UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Cristian Augusto Wülfing

# ANÁLISE E IMPLEMENTAÇÃO DO SUPORTE AO TRANSPORTE DE PACOTES IPv6/UDP EM REDES LORAWAN UTILIZANDO O PROTOCOLO DE COMUNICAÇÃO SCHC APLICADO A LEITORES INTELIGENTES DE ENERGIA ELÉTRICA

Santa Maria, RS
2023

Cristian Augusto Wülfing

# ANÁLISE E IMPLEMENTAÇÃO DO SUPORTE AO TRANSPORTE DE PACOTES IPv6/UDP EM REDES LORAWAN UTILIZANDO O PROTOCOLO DE COMUNICAÇÃO SCHC APLICADO A LEITORES INTELIGENTES DE ENERGIA ELÉTRICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Engenheiro de Computação**.

Orientador: Prof. Carlos Henrique Barriquello

Santa Maria, RS
2023

**Cristian Augusto Wülfing**

# ANÁLISE E IMPLEMENTAÇÃO DO SUPORTE AO TRANSPORTE DE PACOTES IPv6/UDP EM REDES LORAWAN UTILIZANDO O PROTOCOLO DE COMUNICAÇÃO SCHC APLICADO A LEITORES INTELIGENTES DE ENERGIA ELÉTRICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Engenheiro de Computação**.

**Aprovado em 2 de fevereiro de 2023:**

_____

**Carlos Henrique Barriquello, Dr. (UFSM)**
**(Presidente/Orientador)**

_____

**Carlos Raniery Paula dos Santos, Dr. (UFSM)**

_____

**Celio Trois, Dr. (UFSM)**

Santa Maria, RS
2023

## AGRADECIMENTOS

*Primeiramente quero agradecer a minha família por todo o apoio nestes anos, em especial a minha Mãe Lorti por me incentivar, acreditar e estar ao meu lado nos meus projetos desde o princípio, mesmo que distante fisicamente neste período de graduação.*

*Agradeço aos professores e aos funcionários da Universidade Federal de Santa Maria que, de uma forma ou de outra, contribuíram para o meu crescimento pessoal e profissional através das aulas, discussões, materiais extras e horas de laboratório. Este trabalho reflete também o ensino de qualidade o qual tive acesso ao longo desta trajetória.*

*Agradeço a todas as pessoas e amizades que nasceram ao longo das aulas, nos laboratórios, na ITEP Jr., na i9 Liga de Empreendedorismo, nas atividades extraclasse e no grupo "New Age". A minha percepção de relacionamento com pessoas teve um salto de aprendizado imensurável depois de todas as experiências vividas.*

*Agradeço a minha namorada Cássia por todo o apoio e companheirismo nos últimos anos. Sou muito grato pelas conquistas compartilhadas e pelas inúmeras vezes em que as conversas trouxeram leveza e clareza ao longo das dificuldades.*

*Agradeço aos "rapazes" da Fox IoT por todos os momentos e pela confiança ao longo destes oito meses, nos quais pude não somente adquirir uma bagagem técnica gigantesca, mas também crescer muito como profissional. O presente trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES/PROEX) – Código de Financiamento 001 e das empresas Mux Energia e Fox IoT através do projeto de P&D ANEEL PD-00401-0005/2020.*

*Por fim, quero agradecer ao meu Orientador, Carlos Henrique Barriquello, por todas as trocas de experiências e conhecimentos adquiridos nestes últimos três anos, seja em sala de aula ou no projeto de Pesquisa e Extensão onde atuei como Bolsista de Iniciação Científica.*

*Este trabalho reflete não apenas os trabalhos realizados durante os últimos oito meses, mas também toda a trajetória percorrida ao longo destes anos de graduação. Sou muito grato por todas as experiências e todas as pessoas que tive a honra de conhecer e conviver ao longo deste período.*

# RESUMO

## ANÁLISE E IMPLEMENTAÇÃO DO SUPORTE AO TRANSPORTE DE PACOTES IPv6/UDP EM REDES LORAWAN UTILIZANDO O PROTOCOLO DE COMUNICAÇÃO SCHC APLICADO A LEITORES INTELIGENTES DE ENERGIA ELÉTRICA

AUTOR: Cristian Augusto Wülfing
Orientador: Carlos Henrique Barriquello

Com o constante crescimento de dispositivos relacionados à Internet das Coisas, faz-se necessária a implementação de soluções que tornem possível a comunicação eficiente e robusta entre os dispositivos finais e as aplicações que venham a estabelecer conexão com estes. Neste cenário, a implementação das LPWANs está em constante avanço, apesar de que estas tecnologias não suportam o endereçamento IP de forma nativa devido às suas limitações técnicas. Como consequência, estes dispositivos não podem ser acessados diretamente por meio dos padrões de Internet estabelecidos e amplamente utilizados. Este trabalho teve como objetivo a implementação de uma camada adaptativa com o intuito de suprir essa necessidade, considerando a aplicação desta em um ambiente completo que tem como propósito final a comunicação com um medidor inteligente de energia elétrica. Para isso, o protocolo SCHC foi implementado com o intuito de possibilitar a comunicação direta entre uma rede LoRaWAN e a Internet por meio de pacotes IPv6/UDP. Além de estabelecer a comunicação remota com o medidor a partir de uma aplicação responsável pelo envio e recebimento de pacotes DLMS/COSEM, este trabalho apresenta também uma análise do protocolo SCHC operando em uma rede LoRaWAN, de modo a avaliar as implementações dos algoritmos e seu desempenho, considerando a visão de uma rede de comunicação como um todo. Por fim, possíveis melhorias são apresentadas com o intuito de viabilizar melhores resultados acompanhados da continuação do trabalho.

**Palavras-chave:** SCHC. IPv6. LoRaWAN. Medidor inteligente. IoT.

# ABSTRACT

## ANALYSIS AND IMPLEMENTATION OF IPv6/UDP PACKET TRANSPORT SUPPORT IN LORAWAN NETWORKS USING THE SCHC COMMUNICATION PROTOCOL APPLIED TO SMART METERS

AUTHOR: Cristian Augusto Wülfing
ADVISOR: Carlos Henrique Barriquello

With the constant growth of devices related to the Internet of Things, it is necessary to implement solutions that make possible efficient and robust communications between end devices and the applications that may establish a connection with them. In this scenario, the implementation of LPWANs is constantly advancing, although these technologies do not natively support IP addressing due to their technical limitations. As a result, these devices cannot be directly accessed via established and widely used Internet standards. This work aimed to implement an adaptive layer to meet this need, considering its application in a complete environment, in which the final purpose is the communication with a smart meter. For this, the SCHC protocol was implemented to enable direct communication between a LoRaWAN network and the Internet through IPv6/UDP packets. In addition to establishing remote communication with the meter from an application responsible for sending and receiving DLMS/COSEM packets, this project also presents an analysis of the SCHC protocol operating in a LoRaWAN network, in order to to evaluate the algorithms implementations and their performance, considering a complete communication network. Finally, possible improvements are presented in order to enable better results accompanied by the continuation of the work.

**Keywords:** SCHC. IPv6. LoRaWAN. Smart Meter. IoT.

## LISTA DE FIGURAS

## LISTA DE GRÁFICOS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| $ABNT$ | Associação Brasileira de Normas Técnicas |
| $ACK$ | *Acknowledgement* |
| $ADR$ | *Adaptive Data Rate* |
| $AES$ | *Advanced Encryption Standard* |
| $API$ | *Application Programming Interface* |
| $CDA$ | *Compression/Decompression Actions* |
| $CRC$ | *Cyclic redundancy check* |
| $CSS$ | *Chirp Spread Spectrum* |
| $FSM$ | *Finite-state machine* |
| $ID$ | *Identity document* |
| $IETF$ | *Internet Engineering Task Force* |
| $IP$ | *Internet Protocol* |
| $ISM$ | *Industrial, Scientific and Medical radio bands* |
| $IoT$ | *Internet of Things* |
| $LPWAN$ | *Low Power Wide Area Network* |
| $M2M$ | *Machine to Machine* |
| $MQTT$ | *Message Queuing Telemetry Transport* |
| $MTU$ | *Maximum Transmission Unit* |
| $NBR$ | Norma Brasileira |
| $OBIS$ | *Object Identification System* |
| $RCS$ | *Reassembly Check Sequence* |
| $RF$ | Rádio Frequência |
| $RFC$ | *Request for Comments* |
| $SCHC$ | *Static Context Header Compression* |

| | |
|---|---|
| $SF$ | *Spreading factor* |
| $SO$ | Sistema operacional |
| $TCP$ | *Transmission Control Protocol* |
| $TTN$ | *The Things Network* |
| $UDP$ | *User Datagram Protocol* |
| $USB$ | *Universal Serial Bus* |

**SUMÁRIO**

# 1 INTRODUÇÃO

A Internet das Coisas (*Internet of Things* - IoT) é um paradigma que permite a criação de novos modelos de negócios a partir de redes de dispositivos conectados à Internet, sendo capazes de coletar e transferir informações entre si juntamente à interação com mecanismos para cruzamento e análise de dados presentes na nuvem. Neste contexto, torna-se possível o desenvolvimento de soluções de baixo custo e alta interconectividade em diversos contextos como agricultura, cidades inteligentes, automação industrial e residencial, distribuição de energia elétrica, dentre outros (BORGIA, 2014).

As tecnologias voltadas às necessidades destas aplicações estão em constante desenvolvimento, como é o caso das LPWANs (*Low Power Wide Area Networks*). Padronizadas pela RFC 8376 (S. Farrell, 2018), suas principais características envolvem longas áreas de cobertura, baixas taxas de transmissão de dados e operação com longa duração da bateria. Neste contexto, a LoRa Alliance© desenvolveu o protocolo de comunicação chamado LoRaWAN (LoRa Alliance, 2015), que, dentro da categoria LPWAN, implementa funcionalidades das camadas de rede e aplicação entre dispositivos com comunicação LoRa (Francois Sforza, 2013) e a Internet.

Apesar da possibilidade de comunicação bidirecional a longas distâncias e baixo custo, as limitações dessa tecnologia impedem o envio de grandes volumes de dados em um único pacote. Além disso, os dispositivos conectados a essa rede não são acessíveis diretamente pela Internet, de modo que um *gateway* é necessário para o encaminhamento das mensagens entre esses dispositivos e as redes IP (*Internet Protocol*). Consequentemente, não é possível fazer o uso nativo das mais diversas aplicações e protocolos já consolidados neste ambiente, ficando os dispositivos isolados e dependentes de aplicações intermediárias (MUÑOZ et al., 2022).

Com isso, a IETF (*Internet Engineering Task Force*) desenvolveu uma especificação que padroniza o protocolo SCHC (*Static Context Header Compression and Fragmentation*) (MINABURO et al., 2020), que, atuando como uma camada adaptativa acima da camada de rede, trata da compressão e da fragmentação dos pacotes IPv6 com o intuito de possibilitar o transporte destes nas redes LPWAN. Embora as limitações da rede continuem existindo, este protocolo prevê as adequações necessárias de modo a viabilizar o uso de tecnologias amplamente adotadas na Internet, tanto nas aplicações quanto nos dispositivos, incluindo o endereçamento destes para comunicação bidirecional e direta.

A necessidade de desenvolvimento e de utilização desta tecnologia surgiu a partir da realização de um trabalho realizado juntamente com medidores inteligentes de energia elétrica, que analisou o uso do protocolo DLMS/COSEM (*Device Language Message Specification, Companion Specification for Energy Metering*) juntamente com um medidor em dois *setups* (WÜLFING et al., 2022). O primeiro considerou a comunicação entre o me-

didor e um microcontrolador, de modo que este era responsável por coletar informações e enviá-las já tratadas a uma aplicação externa. O segundo *setup* também considerou a comunicação entre estes dois dispositivos, porém os pacotes DLMS/COSEM eram trafegados de forma íntegra, sem tratamento nos dados. Neste segundo cenário, pacotes que eram frequentemente enviados foram identificados e substituídas por códigos de identificação, os quais eram posteriormente transmitidos na rede e remontados no seu destino. A simplicidade deste método não foi suficiente para o uso mais amplo do protocolo DLMS/COSEM, surgindo então o SCHC como uma alternativa interessante. Esta possibilidade vai de encontro à recente norma ABNT NBR 16968(ABNT, 2022), que regulamenta o uso do DLMS/COSEM na comunicação com medidores inteligentes no Brasil a partir do uso do protocolo SCHC em redes LoRaWAN.

Este trabalho tem como contribuição a implementação e o uso do protocolo SCHC com o intuito de possibilitar a troca de pacotes IPv6/UDP em dispositivos que não eram capazes de transmiti-los nativamente até então, de modo que a rede LPWAN torna-se completamente transparente e agnóstica aos pacotes que por ela trafegam (SANCHEZ-GOMEZ et al., 2020). Para isso, é utilizada uma rede LoRaWAN que tem como intuito promover a comunicação com os dispositivos. Conectado a esta rede através da Internet, um *gateway* de aplicação é responsável pela conversão de tecnologia entre a rede LoRaWAN e uma rede IP, que utiliza o IPv6/UDP como protocolo de comunicação. Desta forma, pacotes deste protocolo são gerados a partir de aplicações e posteriormente trafegados neste sistema tendo como destino dispositivos conectados na rede LoRaWAN. Considerando que estes dispositivos não possuem a implementação nativa do protocolo IPv6/UDP, o protocolo SCHC é responsável por promover uma camada de adaptação neste cenário, possibilitando com isso a troca de pacotes IPv6/UDP entre os dispositivos e as aplicações que venham a estabelecer comunicação.

Neste contexto, são utilizados dois dispositivos conectados à rede LoRaWAN, havendo uma aplicação destinada a cada um com o intuito de estabelecer a respectiva comunicação. Os objetivos dos dois dispositivos são diferentes, sendo o primeiro responsável por promover a comunicação com um medidor inteligente de energia elétrica através do protocolo DLMS/COSEM. As solicitações de informações acerca deste medidor são geradas em uma aplicação encarregada por esta função, de modo que o medidor recebe estas solicitações e as responde após o processamento interno. Os pacotes relacionados a este método de solicitação-resposta utilizam o protocolo DLMS/COSEM, de modo que estes são encapsulados em pacotes IPv6/UDP para o transporte ao longo do sistema de comunicação proposto neste trabalho. Já o segundo dispositivo conectado à rede LoRaWAN tem como intuito a produção de tráfego aleatório, tanto em relação às informações quanto ao tamanho dos dados gerados. Isso possibilita a realização de análises do protocolo SCHC operando em uma rede LoRaWAN, de modo a avaliar os modos de operação do protocolo, o desempenho da rede e os custos de tráfego, além de realizar comparações

com uma rede otimizada e com menos perdas de transmissão.

A implementação deste trabalho contribui para que dispositivos relacionados à Internet das Coisas tornem-se aptos a se comunicar de forma semelhante ao restante da Internet, incluindo cenários como medidores de energia elétrica, sensores e componentes de cidades inteligentes (iluminação pública, semáforos, estações de ônibus etc.). A possibilidade de endereçar dispositivos conectados à rede LoRaWAN por meio de endereços IPv6 viabiliza o uso de aplicações e protocolos já consolidados em redes IP, de modo que os dispositivos tornam-se ainda mais aptos a interagirem com outras redes por meio de serviços hospedados na nuvem. Desta forma, facilita-se a criação de ecossistemas envolvendo a integração entre dispositivos, sensores e mecanismos de processamento, armazenamento e análise de dados.

O trabalho está organizado de modo a primeiramente apresentar o embasamento teórico dos principais componentes utilizados ao longo do desenvolvimento, trazendo uma visão que parte da rede utilizada, passando pelo protocolo SCHC e por fim informações acerca da comunicação com o medidor inteligente de energia elétrica. Na sequência, é abordada a implementação e utilização desses elementos aplicados aos interesses do trabalho. Por fim, são apresentados os resultados obtidos a partir dos experimentos, juntamente com as discussões que estes levaram, finalizando então com a conclusão do trabalho. Os algoritmos implementados no trabalho podem ser acessados através do repositório GitHub[1].

[1] Disponível em: https://github.com/CristianWulfing/SCHC-Project

# 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os conceitos utilizados ao longo do trabalho, com o intuito de fundamentar e ambientar os componentes que posteriormente terão suas implementações apresentadas no capítulo de Desenvolvimento. Este capítulo inicia com a introdução das redes LPWAN, utilizadas como meio de comunicação principal nos dispositivos empregados. Na sequência, os protocolos IP e SCHC serão abordados devido às suas importâncias no transporte dos dados a serem trafegados juntamente, com a possibilidade do suporte dos pacotes IP dentro da rede LPWAN. Por fim, será apresentado o protocolo DLMS/COSEM, que é utilizado para a comunicação com o medidor inteligente de energia elétrica empregado no trabalho.

