INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores

Computação na periferia na Internet das Coisas

Gonçalo Correia de Almeida da Silva Esteves

Licenciado

Dissertação para obtenção do Grau de Mestre
em Engenharia de Eletrónica e Telecomunicações

Orientador : Professor Nuno Cruz

Júri:

Presidente: Professor Rui António Policarpo Duarte

Vogais: Professor Nuno Miguel Machado Cruz
Professor Nuno António Juliano Cota

Setembro, 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores

Computação na periferia na Internet das Coisas

Gonçalo Correia de Almeida da Silva Esteves

Licenciado

Dissertação para obtenção do Grau de Mestre
em Engenharia de Eletrónica e Telecomunicações

Orientador : Professor Nuno Cruz

Júri:

Presidente: Professor Rui António Policarpo Duarte

Vogais: Professor Nuno Miguel Machado Cruz
Professor Nuno António Juliano Cota

Setembro, 2023

Agradecimentos

Gostaria de começar por agradecer ao meu Orientador, Professor Doutor Nuno Cruz, por me ter orientado nesta minha nova aventura das redes de longo alcance LoRaWAN e pela disponibilidade que teve ao longo deste projeto.

Agradeço a toda a equipa do Future Internet Technologies, do Instituto Superior de Engenharia de Lisboa, pelo aconselhamento que me foram dando ao longo das reuniões de projeto, em particular aos professores Nuno Cota, Nuno Datia e José Simão, e aos bolseiros que me acompanharam.

Agradeço aos meus pais e irmã Paulo, Mafalda e Mariana e à minha namorada Cláudia que também ficou a saber tudo sobre comunicações LoRaWAN e me ajudou ao longo deste projeto. Agradeço também aos meus familiares.

Não poderia deixar de agradecer à Solvit e à sua equipa, por me autorizar a testar os componentes deste projeto cedendo-me espaço, equipamentos, paciência e ajuda.

A todos, o meu muito obrigado!

Resumo

A tecnologia LoRaWAN foi desenvolvida com o intuito de permitir sensores IoT de baixo custo e baixa capacidade energética, transmitir pequenas mensagens, a longas distâncias. A existência desta tecnologia e o facto de se comprovar ser bastante versátil permite, assim, o afastamento do âmbito apenas do IoT e colocá-la também em sistemas críticos, como, por exemplo, sistemas de segurança associados à infraestrutura ferroviária. No entanto, neste cenário, seria necessário tornar mais eficaz o processamento das mensagens, que, eventualmente, se conseguiria através da computação na periferia. No entanto, a arquitetura das redes lorawan impedem este tipo de acelaração do processamento. Este projeto propõe uma solução para colmatar a dificuldade de integração da tecnologia LoRaWAN com computação na periferia. Através do desenvolvimento de um sistema de coordenação entre múltiplos sistemas de computação na periferia, capazes de funcionar como gateways LoRaWAN, é possível processar as mensagens localmente, sem estas terem de ser enviadas para um elemento central na núvem. De forma a aumentar a resiliência da solução, os sistemas de computação na periferia incluem múltiplas ligações a redes móveis, desenvolvidas no sentido de encaminhar o tráfego do serviço de forma resiliente e de acordo com as prioridades definidas. Com o intuito de aumentar a segurança da solução, este trabalho propõe, também, a implementação do algoritmo *K-nearest neighbors*, para deteção de interferências nos sensores que monitorizam as infraestruturas críticas.

Palavras-chave: IoT, LoRaWAN, Edge, LoRa

Abstract

The LoRaWAN technology was designed with the objective of allowing low cost and low energetic capacity iot sensors to transmit short messages over long distances. The existence of this technology and the fact that it has proven to be quite versatile allows, thus, the deviation from the realm of just IoT and also places it in critical systems, such as, for example, security systems associated with the railway infrastructure. Although, in this scenario, it would be necessary to make message processing more efficient, which could, eventually, be achieved through computing at the edge. However, LoRaWAN network infrastructure hinder this type of processing acceleration. This project presents a solution to overcome the difficulty of integrating LoRaWAN technology with computing at the periphery. By developing a coordination system between multiple edge computing systems, capable of functioning as LoRaWAN gateways, it is possible to process messages locally, without them having to be sent to a central element in the cloud. In order to increase the resilience of the solution, computing systems at the edge include multiple connections to mobile networks, developed to route service traffic in a resilient manner and in accordance with defined priorities. In order to increase the security of the solution, this work also proposes the implementation of the knn algorithm to detect interference in sensors that monitor critical infrastructures.

Keywords: IoT, LoRaWAN, Edge, LoRa

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

3GPP *3rd Generation Partnership Project*. 11, 12

ABP *Activation By Personalization*. 8, 14, 17, 25, 30, 31
API *Application Programming Interface*. 13

devEUI *Device Extended Unique Identifier*. 16
DSCP *Differentiated Services Code Point*. 17, 19

IDS *Intrusion Detection System*. xvi, 3, 34, 35
IoT *Internet of Things*. vii, ix, 1, 2, 4, 5, 11, 20, 45
IP *Internet Protocol*. xii, 17
ISM *Industrial, Scientific and Medical*. 6, 11

JSON *JavaScript Object Notation*. 29

KNN *K-nearest neighbors*. vii, 34, 41, 43, 45

LoRa *Long Range*. xv, 6, 7, 12, 13, 15, 16, 23, 26, 29, 37, 40
LoRaWAN *Long Range Wide Area Network*. vii, ix, xv, 1, 2, 4, 6, 9, 11, 12, 13, 14, 21, 23, 24, 25, 26, 29, 31, 38, 39, 40, 45
LPWAN *Low Power Wide Area Network*. 5
LTE *Long Term Evolution*. xv, 9, 10

M2M *Machine to Machine*. 17
MAC *Media Access Control*. 6

MEC *Multi-Access Edge Computing*. 9, 11, 20, 21
MIC *Message Integrity Code*. 7, 8, 16
MQTT *Message Queuing Telemetry Transport*. xii, 8, 13, 14, 17, 18, 19, 26, 28, 29, 39

NB-IoT *Narrow Band Internet of Things*. 9, 10, 11, 12
NR5G *5G New Radio*. xv, 9, 10, 11

OBU *On Board Unit*. 37
OTAA *Over the Air Activation*. xv, 8, 14, 17, 25, 30, 31, 41

QoS *Quality of Service*. 18, 19

TCP *Transmission Control Protocol*. 18, 19
ToS *Type of Service*. 17
TTN *The Things Network*. 16

UDP *User Datagram Protocol*. 12

VM *Virtual Machine*. xvi, 26, 38
VPN *Virtual Private Network*. 26, 27, 39

1

Introdução

Hoje em dia, com o crescente uso da Internet das Coisas (IoT - *Internet of Things*), cada vez mais existe a necessidade de uma rápida resposta por parte dos sistemas de processamento de dados. A computação instantânea no IoT, ideal para aplicações críticas, deve ser próxima para uma baixa latência, mas também tem de permitir a compatibilidade com os sensores, atuadores e aplicações existentes.

A IoT surge como a interligação de coisas do quotidiano como luzes, interruptores, sensores de temperatura, que podem ser facilmente adquiridos e instalados em ambiente residencial. Este tipo de aplicações existe num ambiente não crítico, onde a falha ou o atraso na comunicação entre estes equipamento não representa riscos, apenas incómodo. Sendo estes equipamentos de baixo custo, existe a tendência de transportar estes tipos de soluções para ambientes mais críticos, onde a latência se mostra um fator crucial para a tomada de decisões, surgindo assim esta tese que propõe o desenvolvimento de um sistema de computação na periferia, para sistemas críticos, onde seja possível manter todas as funcionalidades já existentes em ambiente normal, mas permitindo reduzir a latência na transmissão dos dados e na tomada de decisões.

A tecnologia LoRaWAN, como um método de comunicação IoT de baixo custo, baixo consumo e de baixo débito é uma das tecnologias mais implementada no IoT. A aplicação desta tecnologia representa um avanço significativo no domínio dos métodos de comunicação, particularmente no domínio da Internet das Coisas (IoT). Estas propriedades tornam a tecnologia LoRaWAN particularmente ideal para comunicações com sensores que necessitam de transmitir quantidades reduzidas de informação de cada

vez, tais como acelerómetros, sistemas de geolocalização, sensores de temperatura, entre outros. Por outro lado, computação na periferia representa uma abordagem em que é possível processar e realizar ações com a informação recebida, no local onde esta é recebida, por parte de um equipamento que comunica com este. Esta abordagem permite assim reduzir a latência, usualmente derivada à distância desde o equipamento que fez a receção dos dados até ao equipamento encarregue de os processar; não esquecendo também que, pelo percurso, pode ser necessário a informação passar por outros sistemas para desencriptação/encriptação.

1.1 Motivação

Num mundo cada vez mais interligado e dependente da tecnologia, o IoT emergiu como uma revolução que transforma a maneira como interagimos com o nosso ambiente, desde dispositivos domésticos como fogões e frigoríficos até a máquinas industriais. À medida que se torna uma parte intrínseca das nossas vidas, começa a surgir a ideia do uso do IoT para aplicações mais críticas, onde a latência pode significar riscos para a vida humana e valores avultados de prejuízos. Para enfrentar esses desafios, surge a necessidade de um sistema de comunicação e computação na periferia que seja eficiente e capaz de funcionar de forma autónoma, de forma a reduzir a latência da receção de dados provenientes de equipamentos IoT. Este sistema deve ser capaz de receber, processar e responder a informações sem depender de conexões externas constantes e de implementar um protocolo de comunicação IoT já existente, para que seja compatível com aplicações e sensores já instalados. Existindo já sensores LoRaWAN para aplicações mais críticas como sensores de taludes para a ferrovia que comunicam com uma plataforma central informações críticas, como por exemplo a queda desse talude, aumenta assim a necessidade de ser instalado um sistema de computação na periferia diretamente no comboio, de forma a que esta informação possa ser recebida imediatamente pelo maquinista do comboio que se dirige para o local perigoso.

1.2 Objetivos

O objetivo desta tese é o desenvolvimento dos mecanismos necessários para tornar possível a existência de computação na periferia em redes LoRaWAN, nomeadamente, permitir ao equipamento que recebe o pacote LoRaWAN conseguir fazer o processamento da informação e despoletar ações. Para isso, será necessário desenvolver mecanismos de coordenação entre os gateways LoRaWAN, que passam a estar ligados ao

computador de fronteira de forma a permitir o registo e decifra de mensagens oriundas de sensores e de não só processar essas mensagens, como enviar também para um sistema centralizado através de mecanismos resilientes, que possibilitem o encaminhamento do tráfego por diversas rotas. Por último, pretende-se aumentar a segurança da rede com a implementação de um sistema de deteção de intrusões (IDS), de forma a aumentar a segurança dos pacotes recebidos pelo computador de periferia.

1.3 Avaliação

A avaliação deste trabalho irá consistir em testes práticos, com dois computadores de periferia, um servidor central e sensores. Com estes equipamentos em funcionamento, irá ser possível avaliar as autenticações efetuadas pelos sensores, sincronizações entre o servidor central e os computadores de periferia. Com estes equipamentos, será também instalado e treinado o sistema de deteção de intrusões, sendo este sistema avaliado conforme os parâmetros de sistemas de machine learning para este fim, nomeadamente F1 Score, Recall e Precisão.

1.4 Contribuições

A Agência Nacional de Inovação financiou o projeto Ferrovia 4.0 (Ref LISBOA-01-0247-FEDER-046111 & POCI-01-0247-FEDER-046111), que apoiou a elaboração desta tese aqui apresentada, disponibilizando as ferramentas de hardware necessárias à prova de conceito e testes às tecnologias inerentes a este projeto de investigação. Esta infraestrutura esteve exposta no Portugal Railway Summit 2023 no Entroncamento e na apresentação final do projeto, no ISEP, onde obteve uma avaliação positiva por parte do público.

Existe um artigo com o títluo "Implementing LoRaWAN Edge Computing Intrusion Detection to Safeguard Tourist Trail Emergency Response", submetido e em estado de avaliação pela revista MDPI Sensors, que resulta da implementação e dos testes efetuados ao sistema de IDS no decorrer desta tese.

1.5 Estrutura do documento

Este documento encontra-se organizado em 5 capítulos, sendo o primeiro capítulo destinado à introdução. No Capítulo 2 é descrito o estado da arte, onde são apresentadas

várias tecnologias de suporte a dispositivos IoT e a redes de transporte no caso do protocolo LoRaWAN. No Capítulo 3 é apresentado o trabalho realizado, onde são referidas as várias partes que foram desenvolvidas para ser possível um sistema de computação na periferia usando a tecnologia LoRaWAN. Por fim, no Capítulo 4, são apresentados os resultados da implementação do sistema desenvolvido.

2

Estado da Arte

Neste capítulo, serão identificadas as tecnologias existentes que permitem criar redes de sensores e atuadores IoT. Tendo como objetivo o desenvolvimento e demonstração de uma federação de sistemas de computação na periferia, é importante a existência de um sistema de rede de sensores que permita não só que o número de dispositivos intermédios entre o sensor e o computador de fronteira seja o menor possível, como também que seja de baixo custo e compatível com dispositivos já existentes, alguns já instalados. Será também devidamente estudada a forma de comunicação entre o computador de fronteira (um computador que se encontra instalado no local de receção dos dados) e o nó central. Isto resulta da necessidade identificada de haver uma coordenação central entre todos os computadores de fronteira instalados. Serão abordadas as tecnologias de rede LPWAN, as redes de transporte capazes de receber os dados dos sensores e fazer o seu envio para outros sistemas e a tecnologia atualmente existente, que de alguma forma conseguiria fazer alguma realização de computação na periferia.

2.1 Redes LPWAN

As redes LPWAN, são redes de longo alcance e de baixo consumo energético para dispositivos IoT transmitirem dados para um sistema central que os irá processar posteriormente. Este tipo de redes surgiu da necessidade de tecnologias que permitissem não só uma transmissão de longo alcance, mas também que não necessitassem de muita energia e que fossem de baixo custo. É possível com este conceito de rede abranger

sensores com grande alcance e, por ser uma tecnologia de baixo consumo, pode permitir que um sensor tenha uma duração de bateria muito longa, que pode chegar até aos 10 anos [1].

