004.

KOUBâA, A.; ALVES, M.; TOVAR, E. Ieee 802.15.4 for wireless sensor networks: A technical overview. 2005.

LEE, B.-H. et al. Study on additional carrier sensing for ieee 802.15.4 wireless sensor networks. **Sensors**, v. 10, n. 7, p. 6275–6289, 2010. ISSN 1424-8220.

LU, G.; KRISHNAMACHARI, B.; RAGHAVENDRA, C. Performance evaluation of the ieee 802.15.4 mac for low-rate low-power wireless networks. In: **Performance, Computing, and Communications, 2004 IEEE International Conference on**. [S.l.: s.n.], 2004a. p. 701–706.

LU, G.; KRISHNAMACHARI, B.; RAGHAVENDRA, C. Performance evaluation of the ieee 802.15.4 mac for low-rate low-power wireless networks. In: **Performance, Computing, and Communications, 2004 IEEE International Conference on**. [S.l.: s.n.], 2004b. p. 701–706.

MANJESHWAR, A.; AGRAWAL, D. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In: **Parallel and Distributed Processing Symposium., Proceedings 15th International**. [S.l.: s.n.], 2001. p. 2009–2015. ISSN 1530-2075.

MATHIS JAMSHID MAHDAVI, S. F. A. R. M. **TCP Selective Acknowledgement Options**. [S.l.], October 1996. Disponível em: <http://www.ietf.org/rfc/rfc793.txt>.

MORI, O. N. **TCP HollyWood**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2005.

PUCCINELLI, D.; HAENGGI, M. Wireless sensor networks: applications and challenges of ubiquitous sensing. **Circuits and Systems Magazine, IEEE**, v. 5, n. 3, p. 19–31, 2005. ISSN 1531-636X.

RAZA, S. et al. Securing communication in 6lowpan with compressed ipsec. In: **Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on**. [S.l.: s.n.], 2011. p. 1–8.

SHAH R.C., R. S. J. S.; BRUNETTE, W. **Data MULEs: modelling a three-tier architecture for sparse sensor networks**. [S.l.], 2003.

SHELBY, Z.; BORMANN, C. **6LoWPAN: The Wireless Embedded Internet**. [S.l.]: Wiley Publishing, 2010. ISBN 0470747994, 9780470747995.

TANENBAUM, A. **Computer Networks**. 4th. ed. [S.l.]: Prentice Hall Professional Technical Reference, 2002. ISBN 0130661023.

TEH, S. K. et al. Experiments in integrating autonomous uninhabited aerial vehicles (uavs) and wireless sensor networks. The Australian Robotics and Automation Association Inc., 2008.

YE, D.; GONG, D.; WANG, W. Application of wireless sensor networks in environmental monitoring. In: **Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference on**. [S.l.: s.n.], 2009. v. 1, p. 205–208.

YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. **Computer Networks**, v. 52, n. 12, p. 2292 – 2330, 2008. ISSN 1389-1286.

ZHANG, P. et al. Hardware design experiences in zebranet. In: **Proceedings of the 2nd international conference on Embedded networked sensor systems**. New York, NY, USA: ACM, 2004. (SenSys '04), p. 227–238. ISBN 1-58113-879-2. Disponível em: <http://doi.acm.org/10.1145/1031495.1031522>.

ZHAO, F.; GUIBAS, L. **Wireless Sensor Networks: An Information Processing Approach**. [S.l.]: Morgan Kaufmann, 2004. (Electronics & Electrical). ISBN 9781558609143.

ZHENG, J.; LEE, M. J. **A comprehensive performance study of IEEE 802.15.4**. [S.l.]: IEEE Press Book Los Alamitos, 2004.