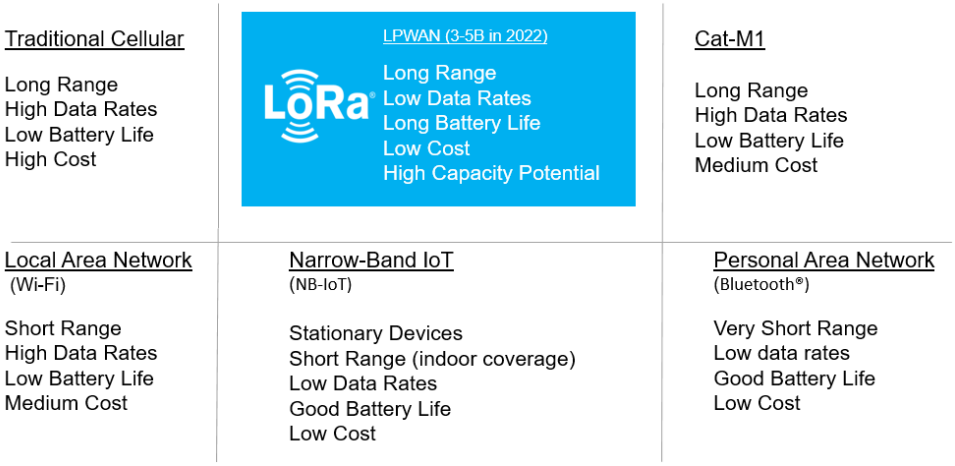
## 2.1 LPWAN

A tecnologia LPWAN é definida como uma classe de padrões e soluções de comunicação sem fio voltadas à Internet das Coisas, caracterizada pelas grandes áreas de cobertura, baixas taxas de transmissão de dados com pequenos tamanhos de pacotes e operação com longa vida útil de baterias. Ganhando cada vez mais popularidade em comunidades industriais e de pesquisa, é capaz de alcançar comunicação com distâncias de 10 a 40km no meio rural e 1 a 5km em ambientes urbanos, tendo um custo médio de US$3 a US$7 por dispositivo, duração de bateria de mais de 10 anos e suporte a mais de 100 mil dispositivos conectados simultaneamente (S. Farrell, 2018; MEKKI et al., 2019; CENTENARO et al., 2016; Margaret Rouse, 2017).

De forma geral, as aplicações de IoT requerem maior eficiência energética e menor complexidade na comunicação dos dispositivos, que são implantados em redes escaláveis. Atualmente, tecnologias como WLANs (IEEE 802.11), Bluetooth (IEEE 802.15.1) e ZigBee(IEEE 802.15.4) são utilizadas para aplicações em ambientes internos e de curto alcance devido as suas limitações (apesar das duas primeiras suportarem comunicação de dados de alta velocidade). Não obstante, as soluções baseadas em comunicação celular como 2G, 3G e 4G foram projetadas inicialmente para a comunicação de voz e dados, não atendendo às métricas de desempenho necessárias por consumirem energia excessiva do dispositivo e possuírem elevado custo de operação (CHAUDHARI; ZENNARO; BORKAR, 2020). A Figura 1 mostra a comparação das redes LPWAN com tecnologias conhecidas, tendo a tecnologia LoRa como referência para o primeiro caso.

Neste contexto, as redes LPWAN podem ser customizadas e empregadas em uma grande variedade de aplicações, demandando diferentes configurações e infraestruturas em cada situação. A exemplo disso, tem-se as aplicações de cidades inteligentes, que

Figura 1 – Comparação entre LoRa e outras opções de comunicação sem fio.




Fonte: (SEMTECH, 2022).

podem necessitar de ampla cobertura de comunicação com o intuito de abranger a maior área possível da cidade. Neste cenário se aplicaria a leitura de parâmetros como umidade, temperatura, níveis de poluição e de ruído, a medição de consumo de energia elétrica, gás e água, o controle de tráfego como densidade por área, estacionamento inteligente e seleção de melhor rota, serviços essenciais como coleta de lixo, gerenciamento de resíduos e muito mais (CHAUDHARI; ZENNARO; BORKAR, 2020).

### 2.1.1 LoRa

LoRa é uma tecnologia de camada física para comunicação sem fio bidirecional LPWAN desenvolvida e patenteada pela Semtech (Francois Sforza, 2013), que opera em uma faixa do espectro de frequências conhecida como banda livre de licenças ISM (*Industrial, Scientific and Medical*). Dentre as possíveis frequências de operação utilizadas no mundo, o Brasil adota o padrão Australiano AU915 de 915 MHz (variando de 902 MHz a 907,5 MHz e de 915 MHz a 928 MHz), homologado em 2018 pela Agência Nacional de Telecomunicações (ANATEL) a partir da lei número 6.506, que aprova os procedimentos para avaliar a conformidade de equipamentos de radiocomunicação com radiação restrita, permitindo a operação de dispositivos LoRa no território nacional (ANATEL, 2018).

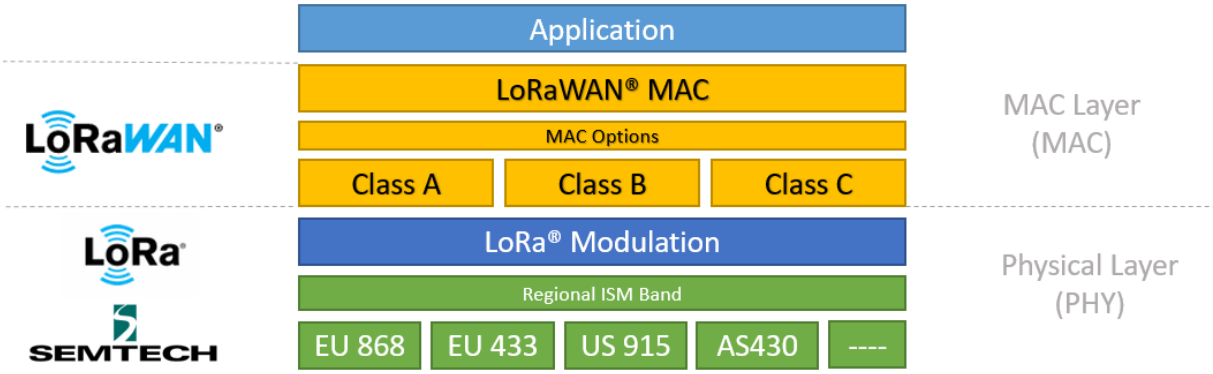
O nome LoRa é uma referência a *"Long Range"* (Longo Alcance), sendo caracterizada pela modulação de rádio frequências (RF) baseadas em *Chirp Spread Spectrum* (CSS), permitindo a cobertura de longas distâncias com baixos níveis de interferência (Semtech, 2015). Esta tecnologia opera em uma faixa de banda linear na qual a frequência da portadora varia no período de tempo definido. De acordo com (SEMTECH, 2022), LoRa utiliza seis fatores de espalhamento (*Spreading factors* - SF) para adaptar a taxa de dados e a compensação de alcance - quanto maior o fator de espalhamento usado, mais longe o

sinal poderá viajar e ainda ser recebido sem erros pelo receptor. Neste contexto, a taxa de dados pode variar entre 300 bps e 50 kbps em decorrência do SF e da largura de banda do canal de RF utilizados, permitindo o envio de mensagens com tamanho de até 243 *bytes*.

### 2.1.2 LoRaWAN

LoRaWAN é uma camada de rede lançada em 2015 pela LoRa Alliance©. Baseada na tecnologia LoRa, proporciona a criação de uma rede em topologia estrela entre um *gateway* e dispositivos finais. Utilizando puramente os princípios do protocolo de comunicação ALOHA, uma rede LoRaWAN utiliza o modo de comunicação *"single-hop"*, que transmite as mensagens dos dispositivos para um servidor central por meio de *gateways* (SEMTECH, 2022). A Figura 2 mostra a pilha LoRaWAN fundamentada a partir da tecnologia LoRa. Considerando que parte da mensagem transmitida é reservada para fins da pilha LoRaWAN, o MTU máximo das mensagens também fica atrelado ao fator de espalhamento utilizado. Desta forma, defini-se neste trabalho o limite de 51 *bytes* com o intuito de atender às possíveis variações de SF na rede LoRaWAN.

Figura 2 – Pilha LoraWAN.



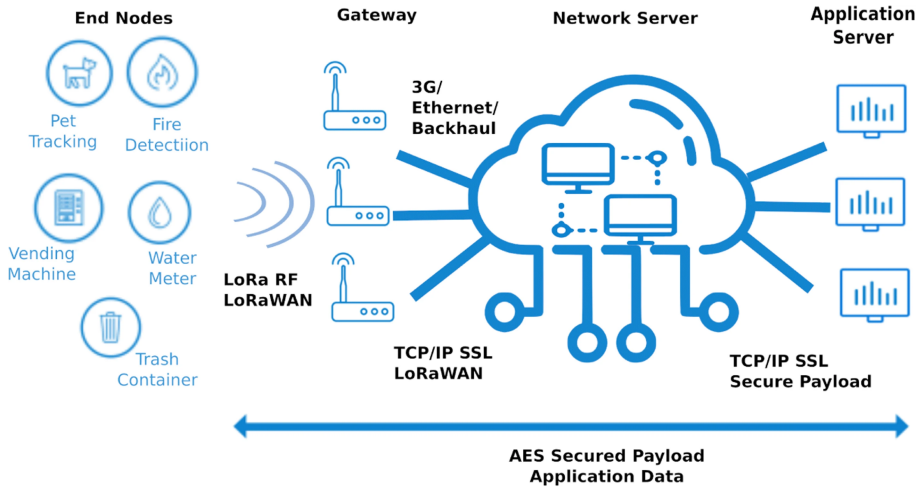

Fonte: (SEMTECH, 2022).

As mensagens trafegadas podem ser de *uplink* (enviada pelo dispositivo ao *gateway*) ou *downlink* (enviada pelo *gateway* ao dispositivo), de modo a estarem atreladas a uma porta definida pelo remetente com o intuito de possibilitar a segmentação do conteúdo de acordo com diferentes aplicações simultâneas. Apesar da estrutura semelhante nos dois sentidos de transmissão, apenas o *uplink* conta com a verificação CRC (*Cyclic redundancy check*) que garante a integridade do *payload* transmitido (CAMARGO; SPANHOL; SOUZA, 2021). Neste contexto, cada mensagem transmitida por um dispositivo é recebida por todos os *gateways* presentes na região, sendo estas posteriormente filtradas

por um servidor de rede com o intuito de remover mensagens repetidas. Este sistema redundante garante maiores taxas de sucesso no envio de mensagens de *uplink* (MEKKI et al., 2019).

Diferente da telefonia celular, uma rede LoRaWAN pode ser criada de forma particular sem a necessidade de pagamento de licenças para instalação e manutenção das operações, apesar de que já existem soluções comerciais no fornecimento de redes neutras, a exemplo de American Tower no Brasil. Neste cenário, tem-se que a arquitetura de uma rede LoRaWAN é composta pelos *gateways* (responsáveis pela comunicação com os dispositivos através do protocolo LoRa), o servidor de rede (responsável por gerenciar a rede LoRaWAN e encaminhar as mensagens) e os servidores de aplicação (que funciona como interlocutor entre a rede LoRaWAN e as aplicações que se comunicam com os dispositivos). A comunicação entre estes é realizada por meio do IP, de modo a possibilitar a integração entre os dispositivos com comunicação LoRa e o restante da Internet. A rede LoRaWAN possui a inteligência necessária para verificar a segurança, enviar confirmações para o dispositivo final e encaminhar a mensagem para o servidor de aplicação correspondente (MEKKI et al., 2019). Essa estrutura pode ser visualizada na Figura 3.

Figura 3 – Visão geral da arquitetura LoRaWAN.




Fonte: (CAMARGO; SPANHOL; SOUZA, 2021).

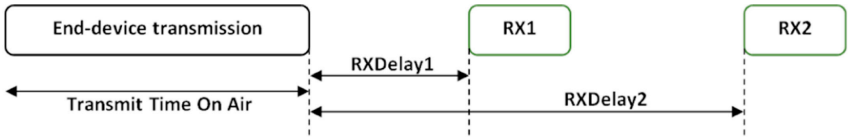
Neste trabalho, será utilizado o servidor de rede fornecido pela TTN (*The Things Network*), que é responsável pela comunicação com os dispositivos através dos *gateways* LoRaWAN. Este serviço engloba a gerência da rede LoRaWAN com todos os mecanismos citados acima, sem a necessidade de intervenção por parte do usuário. Desta forma, a comunicação com o servidor de rede é realizada através do protocolo MQTT (*Message Queuing Telemetry Transport*), conhecido como um protocolo M2M (Machine to Machine), que baseia-se na inscrição e na publicação em tópicos com o objetivo de respectivamente receber e enviar mensagens aos dispositivos da rede LoRaWAN.

A arquitetura LoRaWAN fornece três classes de comunicação com o intuito de aten-

der a diferentes aplicações de IoT, impactando principalmente em questões como latência e consumo de energia. As classes são denominadas A, B e C, apesar de que neste trabalho será considerada apenas a classe A, devido ao fato desta ser base e obrigatória em qualquer dispositivo. Esta classe possibilita a comunicação bidirecional *Half-duplex*, sendo o dispositivo responsável pela inicialização de cada sessão. A Figura 4 representa o processo de comunicaão desta classe, de modo que uma mensagem de *uplink* é enviada ("End-device transmission") e na sequência duas janelas de *downlink* ("RX1" e "RX2") são abertas após os respectivos intervalos de tempo "RXDelay1" e "RXDelay2" (ambos de 5 segundos). Caso hajam mensagens agendadas e destinadas ao dispositivo no servidor de rede, o *gateway* encaminha uma das mensagens (S. Farrell, 2018), podendo o dispositivo recebê-las em uma das duas janelas de *downlink* abertas no processo.

Na comunicação classe A, as janelas de transmissão são sempre inicializadas pelo dispositivo com base em suas próprias necessidades de comunicação, podendo ter transmissões programadas para a verificação de *downlinks* pendentes caso faça sentido para a aplicação. As comunicações deste tipo em qualquer outro momento terão que esperar sempre até a próxima sessão. Desta forma, a eficiência energética do dispositivo é elevada pois a comunicação ocorre apenas em momentos específicos, de modo que a janela de recepção fica fechada na maior parte do tempo. Em contrapartida, a latência entre o agendamento de um *downlink* e o recebimento deste aumenta consideravelmente.

Figura 4 – Comunicação bidirecional com LoRaWAN classe A.




Fonte: (SEMTECH, 2022).

As classes B e C diferenciam-se na abertura das janelas de *downlink*, de modo que a primeira realiza a abertura em períodos agendados pela rede, sem a dependência de uma janela de *uplink* para a sua ocorrência. Já a classe C mantém uma janela de recepção aberta durante todo o tempo, com exceção dos momentos em que transmissões de *uplink* são realizadas. Nestes dois casos, o dispositivo receberá as mensagens de *downlink* ao passo que o envio destas é agendado, juntamente com a abertura das janelas de recebimento em cada classe.

Além do uso da rede LoRaWAN em alto nível (a exemplo do envio e recebimento de mensagens), sua estruturação comporta também o ajuste dinâmico de configurações referentes à transmissão ao longo do tempo. Isso torna possível o ajuste de parâmetros como fator de espalhamento, largura de banda e potência de transmissão. As redes LoRaWAN contam com um mecanismo chamado ADR (*Adaptive Data Rate*) que faz justamente o ajuste desses três parâmetros conforme o servidor de rede percebe a necessidade, que é calculada a partir da análise das mensagens trocadas entre *gateway* e dispositivos. Isso

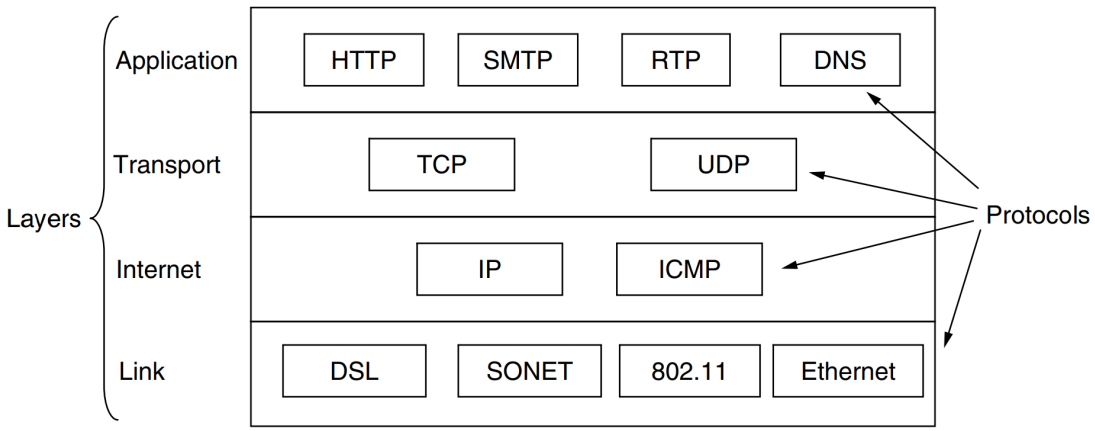
permite que em situações onde a rede está com bons níveis de sinais, menores tempos de transmissão sejam utilizados e consequentemente menores níveis de energia sejam despendidos pelos dispositivos. Situações opostas também são válidas, com o intuito de garantir a entrega das mensagens e otimizar a rede.