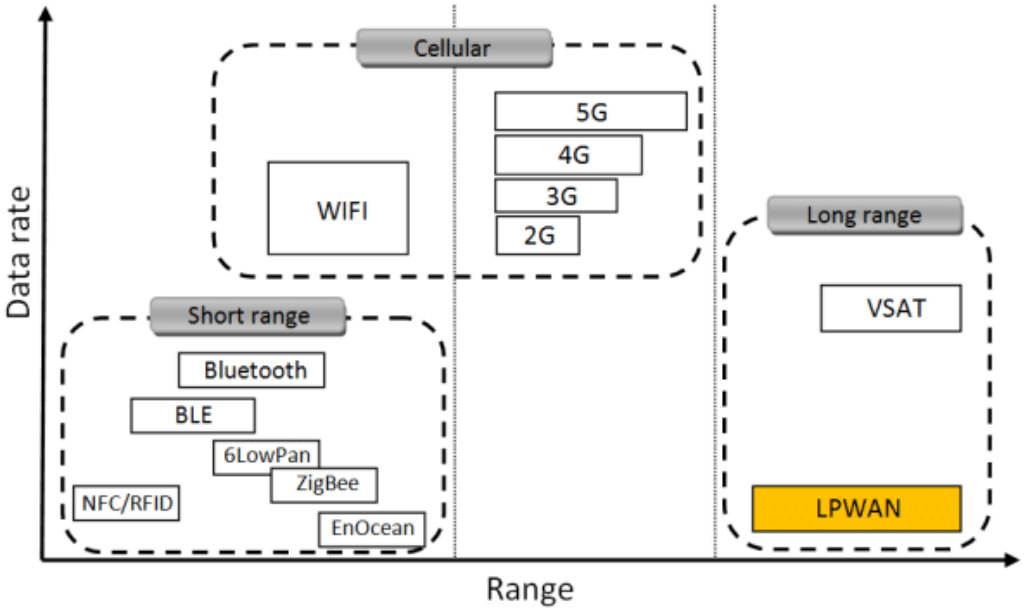

Figura 2.1: Comparação das tecnologias LPWAN. (Retirado de [2]).

Na figura 2.1, é possível observar as diferenças nas distância e ritmo de transmissão das diversas tecnologias assentes principalmente em comunicação rádio no geral, podendo assim entender a diferença entre elas.

2.1.1 LoRaWAN

LoRaWAN é uma tecnologia de transmissão de dados baseada em LoRa. Esta tecnologia refere-se à camada MAC, sendo LoRa como nome, apenas referente à camada física [3]. Esta rede funciona numa banda não licenciada, reservada a *Industrial, Scientific and Medical* (ISM), nomeadamente na banda de 868MHz ou 433MHz, ambas na Europa, sendo a mais popular a primeira, permitindo assim criar uma rede de baixo custo e de longo alcance, ótima para transferir em baixo débito pequenos dados de sensores.

Na figura 2.2, é possível observar uma instalação de um serviço de rede LoRaWAN, onde existe um ponto central que possui toda a coordenação da infraestrutura.

Em LoRaWAN, existem 3 classes de dispositivos, relacionadas com a forma como os dispositivos irão receber mensagens de downlink. Estas classes influenciam diretamente a quantidade de energia necessária ao dispositivo, determinando assim a vida útil da sua bateria [2]:

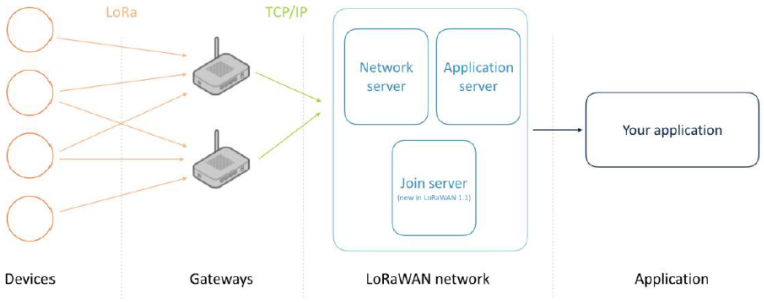

Figura 2.2: Arquitetura da rede LoRaWAN comum.

- **Classe A** - Os dispositivos são maioritariamente desta classe. O equipamento permite comunicação bi-direcional, podendo transmitir os dados em qualquer instante, desde que cumprindo o duty-cycle característico. Esta classe é a mais usada por não estar sempre à escuta de mensagens da rede, permitindo uma maior redução de consumo. Como desvantagem, apenas recebe mensagens durante um curto espaço de tempo após o envio.
- **Classe B** - Esta classe de dispositivos é bastante semelhante à primeira, possuindo todas as suas características, mas possui também uma forma de abrir temporariamente janelas de receção num horário pré-definido. Para que as janelas de transmissão sejam abertas, é necessário que seja transmitido pelo *gateway* uma informação de sincronização para que todos os sensores de classe B estejam sincronizados e assim o controlador conseguir saber quando estão à escuta e enviar as respetivas mensagens.
- **Classe C** - Os dispositivos de classe C são dispositivos com mais necessidades energéticas, sendo esta classe normalmente usada apenas em dispositivos que possuem uma alimentação continua ligada à rede pública, por exemplo. Nesta classe, os dispositivos estão quase permanentemente à escuta de mensagens LoRa, sendo possível comunicar com eles em tempo real, mas não sendo possível a violação dos valores legais de *duty-cycle*, não permitindo assim fazer desta tecnologia um meio de comunicação rápido e instantâneo.

Os dados transmitidos por estes dispositivos são recebidos por todos os *gateways* na zona. Como a rede atribui ao dispositivo uma network session key, outros *gateways* de outras redes não serão capazes de calcular um valor de MIC - *Message Integrity Code* válido, significando que a mensagem não está correta. Neste caso porque considera a chave de rede inválida e irá ignorar o pacote. Nas mensagens que conseguem calcular o MIC, são abertas duas janelas de receção de mensagens, conforme a norma, para possibilitar a receção de dados por parte do sensor, enviados pela rede.

Sendo os pacotes rádio transmitidos numa frequência não licenciada, mesmo com as diferentes classes, há necessidade manter algum controlo na quantidade de informação enviada de forma a não saturar o espectro. Assim, o quadro nacional de telecomunicações [4] indica a necessidade de manter um duty-cycle de 10%, 1% ou 0.1% dependendo da banda desejada. Este parâmetro é especialmente importante pois terá de ser controlado pela programação do sensor, não existindo qualquer mecanismo físico que avise ou impeça a ultrapassagem do valor duty-cycle imposto.

Estes dispositivos possuem, porém, uma forma de segurança para comunicação com a rede, recorrendo ao uso de chaves de aplicação e de rede [5]. Estas chaves têm como princípio a segurança das mensagens entre a aplicação final e o sensor, servindo a chave de rede para calcular e validar o MIC pela rede, ficando assim a saber que a mensagem é válida e é para a rede certa. Os dados aplicacionais (mensagem) permanecem encriptados, sendo estes apenas desencriptados no *application server*, disponibilizando-os posteriormente via MQTT para aplicações no *edge* as processarem.

Existem duas formas dos dispositivos obterem as suas chaves:

- **OTAA** - *Over the Air Activation* - Modo de ativação em que as chaves são trocadas automaticamente entre o *network server*/*application server* e o sensor. Neste modo, apenas é necessário ter conhecimento de uma chave, a chave de ativação, que por sua vez, através desta, podem ser negociados entre a rede e o dispositivo final, as chaves de rede e aplicacional.

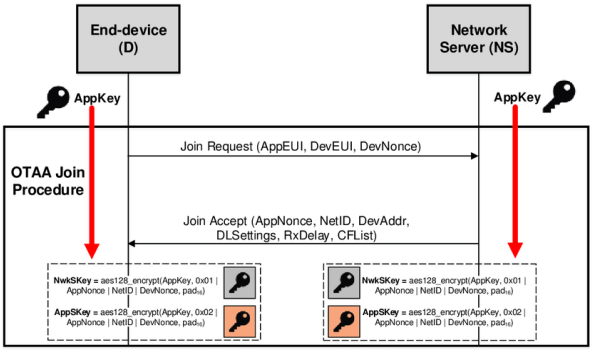

Figura 2.3: Procedimento de troca de chaves OTAA entre a rede e o dispositivo final (Retirado de [6]).

- **ABP** - *Activation By Personalization* - Neste modo de ativação, as chaves de rede e de aplicação são pré-definidas no dispositivo final, que as irá usar para comunicar com a rede, não havendo lugar a qualquer troca de chaves entre o sensor e o *network server*/*application server*.

2.1.2 NB-IoT

Narrow Band Internet of Things (NB-IoT) é uma tecnologia assente em LTE ou NR5G, para estabelecimento de ligações *machine to machine* [7]. Esta tecnologia conta com células de comunicações móveis dos operadores já existentes em quase todo o território português, mas, ao contrário do LoRaWAN, possui um custo associado à utilização da rede.

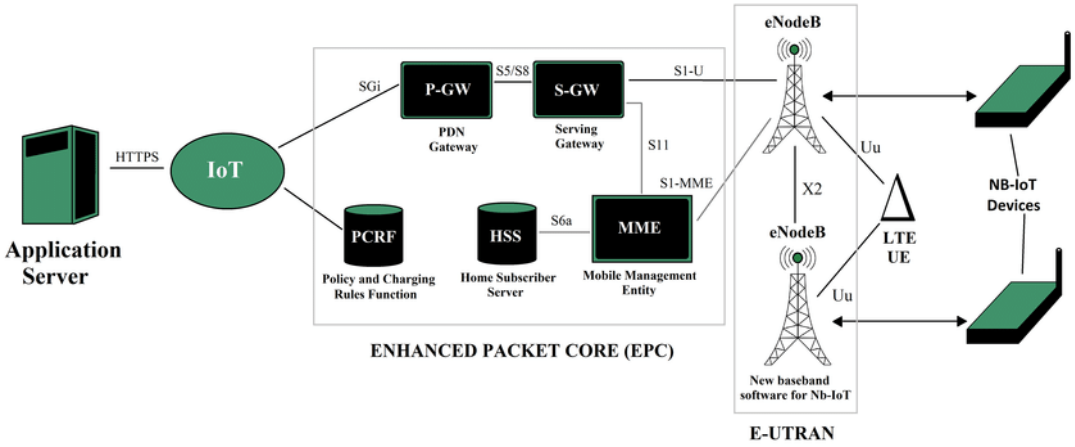

Figura 2.4: Arquitetura da rede NB-IoT em LTE (Retirado de [8]).

A figura 2.4 representa a arquitetura de rede em NB-IoT em LTE para uma computação na nuvem. Como é possível verificar, existe latência associada à passagem dos pacotes pelo core da rede móvel por não ser possível nesta tecnologia a instalação nas estações base eNodeB de um computador para processamento de dados na fronteira. Este problema é mitigado na tecnologia 5G, em que é proposta a criação de um *Multi-Access Edge Computing* (MEC), que permite o processamento de dados imediatamente após a receção rádio da informação.

Na figura 2.5, é possível reparar na existência em cada estação base 5G, ou gNode-b, de um computador Edge. Esta configuração permite, com muito baixa latência na receção dos dados dos sensores conectados, o seu processamento e o acionamento de ações associadas. Mesmo assim, o envio dos dados pela rede até um ponto central, como por exemplo um servidor em *Cloud*, não é removido, estando presente na tecnologia e permitindo a receção dos dados num ponto único. Novamente e tal como em LTE, a desvantagem deste sistema quando comparado ao LoRaWAN é o custo da utilização e dependência de operadores de telecomunicações.

Esta tecnologia, tal como as outras referenciadas, é também de baixo consumo, sendo a vida útil da bateria capaz de, em certas condições, durar mais de 10 anos [10]. No entanto, como já indicado, possui custos de operação pois o detentor do dispositivo,

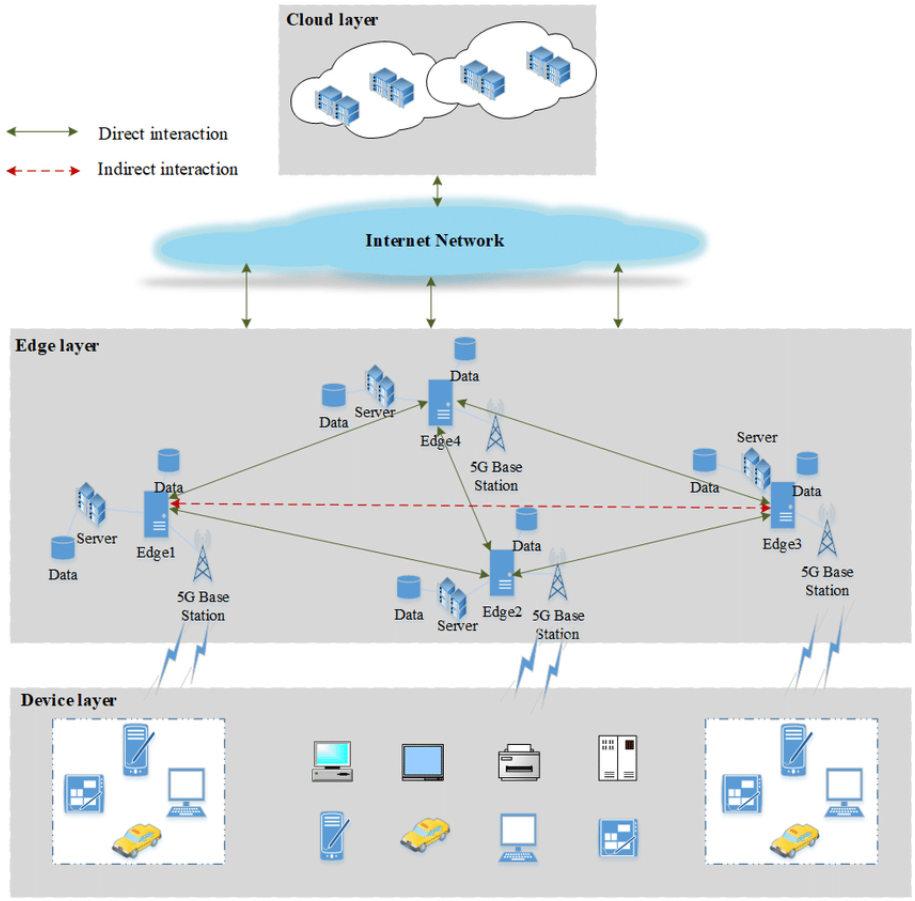

Figura 2.5: Arquitetura da rede NB-IoT com Edge Computing em NR5G (Retirado de [9]).

quando não é um operador de telecomunicações móveis, terá de realizar o respetivo pagamento do uso da rede. Apesar de, por um lado ser uma desvantagem comparando com outras tecnologias, pode ser vantajoso quando há a necessidade de espalhar sensores por uma área grande, mas não há possibilidade de desenvolver uma rede própria.

Os sensores necessitam de estar dotados de modems capazes de realizar comunicações NB-IoT, não bastando um modem simples LTE ou NR5G. Isto deve-se ao facto desta tecnologia se basear numa largura de banda de 200 kHz [10], ao contrário de outras tecnologias que usam uma largura de banda na escala dos MHz [11].

Ao contrário do convencional em LTE ou 5G, em NB-IoT não é possível realizar *handovers* em comunicações ativas [10], o que pode ser um ponto negativo nesta tecnologia. Uma comunicação, quando estabelecida entre o dispositivo e um *gateway*, é necessário que a ligação seja fechada de forma a conseguir comunicar com outro *gateway*.