Na seção a seguir será apresentado o protocolo IP, utilizado neste trabalho para o transporte dos dados a partir do uso de uma rede LoRaWAN, introduzida nesta seção.

## 2.2 MODELO TCP/IP

TCP/IP (M. Rose and K. McCloghrie, 1998) é um conjunto de protocolos que fornece a estrutura para o desenvolvimento de uma gama completa de padrões de comunicação entre dispositivos. Desenvolvido a partir da simplificação do modelo OSI, o modelo TCP/IP conta com as camadas de aplicação, transporte, Internet e rede. Esta composição dá a base para o que hoje conhecemos como a Internet, de modo a atuar como um padrão mundial para a comunicação entre diferentes sistemas. Essa estrutura pode ser visualizada na Figura 5.

Figura 5 – Modelo TCP/IP com exemplos de protocolos.




Fonte: (TANENBAUM, 2011).

Considerando que a comunicação entre computadores se resume em aplicações, dispositivos e redes, tem-se que o modelo TCP/IP é fundamentado em camadas, sendo cada uma responsável por funcionalidades distintas. A mais inferior, denominada rede, representa os elementos físicos envolvidos no meio de transmissão de dados, como a natureza dos sinais, as taxas de dados, dentre outros fatores.

A camada de rede é responsável pelo roteamento da rede, comumente relacionado ao IP. Presente tanto nos dispositivos finais quanto nos roteadores de rede, esta camada fornece o endereçamento e as rotas entre os pontos de comunicação, que podem estar presentes em uma rede local ou então em redes externas que estão interconectadas.

Junto a isso, a camada de transporte originalmente visa garantir que a comunicação entre processos está ocorrendo de forma confiável. Isso significa que o destinatário recebe as mensagens com garantia de integridade e ordem correta em todos os dados.

Por fim, a camada de aplicação fornece o suporte necessário a diferentes aplicações que venham a ser executadas na rede, como troca de mensagens, transferência de arquivos, telefonia digital, dentre outros. Respeitando as peculiaridades de diferentes possibilidades de emprego da rede, o modelo TCP/IP ganhou notoriedade devido à sua característica multiplataforma e transparente, podendo ser utilizada em diferentes sistemas operacionais devido ao vasto suporte disponível.

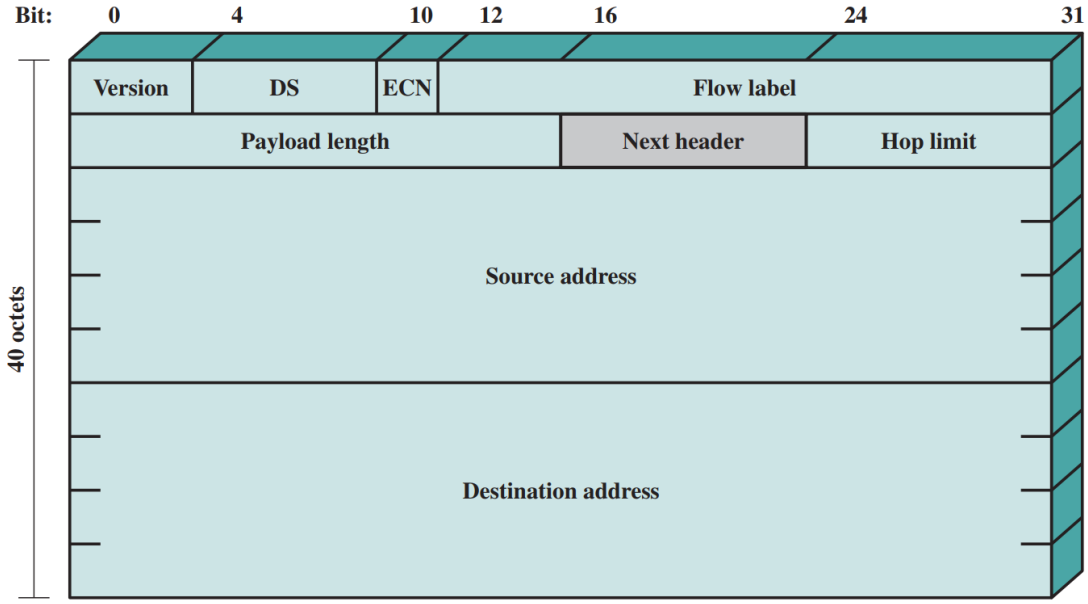
### 2.2.1 Protocolo IP

Pertencente à camada de rede no modelo TCP/IP, o Protocolo de Internet (IP) é definido como o sistema de endereços da Internet e tem como função transportar pacotes de informações vindos de um dispositivo de origem a outro de destino. Desta forma, um caminho entre os pontos inicial e final é composto pela comunicação entre diferentes IPs, que representam pontos distintos na rede e roteados de acordo com as definições tomadas pelos operadores de rede e com a disponibilidade apresentada pela rede.

Por décadas, o protocolo IPv4 foi o princípio utilizado no modelo TCP/IP. Entretanto, o crescimento exponencial do número de dispositivos conectados à Internet tornou o IPv4 cada vez mais insuficiente para atender às novas demandas, principalmente no que diz respeito ao número máximo de dispositivos que poderiam ser endereçados nessa tecnologia. Desta forma, uma nova geração IP foi desenvolvida em 1995 com o intuito de resolver este problema, a qual foi posteriormente nomeada como IPv6 (S. Deering and R. Hinden, 1998). A migração entre as versões IPv4 e IPv6 já era prevista como necessária no início do século, porém atingiu um patamar urgente em 2011 quando as reservas iniciais de IPv4 foram decretadas como esgotadas pela IANA (Internet Assigned Numbers Authority) - órgão global que gerencia a distribuição de blocos de endereço ao longo do mundo (BEEHARRY; NOWBUTSING, 2016).

De acordo com (STALLINGS, 2014), esta nova especificação nomeada IPv6 fornece várias melhorias funcionais em relação à versão anterior, de modo a suportar maiores velocidades nas redes e diferentes fluxos de dados cada vez mais predominantes, como *streaming* de áudio e vídeo. Além disso, os endereços limitados anteriormente a 32 bits agora suportam 128 bits, acomodando uma quantidade maior de dispositivos simultaneamente conectados. A Figura 6 ilustra a estrutura de um cabeçalho IPv6 com todos as suas especificações.

Figura 6 – Cabeçalho IPv6.

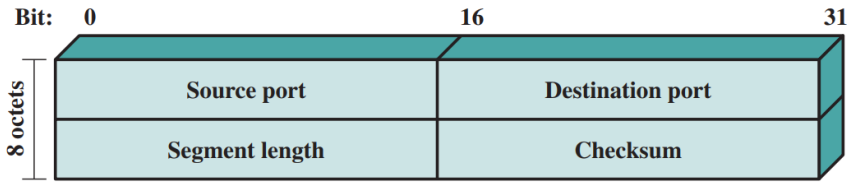



Fonte: (STALLINGS, 2014).

### 2.2.2 Protocolo UDP

O UDP (J. Postel, 1980) é a opção mais simples dentre as utilizadas na camada de transporte, de modo a não estabelecer e não manter uma conexão entre cliente e servidor para a troca de pacotes (TANENBAUM, 2011). Composto por um cabeçalho de 8 bytes seguido do *payload*, sua função é identificar as duas portas referentes às aplicações de origem e de destino, de modo a garantir que o pacote será encaminhado à aplicação correta quando chegar no dispositivo destino. A Figura 7 ilustra o cabeçalho deste protocolo.

Figura 7 – Cabeçalho UDP.




Fonte: (STALLINGS, 2014).

O protocolo UDP - por ser menos complexo e consequentemente utilizar menos tráfego na rede - não garante a confiabilidade da conexão devido à ausência de mecanismos de controle de fluxo e de erros nos dados transmitidos. Assim, todos esses mecanismos precisam ser implementados em uma camada superior.

Na seção a seguir será apresentado o protocolo de comunicação SCHC, responsável pela compressão e fragmentação dos pacotes IPv6/UDP com o intuito de possibilitar o tráfego destes em uma rede LPWAN.

## 2.3 SCHC

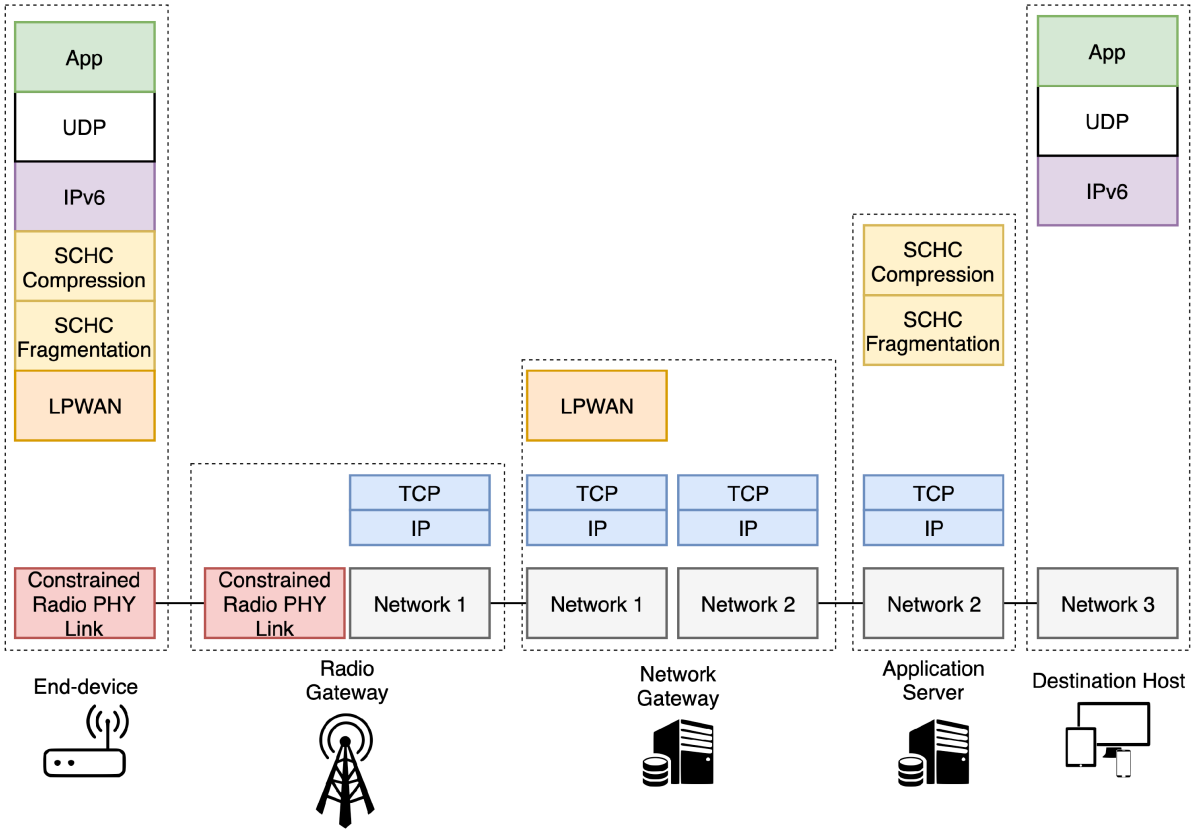
SCHC é um protocolo de comunicação especificado pela RFC 8724, de modo a propor funcionalidades para compressão e fragmentação de dados tendo em mente o uso em tecnologias da família LPWAN, como Sigfox, LoRaWAN, e NB-IoT (MINABURO et al., 2020). O SCHC é considerado uma subcamada adaptativa localizada acima da camada de rede, promovendo o transporte de pacotes IPv6 em redes LPWAN (MUÑOZ et al., 2022). Apesar da padronização recente do protocolo, já existem soluções comerciais que propõem a implementação do SCHC em forma de serviço[1], de modo que a comunicação com os dispositivos finais ocorre já por meio da Internet sem a necessidade de gerenciar implementações do protocolo SCHC.

O funcionamento do SCHC depende da sua implementação em uma rede LPWAN, juntamente com os demais componentes que formam o sistema de comunicação. O diagrama apresentado na Figura 8 representa uma possível implementação do sistema juntamente com toda a pilha de rede, composto por dispositivos finais (*"End-device"*), rede LPWAN (*"Radio Gateway"* e *"Network Gateway"*), servidor de aplicação (*"Application Server"*) e *hosts* de destino (*"Destination Host"*). Observa-se que a pilha IPv6/UDP juntamente com a camada de aplicação (representados na figura por "IPv6", "UDP"e "App"), estão presentes apenas na origem e no destino do sistema de comunicação, de modo que o SCHC fica responsável pela adaptação ao contexto LPWAN fornecendo ferramentas de compressão e fragmentação dos pacotes originalmente IPv6/UDP. O transporte destes fragmentos fica sob responsabilidade da rede LPWAN juntamente com a Internet, que viabilizam a comunicação entre os dispositivos e o servidor de aplicação.

A partir de um pacote IPv6 gerado, por exemplo, por uma biblioteca *socket*, a subcamada de compressão faz o seu processamento gerando um novo pacote no formato SCHC relacionando as regras implementadas e as informações recebidas. Caso o tamanho final dos dados gerados pela aplicação seja igual ou menor ao MTU utilizado na aplicação - que deve sempre respeitar os limites impostos pela tecnologia LPWAN em uso - é realizado o envio em direção ao destinatário. Caso o tamanho seja maior, a subcamada de fragmentação divide e organiza os pacotes em fragmentos menores, de modo a serem posteriormente enviados em mais de uma transmissão. A Figura 9 ilustra esse processo tanto no lado do transmissor (que realiza a compressão e a fragmentação), quanto no lado do receptor (que realiza os processos inversos).

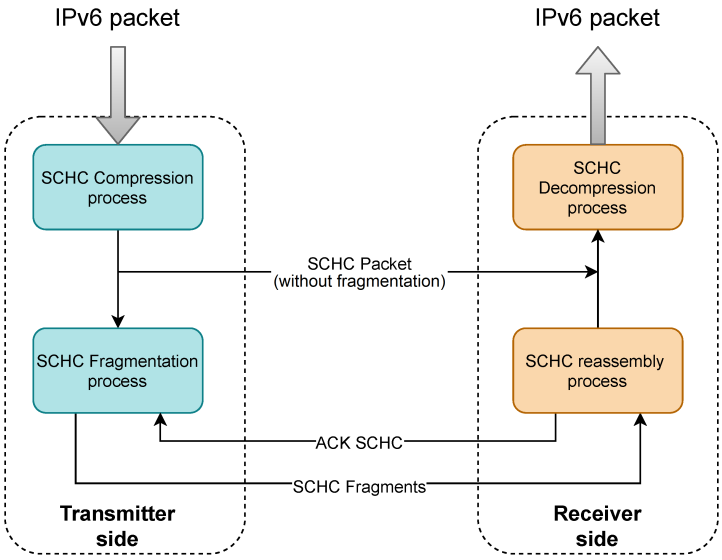
[1] Disponível em: https://www.ackl.io/developer/ipv6-for-lorawan-developers

Figura 8 – Visão geral de uma possível aplicação do SCHC em rede LPWAN




Fonte: (SANCHEZ-GOMEZ et al., 2020).

Figura 9 – Processos de compressão e fragmentação do SCHC.




Fonte: (MUÑOZ et al., 2022).

### 2.3.1 Compressão

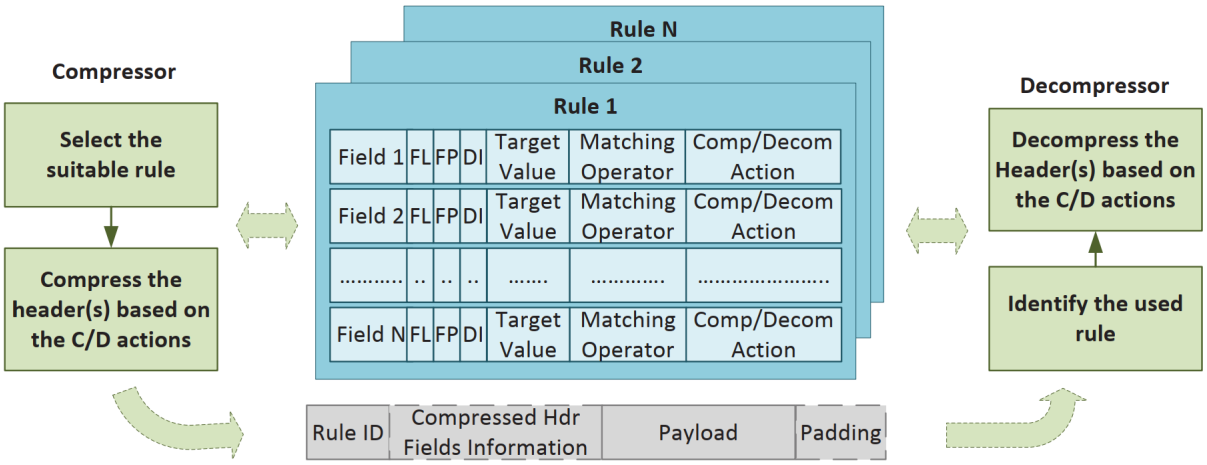
Segundo Sanchez-Gomez e colaboradores, a compressão de cabeçalhos é dividida em três tipos: sem estado, com estado e híbrida. O primeiro não exige que o transmissor e o receptor mantenham sequências de operações sobre como proceder com os pacotes de dados. A compressão ocorre com base em informações constantes ou utilizadas com frequência, sendo sua eficiência menor diante de tráfegos que não foram previamente identificados ou projetados. Por outro lado, o segundo caso é caracterizado pelo tabelamento de contextos locais que devem ser frequentemente sincronizados entre os pontos de origem e destino, promovendo maiores índices de compressão. Essas atualizações provocam sobrecargas na rede de comunicação, além de requererem mais memória, tempo de processador e energia dos dispositivos (SANCHEZ-GOMEZ et al., 2020).

No caso do SCHC, uma forma híbrida de compressão é adotada baseando-se em regras de compressão. Assim, o transmissor e o receptor armazenam contextos comuns e frequentes definidos com antecedência, sendo estes relacionados aos cabeçalhos dos pacotes; estas informações são estáticas e visam evitar a necessidade de procedimentos de sincronização. Entretanto, o SCHC também possibilita o uso de diferentes configurações que visam definir operações a serem realizadas com o pacote, sendo elas sempre atreladas a um código de identificação de regra correspondente àquela situação.

O processo de compressão do SCHC visa reduzir o tráfego na rede LPWAN por meio do mapeamento de informações estáticas presentes nos cabeçalhos dos pacotes IPv6/UDP. Este procedimento é realizado através da configuração de regras de compressão, onde todos os componentes presentes nos cabeçalhos IPv6 e UDP são listados e passíveis pelo processo. A partir do uso de operadores de compressão, cada parâmetro do cabeçalho pode receber um valor totalmente fixo, parcial, ou então um valor ignorado pelas regras. No primeiro caso, a informação não é inserida dentro do pacote resultante, de modo que este dado ficará atrelado a uma regra. Posteriormente, o código de identificação da regra (inserido no pacote) será utilizado para a recuperação da informação no processo de descompressão. No segundo caso (parcial), parte da informação fica atrelada à regra, sendo a outra parte inserida no pacote resultante. No último caso, quando o valor de algum parâmetro é ignorado pelo sistema de regras, significa que esta informação será obtida do pacote IPv6/UDP e inserida integralmente dentro do pacote SCHC, sem passar pelo processo de compressão. A Figura 10 ilustra o processo de compressão como um todo, de modo que a regra de compressão é escolhida com base nas informações presentes no pacote IPv6/UDP e nos parâmetros de compressão definidos nas regras configuradas.

Para este processo funcionar, é necessário que as mesmas regras de compressão do SCHC estejam configuradas tanto no ponto em que o pacote será comprimido quanto no local onde este pacote for descomprimido, de modo que o mesmo pacote IPv6/UDP inserido na compressão será entregue pelo processo de descompressão. A seguir serão apresentados os operadores de compressão presentes nas regras citadas.

Figura 10 – Processo de seleção de regra para compressão SCHC.




Fonte: (ABDELFADEEL; CIONCA; PESCH, 2018).

#### 2.3.1.1 Operadores de compressão

Especificados na RFC 8724, o processo de compressão conta com os operadores de equivalência e os operações para compressão/descompressão (CDA), sendo suas funções descritas nas regras e aplicadas no processo de compressão dos pacotes SCHC. Primeiramente, os operadores de equivalência são compostos pelos seguintes itens:

- Operador *equal*: visa substituir informações de forma exata, sem variações;
- Operador *ignore*: não compara valores de regra com os valores do pacote, os quais normalmente são transmitidos de forma inteira neste caso;
- Operador *MSB(x)*: define que os x bits mais significativos da informação devem corresponder ao valor presente na regra;
- Operador *match-mapping*: utiliza um índice de valores tabelados, de modo que apenas este número é inserido no pacote SCHC e posteriormente substituído.

Já os operações para compressão/descompressão (CDA) são descritos por:

- Not-Sent CDA: o valor do campo é totalmente especificado na regra, logo não precisa ser montado no pacote final. Esta operação é normalmente utilizada junto com o operador *equal*;
- Value-Sent CDA: sendo o valor do campo não definido previamente, essa operação é utilizada quando a informação é integralmente inserida no pacote. O operador *ignore* é geralmente utilizado em conjunto;

- Mapping-Sent CDA: utilizado juntamente com o operador *match-mapping*, essa operação realiza a substituição do valor com base no índice presente no pacote, sendo este previamente definido nas regras juntamente com as demais possibilidades;
- LSB CDA: essa operação é utilizada juntamente com o operador *MSB(x)* com o intuito de evitar que a parte mais significativa das informações sejam consideradas. Assim, somente o restante (a parte menos significativa) é montada no pacote;
- Compute-*: alguns campos podem ser omitidos no compressor e recalculados pelo descompressor, a exemplo do *IPv6 length* que pode ser calculado pelo próprio algoritmo referente ao protocolo em questão.

#### 2.3.1.2 Montagem do pacote comprimido

O SCHC foi projetado com a ideia de que os dispositivos trabalham com aplicações integradas, sendo o tráfego gerado por estes altamente previsível e configurável com antecedência nas regras de compressão. Desta forma, a proposta de uma subcamada de adaptação trabalha de forma completamente agnóstica aos pacotes que por ali trafegam.

Como apresentado na Seção 2.3.1, o tratamento dos pacotes é realizado com base em códigos de identificação de regras referentes aos valores de seus campos, que são mapeados com antecedência. Caso a correspondência entre os valores não seja possível, estes são enviados sem compactação. Desta forma, o pacote SCHC é composto pelo cabeçalho compactado representado pelo código de identificação da sua regra e os valores remanescentes, juntamente com o *payload* do pacote referente a uma camada superior. Posteriormente, a fragmentação é realizada com o intuito de adequar o tamanho do pacote com as limitações da tecnologia LPWAN em uso.

### 2.3.2 Fragmentação

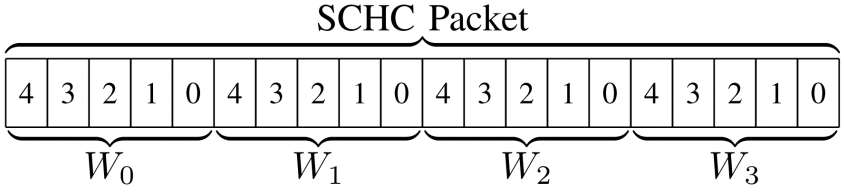
O processo de fragmentação corresponde à adequação do pacote SCHC comprimido na tecnologia LPWAN em uso, com o intuito de torná-lo apto a ser transmitido em detrimento do MTU de cada mensagem. Esse processo não é obrigatório, sendo ignorado caso o tamanho do pacote seja igual ou inferior ao limite proposto. Originalmente, o pacote IPv6 possui um MTU mínimo de 1280 *bytes* (S. Deering and R. Hinden, 1998), sendo este tamanho normalmente reduzido no processo de compressão SCHC. Desta forma, apesar da compressão dos cabeçalhos IPv6, é provável que a fragmentação seja necessária em virtude da carga útil presente no pacote (cuja compactação não é abrangida pelo SCHC).

De acordo com a RFC 8724, o protocolo SCHC foi desenvolvido tendo em mente as redes LPWAN com o intuito de melhorar a eficiência em cada uma das tecnologias devido as suas diferentes especificações, sendo que cada uma deve adotar um perfil distinto de operação com parâmetros e modos de operação específicos. Nisso, tem-se que a RFC 9011 define as especificações para uso em uma rede LoRaWAN, sendo neste contexto considerados os modos ACK *On Error* (que exige um ACK após a transmissão de todos os fragmentos ou na detecção de falhas) e ACK *Always* (que exige um ACK após a transmissão de cada fragmento). Estes dois modos são utilizados para *uplinks* e *downlinks* respectivamente, formando assim uma comunicação bidirecional entre os dispositivos e o servidor de aplicação (O. Gimenez and I. Petrov, 2021). Ressalta-se que ambos podem assumir as posições de receptor e transmissor, sendo elas definidas com base no sentido do tráfego em andamento. Explicações mais detalhadas sobre estes dois modos serão apresentadas na sequência.

*Uplinks* e *downlinks* são termos que podem ser referentes tanto à rede LoRaWAN como ao protocolo SCHC, conforme citado acima e na Seção 2.1. Com isso, define-se que os termos daqui em diante serão preferencialmente referenciados ao SCHC, de modo que cada *uplink* e *downlink* do protocolo é composto por vários *uplinks* e *downlinks* na rede LoRaWAN. Quando necessário, a diferenciação dos dois casos será enfatizada para promover maior clareza na explicação.

A fragmentação baseia-se na suposição de que o tamanho dos fragmentos pode variar ao longo das transmissões e que erros na camada de rede possam vir a comprometer a entrega de alguns fragmentos. Nestes casos, são previstos procedimentos de retransmissão que visam corrigir as partes faltantes. O comportamento diante dessas possibilidades está atrelado ao modo de operação, que nos casos ACK *On Error* e ACK *Always* comportam-se de maneiras diferentes, sendo suas diferenças apresentadas a seguir. De qualquer forma, a base do pacote SCHC é a mesma para ambos os casos: dependendo do tamanho do pacote, ele é dividido em uma ou mais janelas, que por sua vez possui um ou mais fragmentos dependendo do modo de operação. A Figura 11 ilustra um exemplo de pacote no modo ACK *On Error*, estando ele agrupado em 4 janelas e 5 fragmentos por janela.

Figura 11 – Pacote SCHC dividido em janelas e fragmentos no modo ACK *On Error*.

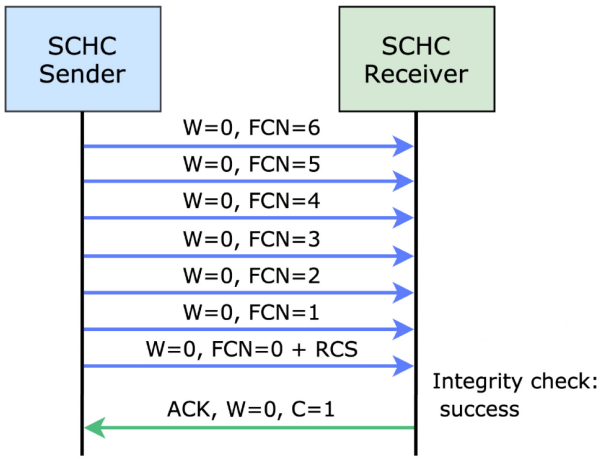



Fonte: (AGUILAR et al., 2020).

#### 2.3.2.1 Modo ACK On Error

De acordo com a RFC 9011, todo o tráfego de *uplink* deve ser conduzido utilizando o modo *On Error*. Neste contexto, o dispositivo passa a ser o transmissor (*Sender*) e o servidor de aplicação passa a ser o receptor (*Receiver*). Dessa forma, o transmissor envia um conjunto de fragmentos referentes a uma janela, sem a necessidade de confirmações ACK durante esse processo. Ao final, é esperado que o receptor envie uma mensagem SCHC ACK confirmando o recebimento e a integridade da mensagem. Este é um caso ideal onde não ocorrem falhas na transmissão, sendo ele ilustrado na Figura 12.

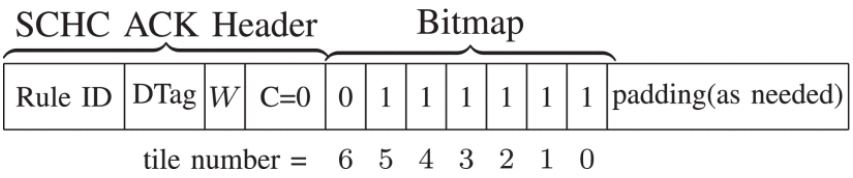
Figura 12 – Exemplo de interação do modo ACK *On Error*.




Fonte: Adaptado de (AGUILAR et al., 2020).

Considerando uma situação onde um fragmento não chegou ao receptor, este deve enviar uma mensagem SCHC ACK anexando o bitmap atual (representado por uma sequência comporta por números zero e um, que reflete quais fragmentos foram ou não recebidos). Ao receber essa mensagem, o transmissor verifica a necessidade e envia novamente as partes faltantes, aguardando então uma nova confirmação. Para ilustrar essa situação, a Figura 13 apresenta uma mensagem ACK com *bitmap* incompleto (enviada pelo receptor), indicando que o fragmento de número 6 não foi recebido. Dessa forma, o modo *On Error* permite a entrega de fragmentos fora da sua ordem original, sendo sua organização também prevista no protocolo.

Figura 13 – Exemplo SCHC ACK com *bitmap* incompleto.

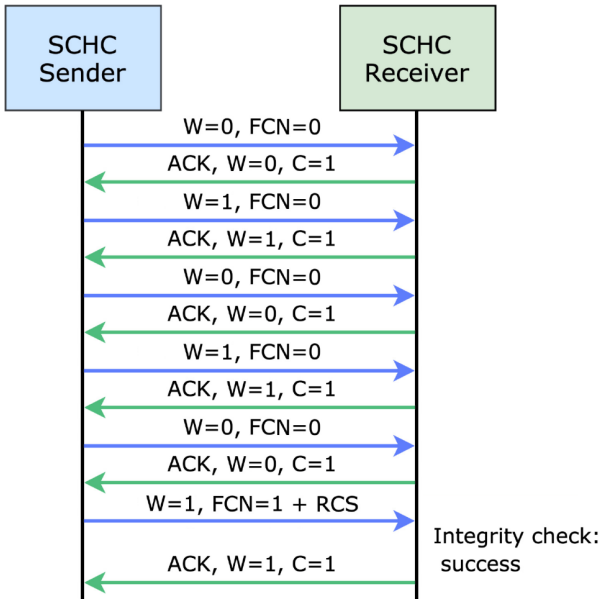



Fonte: (AGUILAR et al., 2020).

#### 2.3.2.2 Modo ACK Always

Diante de um tráfego de *downlink*, o servidor de aplicação passa a ser o transmissor e o dispositivo passa a ser o receptor, ambos operando em modo ACK *Always*. Considerando as normativas da RFC 9011 quanto ao uso do SCHC em uma rede LoRaWAN, tem-se que tanto janelas quanto fragmentos terão o tamanho de 1 *bit*. Com isso, faz-se necessária a confirmação diante do recebimento de qualquer fragmento SCHC, de modo que o transmissor avançará na ordem dos fragmentos apenas após o recebimento de um ACK referente ao fragmento anterior. A interação entre os dois extremos está representada na Figura 14. Assim, a entrega de fragmentos fora de ordem não é permitida justamente pela necessidade de confirmações frequentes.

Figura 14 – Exemplo de interação do modo ACK *Always*.



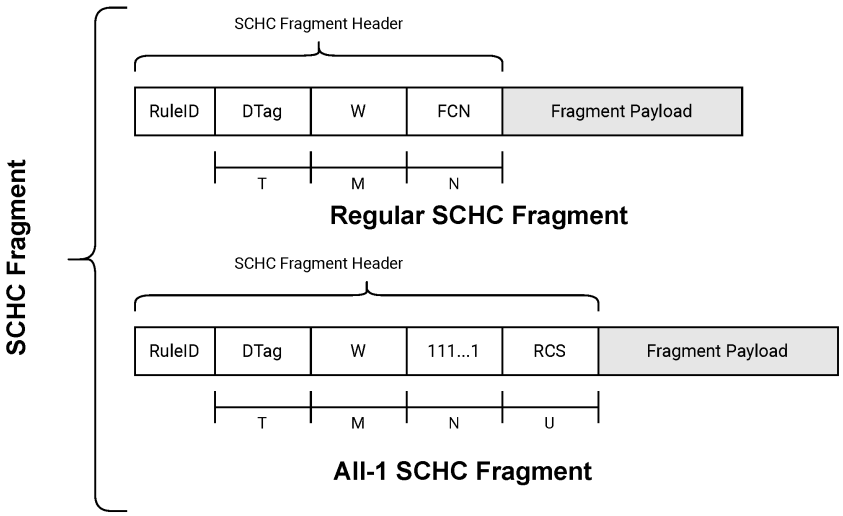

Fonte: Adaptado de (AGUILAR et al., 2020).