2.1.3 Sigfox

Sigfox é uma tecnologia proprietária da empresa que possui o mesmo nome, em que o seu funcionamento é idêntico ao das redes 3GPP, na medida em que são colocadas estações base pela empresa e os utilizadores pagam pelo uso da mesma. Esta rede possui diversas desvantagens quando comparado com as outras tecnologias, como por exemplo o desenvolvimento de dispositivos, devido ao facto destes terem de ser licenciados pelo operador de rede.

Figura 2.6: Arquitetura de rede sigfox (Retirado de [12]).

Nesta rede, conforme a figura 2.6, não é possível realizar-se computação na periferia, não sendo possível assim fazer um processamento imediato como é possível por exemplo em NB-IoT, mais particularmente em NR5G com a existência do MEC.

Esta tecnologia também trabalha em frequências não licenciadas ISM como a tecnologia LoRaWAN, mas ao contrário das outras tecnologias referidas, esta possui um limite máximo de mensagens diárias e um *payload* máximo de mensagens muito reduzido [13].

2.1.4 Comparação entre as tecnologias

As tecnologias referidas no corrente capítulo, nomeadamente LoRaWAN, NB-IoT e Sifox são possíveis de ser usadas em dispositivos de baixo consumo como sensores e atuadores, sendo assim compatíveis com dispositivos IoT. Porém, existem vantagens e desvantagens face a cada uma delas, conforme a tabela 2.1.

Tabela 2.1: Comparação entre tecnologias de rede.

	Facilidade de Desenvolvimento de dispositivos	Possiblita nativamente computação de fronteira	Permite instalações de gateways sem licenciamento	Controlo sobre a rede
LoRaWAN	SIM	NÃO	SIM	SIM
NB-IoT	SIM	SIM (em 5G)	NÃO	NÃO
SIGFOX	NÃO	NÃO	NÃO	NÃO

Na tabela 2.1, é possível observar uma comparação entre as 3 tecnologias faladas anteriormente. Tanto em LoRaWAN como em NB-IoT, existe a facilidade de desenvolvimento de dispositivos sem necessidade de que exista uma certificação pelo dono da rede. Note-se que em NB-IoT, o modem necessita de ter as especificações 3GPP, mas não existe nenhum bloqueio na rede, estando a escolha do modem nas mãos do criador do dispositivo. Também se percebe que nenhuma das tecnologias permite de forma nativa computação na periferia, à exceção do NB-IoT, quando a ser usado em 5G e, ainda assim, esta computação necessita de ser acordada com o fornecedor de serviços de rede. Por fim, apenas a tecnologia LoRaWAN permite instalações de *gateways* não licenciados por operar a frequências menos reguladas e, por inerência, o controlo da rede. Esta tecnologia, ao permitir o total controlo da infraestrutura desde o sensor ao nó final, é a tecnologia ideal para se desenvolver novas soluções de baixo custo e acessíveis.

2.2 Rede de transporte

A Rede de Transporte é a infraestrutura que irá gerir todo o tráfego LoRaWAN, na camada de rede. Nesta camada, os pacotes já provenientes do *stack* responsável pela gestão do hardware são lidos, decifrados e entregues às aplicações finais. Assim, é necessário identificar softwares compatível com a tecnologia LoRaWAN que seja capaz de receber os pacotes UDP provenientes do *stack* LoRa presente nos gateways, descodificá-los e disponibilizá-los para que *sofware* de terceiros os consigam processar. Este *stack* LoRa presente é a base do LoRaWAN, sendo responsável por receber as mensagens dos sensores LoRa, para posteriormente reencaminhar para uma aplicações de rede como The Things Network ou Chirpstack. Torna-se também necessário, na sequência do software anterior, encontrar outros softwares, preferencialmente com uma compatibilidade elevada, para que um programador o possa incorporar no seu desenvolvimento e assim receber as mensagens provenientes do sistema.

2.2.1 Chirpstack

O Chirpstack é um software livre que ajuda na implementação aberta dos componentes de software central de uma rede LoRaWAN. Esta tecnologia foi escolhida para a tese pelo facto de, para além de possuir código aberto, possuir uma API de ligação que permite o desenvolvimento de outros softwares complementares, que permitem aumentar as funcionalidades e implementar novas características, nomeadamente a possibilidade de interligação entre diferentes instâncias completas, fazendo com que múltiplos *gateways* independentes interajam como se de uma rede completa se tratasse.

O software base é composto por 3 componentes que podem ser instalados de forma distribuída em rede ou numa máquina apenas:

- Gateway Bridge
- Network Server
- Application Server

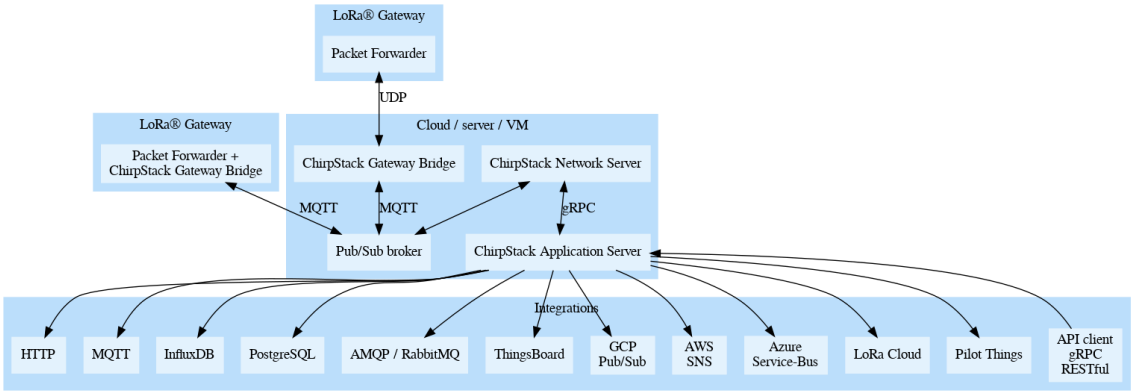

Figura 2.7: Arquitetura de implementação dos componentes do software chirpstack[1].

Este software está originalmente desenhado para que os múltiplos *gateways* LoRa estejam permanentemente conectados à rede, de forma a haver apenas um único conjunto de Network Server e Application Server.

2.2.1.1 Gateway Bridge

O Gateway Bridge é o componente responsável pela interligação do *gateway* LoRa físico e o network server, por intermédio de um servidor MQTT[1]. Este componente

[1]https://www.chirpstack.io/project/architecture/ acedido a 2023-02-20

recebe em pacotes UDP mensagens JSON, normalizadas com o padrão do protocolo Semtech LoRa Packet Forwarder e reencaminha-as via MQTT para o Network Server as interpretar. Estas mensagens encontram-se num formato encriptado, pelo que não é possível, sem as chaves, nesta ligação, fazer alguma interceção e processamento destas mensagens.

Na implementação deste projeto, graças a este módulo, passa a haver uma abstração entre a camada IP/UDP do protocolo Semtech, sendo a partir deste ponto todas as mensagens enviadas via MQTT. Assim, é possível efetuar decisões sobre quem escuta estas mensagens, sendo possível colocar intermediários ou até mesmo enviar estas mensagens para um ponto central, cumprindo assim o objetivo de receber as mensagens tanto no *edge* como na *cloud*.

2.2.1.2 Network Server

O Network Server é o componente essencial de uma rede LoRaWAN, sendo este o nó central responsável por receber as mensagens dos *gateways*, obter os IDs e chaves de encriptação/desencriptação dos dispositivos através do Application Server, realizar a desduplicação de mensagens quando existe mais que um *gateway* registado, enviar as mensagens de uplink sem encriptação de rede para o application server e agendar as mensagens downlink para a janela de receção dos sensores.

2.2.1.3 Application Server

O Application Server é o componente do conjunto que agrupa as mensagens por sensor e por aplicação, organizando-as em tópicos e enviando-as para o broker MQTT para serem processadas pelas aplicações. Este componente é responsável também pela criação das chaves referentes ao LoRaWAN tanto em modo OTAA como em modo ABP e a partilha da chave de Rede com o Network Server.

2.2.2 The Things Stack

The Things Stack é um pacote de software usado pela The Things Network, que é uma rede pública de *gateways* LoRaWAN que permite a um sensor enviar mensagens, que serão recebidas e descodificadas por servidores públicos, sendo depois as mesmas encaminhadas para um servidor de receção de mensagens, como MQTT, para possibilidade de processamento das mensagens por aplicações externas.

É composto por 6 componentes, necessários à receção, descodificação e transmissão das mensagens provenientes de sensores, conforme a figura 2.8[1]:

- Network Server
- Broker
- Handler
- Discovery
- Gateway Bridge
- Router

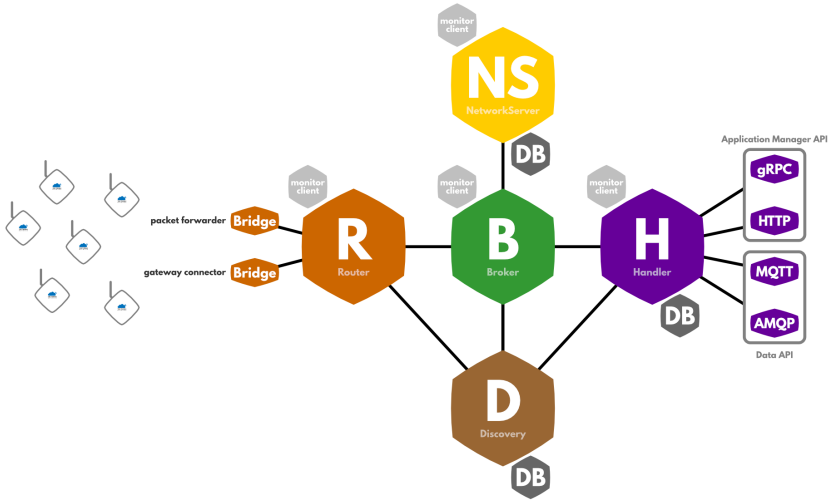

Figura 2.8: Arquitetura da rede The Things Network.

Este sistema de rede, apesar de possibilidade de instalação própria por ser open-source[2], é um sistema constituído por um único pacote que contém todas as funcionalidades, sendo este software pensado para que os *gateways* sejam simples transmissores/recetores LoRa, ligados permanentemente à rede com acesso à Internet, que por sua vez permite o acesso ao serviço a correr num fornecedor *cloud*. Possui também uma API que possibilita as funcionalidades de adicionar, remover, editar, registar dispositivos, permindo assim uma integração com software externo.

Este software possui uma versão em *cloud*, mas que, no entanto é baseada numa rede pública onde não existe controlo algum sobre os *gateways*, que são propriedade de pessoas particulares ou coletivas, como por exemplo, os dois *gateways* do ISEL que se encontram ligados a esta rede pública.

[1]https://www.thethingsnetwork.org/docs/network/architecture/ acedido a 2023-03-13
[2]https://www.thethingsindustries.com/docs/getting-started/what-is-tts/ acedido a 2023-02-12

2.2.2.1 Network Server

O Network Server no The Things Stack, é o elemento responsável por validar as mensagens, fazendo as verificações do MIC, atribuição de endereços de rede e disponibilizando aos outros componentes a relação entre o devEUI e o endereço de rede .

2.2.2.2 Broker

O broker é o elemento principal da rede TTN, tendo a responsabilidade de interligar os dispositivos com as aplicações e reencaminhar mensagens de uplink e downlink entre as aplicações corretas e entre o Router[2].

2.2.2.3 Handler

O Handler é responsável pelo envio das mensagens dos sensores às aplicações. Neste ponto, a mensagens ainda se encontra encriptada, sendo feita a sua desencriptação e envio para as aplicações clientes a correr[2].

2.2.2.4 Discovery

O Servidor Discovery é o componente que permite a todos os componentes da rede de se descobrirem uns aos outros, fazendo com que em redes muito complexas não haja necessidade de colocar manualmente todos os endereços de *routers* e *brokers* [1].

2.2.2.5 Gateway Bridge

O Gateway Bridge é o componente responsável por receber as mensagens em bruto provenientes de *gateways* e colocá-las no sistema, nomeadamente através do Router. Este componente é também capaz de receber mensagens da rede e fazer a sua transmissão via LoRa, para assim possibilitar a forma de comunicação por downlink.

2.2.2.6 Router

O Router está responsável pela gestão dos *gateways*, nomeadamente a tratar mensagens de estado e mensagens de sensores. Este componente recebe mensagens de estado dos *gateways*, fazendo assim a gestão dos *gateways* que estão ativos. Recebe mensagens de

[1]https://www.thethingsnetwork.org/docs/network/discovery/ acedido a 2023-03-13

sensores, processa o conteúdo verificando o payload por mensgens ou ativações ABP ou OTAA e também recebe da rede mensagens downlink, sendo responsável por fazer a manutenção da fila de espera, aguardando que o sensor esteja perto de um *gateway* da rede e fazendo a transmissão da mensagem para esse *gateway*[2].

2.2.3 Marcação de pacotes IP

O facto de os *gateways* necessitarem de ser coordenados por um nó central e de circular por estes nós mensagens críticas, torna-se necessário ter algum cuidado com a forma de encaminhamento deste tráfego pela rede. Por vezes, estes *gateways* podem ter diversas formas de comunicação IP, modems de comunicação com a internet, podendo ser importante a realização de algum tipo de marcação dos pacotes, fazendo assim distinção de outros pacotes não críticos.

Tomou-se então como princípio a possibilidade de marcação de pacotes usando *Differentiated Services Code Point* (DSCP) [14], uma tecnologia compatível com bastantes equipamentos de rede do âmbito comercial e que permite ao nível do sistema o encaminhamento do tráfego por caminhos específicos, consoante por exemplo a qualidade de serviço de rede IP do *gateway*.

Esta tecnologia baseia-se nas aplicações ou routers/firewalls, na altura em que respetivamente criam ou processam os pacotes, modificam 8 bits do header correspondentes ao *Type of Service* (ToS) por outro valor, sendo que, por omissão, o valor será 8 bits a zero.

2.2.4 MQTT

Message Queuing Telemetry Transport (MQTT) é um protocolo de mensagens baseado em subscrições e publicações em tópicos, desenhado para operar em redes com baixa largura de banda, alta latência e para aplicações *Machine to Machine* (M2M) [15].

Os tópicos são títulos de mensagens, com um formato de palavras separadas por barras, por exemplo "chirpstack/syncronization". Este formato torna mais fácil a subscrição de tópicos, ou conjuntos de tópicos, pois é possível com a utilização do símbolo # fazer uma subscrição a um conjunto de tópicos, por exemplo, "chirpstack/#"fará com que o cliente receba todas as mensagens começadas por chirpstack.