#### 2.3.2.3 Fragmentos SCHC

Os fragmentos SCHC representam o transporte de partes do pacote SCHC, compostos pelo código de identificação da regra de compressão, pelo cabeçalho de fragmentação e pela carga útil referente às fatias do pacote IPv6, as quais podem conter resíduos do processo de compressão e/ou o *payload* da aplicação de camada superior. Cada fragmento pode ser do tipo Regular ou All-1, sendo a diferença destes representada por seu tamanho e conteúdo: o primeiro possui tamanho fixo e carrega os fragmentos do pacote com exceção do último. Já o fragmento All-1 possui tamanho variável e sinaliza a última parte do pacote, carregando consigo o último fragmento e também o valor de RCS refe-

rente ao pacote completo; este valor é baseado em CRC32 utilizado para validação no processo de desfragmentação.

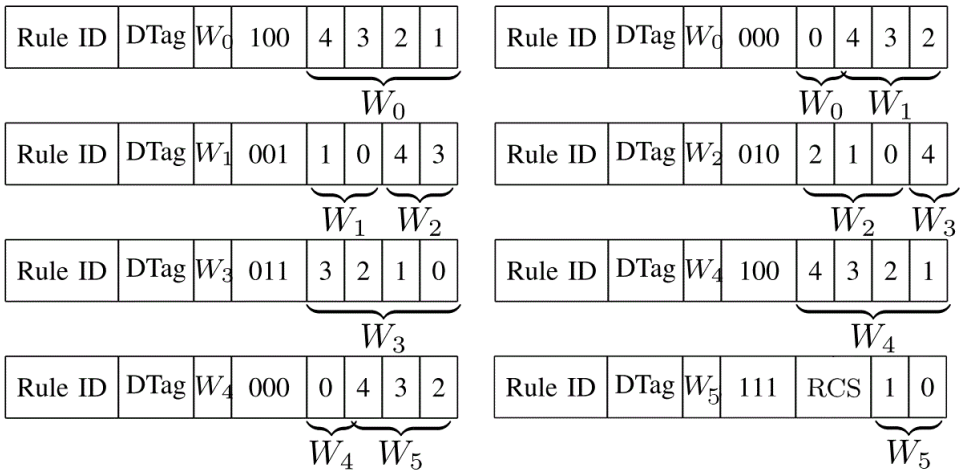
Figura 15 – Fragmentos SCHC.




Fonte: (MUÑOZ et al., 2022).

Com base no exemplo apresentado na Figura 11, a Figura 16 ilustra a organização do pacote em fragmentos SCHC, sendo todos Regulares com exceção do último que é All-1.

Figura 16 – Exemplo de pacote SCHC organizado em fragmentos.



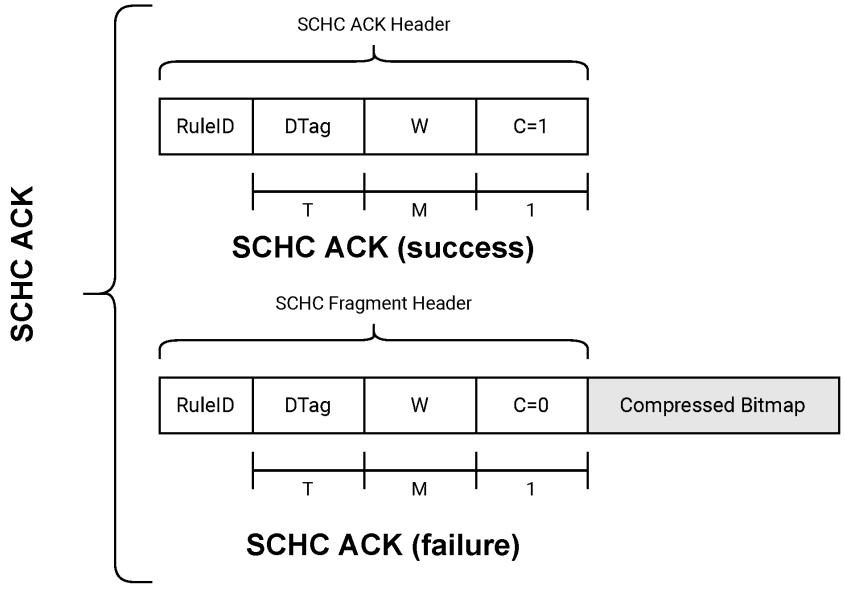

Fonte: Adaptado de (AGUILAR et al., 2020).

#### 2.3.2.4 SCHC ACK

A mensagem SCHC ACK é utilizada pelo receptor para informar ao transmissor sobre o recebimento de um ou mais fragmentos. Em caso de sucesso, o campo C é

definido com o valor 1. Do contrário, recebe o valor 0 juntamente com o *bitmap* inserido no final da mensagem, o qual informa quais fragmentos não foram recebidos corretamente. Essa estrutura pode ser visualizada na Figura 17.

Figura 17 – Estrutura de uma mensagem SCHC ACK.

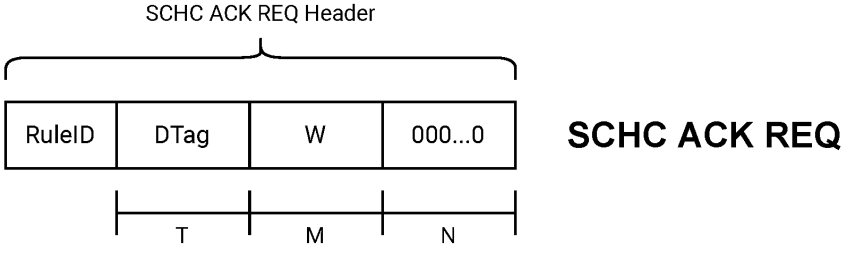



Fonte: (MUÑOZ et al., 2022).

#### 2.3.2.5 SCHC ACK REQ

A mensagem SCHC ACK REQ é enviada pelo transmissor com o intuito de solicitar a confirmação de recebimento por parte do receptor. Seu uso pode ocorrer após o envio de uma sequência de fragmentos ou então após um período definido onde não houve o recebimento de nenhuma resposta quando aguardada. A estrutura dessa mensagem pode ser visualizada na Figura 18.

Figura 18 – Estrutura de uma mensagem SCHC ACK REQ.

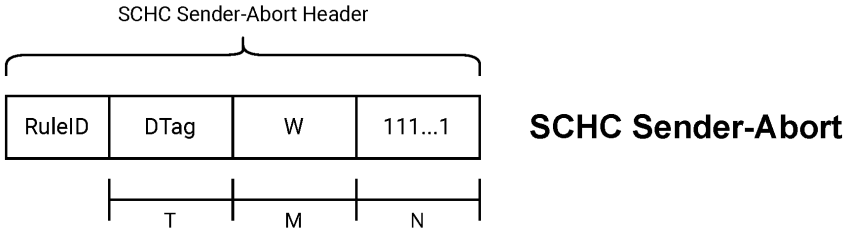



Fonte: (MUÑOZ et al., 2022).

#### 2.3.2.6 SCHC Sender-Abort

A mensagem SCHC Sender-Abort é utilizada pelo transmissor quando há a necessidade de descartar um pacote com transmissão em vigência. Ela normalmente é utilizada quando o transmissor percebe que não há como identificar quais fragmentos foram perdidos naquela transmissão, sendo mais vantajoso abortá-la e iniciar novamente. A estrutura dessa mensagem pode ser visualizada na Figura 19.

Figura 19 – Estrutura de uma mensagem SCHC Sender-Abort.

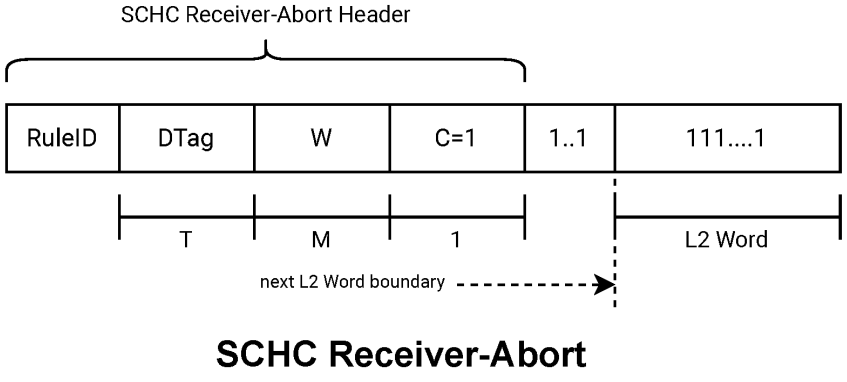



Fonte: (MUÑOZ et al., 2022).

#### 2.3.2.7 SCHC Receiver-Abort

Da mesma forma como um SCHC Sender-Abort, a mensagem SCHC Receiver-Abort parte da necessidade de descartar uma transmissão em andamento, com a diferença de que o pedido parte do receptor. Além disso, esse tipo de mensagem pode ser utilizado para confirmar o recebimento de um Sender-Abort previamente enviado pelo transmissor. A estrutura do Receiver-Abort pode ser visualizada na Figura 20.

Figura 20 – Estrutura de uma mensagem SCHC Receiver-Abort.




Fonte: (MUÑOZ et al., 2022).

#### 2.3.2.8 SCHC Header

Ao longo do processo de transmissão de um pacote SCHC, diferentes tipos de mensagem são utilizados em suas respectivas funcionalidades. Junto a estes, é enviado um cabeçalho SCHC com informações do próprio protocolo, as quais podem ser um ou mais dos itens apresentados a seguir.

- *Rule* ID: presente em todas as transmissões, sinaliza que é uma mensagem SCHC a partir do código do modo em uso (ACK *On Error* ou ACK *Always*);
- *Datagram Tag*(DTag): este campo diferencia dois ou mais pacotes SCHC que estejam sendo enviados simultaneamente no mesmo modo de operação. Entretanto, sua utilização não é realizada quando aplicado a LoRaWAN;
- W: com tamanho de M *bits*, referencia a janela usada naquele momento;
- *Fragment Compressed Number* (FCN): em cada janela, este campo identifica o número do fragmento em questão. O tamanho máximo é definido pelo parâmetro N;
- *Reassembly Check Sequence* (RCS): presente apenas em fragmentos do tipo All-1, informa o código relacionado ao cálculo do polinômio CRC32. Ele serve para validação do pacote durante o processo de desfragmentação, tendo seu tamanho limitado pelo parâmetro U;
- C *(Integrity Check)*: utilizado em mensagens do tipo SCHC ACK, informa se o pacote desfragmentado passou ou não no teste de integridade. O valor 1 indica sucesso e 0 indica falha;
- *Compressed Bitmap*: enviado pelo receptor, é responsável por informar ao transmissor quais fragmentos foram recebidos, de modo que em caso de falhas, torna-se possível identificar quais partes devem ser retransmitidas.

Na seção a seguir será apresentado o serviço DLMS/COSEM, responsável pela padronização na comunicação com medidores inteligentes de energia elétrica. Os dados gerados serão inseridos em pacotes IPv6/UDP e manipulados pelo protocolo SCHC, possibilitando com isso a comunicação cliente-servidor com o uso da rede LoRaWAN.

## 2.4 DLMS/COSEM

O DLMS/COSEM (DLMS, 2022) é um serviço da camada de aplicação que especifica um modelo de dados orientado a objetos voltada à comunicação com dispositivos

inteligentes, como medidores de energia, gás, água, dentre outros. Considerado um padrão reconhecido mundialmente, permite acesso remoto a dados e procedimentos garantindo interoperabilidade, eficiência e segurança entre diferentes modelos de dispositivos e fabricantes. Todas as permissões de acesso são previamente definidas pelos fabricantes seguindo a padronização OBIS (Object Identification System), que reserva códigos relacionados a objetos COSEM padronizados, além de códigos com finalidades específicas de acordo com o fabricante, país e características dos dados.

Executando com base no paradigma cliente-servidor (onde o dispositivo é o servidor e o usuário externo é o cliente), existem diferentes formas de comunicação - as quais dependem de definições de projeto do fabricante. No Brasil, a ABNT NBR 16968(ABNT, 2022) regulamenta o DLMS/COSEM para medidores inteligentes de energia elétrica de modo a cobrir suas funcionalidades e a comunicação com acesso local e remoto. A norma brasileira não altera o que foi originalmente especificado pela IEC 62056, porém recomenda que a comunicação em rede LoRaWAN seja realizada com o uso do SCHC. O perfil completo de comunicação está representado na Figura 21.

Figura 21 – Perfil de comunicação RF-LoRaWAN segundo a ABNT NBR 16968.

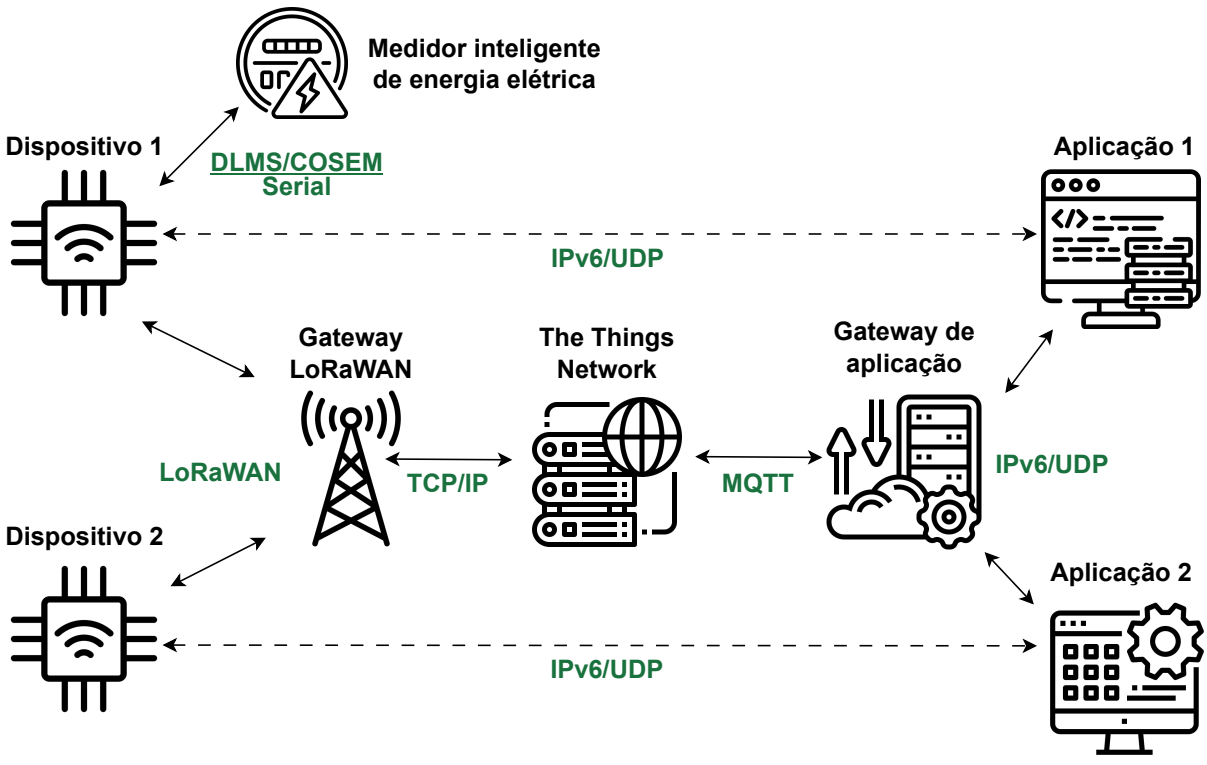

Fonte: Adaptado de (ABNT, 2022).

O próximo capítulo tratará do desenvolvimento deste trabalho, aplicando os conceitos apresentados até aqui em um sistema composto pela implementação do protocolo SCHC juntamente com os demais componentes necessários para estabelecer uma rede de comunicação utilizando o SCHC e uma rede LPWAN.

# 3 DESENVOLVIMENTO

Neste capítulo serão apresentados todos os processos de estruturação e implementação deste trabalho, tendo como objetivos desenvolver um meio de comunicação IPv6/UDP ponto a ponto entre um medidor inteligente de energia elétrica e um *notebook* por meio de uma rede LoRaWAN, possibilitando também análises de todo o sistema. A Figura 22 ilustra o sistema proposto neste trabalho, que parte de dois dispositivos os quais possuem o protocolo SCHC implementado, possibilitando o envio e o recebimento de pacotes IPv6/UDP. O primeiro dispositivo está conectado a um medidor inteligente de energia elétrica, de modo que a comunicação com este ocorre através do protocolo DLMS/COSEM por meio de uma comunicação serial. O segundo dispositivo não possui um medidor conectado, sendo sua função gerar tráfego para análises de rede. Estes dois dispositivos estão conectados a uma rede LoRaWAN por meio do *Gateway* LoRaWAN, que por sua vez está conectado via Internet ao servidor de rede fornecido pela TTN (*The Things Network*).

Figura 22 – Ilustração de todos os componentes situados no contexto de aplicação do trabalho.




Fonte: Autor.