O funcionamento deste protocolo consiste na existência de um broker, que é um software a correr o sistema de mensagens e que recebe subscrições e publicações. Quando

[2]https://www.thethingsnetwork.org/docs/gateways/start/connection/%7D acedido a 2023-03-13

um cliente faz uma subscrição de um tópico ou de um conjunto de tópicos, o servidor guarda a informação do ID do cliente e, sempre que houver uma publicação por um publicador, o subscritor receberá a mensagem sem que nunca haja contacto direto entre o publicador e o subscritor.

Nas mensagens enviadas, podem ser colocados parâmetros internos ao broker, que possibilitam a existência de *Quality of Service* (QoS) e de retenção nas mensagens. Em MQTT, o QoS não é o da rede, colocado nos pacotes TCP, mas sim um QoS interno ao broker. Em termos de QoS, existem 3 níveis:

- **QoS 0** - *At most once* - Este é o valor QoS por omissão. Neste caso, o broker funcionará por *best-effort*, o que implicará não existir garantias de entrega, sendo a mensagem enviada sem qualquer tipo de confirmação, conforme a figura 2.9.

Figura 2.9: Fluxo MQTT com QoS 0 [3].

- **QoS 1** - *At least once* - Neste valor, o publicador irá obter a confirmação de entrega (PUBACK) do servidor, que irá garantir assim a entrega pelo menos uma vez da mensagem ao destinatário. Neste valor de QoS, não é garantido que não haja duplicação de mensagens, ou seja, o cliente pode receber a mensagem mais que uma vez, conforme a figura 2.10.

Figura 2.10: Fluxo MQTT com QoS 1 [3].

[3]https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/ acedido a 2023-02-02

- **QoS 2** - *Exactly once* - Neste nível, existe a garantia de que a mensagem foi enviada e de que esta será recebida apenas uma vez pelo destino. Apesar desta opção tornar o sistema mais lento, torna-o mais seguro pois o cliente é obrigado a confirmar que recebeu a mensagem através dum pacote (PUBREL), tomando assim o servidor conhecimento de que o cliente recebeu efetivamente a mensagem, conforme a figura 2.11.

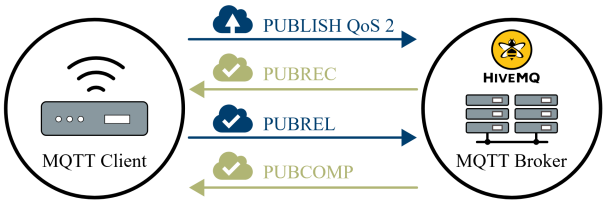

Figura 2.11: Fluxo MQTT com QoS 2 [3].

É possível também colocar numa publicação a informação de RETAIN, que consiste em informar o servidor que deve reter a mensagem do tópico. O servidor apenas tem capacidade de reter uma mensagem por tópico, o que implica que a próxima mensagem com a flag RETAIN irá eliminar a mensagem anterior. Assim, quando um cliente fizer uma subscrição ao tópico, irá receber imediatamente a última mensagem em RETAIN.

Através das instruções de QoS e RETAIN, é possível determinar um valor mínimo de qualidade de serviço do broker MQTT, no entanto, não é possível ter qualquer controlo sobre a rede, havendo possibilidade de um router ou firewall colocar o pacote em espera consoante o tráfego existente na rede. De forma a minimizar esse problema, podem ser aplicadas marcações DSCP nos pacotes TCP com origem ou destino ao broker, de forma a ser possível manter a qualidade de serviço na camada a cima, sendo estas marcações processadas pelos equipamentos de rede.

2.2.5 Apache Kafka

O Apache Kafka é um software de distribuição de mensagens escalável que permite a transmissão de um grande volume de mensagens por segundo. [16] Desenvolvido originalmente pela equipa do Linkedin, devido à necessidade de uma transmissão de dezenas de milhões de leituras e escritas por segundo [17]. O seu funcionamento consiste na existência de um servidor, capaz de realizar o processamento anteriormente referido, sendo os dados enviados para o mesmo por produtores e recebidos por subscritores.

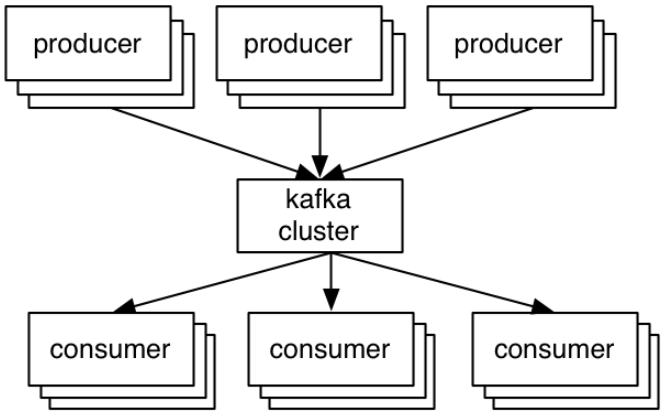

Figura 2.12: Arquitetura de consumidores e produtores KAFKA (Retirado de [17]).

Neste sistema, os produtores ao necessitarem de enviar mensagens, enviam-nas fazendo referência a um tópico que possui um formato de palavras separadas por pontos, por exemplo, "chirpstack.syncronization".

O Kafka está pensado para um grande fluxo de dados e permite também a persistência de uma quantidade limitada de dados, fazendo com que assim que haja uma interrupção momentânea na transmissão de dados, o cliente na reconexão pode receber todas as mensagens que não recebeu quando existia a falha de ligação.

2.3 Trabalho Relacionado

O Trabalho Relacionado com este projeto refere-se a tecnologias IoT onde já esteja implementada uma computação na periferia, podendo as mensagens dos sensores serem processadas diretamente no local de receção dos dados. Atualmente, em IoT, através do MEC em 5G, é possível a existência deste tipo de computação com dispositivos. Esta tecnologia permite a receção e processamento dos dados, no mesmo local, reduzindo em muito a latência nas respostas.

O MEC, Multi-Access Edge Computing, é uma extensão da computação na nuvem [18] que é colocada junto ao local de receção dos dados, para que o processamento da informação seja mais rápido, reduzindo a latência na resposta inerente à rede. O MEC é apenas usado na tecnologia 5G, sendo apenas possível de instalar por operadores desta tecnologia.

Na figura 2.13, é possível observar que o MEC se encontra ligado diretamente à rede de acesso, neste caso, encontra-se normalmente na estação base. Isto permite assim uma comunicação quase imediata entre o MEC e os dispositivos finais, permitindo o

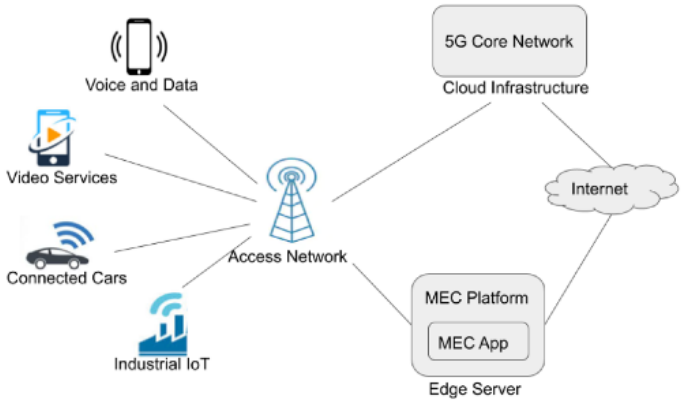

Figura 2.13: Localização do MEC numa rede 5G (Retirado de [18]).

seu uso para situações mais críticas como veículos autónomos ou situações médicas.

Dentro deste MEC, é possível a clientes dos operadores instalarem as suas aplicações e assim tirarem partido da baixa latência existente e apesar do processamento poder ser local, não invalida a possibilidade do envio dos dados para um servidor central.

Existe também o artigo [19] onde são referidos os problemas do LoRaWAN em computação na periferia. Este artigo, à semelhança desta tese, tem a finalidade de criar a possibilidade desta computação com a tecnologia LoRaWAN, porém, é feita uma reimplantação do protocolo tanto no dispositivo, como nos *gateways*, o que não acontece nesta tese, onde o sistema é totalmente transparente para os dispositivos.

3

Trabalho Realizado

Na sequência do estado da arte, foi feita a implementação do sistema proposto, nomeadamente no que diz respeito à receção de mensagens LoRaWAN em múltiplos sistemas de computação de fronteira. Inicialmente, é referência ao hardware e às aplicações usadas num computador de computação de periferia de exemplo, o Network Server e o Application Server, fazendo seguimento para a infraestrutura usada para ligação entre múltiplos computadores de computação de periferia. Sendo descrita a forma como todos os sistemas são interligados, é explicada a implementação de segurança na rede, tanto na infraestrutura de rede de suporte como na possibilidade de integração das atuais formas de segurança do protocolo LoRaWAN em computação de periferia.

3.1 LoRaWAN Network Server e Application Server

Para este projeto, foi decidido usar o Chirpstack por este ser constituído por módulos que, apesar de relacionados, são independentes entre si. Possui uma API bastante simples e dirigida ao projeto e uma simplicidade do modelo de dados. Tal não acontece com o The Things Stack, que possui em conjunto todos os módulos, tanto network server como application server e gateway bridge, fazendo com que os *gateways* comuniquem todos com o um único módulo, sem uma cadeia como existe no Chirpstack e permite de forma acessível a "interceção"das mensagens e modificação. Assim, foi projetada a instalação de pelo menos dois *gateways* móveis, ambos com um módulo LoRa

MikroTik e um *gateway* fixo com um módulo pycom, que permitiu avaliar a aplicabilidade do conceito estudado com diferentes fornecedores de hardware e em diferentes aplicações. Estes fornecedores de hardware devem comunicar usando o protocolo UDP normalizado pela Semtech.

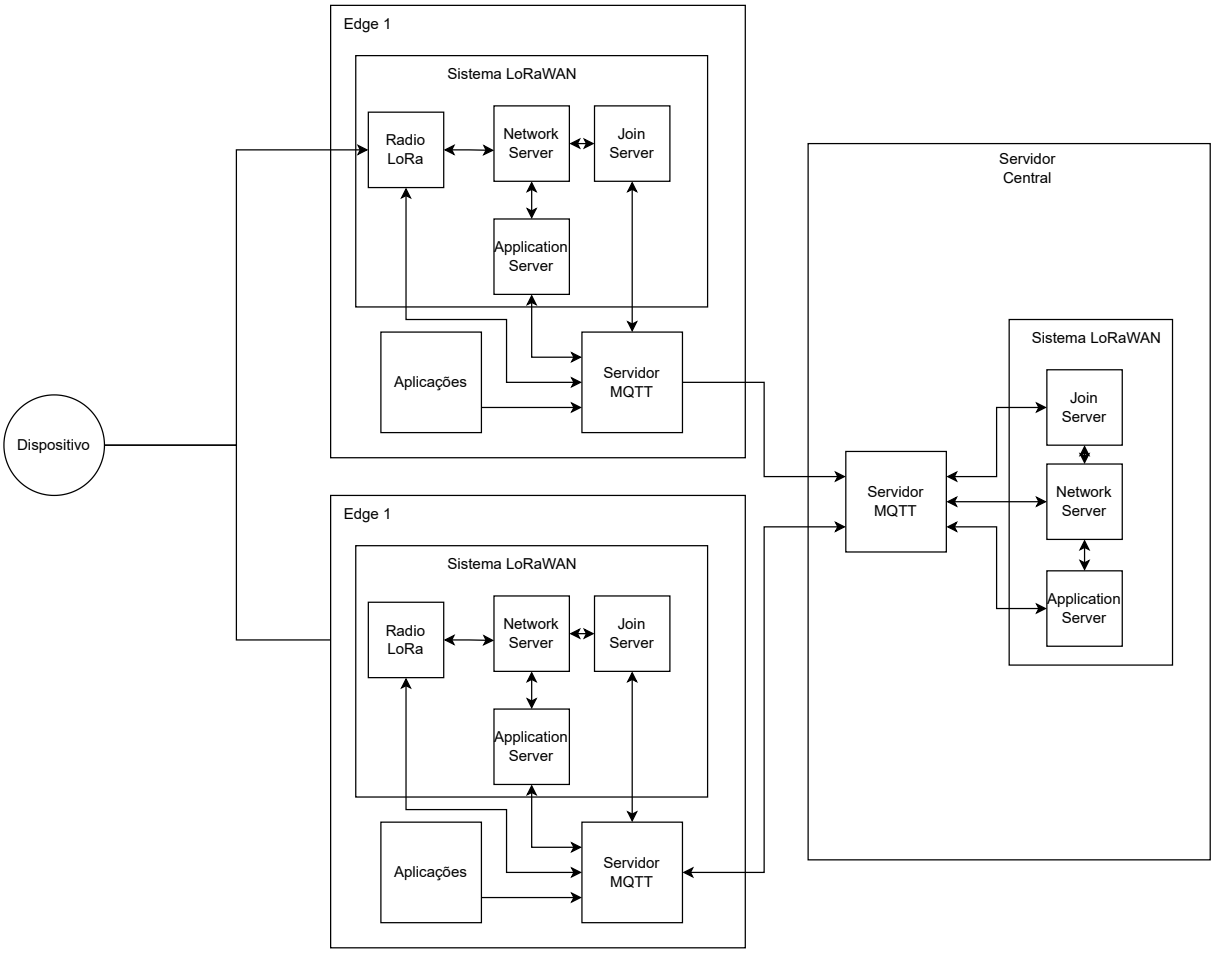

Figura 3.1: Arquitetura LoRaWAN desenvolvida.

Inicialmente, procedeu-se à verificação do correto funcionamento de um *gateway* conectado a uma instância do Chirpstack. Nesse contexto, foram examinadas as interações entre todos os componentes envolvidos. Esta análise incluiu a análise das mensagens trocadas entre esses componentes no interior do *gateway*, conforme representado na figura 3.1, onde é apresentado o diagrama final de ligações entre os módulos. Foi também verificada a etapa de adição do dispositivo à rede na interface, bem como durante o processo de associação (join) do dispositivo à rede, que implica o registo do dispositivo na infraestrutura, além da transmissão de mensagens para as aplicações finais.

Assim, foi verificado que seria necessário que todas as *gateways* possuíssem uma instância completa deste software e fazer um ponto central da rede onde se encontrassem

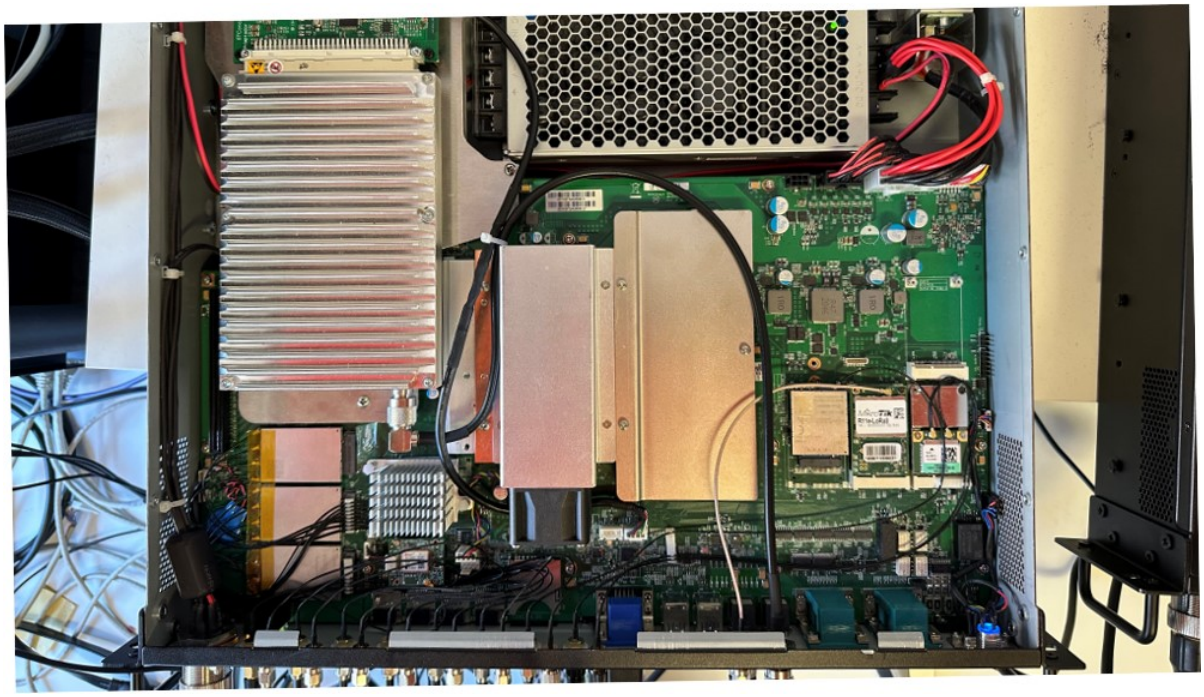

Figura 3.2: Interior de um *gateway* com computação na periferia, com o modem *MikroTik* visível do lado direito.

todas as configurações. Para isso, foi também instalado no servidor central apenas dois componentes de software referentes ao Chirpstack, nomeadamente o Application Server, responsável por guardar a informação das aplicações e dos sensores e o Network Server, que será usado para fazer a desduplicação de mensagens das *gateways*, quando enviadas para o servidor central.

Um ponto importante também verificado foi a forma como os sensores se registavam na rede e efetuavam a troca de chaves, sendo que o serviço em *edge* necessitaria de suportar registos tanto em ABP como em OTAA.

3.2 Infraestrutura de Rede IoT para LoRAWAN em Edge Computing

Um *gateway* é um componente de hardware e software que permitirá o encaminhamento de dados entre dois sistemas, sendo neste caso o sistema LoRaWAN e a rede IP. Será composto pelos componentes referentes ao Chirpstack falados anteriormente e suportará também aplicações de forma a ser possível funcionar em modo de *edge*, havendo possibilidade de por vezes este ser desconectado da rede ou por falta de rede no caso do acesso ser através de operadores móveis. Este comunicará via rede com um

servidor central, permitindo assim a passagem de dados entre os sensores/aplicações e uma cloud, também para sincronizar as aplicações e as chaves de comunicação com os sensores LoRaWAN.

Para desenvolvimento e testes, foram instalados tanto o chirpstack como uma placa LoRa em dois computadores de forma a simular um computador de processamento de periferia móveis. Numa VM foi apenas instalado o chirpstack que recebe os pacotes LoRa através de um módulo de rede da marca Pycom. Em ambas as modalidades de instalação, havia a possibililidade de serem instaladas aplicações de terceiros, para serem processadas as mensagens vindas de sensores. Dado esta tese estar integrada no projeto Ferrovia 4.0, do qual o ISEL é parceiro, nos computadores de periferia estiveram a funcionar aplicações de recolha de dados oriundos de sensores de Taludes.

Assim, o produto final será um bloco de hardware e software, ligado a uma rede IP, capaz de receber pacotes LoRa, processá-los com recurso a aplicações externas e também de os enviar para um sistema central para gravação ou posterior processamento.

Na figura 3.1, é possível verificar a ideia de funcionamento usando como exemplo dois *gateways*. Este número de *gateways* pode variar, estando o número máximo de *gateways* relacionado com o número de dispositivos (sensores) na rede, pois as alterações necessitam de ser propagadas entre os computadores de periferia. Assim estima-se que esse número equivale ao número máximo de mensagens MQTT por segundo suportadas pelo servidor central, podendo chegar a mais de 4 mil mensagens por segundo [20].

3.3 Meio de comunicação entre Servidor e Gateways

Como demonstrado na Figura 3.1, é evidente a importância da interconexão entre os *gateways* e o servidor central. Essa interconexão é fundamental devido à necessidade de sincronizar os as informações dos dispositivos, nomeadamente nomes, endereços e chaves de cifra, os quais devem estar disponíveis para todas os *gateways*. Nesta secção, será abordada a maneira como essa ligação entre o servidor central e os *gateways* é estabelecida de forma segura e eficiente e, também, as medidas efetuadas para tornar esta ligação resiliente [14].

3.3.1 VPN

Uma VPN, Rede Privada Virtual, é um mecanismo que permite de forma segura a transferência de pacotes IP entre dois dispositivos, mesmo quando esses pacotes atravessam uma rede considerada insegura, como é o caso da Internet. É relevante frisar

que as plataformas de computação periférica devem estar ligadas à plataforma central através de uma VPN para que o seu funcionamento não seja limitado tanto por restrições geográficas e para garantir a segurança do trafego. Assim, fica disponível uma comunicação continua entre todos os *gateways* e a rede da plataforma central.

De forma a permitir uma ligação resiliente e dando continuidade a [14], foi feita a instalação de instâncias do software Wireguard tanto na plataforma central, como nos *gateways*.

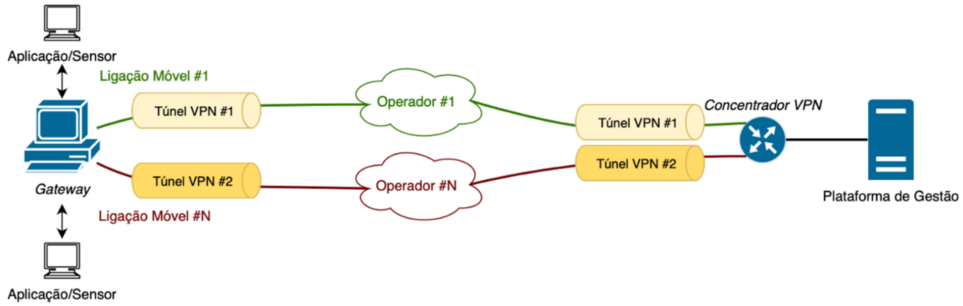

Figura 3.3: Esquema de ligação entre *gateway* e Plataforma central (de gestão) (Retirado de [14]).

Na figura 3.3, é possível ver o esquema de ligação proposto por [14] para a ligação resiliente entre um *gateway* e um nó central, neste caso, na Plataforma central. Assim, foi instalado em cada *gateway* duas instâncias do software Wireguard, que possibilitam a ligação entre duas redes, que para testes foi a rede ethernet recebida por cabo de rede fazendo a simulação de uma rede wifi instável e a rede móvel com recurso a um cartão SIM.

3.3.2 Seleção de Rede

Com a possibilidade de coexistência de duas redes simultaneamente no mesmo gateway, tomou-se a decisão de direcionar o tráfego com base em prioridades. Dado que a rede Wi-Fi é menos estável, optou-se por utilizá-la para tráfego de menor importância, mas com uma elevada largura de banda disponível. Isso é exemplificado quando um *gateway* é móvel e possui apenas acesso à rede Wi-Fi em determinadas situações. Por outro lado, a rede móvel foi reservada para o tráfego de maior importância, como é o caso das aplicações mencionadas nesta tese.

Em caso de falha completa em uma das redes, é possível que todo o tráfego considerado crítico passe a ser direcionado exclusivamente por uma das vias disponíveis, independentemente das prioridades inicialmente estabelecidas.

3.3.3 MQTT

O software Chirpstack indica obrigatoriamente o uso do MQTT para a troca de mensagens entre componentes, nomeadamente entre o gateway bridge, responsável pela receção de pacotes UDP no formato normalizado pela Semtech e, posteriormente, reencaminhamento dos mesmos via MQTT para o Network Server. O Network Server, que por sua vez comunica com o application server via gRPC, recebe a mensagem e envia-a para o Application Server, que a reencaminha novamente a mensagem desencriptada para o servidor MQTT. As aplicações centrais, tendo acesso ao servidor MQTT, conseguem receber as mensagens desencriptadas.

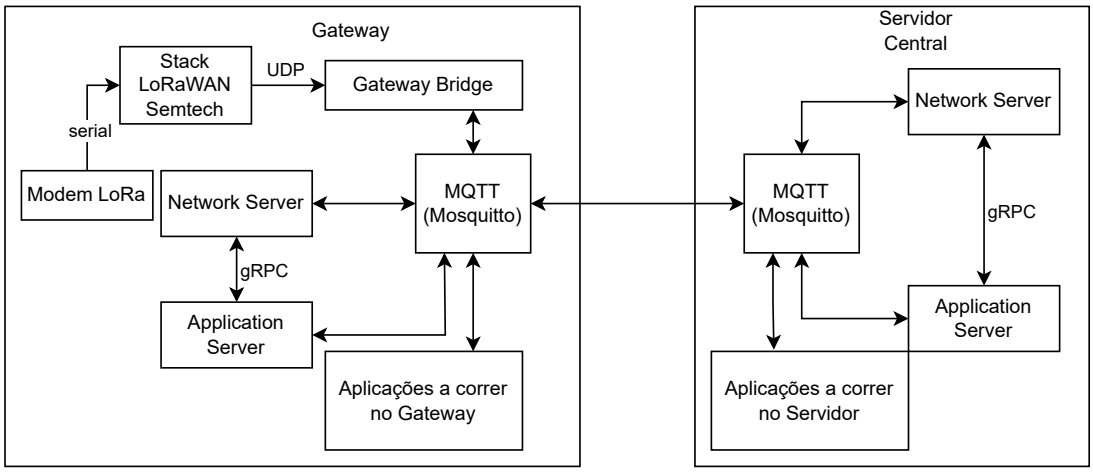

Figura 3.4: Representação das ligações entre Edge e Central para troca de mensagens.

Sendo o servidor MQTT o nó central por onde todas as mensagens dos dispositivos transitam, considerou-se crucial encontrar uma solução que não apenas permitisse a publicação e subscrição de mensagens, mas também possibilitasse o encaminhamento dessas mensagens para outros sistemas, conforme a figura 3.4. Um exemplo que atende a essas necessidades é a aplicação Mosquitto, que com a funcionalidade de bridges [1], permite tanto o envio e subscrição de mensagens, quanto a capacidade de republicar essas mensagens em outros servidores MQTT mantendo os mesmos parâmetros locais, nomeadamente mantendo as classes de qualidade de serviço e retenção. A escolha deste software é justificada pela sua facilidade de uso, leve e permitir a criação de bridges.

Havendo a necessidade de que as mensagens dos *gateways* fossem também reencaminhadas para o servidor central e sendo cada *gateway* independente, poderia existir o risco da duplicação de mensagens quando as mesmas chegassem ao servidor central. Assim, existindo já um mecanismo que previne a duplicação de mensagens das redes

[1] https://mosquitto.org/man/mosquitto-conf-5.html acedido a 2023-09-12

LoRaWAN, o Network Server, decidiu-se fazer a sua instalação no servidor central e fazendo com que os *gateways* reencaminhassem as mensagens por desencriptar do gateway bridge para o network server do servidor central. Tendo no Servidor Central uma cópia de todos os endereços e chaves dos dispositivos, este tem capacidade de desduplicar, decifrar e disponibilizar as mensagens, conforme representado na figura 3.5, onde a linha representa o fluxo de mensagens desde o modem LoRa até às aplicações finais.

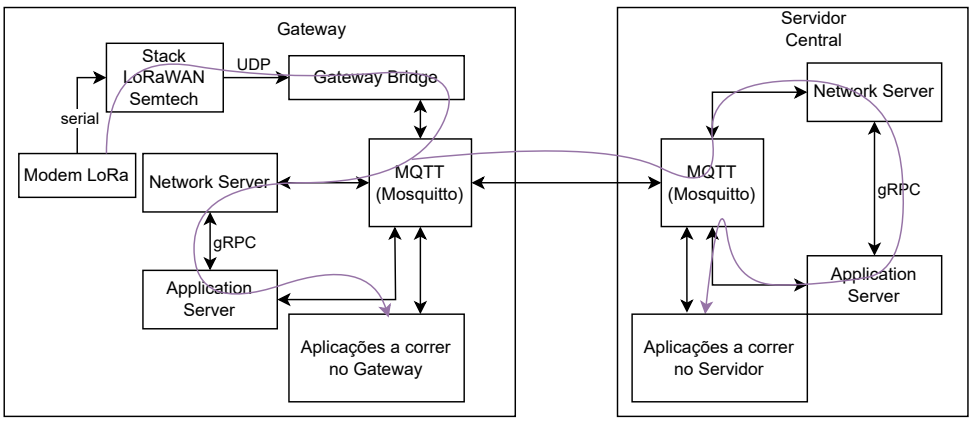

Figura 3.5: Fluxo de mensagens desde o Modem LoRa dos *gateways* até às aplicações finais.

3.4 Sincronização de sensores e aplicações

Sendo necessário a existência de um Application Server e de um Network Server para a interpretação de mensagens LoRa, é então necessário que estes dois componentes se encontrem o mais perto possível do edge, onde será feito o processamento dos dados, sendo recomendado até que esteja na mesma máquina, como vimos anteriormente. Sendo a instalação de um network server e application server a essência de uma rede, isto significa que cada *gateway* terá a sua própria rede, independente de *gateway* em *gateway*. Isto é bastante vantajoso na medida em que mesmo sem conexão à Internet, é possível ao *gateway* receber e interpertar mensagens e fazer com que estas sejam recebidas por outras aplicações locais.

Assim, torna-se necessário o desenvolvimento de uma aplicação que realize a sincronização entre os dados existentes na plataforma central e os dados das gateways. Estas mensagens devem devem conter a informação das aplicações e dos sensores que foram introduzidos na plataforma central, sendo os mesmos enviados num formato JSON.