Na sequência, um servidor de aplicação (chamado na figura de *Gateway* de aplicação) é conectado na TTN por meio do protocolo MQTT, que fica responsável pelo envio e recebimento de mensagens em direção aos dispositivos através da rede LoRaWAN. Este *Gateway* de aplicação de agora em diante será chamado de *gateway* (a rede LoRaWAN

também possui um componente com esse termo e seu uso será diferenciado quando necessário), estando ele localizado em um *notebook* (podendo ser um servidor). A função do *gateway* é comunicar-se com a rede LoRaWAN, além de gerenciar os pacotes SCHC gerados a partir da comunicação com uma rede IP composta pelas aplicações. Estas aplicações são responsáveis por enviar solicitações aos dispositivos, de modo que todo o sistema proposto atua com o objetivo de possibilitar essa comunicação. Maiores detalhes dos componentes envolvidos neste sistema serão abordados ao longo deste capítulo.

Apesar de desempenhar um papel fundamental neste trabalho e ser a base de comunicação entre os dois pontos das conexões, a rede LoRaWAN e suas componentes não serão o foco das discussões nas seções a seguir, considerando assim um nível de abstração maior quanto ao seu funcionamento. Conforme a pilha da rede LoRaWAN apresentada na Figura 2, será considerada a camada de aplicação com o intuito de transportar os fragmentos referentes ao SCHC que posteriormente serão traduzidos em pacotes IPv6/UDP.

As seções a seguir foram divididas no detalhamento da implementação dos algoritmos referentes ao SCHC, no uso destes em um servidor de aplicação conectado às redes LoRaWAN e IP, e na apresentação dos dispositivos conectados à rede LoRaWAN que também usam o SCHC junto à comunicação com o medidor inteligente de energia elétrica.

## 3.1 SCHC

A implementação de um ambiente com o SCHC é o ponto principal deste trabalho. A partir do tratamento dos pacotes IPv6/UDP pelo SCHC, serão conectados os demais componentes responsáveis pelo encaminhamento das informações a serem trafegadas. Nesta seção, serão abordadas as implementações realizadas com base nas padronizações estabelecidas pelas RFC 8724 e 9011, juntamente com o uso da biblioteca PySCHC[1].

### 3.1.1 Biblioteca PySCHC

PySCHC (NIC Chile Research Labs, 2021) é uma biblioteca de código aberto na linguagem de programação Python, que implementa de forma experimental as ferramentas de compressão e fragmentação do SCHC com base nas RFC 8724 e 9011. Essa biblioteca foi utilizada para as avaliações propostas em (MUÑOZ et al., 2022). Sendo ela experimental e com ausência de atualizações desde 2021, foram necessárias várias melhorias para torná-la funcional e escalável, as quais serão abordadas nos itens a seguir, contendo a forma como foram implementadas e também as decisões tomadas a partir dos testes realizados ao longo do desenvolvimento.

[1] Disponível em: https://github.com/niclabs/PySCHC

### 3.1.2 Compressão

A compressão dos pacotes IPv6/UDP é baseada em regras de compressão que definem a estruturação e a maneira como os cabeçalhos serão tratados, podendo reduzir as informações trafegadas relacionadas tanto à parte IPv6/UDP quanto ao protocolo utilizado na camada de aplicação. Neste trabalho será considerado apenas o primeiro caso, de modo que o protocolo posteriormente utilizado para comunicação com o medidor de energia não passará pelo processo de compressão.

O processo de definição das regras é de extrema importância para a compressão, visto que isso impactará diretamente na quantidade de dados a serem trafegados na rede LoRaWAN. Desta forma, foram definidas duas regras referentes a dois dispositivos distintos. Em cada situação, foram fixados o endereço do *Gateway* operador da rede SCHC (appPrefix e appIID) e os endereços dos dispositivos (devPrefix e devIID). A definição destes últimos depende do endereço do dispositivo atrelado à rede LoRaWAN, processo esse que será explicado ainda nesse capítulo.

Optou-se por somente comprimir os endereços indicados devido a falhas de implementação presentes nos algoritmos de compressão da biblioteca PySCHC, que não estavam processando corretamente as informações definidas nas regras quando aplicadas aos pacotes IPv6/UDP. Devido a limitações de tempo de desenvolvimento do trabalho, este problema não foi resolvido, ficando assim como melhoria futura na biblioteca.

### 3.1.3 Fragmentação

O processo de fragmentação do SCHC desempenha a função de possibilitar que o pacote IPv6/UDP comprimido possa ser transmitido através da rede LoRaWAN, visando nisso respeitar o MTU limitado pela rede. Nesta capítulo será apresentada a organização deste processo na implementação do SCHC juntamente com a aplicação no trabalho, considerando as máquinas de estados que orquestram as transmissões e os modos de transmissão que organizam o sentido de envio dos pacotes.

#### 3.1.3.1 Máquinas de estado

Tanto para o modo ACK *Always* quanto para o modo ACK *On Error*, são utilizadas máquinas de estados finitas (do inglês *Finite State Machine* - FSM) com o intuito de assegurar o gerenciamento correto de cada pacote SCHC transmitido e/ou recebido. De acordo com a RFC 8724, são dois componentes que diferenciam a FSM utilizada em cada transmissão de pacotes SCHC: o modo (ACK *On Error* ou ACK *Always*) e a atribuição (transmissor ou receptor). Dessa forma, fica estabelecido que dependendo do sentido

da comunicação (*uplink* ou *downlink*), diferentes máquinas de estados serão utilizadas. Isso garante que conexões com direções opostas possam ocorrer simultaneamente sem a ocorrência de conflitos no processamento das informações.

A organização desse processo é realizada a partir do uso do parâmetro "RuleID", projetado justamente com o intuito de garantir a interoperabilidade da rede. Ainda segundo a RFC 8724, fica estabelecido que esse parâmetro estará atrelado aos modos ACK *On Error* e ACK *Always* com os valores "20"e "21"respectivamente, diferenciando assim o sentido da comunicação. Já a RFC 9011 complementa esse ponto abordando a questão da transmissão simultânea de pacotes com o mesmo sentido, função atrelada ao parâmetro "DTag". Embora previsto na implementação inicial do SCHC, esse paralelismo de fragmentos transmitidos simultaneamente com a mesma "RuleID" não está previsto no uso do SCHC em LoRaWAN. Desta forma, "DTag" será considerado sempre como valor nulo.

Devido ao fato de que o envio/recebimento de um pacote SCHC envolve vários *uplinks* e *downlinks* na rede LoRaWAN, faz-se necessário que uma máquina de estados seja instanciada para cada pacote IPv6/UDP transmitido, de modo que todos os fragmentos do SCHC referentes a esse pacote serão processados pela mesma máquina de estados. Desta forma, optou-se pelo uso de sessões tanto no dispositivo quanto no *gateway* com o intuito de assegurar a existência de uma FSM durante todo o período de vida útil atrelado à transferência de um pacote. Assim, no momento em que uma transmissão é finalizada, a sessão e a respectiva FSM são descartadas.

Essa abordagem mostrou-se funcional em um primeiro momento, porém houveram situações em que fez-se necessária uma análise mais profunda. Tanto no modo ACK *On Error* quanto ACK *Always*, o descarte da sessão ao final do transporte de cada pacote IPv6/UDP estava atrelado à confirmação de um ACK SCHC. Entretanto existe a probabilidade de que essa mensagem não chegue ao destinatário, que como consequência seguirá solicitando confirmações. Ao receber essa solicitação, o receptor não estará ambientado pois a sessão com a FSM em questão foi descartada anteriormente. Apesar de não previsto em norma, optou-se pela resposta positiva para solicitações de ACK sem um "contexto conhecido", que na verdade se referem a uma transmissão já finalizada. Essa solução possibilitou o encerramento correto de transmissões diante dessas condições, porém comprometeu o envio de pacotes pequenos. Estes, normalmente compostos por poucos fragmentos Regulares e um All-1 seguido de uma solicitação de ACK, podem ser perdidos pelo fato de que o *gateway* pode receber apenas a solicitação de ACK diante de instabilidades na rede, que será respondida de forma positiva devido à estratégia escolhida.

Nestes casos, a camada de transporte seria comumente responsável pelo controle e gerenciamento dos pacotes perdidos. Entretanto, o SCHC prevê o uso do protocolo UDP devido à sua simplicidade e economia de tráfego se comparado ao TCP. A consequência disso é que a responsabilidade pela confiabilidade da rede IP passa a ser da camada de aplicação com um protocolo adequado.

A seguir serão apresentados os dois modos de operação implementados no trabalho, juntamente com suas máquinas de estados responsáveis pelo gerenciamento do fluxo de fragmentos e mensagens SCHC.

#### 3.1.3.2 ACK On Error

O modo ACK *On Error* é utilizado para o envio de pacotes *uplink* partindo do dispositivo em direção ao *Gateway*, sendo encontrado com mais frequência em análises presentes na literatura. Este modo é caracterizado pelo envio de mensagens em rajada, de modo que uma requisição ACK é realizada apenas após a transmissão de uma janela completa.

Neste caso, o dispositivo comporta-se como transmissor e o *gateway* como receptor. Partindo do dispositivo, um pacote IPv6/UDP é comprimido e fragmentado, sendo posteriormente transmitido em fragmentos do formato SCHC Regular (conforme apresentado na Seção 2.3.2.7). O último fragmento é enviado em formato SCHC All-1, justamente para indicar o final da janela em questão. Neste ponto, o transmissor solicita um ACK, sendo a resposta deste processada pelo receptor após a remontagem do pacote. Em caso de sucesso, a resposta é positiva. Caso contrário, o receptor envia o *bitmap* dos fragmentos recebidos juntamente com o ACK, de modo que o transmissor consegue a partir disso identificar os fragmentos faltantes e retransmiti-los. Esse processo pode acontecer mais de uma vez para um único pacote SCHC dependendo da taxa de sucesso das transmissões na rede LoRaWAN naquele momento. As Figuras 23 e 24 apresentam o mapeamento das etapas referentes às máquinas de estado do transmissor e do receptor.

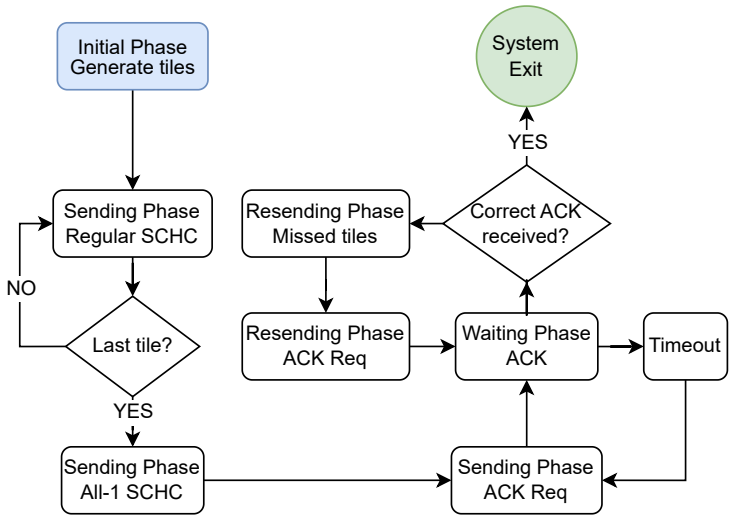
A partir da biblioteca PySCHC, foi possível operar com o modo ACK *On Error* logo no início do trabalho, visto que as implementações deste modo já estavam funcionais. Entretanto, foram necessários alguns ajustes na máquina de estados do receptor pois a finalização do processo de recebimento não estava acontecendo de forma correta.

#### 3.1.3.3 ACK Always

O modo ACK *Always* é utilizado para o envio de pacotes *downlink*, partindo do *gateway* em direção ao dispositivo. Esta operação é mais custosa para a rede LoRaWAN pois é necessário o envio de um ACK para cada fragmento recebido. Nesta situação, o dispositivo comporta-se como receptor e o *gateway* como transmissor.

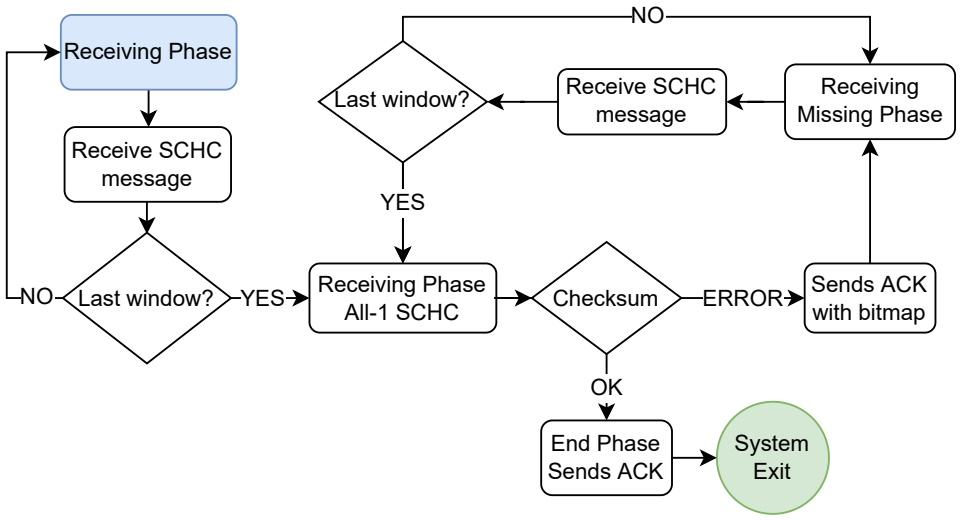
Este trabalho utiliza a comunicação Classe A da rede LoRaWAN conforme citado anteriormente na Seção 2.1.2. Desta forma, toda e qualquer transmissão deve ser inicializada pelo dispositivo. Sabendo que este comporta-se como receptor, existem duas formas de uma nova transmissão de *downlink* SCHC acontecer: (i) aproveitar uma comunicação

Figura 23 – FSM do transmissor ACK *On Error*.




Fonte: Autor.

Figura 24 – FSM do receptor ACK *On Error*.



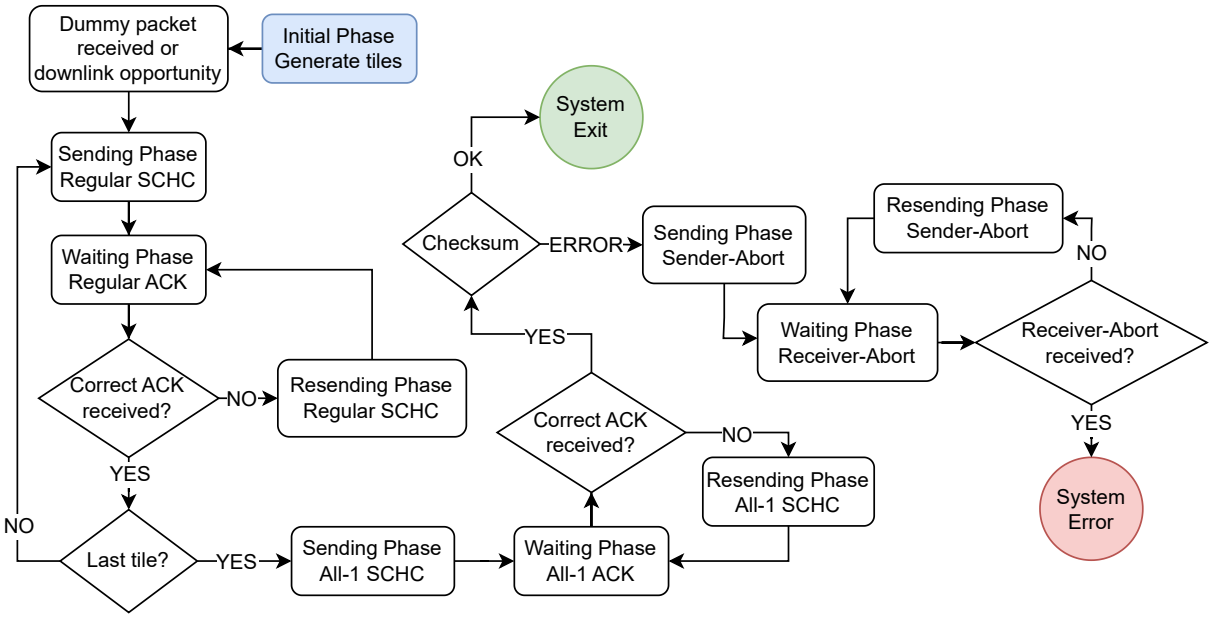

Fonte: Autor.

de *uplink* que está em curso paralelamente ou (ii) a partir do envio de um pacote *dummy* por parte do dispositivo. Essa última opção envia uma mensagem vazia na rede LoRaWAN com o intuito de capturar possíveis mensagens de *downlink* que estão em espera.