Na figura 3.6, é representado o fluxo de mensagens de atualização desde a sua origem, até aos computadores na periferia e as configurações de envio MQTT da mensagem

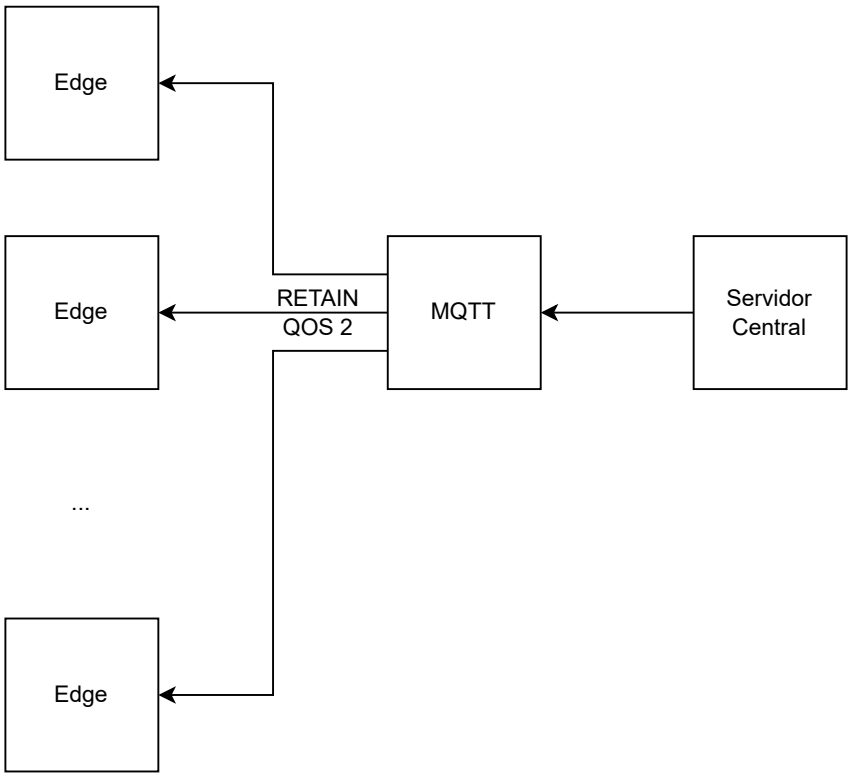

Figura 3.6: Esquema de propagação de mensagens *json* da plataforma central ao *edge*.

(Retain e QOS). A estrutura de dados, representada no Anexo A, permite aos computadores de periferia conhecerem as aplicações e respetivos dispositivos, juntamente com as chaves de autenticação. A publicação destas mensagens pelo servidor central nunca ocorre diretamente nas plataformas de computação na periferia, mas sim diretamente no servidor MQTT em funcionamento no servidor central e, por sua vez, com recurso a bridges, reencaminha a mensagem para o computador de periferia. Este fluxo permite que caso o sistema se encontre momentaneamente desconectado da rede internet, é possível reter a mensagem até que a ligação seja reconectada.

3.4.1 Registo dos sensores na rede

De acordo com as especificações do protocolo LoRaWAN e falado anteriormente, existem duas formas de registar os sensores, que podem ser via ABP ou OTAA. É bastante importante que estes métodos de registo não sejam comprometidos com as alterações efetuadas ao método de funcionamento do sistema LoRaWAN. Assim, este trabalho foi dividido em duas partes, conforme o método de registo, começando pelo sistema de registo ABP, o mais simples de implementar devido ao facto de não ser necessário comunicar com o dispositivo final as chaves e terminando no sistema de registo OTAA, mais complexo de implementar devido ao facto de ser necessário trocar informações

com o dispositivo final.

3.4.1.1 Método de autenticação ABP

O Método de autenticação ABP é o método mais simples de implementar, pois funciona juntamente com a sincronização dos dados das aplicações e sensores do servidor central para os *gateways*

O utilizador, na interface, ao preencher as chaves e guardar, as chaves são guardadas no servidor central e é despoletado o processo de atualização dos gateways. Os gateways, se ligados à rede, irão receber as chaves tão rápido quanto possível, dependendo esta velocidade da latência da rede do *gateway*. Os gateways que possuem a ligação à rede interrompida, assim que a sua ligação for reestabelecida, irão atualizar as chaves dos sensores, permitindo assim aos gateways descodificar as mensagens do sensores.

3.4.1.2 Método de autenticação OTAA

O método de autenticação OTAA é o método mais desafiante de implementar num contexto de computação de fronteira, pois existe a necessidade de gerar e trocar chaves entre o dispositivo e a rede. Sendo que cada *gateway* se comporta como uma rede independente e como apenas um *gateway* pode comunicar com um dispositivo, esta autenticação é a única fase da rede em que existe a necessidade obrigatória de conecção ao servidor central, de forma a haver sincronismo entre todos os gateways, pois caso mais que um *gateway* recebesse um pedido de autenticação, estes não iriam gerar chaves iguais. Ao não serem geradas chaves iguais, não é possível entender qual chave foi gravada no dispositivo final e, assim, não seria possível comunicar com o mesmo, como o exemplo da figura 3.7 onde dois gateways respondem ao pedido de "Join Request". Os gateways sem comunicação rejeitam estes pedidos de autenticação.

Este método consiste no envio pelo dispositivo de uma mensagem do tipo "Join Request", sendo recebida pelo *gateway* e dado como resposta um "Join Accept"caso o dispositivo exista, a chave de ativação esteja correta e as chaves tenham sido geradas com sucesso. No Chirpstack, o componente responsável pela gestão dos pedidos de join, é o Network Server através do Application Server. O Network Server envia via HTTP para o Application Server que por sua vez valida a chave de autenticação enviada pelo dispositivo, gera as chaves e responde ao pedido com as chaves geradas. Se a chave não for válida, responde com a informação de erro. No protocolo LoRaWAN, não

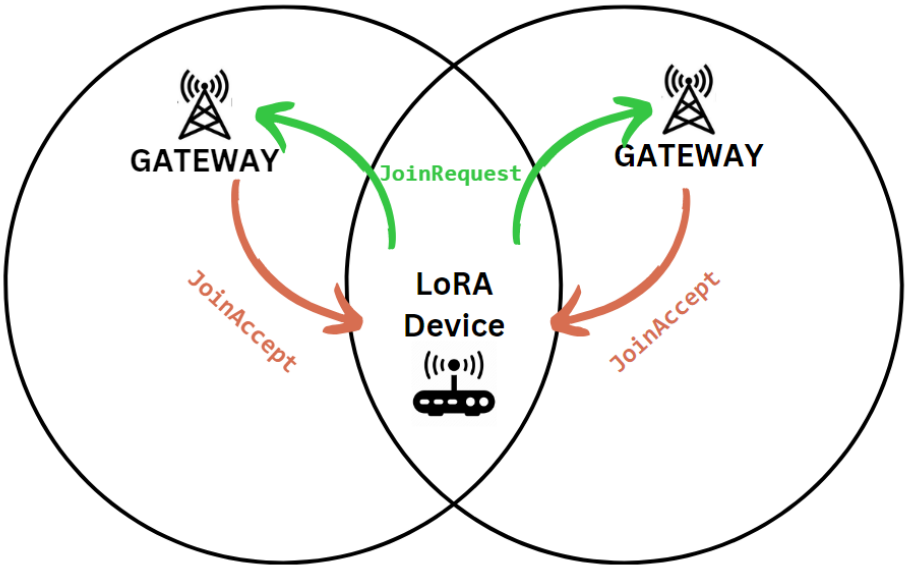

Figura 3.7: Caso em que um dispositivo está no alcance de dois *gateways*.

existe o conceito de "Join Reject", pelo que a rejeição de um "Join Requst"é o não envio de "Join Accept".

A forma de implementação deste pedido de join foi através do desenvolvimento de um intermediário entre o Network Server, em funcionamento nos computadores de periferia e o Application Server, no servidor central. Como existe a necessidade de haver conexão com a rede para haver sincronismo entre gateways, é vantajoso que quem gere as chaves seja o Application Server do servidor central, para que depois as chaves geradas possam ser partilhadas pelos gateways.

Este intermediário recebe os pedidos de join dos network servers dos gateways, através de pedidos HTTP, que são recebidos por este intermediário onde é validado o endereço do dispositivo, de forma a saber que existe um pedido de join aberto para este. Qualquer outro pedido recebido posteriormente para este dispositivo está sujeito a um sistema de semáforos e, num período de 5 segundos, o pedido é rejeitado através de uma resposta de erro. Este período foi testado e é o ideal para acompanhar qualquer latência existente por qualquer um dos *gateways*.

O Intermediário neste processo reencaminha o pedido para o Application Server, que responde com as chaves geradas para o dispositivo, que são reencaminhadas para o *gateway* que fez o pedido. Este *gateway* por fim responde ao dispositivo com uma resposta "Join Accept", com as chaves do dispositivo, ficando este registado na rede. O servidor central inicia uma atualização das chaves em todos os *gateways*, para que estes

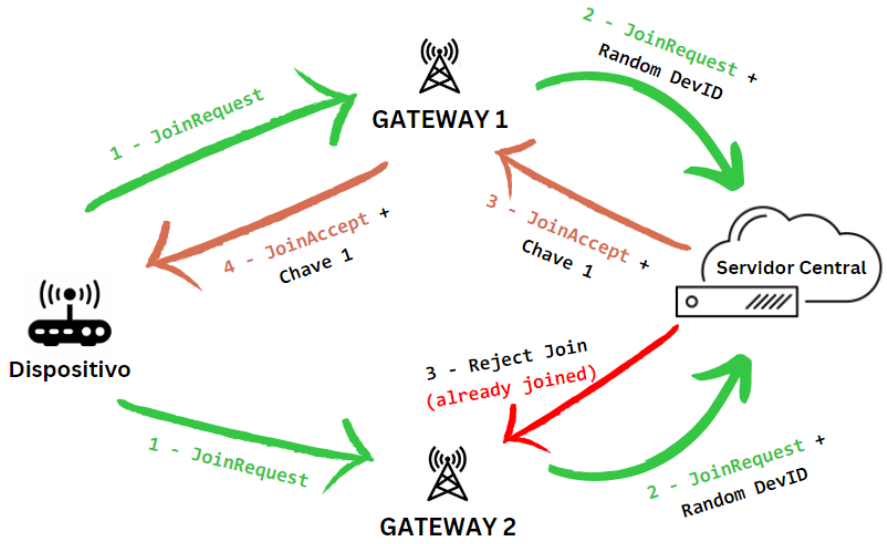

Figura 3.8: Processo join de um dispositivo com dois *gateways* ao alcance.

tenham conhecimento da nova chave do dispositivo e para que todos possam comunicar com este. Este processo é descrito na figura 3.8, onde é exemplificado o processo de associação na rede de um dispositivo com dois *gateways* ao alcance.

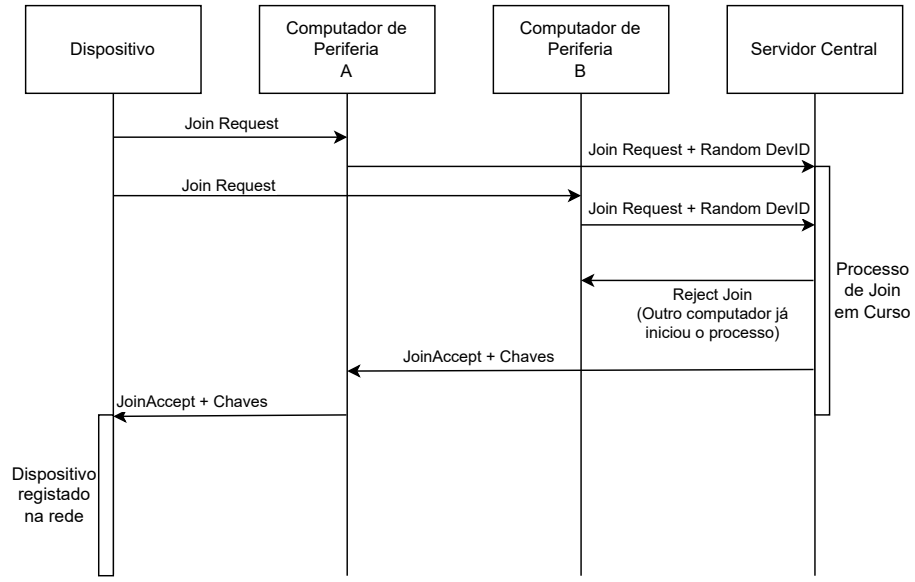

Figura 3.9: Fluxo de mensagens de registo de sensores, em OTAA.

O fluxo representado na figura 3.9 representa a circulação dos dados entre o dispositivo e os *gateways* ao alcance, neste caso, dois. Os computadores de periferia ao alcance necessitam de ter ligação com o Servidor central, de forma a serem autorizados a fazer o registo do sensor na rede.

3.5 Deteção de intrusões

Um sistema de deteção de intrusões (IDS) é um sistema que permite detetar dispositivos maliciosos na rede, que se podem estar a fazer passar por outros. Para detetar intrusões, foi parcialmente implementado o sistema de intrusões referido por [21] através da aplicação do algoritmo de classificação KNN (*K-nearest neighbors*).

O KNN é um algoritmo de classificação que, através de dados anteriormente conhecidos, determina a proximidade dos parâmetros que novos dados têm comparados com anteriores.

Este algoritmo obtém assim parâmetros do *gateway*, nomeadamente a sua localização e parâmetros da mensagem recebida, sendo estes o RSSI, SNR e a mensagem. Com estes valores e mensagens válidas, é possível gerar um modelo de mensagens provenientes de um determinado sensor e, por sua vez, executar o modelo aquando da receção de novas mensagens.

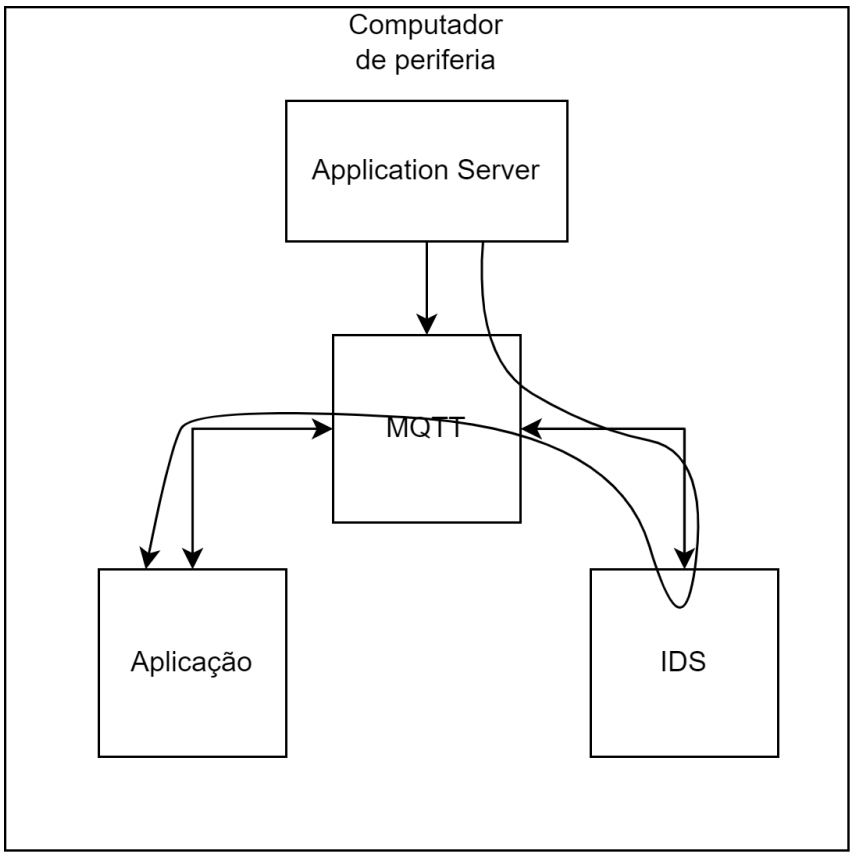

Figura 3.10: Representação do fluxo de uma mensagem para uma aplicação, com IDS.

A implementação deste sistema segue o fluxo representado na Figura 3.10. Nesse fluxo, todas as mensagens provenientes de um dispositivo passam pelo IDS.

Quando o sistema IDS deteta alguma atividade que considere uma intrusão, ele gera um alerta. Esse alerta é então encaminhado para o servidor MQTT existente no gateway, que por sua vez comunica com o servidor central.

Isso permite que o servidor MQTT receba informações de alerta sempre que o sistema IDS identifica uma possível intrusão. Esses alertas ao serem recebidos pelo servidor MQTT, este irá enviar o alerta para outros sistemas ou destinatários relevantes para a tomada de ações apropriadas em resposta à intrusão detetada.

4

Avaliação e resultados

Sendo esta tese apoiada pelo projeto Ferrovia 4.0 do qual o ISEL foi parceiro, foi possível a implementação de todo o conceito de rede e posteriores testes em ambiente perto do real. Assim, foi instalada num provedor de serviços cloud, a Oracle, uma máquina virtual de suporte à plataforma central e dois computadores preparados para instalação num veículo móvel como um comboio ou um carro (OBU). Com estes equipamentos, foi possível instalar, testar e posteriormente verificar e corrigir falhas na implementação. Irei de seguida explicar em detalhe os equipamentos e os testes realizados, começando desde a plataforma central, seguindo para os computadores de periferia e a forma de comunicação entre eles. Por fim, é descrita a forma de segurança implementada para as mensagens recebidas.

4.1 Rede de Comunicação

A rede de comunicação, neste contexto, consiste na ligação desde os dispositivos LoRa, passando pela plataforma central, até chegar às aplicações finais que podem ser executadas tanto nos dois computadores de fronteira como na nuvem, onde a plataforma central está instalada. Esta rede desempenha um papel crucial na transferência de dados e informações entre todos esses componentes do sistema, garantindo a integração e o funcionamento harmonioso de toda a infraestrutura. É através desta rede que os dados são recolhidos, processados e distribuídos para atender às necessidades das aplicações e dos utilizadores finais. Portanto, uma infraestrutura de rede robusta e fiável

é essencial para o sucesso e desempenho eficiente desse ecossistema de comunicação e computação distribuída. Assim, a implementação e avaliação do sistema encontra-se dividida em 4 partes: A Plataforma Central, instalada na Oracle, a Computação de Fronteira, instalada em computadores industriais e a ligação IP entre os diferentes equipamentos, que permite a troca de informações entre os computadores de fronteira e a plataforma central de forma segura. Por fim, será avaliado um método de segurança a implementar na comunicação LoRaWAN entre dispositivos e os *gateways*.

4.2 Plataforma central

Através dos recursos disponibilizados pelo projeto Ferrovia 4,0, foi possível concluir a conclusão da implementação dos túneis Wireguard falados anteriormente e também a implementação de um nó central da rede LoRaWAN, ficando composto assim o servidor central.

Foram criadas três máquinas virtuais, VM, com os nomes "Wireguard", "MQTT"e "QoS".

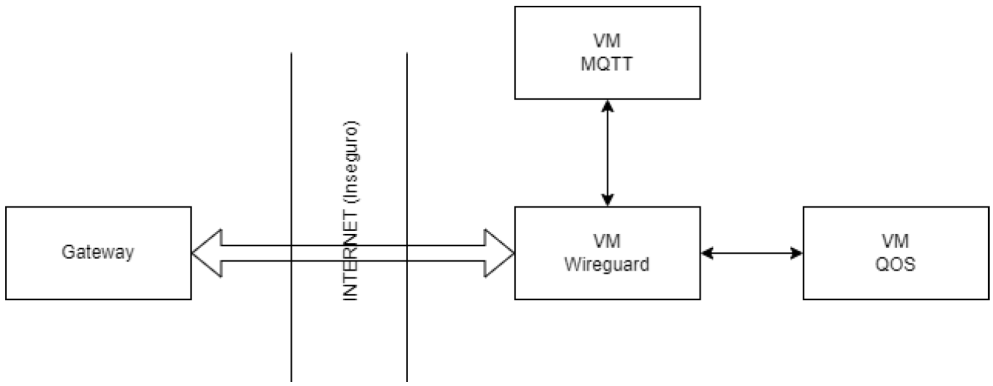

Figura 4.1: Esquema de ligação entre todas as VMs criadas

Na figura 4.1, está representada toda a ligação entre os *gateways* do lado esquerdo e da plataforma central do lado direito. No meio, encontra-se uma rede insegura, a Internet, por onde o tráfego seguro necessita de passar. A Internet é considerada uma rede insegura, pois é partilhada com outros sistemas e pessoas, tanto fidedignas, como prejudiciais. As três máquinas virtuais possuem as características representadas na tabela 4.1.

Na tabela 4.1, encontram-se representadas todas as máquinas virtuais usadas na implementação da plataforma central. A máquina "QoS", que detém este nome devido ao facto de ser maioritariamente usada por outro software, é onde foi instalada todo o

Tabela 4.1: Tabela de VMs, características e endereços IPs virtuais para comunicação entre os *gateways* e a plataforma central.

Nome	OS	Interfaces de Rede	Tipo de acesso das interfaces de rede	IPs Públicos	IPs Privados	CPU	RAM	HDD
QoS	Ubuntu 22.04	1	Interno	N/A	192.168.217.76	8 vCPU	16 GiB	100 GiB
Wireguard	Ubuntu 22.04	2	Externo	130.61.185.194 130.162.214.229	192.168.216.123 192.168.216.16	2 vCPU	2 GiB	20 GiB
MQTT	Ubuntu 22.04	1	Externo	141.144.235.37	192.168.216.9	2 vCPU	2 GiB	20 GiB

software Chirsptack, à exceção do MQTT, onde foi instalado numa máquina própria para o efeito por ser também partilhada por outros *softwares* onde é necessário o uso deste protocolo e também para facilitar a ponte entre os servidores MQTT existentes no *gateway* e no servidor central, podendo assim ser usado não só exclusivamente para a transferência de mensagens relacionadas com o sistema LoRaWAN.

4.3 VPN para comunicação

Na representação da infraestrutura ilustrada na figura 4.1, a máquina virtual "Wireguard"é a responsável pela gestão dos túneis VPN entre os *gateways* e a infraestrutura em nuvem, a plataforma central. Essa máquina efetua a gestão de dois túneis, sendo um deles dedicado às conexões mais críticas. Nesse primeiro túnel, a conexão física utilizada entre a *Gateway* e a Internet é baseada em redes móveis (2G-5G), conhecidas por sua alta disponibilidade e confiabilidade. No segundo túnel, é utilizada uma conexão física via Wi-Fi. Esse último, a comunicação é disponibilizada somente em localizações específicas de um *gateway* em movimento, garantindo a continuidade das comunicações durante esses momentos. O *gateway*, tal como a cloud, é também esta equipada com dois túneis Wireguard que garantem comunicações seguras entre a rede local e a nuvem. Esses túneis desempenham um papel fundamental ao possibilitar a comunicação segura entre os *gateways* e a plataforma central. Na tabela 4.1, são representados os IPs virtuais usados pelos *gateways*, tanto em rede móvel, como em rede wifi.

Todos os *gateways* possuem dois IPs da rede VPN, um da rede Celular e outro da rede Wi-Fi, independentes entre si, que permitem manter as comunicações ativas quando há rede móvel no alcance (mais provável, para comunicações críticas, mas baixa largura de banda), ou quando há Wi-Fi (menos provável, para comunicações menos críticas, mas alta largura de banda).

4.4 Computação de Fronteira

Os computadores usados foram provenientes do mesmo projeto, um deles estando representado na figura 4.2. Neste equipamento, foi possível instalar os componentes necessários ao funcionamento do Chirpstack, incluindo o *stack* LoRa refrente ao hardware e os componentes necessários para realizar as sincronizações das aplicações entre o servidor central. Esta aplicação de sincronização/coordenação dos *gateways* irá resultar tanto para partilha de aplicações e dispositivos, como também para efetuar autenticações.

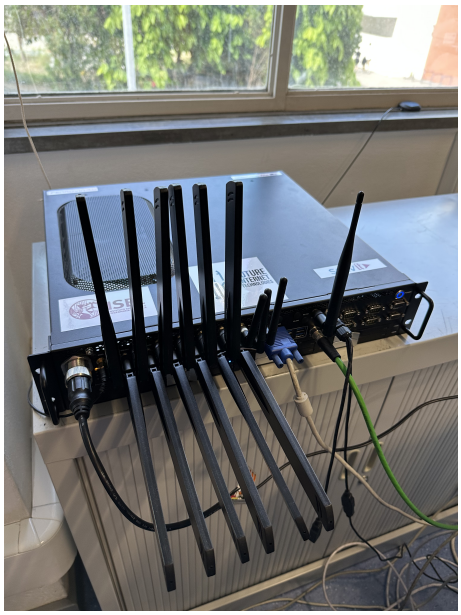

Figura 4.2: Fotografia de uma das gateways usadas para avaliação da implementação desta tese.

Estes *gateways* possuem mais que uma entrada de rede, sendo possível ter tanto rede wifi, como rede cabo e ainda 3 modems 5G e 1 modem LTE. Para a verificação do sistema, bastou usar uma rede por cabo, simulando uma rede móvel onde todo o tráfego crítico circula e uma rede wifi, onde o tráfego menos crítico circula.

A possibilidade de utilização de dois *gateways*, afastados um do outro, permitiu verificar o correto funcionamento da autenticação dos dispositivos LoRaWAN, a fase mais crítica do processo devido ao facto de com autenticações OTAA, ser necessário que as gateways sincronizassem com a plataforma central de forma a impedir a geração de mais que uma chave em simultâneo. A colocação de um dispositivo no espaço intermédio entre as duas gateways deu para comprovar o correto funcionamento do sistema, pois foi realizada uma autenticação apenas por um *gateway*. No entanto, após a autenticação do dispositivo, ambos se sincronizaram com a rede e foi possível de

imediato começar a receber mensagens por parte desses dispositivos tanto no *gateway* que realizou a autenticação, como no outro.

Com os *gateways* a gerar eventos sempre que algum dispositivo se registava e também sempre que realizava uma sincronização, foi possível verificar que num período de Fevereiro de 2023 a Agosto de 2023, um *gateways* realizou 37042 sincronizações com a plataforma central, 996 autenticações OTAA com sucesso. Estas autenticações são validadas no segundo *gateway*, onde é efetuada uma sincronização quando existe uma autenticação OTAA realizada por outro *gateway*.

4.5 Sistema de deteção de intrusões

Foi idealizada a implementação de um sistema de deteção de intrusões através do uso do algoritmo *K-nearest neighbors*, onde existe o conceito do uso de inteligência artificial para utilização de um sistema de aprendizagem, que faria a sua aprendizagem consoante as mensagens recebidas dos dispositivos [21].

Não foi possível a sua integração plena automatizada no funcionamento da infraestrutura como planeado, porém, foi possível realizar testes do algoritmo de aprendizagem, fazendo com que aprendesse determinadas mensagens desencriptadas e que fizesse deteção de intrusões, sendo possível chegar a conclusões de um *gateway* parado e em movimento.

Foi assim usado um dataset de 595 mensagens, 178 mensagens para testar o sistema e 417 para treino. De seguida, foram alteradas 25 mensagens com valores de mensagens anormais, para que fosse detetada uma eventual intrusão.

Na figura 4.3, verifica-se uma validação inicial dos dados, em que 1 é uma mensagem válida, 2 é uma mensagem possivelmente intrusiva e 3 são mensagens onde o sistema de deteção tem duvidas acerca da sua veracidade. Com estes resultados, podemos concluir que este método também funcionaria a detetar intrusões no *edge*, sendo que nenhuma das intrusões introduzidas no dataset foi classificada como "normal". No entanto, como certos parâmetros críticos na mensagem alteram, como o RSSI, SNR e localização (Figura 4.4), o modelo poderia tornar-se menos eficaz na adaptação a novas mensagens, pois estas poderiam ser consideradas duvidas. Nesta experiência, não foi possível criar um movimento contínuo do *gateway* e do sensor, sendo apenas possível mudar a posição da *gateway* para locais fixos. Potenciais soluções para este problema passariam pelo uso deste sistema exclusivamente em gateways estacionários ou pela criação de modelos nas localizações para onde este se irá deslocar. Num cenário na

ferrovia, poderiam existir múltiplos modelos ligados à localização do comboio, pois os parâmetros críticos deixariam de ter grandes alterações.

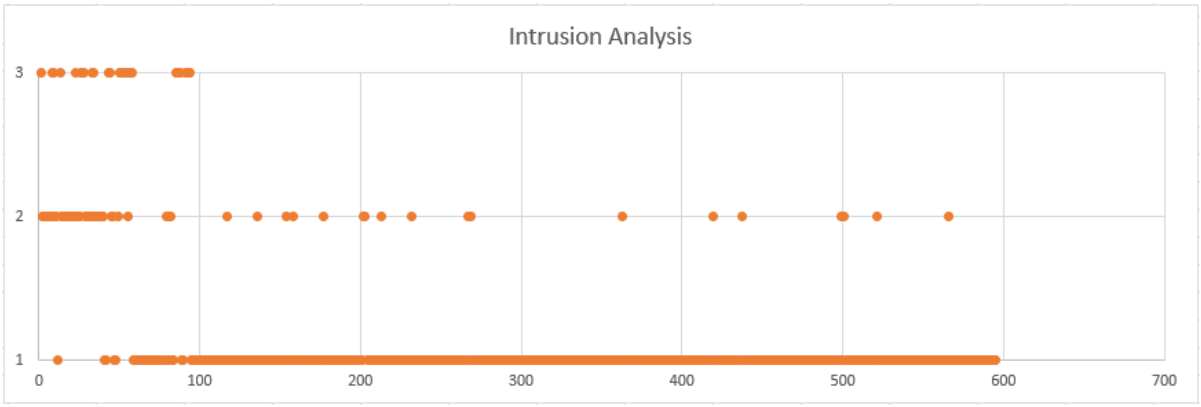

Figura 4.3: Dados provenientes de uma análise de uma série de dados provenientes de um *gateway*.