Apesar da biblioteca PySCHC fornecer um embasamento com o SCHC parcialmente desenvolvido, foi necessário aperfeiçoá-la a partir da implementação tanto do transmissor quanto do receptor do modo ACK *Always*. De acordo com a RFC 9011, cada janela possui apenas um fragmento, forçando com isso a confirmação no recebimento de cada mensagem (ao contrário do primeiro modo que faz o envio em rajada). Desta forma, os algoritmos do modo ACK *On Error* foram utilizados como base para as implementações necessárias. As Figuras 25 e 26 demonstram que a diferença entre as duas FSM, desse modo, está na transmissão seguida de uma confirmação, o que garante também maior

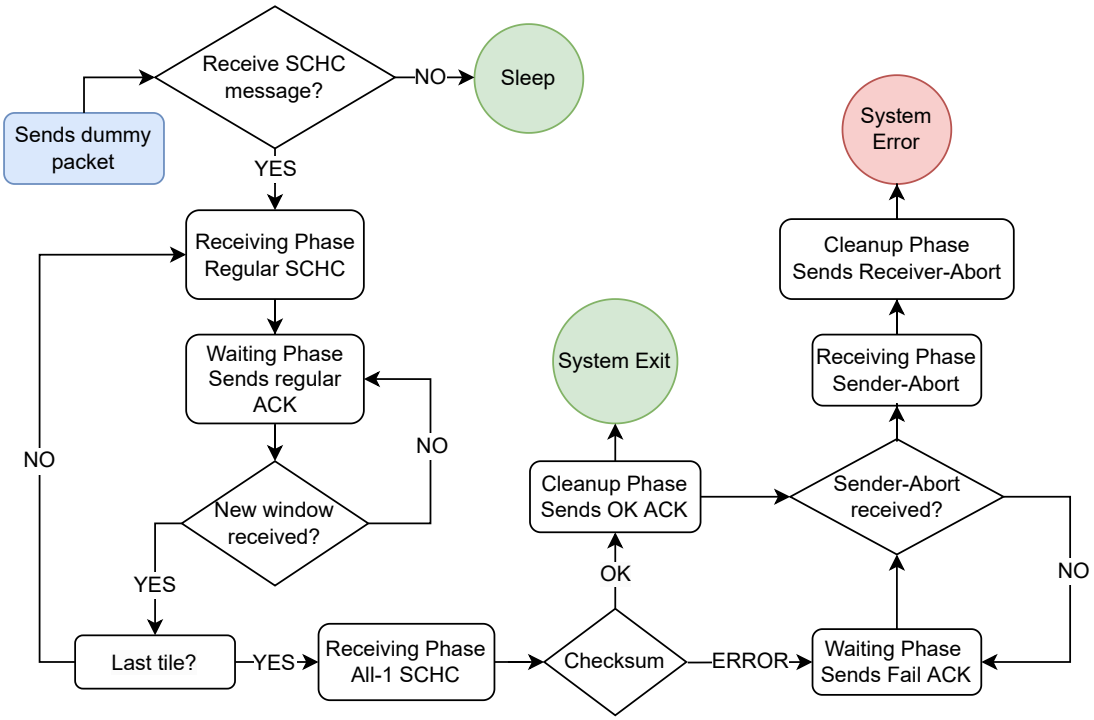
confiabilidade no recebimento correto de cada fragmento. Caso no final da transmissão o processo de verificação ainda assim falhe, optou-se pelo descarte total do pacote.

Figura 25 – FSM do transmissor ACK *Always*.




Fonte: Autor.

Figura 26 – FSM do receptor ACK *Always*.




Fonte: Autor.

## 3.2 GATEWAY

O *gateway* é um servidor de aplicação que, em uma situação real, deve ser executado em uma máquina/servidor com acesso à Internet. Sua responsabilidade é promover a conversão de protocolos entre os dispositivos conectados à Internet - ou uma rede local - e os dispositivos conectados na rede LoRaWAN. Desta forma, o *gateway* comunica-se via protocolo IP com as aplicações de modo a processar os pacotes recebidos com o uso do SCHC e encaminhá-los para os dispositivos através da rede LoRaWAN. O caminho oposto também é válido, sendo ele utilizado para respostas de solicitações oriundas da Internet ou de conexões inicializadas nos próprios dispositivos. Este processamento de pacotes IP é realizado de forma completamente agnóstica ao seu *payload*, que não passa por análises ou alterações.

Considerando que o *gateway* possui comunicação tanto com a rede LoRaWAN para comunicar-se com os dispositivos quanto com uma rede IP para comunicar-se com as aplicações, serão apresentadas a seguir estas duas interfaces juntamente com suas implementações.

### 3.2.1 Comunicação LoRaWAN

A partir do serviço de servidor de rede fornecido pela TTN, é possível comunicar-se com os dispositivos através da Internet, embora estes não possuam nativamente endereços IP para acesso direto e individual. Para isso, foi utilizado o protocolo MQTT para a troca de mensagens com o servidor de rede da TTN, de modo que posteriormente a rede LoRaWAN utilizada fica encarregada de encaminhar as informações aos dispositivos finais.

Neste contexto, foi utilizada a biblioteca Python chamada "Paho"[2], que fornece um conjunto de APIs (*Application Programming Interface*) para o envio e recebimento de mensagens. Para cada dispositivo conectado na rede, são utilizados dois tópicos de comunicação: um para *uplinks* e outro para *downlinks*. Dessa forma, a aplicação deve realizar a inscrição no primeiro para receber as mensagens do dispositivo, e realizar a publicação no segundo para enviar mensagens ao dispositivo. Este método garante que a individualidade dos dispositivos será mantida, assim como o sentido da comunicação será respeitado.

Todas as mensagens trafegadas nesse cenário partem das máquinas de estado do SCHC, responsável pelo processamento dos fragmentos relacionados aos pacotes trocados. Desta forma, a RFC 9011 estabelece que os pacotes de *uplink* (ACK *On Error*) deverão utilizar a porta 20 da rede LoRaWAN, assim como os pacotes de *downlink* (ACK *Always*) deverão utilizar a porta 21. Assim, será mantida a interoperabilidade citada ante-

[2]Disponível em: https://www.eclipse.org/paho

riormente, permitindo o fluxo simultâneo de pacotes nos dois sentidos. Além disso, o MTU disponível para o transporte de fragmentos SCHC será maior visto que o identificador do fluxo não será mais enviado juntamente com o *payload* da mensagem LoRa.

### 3.2.2 Comunicação IP

A interface IP do *gateway* é responsável pela comunicação com as aplicações finais que enviam e recebem pacotes IPv6/UDP, de modo que neste trabalho essas aplicações estão na mesma máquina onde o *gateway* está situado. Nisso, adotou-se a utilização de uma interface de rede virtual configurada com SO (sistema operacional) Linux. Também conhecida como túnel, ela foi configurada com o endereço IPv6 fixo "abdc::10". Na camada IP, este ficou definido também como o IP do *gateway*, de modo que todas as aplicações conectadas a ele utilizam este endereço como origem. Já os IPs referentes aos dispositivos foram configurados utilizando o túnel em questão como rota padrão. Assim, quando qualquer aplicação presente na máquina conectar-se a um destes IPs como destino, o gerenciador de redes do SO direciona os pacotes para a interface de rede virtual criada. Seus parâmetros podem ser visualizados na Figura 27.

Figura 27 – Visão geral da interface de rede virtual utilizada.

```
foxiot@foxiot-cristian:/$ ifconfig tun0
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet6 fe80::e02e:c9af:9ba2:a466  prefixlen 64  scopeid 0x20<link>
        inet6 abcd::10  prefixlen 128  scopeid 0x0<global>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 15557  bytes 1332209 (1.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17945  bytes 1535028 (1.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fonte: Autor.

A gerência do túnel por parte do *gateway* é realizada através da leitura e da escrita do arquivo referente ao túnel (automaticamente gerado pelo SO), de modo que a leitura deste implica na coleta dos pacotes que chegam e a escrita implica no envio de pacotes pela interface virtual. Desta forma torna-se possível o controle da interface por meio dos algoritmos implementados em Python, sendo que outras aplicações conseguem comunicar-se com ele por meio de *Sockets*. Optou-se pelo processo de manipulação do arquivo do túnel devido à necessidade de injetar nele pacotes IPv6/UDP que possuem como origem IPs referentes aos dispositivos, sendo que estes não estão de fato dentro da rede IP em questão. Os pacotes IPv6/UDP oriundos destes dispositivos são gerados através da biblioteca "Scapy", de modo que os IPs de origem e destino são manipulados neste processo (o IP de origem é definido como o endereço do dispositivo e o IP de destino é igual ao IPv6 do *gateway*). Isso justifica a escolha em injetar pacotes na interface virtual através do seu respectivo arquivo no sistema operacional.

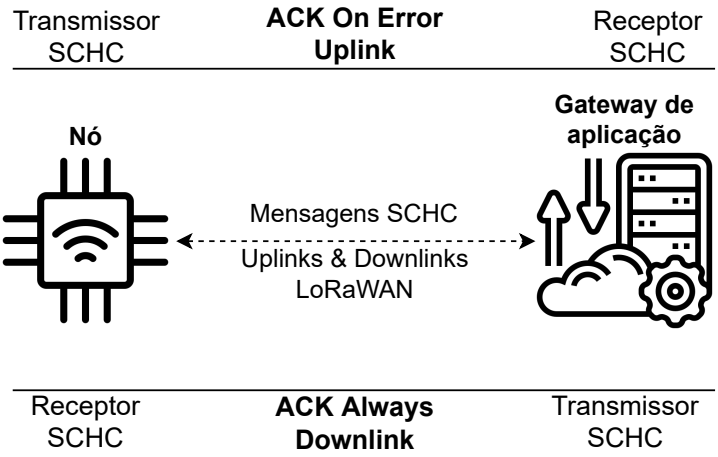
### 3.2.3 Suporte a multi dispositivos

Sendo o *gateway* responsável pela integração entre as redes LoRaWAN e IP, é natural esperar que este seja capaz de comunicar-se com diversos dispositivos simultaneamente em ambos os lados. Os pacotes IPv6/UDP que chegam ao *gateway* através do túnel podem ser oriundos de diversas aplicações, de modo que a constante leitura da interface de rede virtual já contempla a interpretação de todas essas fontes juntamente com a aplicação de filtros que definem quais pacotes serão realmente considerados. A organização destes é realizada por meio de filas de *uplink* (lidas do túnel) e *downlink* (para escrita no túnel). Os pacotes da primeira fila são encaminhados ao SCHC para compressão e fragmentação, tendo como destino os dispositivos da rede LoRaWAN. Ressalta-se que apenas um pacote IPv6/UDP é transmitido por vez quando oriundos ou destinados aos dispositivos. Já os pacotes da fila de *downlink* são encaminhados ao túnel tendo como destino final a respectiva aplicação relacionada à comunicação, através de um *Socket* IP.

Dentro do *gateway*, a comunicação com os dispositivos através do protocolo MQTT conectado à rede LoRaWAN é realizada de forma individual, de modo que uma *thread* é estabelecida para a comunicação via MQTT com cada dispositivo. Desta forma, a biblioteca *Paho* está configurada com uma subscrição para cada dispositivo, juntamente com o uso de uma máquina de estados SCHC dedicada para o gerenciamento da transmissão de cada pacote IPv6/UDP através dos fragmentos SCHC. Isso possibilita que a comunicação com cada dispositivo seja tratada de forma independente, de modo que as sessões atreladas às comunicações em curso não estejam em concorrência com comunicações paralelas.

O resultado disso é que dentro da aplicação do *gateway*, é instanciado em *thread* um objeto atrelado a cada dispositivo. Este objeto, denominado *handler* do *gateway*, possui os métodos necessários para compressão e descompressão, além das máquinas de estado utilizadas no processo de fragmentação dos pacotes enviados e recebidos. Desta forma, o *handler* fica responsável pelo processamento de todos processos referentes ao SCHC, integrando todas as suas funções em uma só instância. Neste contexto, tanto o dispositivo quanto o *gateway* podem assumir o papel de transmissor ou receptor, dependendo do modo SCHC e do sentido da comunicação vigente. A Figura 28 ilustra o sentido atrelado aos possíveis modos ACK *On Error* e ACK *Always*, sendo as funções de transmissor e receptor designadas de acordo com o modo vigente. Para exemplificar a aplicação do *gateway*, a Figura 29 apresenta a inicialização junto ao recebimento de um pacote IPv6/UDP via *Socket*.

Figura 28 – Modos SCHC atrelados ao dispositivo e ao *gateway* de acordo com o sentido da comunicação.




Fonte: Autor.

Figura 29 – Exemplo de informações na aplicação do *gateway*.

```
18/01/23 16:34:21 [INFO] [TUN] Device configured
18/01/23 16:34:21 [INFO] [TUN] Reading loop started
18/01/23 16:34:21 [INFO] [TUN] Writing loop started
18/01/23 16:34:21 [INFO] [SCHC] Started writing loop for DevEUI 70B3D57ED0055D4F IPv6 5454::e8
38:e51b:bc85:898c
18/01/23 16:34:21 [INFO] [SCHC] Started writing loop for DevEUI 70B3D57ED0058962 IPv6 5454::92
5a:bf5d:c5ce:2f9
18/01/23 16:34:21 [INFO] [MQTT] Connected with result code: 0
18/01/23 16:34:21 [INFO] [MQTT] Subscribing on DevEUI 70B3D57ED0055D4F
18/01/23 16:34:21 [INFO] [MQTT] Subscribing on DevEUI 70B3D57ED0058962
18/01/23 16:34:22 [INFO] [TUN] Receiving IPv6 packet | size: 112 bytes | uplink queue size: 1
###[ IPv6 ]###
  version   = 6
  tc        = 0
  fl        = 1005197
  plen      = 72
  nh        = UDP
  hlim      = 64
  src       = abcd::10
  dst       = 5454::e838:e51b:bc85:898c
###[ UDP ]###
     sport     = 34976
     dport     = 33334
     len       = 72
     chksum    = 0xb116
###[ Raw ]###
        load      = '\x00\x01\x00 ?\xff\x008`6\xa1\t\x06\x07`\x85t\x05\x08\x01\x01\x8a\x02\x07
\x80\x8b\x07`\x85t\x05\x08\x02\x01\xac\n\x80\x0800000000\xbe\x10\x04\x0e\x01\x00\x00\x00\x06_\
x1f\x04\x00\x00\x1e]\xff\xff'

18/01/23 16:34:22 [INFO] [SCHC] Scheduling packet: 600f568d00481140abcd000000000000000000000000
0001054540000000000e838e51bbc85898c88a082360048b116000100203fff00386036a10906076085740508010
18a0207808b076085740508020 1ac0a8008303030303030303030be10040e01000000065f1f0400001e5dffff
Length of uncompressed headers: 48 bytes
Length of compressed headers: 16 bytes
Compression: 66.67%
```

Fonte: Autor.

## 3.3 DISPOSITIVOS

Um dispositivo é caracterizado pela sua conexão à rede LoRaWAN, capaz de receber e enviar fragmentos a partir do SCHC e manipulá-los para pacotes IPv6/UDP. Com o objetivo de abstrair níveis mais profundos e específicos de configuração, optou-se pelo uso de um Modem LoRaWAN modelo RisingHF RHF3M076. Sua principal vantagem é a característica *plug & play*, de modo que a pilha LoRaWAN pode ser utilizada por meio de comandos AT via comunicação USB/serial. Desta forma, os canais de comunicação para o padrão AU915 e as chaves para autenticação na rede TTN são configurados apenas uma vez, estando o modem pronto para posterior recebimento dos comandos referentes à transmissão de mensagens.

Em conjunto com a comunicação através do modem LoRaWAN, é necessário que um *gateway* LoRaWAN esteja ao alcance na região, de modo que este comunica-se com os dispositivos através do protocolo LoRa e retransmita as mensagens aos servidores de rede TTN por meio da Internet, operando assim como um *Packet Forwarder*. Para isso, foi utilizado uma *Shield* modelo Radioenge RPi3 GwHat V1.1 acoplada a um Raspberry Pi 3. O desenvolvimento e os testes do trabalho contaram com o *gateway* e os modens LoRaWAN no mesmo ambiente.

A autenticação dos dispositivos na rede TTN é realizada através do modo OTAA (*Over The Air Activation*). O processo conhecido como *Join* realiza a autenticação com a rede a partir do uso de duas chaves simétricas criptografadas em AES 128-bit *Advanced Encryption Standard*, sendo elas conhecidas como "AppKey"e "NwkKey", aumentando com isso a segurança e a flexibilidade para mudanças de rede ao longo do tempo.

Assim como no caso do *gateway*, é instanciado um objeto *handler* em cada dispositivo, porém com funcionalidades respectivas às atribuições destes. Os métodos utilizados são semelhantes, com a diferença de que no modo ACK *On Error* será considerado a atribuição transmissor, enquanto no modo ACK *Always* será considerada a atribuição receptor. Ambos os modos podem operar simultaneamente conforme apresentado anteriormente na Seção 3.1.3.1, sendo que a inicialização do último dentro do dispositivo pode ocorrer por meio de uma comunicação *uplink* já em curso ou pelo envio periódico de mensagens *dummy*. A seguir serão apresentados os dispositivos e suas atribuições.

### 3.3.1 Dispositivo conectado ao medidor inteligente

Este dispositivo é responsável por manipular os pacotes IPv6/UDP transmitidos com auxílio do SCHC pela rede LoRaWAN, de modo a encaminhar o *payload* de cada pacote para um medidor inteligente de energia elétrica. Para isso, utilizou-se um microcomputador Raspberry Pi 3 com sistema operacional Linux, estando o modem LoRaWAN

conectado a ele via interface USB. Junto a isso, uma aplicação em Python ficou responsável pela execução do *handler* do dispositivo com o intuito de executar os algoritmos referentes ao SCHC juntamente com a comunicação LoRaWAN fornecida pelo modem LoRaWAN. A Figura 30 apresenta um breve exemplo de inicialização desta aplicação.

Figura 30 – Exemplo de informações na inicialização do dispositivo 1 via terminal.