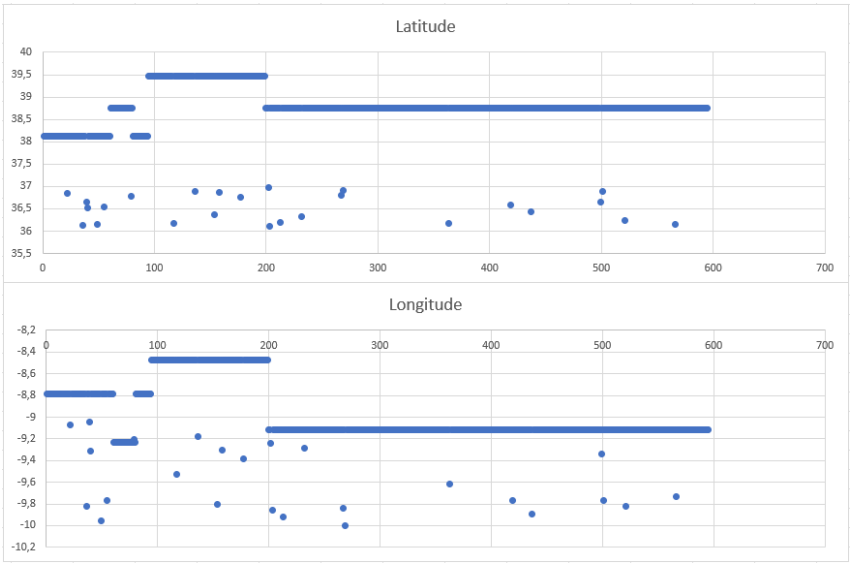

Figura 4.4: Localização das posições de um *gateway*, sincronizado com os dados do detetor de intrusões, onde X é o número da mensagem e Y a latitude/longitude.

A forma mais comum de avaliar um sistema IDS, é através de indicadores como Média de precisão, recall e F1 score, onde as duas primeiras métricas relacionam o número de verdadeiros positivos com o número de verdadeiros negativos e falsos positivos, enquanto que o F1 score é uma média que tem em consideração as duas prévias métricas.

Assim, é possível calcular o valor de precisão médio, sendo este:

$$\frac{NumeroDeMensagensCorretamenteClassificadas}{NumeroTotalDeMensagens} = \frac{513+25}{595} * 100 \approx 90\% \qquad (4.1)$$

O Recall, que relaciona os positivos detados com os positivos existentes, é:

$$\frac{NumeroDeIntrusoesDetetadas}{NumeroDeIntrusoesExistentes} = \frac{25}{25} = 1 \tag{4.2}$$

Por fim, o F1 Score, que é calculado através da precisão:
Precisão:

$$\frac{NumerodeVerdadeirosPositivos}{NumerodeFalsosPositivos} = \frac{25}{29} \approx 0.8620 \tag{4.3}$$

F1 Score:

$$\frac{2 * Precisao * Recall}{Precisao + Recall} = \frac{2 * 0.8620 * 1}{0.8620 + 1} \approx 0.9258 \tag{4.4}$$

Com este teste, conclui-se que o sistema de aprendizagem usando o algoritmo KNN seria bastante útil para *gateways* estáticas, onde seria necessário processamento de fronteira, mas que não funcionaria corretamente em *gateways* em movimento, onde seria necessário retreinar o sistema sempre que este mudava de posição. Apesar de se comprovar que o sistema iria detetar uma mudança no comportamento do sensor, com 90% de precisão, a alteração de posição pode colocar duvidas ao algoritmo se existem mais mensagens de intrusão.

5

Conclusões e Trabalho Futuro

Este projeto possibilitou a avaliação do vasto potencial ainda por explorar e implementar no âmbito da Internet das Coisas (IoT), especialmente no que diz respeito à computação na periferia para aplicações críticas. Nesta tese, demonstraram-se as possibilidades do IoT, utilizando as redes LoRaWAN e implementando computação na periferia com hardware já existente. Ao associar dispositivos LoRaWAN à rede, estes conseguiram autenticar-se e transmitir dados. Os computadores periféricos coordenaram-se eficazmente entre si através da plataforma central, mesmo durante interrupções na ligação à Internet, proporcionando uma operação contínua. Embora não tenha sido possível implementar o sistema em ambiente real conforme inicialmente concebido, foi viável instalar mais de um computador periférico em diferentes localizações geográficas, permitindo simular o movimento de sensores entre os dois *gateways*. Essas simulações resultaram em mais de 37 mil sincronizações em seis meses e mais de 900 autenticações, evidenciando assim a eficácia da arquitetura proposta. No decorrer do projeto, foi também instalado um sistema de deteção de intrusões utilizando o algoritmo KNN, alcançando resultados positivos com uma precisão de 90% na rede LoRaWAN. Como perspetiva futura, seria relevante aprimorar a geolocalização dos *gateways*. O requisito de obter todas as informações de dispositivos e chaves pode gerar sobrecarga no sistema ao lidar com centenas ou milhares de dispositivos. Seria interessante que o sistema reconhecesse a localização típica de dispositivos, recorrendo apenas a uma *cache*. Quando um dispositivo aparece pela primeira vez, o *gateway* tentaria comunicar com a plataforma central e sincronizar apenas os dados essenciais. Outro desenvolvimento futuro relaciona-se com a deteção de intrusões. Apesar dos problemas identificados, o

sistema poderia ser utilizado em *gateways* fixos, enquanto *gateways* móveis necessitariam de conhecer sua localização e adquirir o modelo de aprendizagem aplicável à sua área de atuação.

Referências

[1] Dhaval Patel & Myounggyu Won, "Experimental Study on Low Power Wide Area Networks (LPWAN) for Mobile Internet of Things", 2017, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1705.06926. URL: https://arxiv.org/abs/1705.06926 (acedido em 20/12/2023).

[2] Kais Mekki, Eddy Bajic, Frédéric Chaxel & Fernand Meyer, "Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT", mar. de 2018. DOI: 10.1109/PERCOMW.2018.8480255.

[3] Biswajit Paul, "An Overview of LoRaWAN", *WSEAS TRANSACTIONS ON COMMUNICATIONS*, vol. 19, páginas 231–239, jan. de 2021. DOI: 10.37394/23204.2020.19.27.

[4] Autoridade Nacional de Telecomunicações. "DECISÃO relativa às Estações de pequena potência e curto alcance ("SRD - Short Range Devices")". (1 de jul. de 2021).

[5] Slim Loukil, Lamia Fourati, Anand Nayyar & K-W-A Chee, "Analysis of LoRaWAN 1.0 and 1.1 Protocols Security Mechanisms", *Sensors*, vol. 22, pág. 3717, mai. de 2022. DOI: 10.3390/s22103717.

[6] Ramon Sanchez-Iborra, Jesus Sanchez-Gomez, Salvador Pérez, Pedro Fernandez, José Santa, José Hernández-Ramos & Antonio Skarmeta, "Enhancing LoRaWAN Security through a Lightweight and Authenticated Key Management Approach", *Sensors*, vol. 18, jun. de 2018. DOI: 10.3390/s18061833.

[7] Collins Mwakwata, Hassan Malik, Muhammad Alam, Yannick Le Moullec, Sven Parand & Shahid Mumtaz, "Narrowband internet of things (NB-IoT): From physical (PHY) and media access control (MAC) layers perspectives", *Sensors*, vol. 19, pág. 34, jun. de 2019. DOI: 10.3390/s19112613.

[8] Mehzabien Iqbal, Abu Abdullah & Farzana Shabnam, "An Application Based Comparative Study of LPWAN Technologies for IoT Environment", jan. de 2020, páginas 1857–1860. DOI: 10.1109/TENSYMP50017.2020.9230597.

[9] Wenping Kong, Xiaoyong Li, Liyang Hou & Yanrong Li, "An Efficient and Credible Multi-Source Trust Fusion Mechanism Based on Time Decay for Edge Computing", *Electronics*, vol. 9, pág. 502, mar. de 2020. DOI: 10.3390/electronics9030502.

[10] Min Chen, Yiming Miao, Yixue Hao & Kai Hwang, "Narrow Band Internet of Things", *IEEE Access*, vol. PP, páginas 1–1, set. de 2017. DOI: 10.1109/ACCESS.2017.2751586.

[11] Shihab Jimaa, Michael Chai, Yue Chen & Yasir Alfadhl, "LTE-A an overview and future research areas", out. de 2011, páginas 395–399. DOI: 10.1109/WiMOB.2011.6085370.

[12] Anshuman Dash, Satyajit Pal & Chinmay Hegde, "Ransomware Auto-Detection In IoT Devices Using Machine Learning", dez. de 2018.

[13] Kais Mekki, Eddy Bajic, Frédéric Chaxel & Fernand Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment", vol. 5, páginas 1–7, mar. de 2019. DOI: 10.1016/j.icte.2017.12.005.

[14] Carlos Eduardo Governo Rodrigues, "Sistema de encaminhamento resiliente para serviços críticos sobre redes públicas móveis", Tese de Mestrado, Instituto Superior de Engenharia de Lisboa - ISEL, 2022.

[15] Muhammad Masdani & Denny Darlis, "A comprehensive study on MQTT as a low power protocol for internet of things application", *IOP Conference Series: Materials Science and Engineering*, vol. 434, pág. 012 274, dez. de 2018. DOI: 10.1088/1757-899X/434/1/012274.

[16] Han Wu, Shang Zhihao & Katinka Wolter, "Performance Prediction for the Apache Kafka Messaging System", ago. de 2019, páginas 154–161. DOI: 10.1109/HPCC/SmartCity/DSS.2019.00036.

[17] Jay Kreps. "Benchmarking apache kafka: 2 million writes per second (on three cheap machines)". (27 de abr. de 2014), URL: https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines (acedido em 09/02/2023).

[18] A. Antony Franklin & Supriya Dilip Tambe, "Multi-access edge computing in cellular networks", *CSI Transactions on ICT*, vol. 8, n.º 1, páginas 85–92, mar. de 2020, ISSN: 2277-9078, 2277-9086. DOI: 10.1007/s40012-020-00276-6. URL: https://link.springer.com/10.1007/s40012-020-00276-6 (acedido em 30/09/2023).

[19] Ivan Fardin, Stefano Milani, Francesca Cuomo & Ioannis Chatzigiannakis, "Enabling Edge Computing over LoRaWAN: A Device-Gateway Coordination Protocol", em *Proceedings of the 12th ACM International Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, sér. DIVANet '22, New York, NY, USA: Association for Computing Machinery, 2022, 23–30, ISBN: 9781450394826. DOI: 10.1145/3551662.3560926. URL: https://doi.org/10.1145/3551662.3560926.

[20] Biswajeeban Mishra, Biswaranjan Mishra & Attila Kertesz, "Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements", *Energies*, vol. 14, n.º 18, 2021, ISSN: 1996-1073. DOI: 10.3390/en14185817. URL: https://www.mdpi.com/1996-1073/14/18/5817.

[21] Filipe Fidalgo Torres Costa, "Integração de sistema de deteção de intrusões numa gateway de comunicações para a ferrovia", Tese de Mestrado, Instituto Superior de Engenharia de Lisboa - ISEL, 2022.

Anexo: Estrutura de dados enviados pela plataforma central para sincronização dos gateways


```
{"applications": [{
      "id": "13",
      "name": "Polysense",
      "description": "Polysense-sensors",
      "organizationID": "1",
      "serviceProfileID": "083ce9c1-c053-4787-bbb5-85e117d58971",
      "serviceProfileName": "local_service_profile",
      "devices": [{
          "devEUI": "7a20bdfffe0005c7",
          "name": "Polysense_C7",
          "applicationID": "13",
          "description": "Sensor no cepo",
          "deviceProfileID": "5662be23-15b1-4069-b9a8-ab5ef16ff2df",
          "skipFCntCheck": true,
          "referenceAltitude": 0,
          "variables": {},
          "tags": {},
          "isDisabled": false,
          "keys": {
            "activate": {
              "devEUI": "7a20bdfffe0005c7",
```

```
            "devAddr": "013cc3b9",
            "appSKey": "eaa2071679bd550e591bd53fe01088c4",
            "nwkSEncKey": "97d1750724ccd67091d545e813cf19a1",
            "sNwkSIntKey": "97d1750724ccd67091d545e813cf19a1",
            "fNwkSIntKey": "97d1750724ccd67091d545e813cf19a1",
            "fCntUp": 1559,
            "nFCntDown": 0,
            "aFCntDown": 0
          },
          "keys": {
            "devEUI": "7a20bdfffe0005c7",
            "nwkKey": "11223344556677889900aabbccddeeff",
            "appKey": "00000000000000000000000000000000",
            "genAppKey": ""
          }
        }
      }
    ]
  },
  "device-profiles": [{
      "id": "699e5cf3-a05e-4478-bf5c-a58c33f806df",
      "name": "ABP2",
      "organizationID": "1",
      "networkServerID": "3",
      "supportsClassB": false,
      "classBTimeout": 0,
      "pingSlotPeriod": 0,
      "pingSlotDR": 0,
      "pingSlotFreq": 0,
      "supportsClassC": false,
      "classCTimeout": 0,
      "macVersion": "1.0.4",
      "regParamsRevision": "A",
      "rxDelay1": 0,
      "rxDROffset1": 0,
      "rxDataRate2": 0,
      "rxFreq2": 0,
      "factoryPresetFreqs": [],
      "maxEIRP": 0,
      "maxDutyCycle": 0,
      "supportsJoin": false,
      "rfRegion": "EU868",
      "supports32BitFCnt": false,
      "payloadCodec": "",
```

```
      "payloadEncoderScript": "",
      "payloadDecoderScript": "",
      "geolocBufferTTL": 0,
      "geolocMinBufferSize": 0,
      "tags": {},
      "uplinkInterval": "5s",
      "adrAlgorithmID": "default"
    },{
      "id": "5662be23-15b1-4069-b9a8-ab5ef16ff2df",
      "name": "OTA Device Profile",
      "organizationID": "1",
      "networkServerID": "3",
      "supportsClassB": false,
      "classBTimeout": 0,
      "pingSlotPeriod": 0,
      "pingSlotDR": 0,
      "pingSlotFreq": 0,
      "supportsClassC": false,
      "classCTimeout": 0,
      "macVersion": "1.0.3",
      "regParamsRevision": "A",
      "rxDelay1": 0,
      "rxDROffset1": 0,
      "rxDataRate2": 0,
      "rxFreq2": 0,
      "factoryPresetFreqs": [],
      "maxEIRP": 0,
      "maxDutyCycle": 0,
      "supportsJoin": true,
      "rfRegion": "EU868",
      "supports32BitFCnt": false,
      "payloadCodec": "",
      "payloadEncoderScript": "",
      "payloadDecoderScript": "",
      "geolocBufferTTL": 0,
      "geolocMinBufferSize": 0,
      "tags": {},
      "uplinkInterval": "5s",
      "adrAlgorithmID": "default"
    }]}
```