```
b'+JOIN: Joined already\r\n'
18/01/23 16:34:37 [INFO] [SCHC] Scheduling routine packet: 150f
18/01/23 16:34:37 [INFO] [LoRaWAN] Setting up LoRaWAN on port 21
b'+PORT: 21\r\n'
18/01/23 16:34:37 [INFO] [LoRaWAN] Sending on port 21: 0f
b'+MSGHEX: Start LoRaWAN transaction\r\n'
b'+MSGHEX: TX "0F "\r\n'
b'+MSGHEX: PORT: 21; RX: "16 C0 F5 68 D0 04 81 14 08 8A 08 23 60 04 8B 11 60 00 10 02 03 F
F F0 03 86 03 6A 10 90 60 40 " \r\n'
b'+MSGHEX: RXWIN1, RSSI -31, SNR 6\r\n'
b'+MSGHEX: Done\r\n'
18/01/23 16:34:44 [INFO] [LoRaWAN] Receiving on port 21: 16c0f568d0048114088a082360048b116
000100203fff00386036a10906040
DEBUG::[2023-01-18 16:34:44.373870]::SCHC Fragment on 'Ack Always' mode, Receiver on 'Wait
ing phase' state
```

Fonte: Autor.

Ao passo que o dispositivo comunica-se com a rede LoRaWAN, ele é responsável também pela comunicação com o medidor inteligente de energia elétrica da fabricante Nansen modelo NSX P314i (apresentado na Figura 31 juntamente com o restante do *setup* deste dispositivo). Este medidor possui suporte nativo ao protocolo de comunicação DLMS/COSEM por meio de uma porta de comunicação serial RS232, sendo para isso necessário o uso de um conversor serial/USB para a comunicação com o Raspberry, que comporta-se como uma ponte entre o medidor e a rede LoRaWAN. Nesta configuração, o medidor atua como um servidor, de modo que as solicitações são enviadas a ele e posteriormente respondidas. Neste cenário, percebeu-se que em algumas situações as respostas não chegavam no limite de tempo estipulado de 3 segundos. Com isso, decidiu-se realizar até 5 tentativas de leitura serial no medidor ao invés de aumentar o limite de tempo (que aumentaria ainda mais a latência de comunicação na rede IP).

A partir desta estrutura, os pacotes IPv6/UDP são recebidos e enviados por meio das manipulações realizadas pelo SCHC juntamente com a comunicação com o *gateway* através da rede LoRaWAN. A partir da chegada de pacotes IP ou na necessidade de criação de novos, a biblioteca "Scapy" é utilizada para o encaminhamento dos pacotes DLMS/COSEM oriundos ou destinados ao medidor por meio da comunicação serial. Assim, estes pacotes originários de uma aplicação conectada ao *gateway* são extraídos dos pacotes IPv6/UDP para envio ao medidor, sendo que as informações recebidas deste são encapsuladas em um novo pacote IP para resposta à respectiva aplicação de origem.

Figura 31 – Setup do dispositivo 1 juntamente com o medidor inteligente de energia elétrica.


Fonte: Autor.

### 3.3.2 Dispositivo produtor de tráfego aleatório

Diante da disponibilidade de um único medidor inteligente de energia elétrica conectado a um Raspberry, este segundo dispositivo tem como função estabelecer conexão com uma aplicação por meio do tráfego de pacotes IPv6/UDP aleatórios. Para isso, um *handler* também é utilizado no dispositivo juntamente com um segundo modem LoRaWAN, os quais são executados em um *notebook* com sistema operacional Linux. Assim, o *payload* do pacote IPv6/UDP é extraído com auxílio da biblioteca "Scapy", que também é utilizada para a montagem dos pacotes resposta. Optou-se pela implementação desta configuração devido à necessidade de testar o sistema operando com mais de um dispositivo simultaneamente, podendo ser posteriormente escalado para uso com vários medidores em projetos maiores. Além disso, o uso de *payloads* com tamanhos e conteúdos aleatórios possibilita testes do emprego da rede LoRaWAN em conjunto com o protocolo SCHC.

### 3.3.3 Endereçamento IPv6

Cada dispositivo possui um identificador único na rede LoRaWAN chamado DevEUI, que é responsável pela diferenciação dos dispositivos e utilizado para o direcionamento correto no envio e recebimento de mensagens. Já na rede IP, esses mesmos dispositivos não possuem um identificador nativo, como um endereço IPv6. Desta forma, de acordo com a RFC 9011, cada dispositivo receberá um IID (*Interface IDentifier*) calculado a partir do algoritmo de criptografia AES-CMAC com base no DevEUI e na chave AppSKey (ambos atrelados ao dispositivo na rede LoRaWAN). Isso possibilita a criação do identificador DevIID (a ser utilizado dentro da rede IP), de modo a não criar uma correlação com o identificador do *hardware* (DevEUI) devido à possibilidade de alteração da chave AppSKey ao longo do tempo.

Desta forma, foi utilizada a mascara de rede "5454::/64" para endereçamento dos dispositivos, sendo os DevIIDs calculados individualmente. Os endereços IPv6 dos dois dispositivos foram definidos como "5454::e838:e51b:bc85:898c" e "5454::925a:bf5d:c5ce:2f9", respectivamente. Assim, as aplicações se comunicam com os dispositivos utilizando conexões IPv6/UDP por meio do túnel de rede utilizado, que por sua vez é gerenciado pelo *gateway* para processamento dos pacotes utilizando o SCHC e a rede LoRaWAN.

## 3.4 COMUNICAÇÃO COM O MEDIDOR INTELIGENTE DE ENERGIA ELÉTRICA

Além de implementar um ambiente para execução do protocolo SCHC, este trabalho tem como objetivo também possibilitar a comunicação entre uma aplicação e um medidor inteligente de energia elétrica por meio da rede desenvolvida. Nesta seção será apresentada a comunicação com o medidor a partir do protocolo DLMS/COSEM, que utiliza o método de solicitação-resposta a partir de uma aplicação responsável pelo gerenciamento dos pacotes DLMS/COSEM.

### 3.4.1 Pacotes DLMS/COSEM

Embora a análise do protocolo DLMS/COSEM não seja o foco deste trabalho, foi necessário definir quais pacotes seriam enviados ao medidor inteligente de energia elétrica com o intuito de obter informações reais da rede elétrica no momento da solicitação. Desta forma, foi utilizado o software "Gurux Director" (Gurux, 2022) com o objetivo inicial de estabelecer a comunicação direta com o medidor, a partir do uso de um *notebook* e um conversor serial/USB. Vale ressaltar que o DLMS/COSEM possui sua própria camada de segurança e cada modelo de medidor possui seus próprios parâmetros para conexão

e chaves para autenticação, de modo que essas informações são normalmente repassadas apenas às concessionárias que adquirem o produto. Assim, considerando diferentes níveis de acesso ao medidor via protocolo DLMS/COSEM, foi utilizado o acesso menos privilegiado, sendo possível apenas a leitura de informações referentes às coletas realizadas pelo medidor na rede elétrica (funcionalidades como escrita de configuração não estavam permitidas neste caso). Os dados acessados a partir deste método partem da memória interna do medidor, que por sua vez realiza o processo constante de leitura e armazenamento das informações de forma automática. Assim, em caso de falha na comunicação com entidades externas, não há prejuízo quanto às informações armazenadas.

Definiu-se a partir disso que os pacotes utilizados seriam referentes às solicitações de autenticação com o medidor e de tensão da rede elétrica. O envio destes é realizado a partir de uma aplicação conectada ao *gateway*, sendo ela apresentada a seguir.

### 3.4.2 Aplicação cliente DLMS/COSEM

Responsável pela comunicação com o medidor inteligente de energia elétrica, a aplicação cliente DLMS/COSEM utiliza a camada de aplicação do trabalho para a troca de informações. Implementada em Python, a aplicação estabelece uma comunicação por meio de um cliente Socket IPv6/UDP, utilizando o IP do respectivo dispositivo como destino. Neste caso, foi adotada a porta "33334" para as aplicações deste tipo, de modo que o dispositivo receberá os pacotes nesta e posteriormente responderá na porta gerada pelo *Socket*.

Representada pela Figura 32, a aplicação envia inicialmente ao medidor um pacote DLMS/COSEM contendo a solicitação, os parâmetros e a senha de acesso "00000000", inicializando com isso uma nova sessão (primeira informação enviada ao dispositivo, conforme Figura). A partir da resposta positiva por parte do medidor (primeira informação recebida do dispositivo, conforme mesma figura), a aplicação solicita a tensão da rede elétrica naquele momento, processo esse que é repetido constantemente após o recebimento das respectivas respostas por parte do medidor. Caso uma resposta fora do esperado seja recebida, a aplicação envia uma solicitação de autenticação para uma nova sessão, repetindo então o procedimento.

O intuito deste processo é gerar tráfego constante com base no método de solicitação-resposta, servindo de exemplo para futuras expansões e implementações em situações reais como a medição remota por parte das concessionárias de energia elétrica.

Figura 32 – Exemplo de informações na aplicação DLMS/COSEM.

```
18/01/23 16:34:22 [INFO] [SOCKET] Sending to 5454::e838:e51b:bc85:898c on port 33334:
    000100203fff00386036a1090607608574050801018a0207808b0760857405080201ac0a80083030303030
303030be10040e01000000065f1f0400001e5dffff
18/01/23 16:35:56 [INFO] [SOCKET] Receiving from 5454::e838:e51b:bc85:898c on port 33334:
    00013fff0020002b6129a1090607608574050801018a203020100a305a103020100be10040e0800065f1f04
0000181d01400007
18/01/23 16:35:56 [INFO] [SOCKET] Sending to 5454::e838:e51b:bc85:898c on port 33334:
    000100203fff000dc001c100030100480700ff0200
18/01/23 16:38:17 [INFO] [SOCKET] Receiving from 5454::e838:e51b:bc85:898c on port 33334:
    00013fff00200009c401c1000600005649
18/01/23 16:38:17 [INFO] [SOCKET] Sending to 5454::e838:e51b:bc85:898c on port 33334:
    000100203fff000dc001c100030100480700ff0200
18/01/23 16:39:39 [INFO] [SOCKET] Receiving from 5454::e838:e51b:bc85:898c on port 33334:
    00013fff00200009c401c100060000563a
```

Fonte: Autor.

## 3.5 APLICAÇÃO PRODUTORA DE TRÁFEGO ALEATÓRIO

O segundo dispositivo está atrelado a uma aplicação que tem como intuito gerar o tráfego de pacotes aleatórios para análise da rede, conforme citado na Seção 3.3.2. Semelhante ao processo apresentado no dispositivo anterior, a Figura 33 demonstra que inicialmente um pacote aleatório é enviado para o dispositivo, que por sua vez responde enviado um pacote nos mesmos moldes. Este processo é repetido ao longo do tempo, tendo como princípio a comunicação através do envio de pacotes com dados aleatórios e tamanhos diversos.

Figura 33 – Exemplo de informações na aplicação produtora de tráfego aleatório.

```
18/01/23 16:40:52 [INFO] [SOCKET] Sending to 5454::925a:bf5d:c5ce:2f9 on port 33334:
    44504a6f35386368624d55315a336d6144556b493879756b4f49437055357858776d4a41684130534670757575 6c494d
62777a4e62685346747a52704152463965704566 4c6a475946386e61424e50394a477a5775304d5373334456715759466754
7134705a6e32306f664b54394b31477467575435794e6c336c617852723950784247437946457 96f38476f586f6b69434c6a
18/01/23 16:42:18 [INFO] [SOCKET] Receiving from 5454::925a:bf5d:c5ce:2f9 on port 33334:
    503766664142434 44f7869747255584c59306676555767726b557242546b4b514453504332377a455066546f474a5559
4a52746e5272316e6b31314c366257644e7631613978396c3859473538763835376368785370486c4d414f71675359745852
697675747667467847 4d43476f70326c323369503230654b513766715963674c4d4733716756536347564 36c37664b654f49
56516b5874415442775576 6444f4357586c766a3656484f66316c436e4f4967726a474641796638457464724975637 6456a
6463456e593058337a68417a41554c5232414341665147337644757865533373353 84d623832676b6d583267636262457077
6f67504c6d4a4634443039714343783578475 76b554852705156384a57787348466272476345616a4471654145 6d3438
18/01/23 16:42:18 [INFO] [SOCKET] Sending to 5454::925a:bf5d:c5ce:2f9 on port 33334:
    374355394130416c6749304c5a694f73704e5a704371754b6a48796932616d5432443754425755534b414b6844755745
483334794f4b68504a6e7a635068414c56534a7243755a766f58336d6f4c556b334675734f4d7974376f4347436a4a764f48
3354306d4c566e674a4447706b536965517338386554726d636d4c4e677966747356747769477742656b554a56314c78554a
7535665766747649 4d32496b38614d315075677a6e5371467a79475a4e77763244684d654549583436744738455a41444c78
4e42475844527a654e6b396430583163307936547a584a713653785263643768543668555549 6d39457
```

Fonte: Autor.

# 4 RESULTADOS E DISCUSSÕES

Após a implementação do sistema e o desenvolvimento dos dois dispositivos apresentados, foram realizados testes com a intenção de analisar o funcionamento da rede como um todo, além de tornar possível a identificação de problemas causados a partir de comportamentos na rede LoRaWAN que não estavam previstos anteriormente. Os experimentos foram realizados em diversas rodadas ao longo do período de um mês, tendo cada um a duração de horas ou até dias. Ao passo que novas situações foram acontecendo e com isso novos problemas foram identificados, realizou-se o processo de correção nos algoritmos com o intuito de tornar o sistema mais robusto.

Os experimentos e seus respectivos resultados a serem apresentados a seguir contaram com um intervalo total de 24 horas de execução. O primeiro teste foca na comunicação com o medidor inteligente de energia elétrica, com ênfase na análise da relação entre o protocolo SCHC e os pacotes DLMS/COSEM. Já o segundo teste possui ênfase na análise do desempenho da rede, possuindo uma relação maior entre o SCHC e o seu desempenho nas redes LoRaWAN e IP. Para ambos foi utilizado um MTU fixo de 32 *bytes*, uma vez que os dados trafegados no primeiro dispositivo são relativamente pequenos, tendo assim o objetivo de gerar vários fragmentos a partir do mesmo pacote.

## 4.1 DISPOSITIVO CONECTADO AO MEDIDOR INTELIGENTE

O primeiro dispositivo realiza a comunicação em tempo real com o medidor inteligente de energia elétrica, de modo que a aplicação solicita informações e este as responde. Neste ambiente, definiu-se a não utilização do mecanismo ADR da rede LoRaWAN, fazendo com que a comunicação entre os rádios não estivesse otimizada. Em termos práticos, essa configuração provocou uma frequente perda de pacotes, apesar de que optou-se por esse cenário justamente pelo intuito de forçar que o SCHC obtivesse sucesso nas transmissões a partir da análise automática das situações e consequente reenvio dos fragmentos perdidos. Desta forma, foi possível realizar os experimentos em um ambiente mais próximo de uma aplicação real.

### 4.1.1 Compressão

A compressão dos conteúdos estáticos dos cabeçalhos IPv6/UDP depende das regras de compressão previamente configuradas, de modo que neste trabalho optou-se pela compressão dos IPs de origem e destino conforme comentado na Seção 3.1.2. Como re-

sultado, o índice de compressão permaneceu constante ao longo de todos os pacotes. Isso se deve ao fato de que o conteúdo alterado (IPs) esteve sempre estático, diferente de situações que poderiam ser consideradas como o cálculo do código de *checksum* referente ao protocolo UDP (o qual é previsto no SCHC como cálculo local no transmissor e no receptor ao invés da sua inclusão no pacote). Ainda assim, a compressão de 66% já considera uma expressiva redução de tráfego se considerada um longo período, sendo o restante dos pacotes atrelados aos seus respectivos *payloads*. Ressalta-se que estes procedimentos de compressão foram definidos e utilizados em todos os testes deste trabalho.

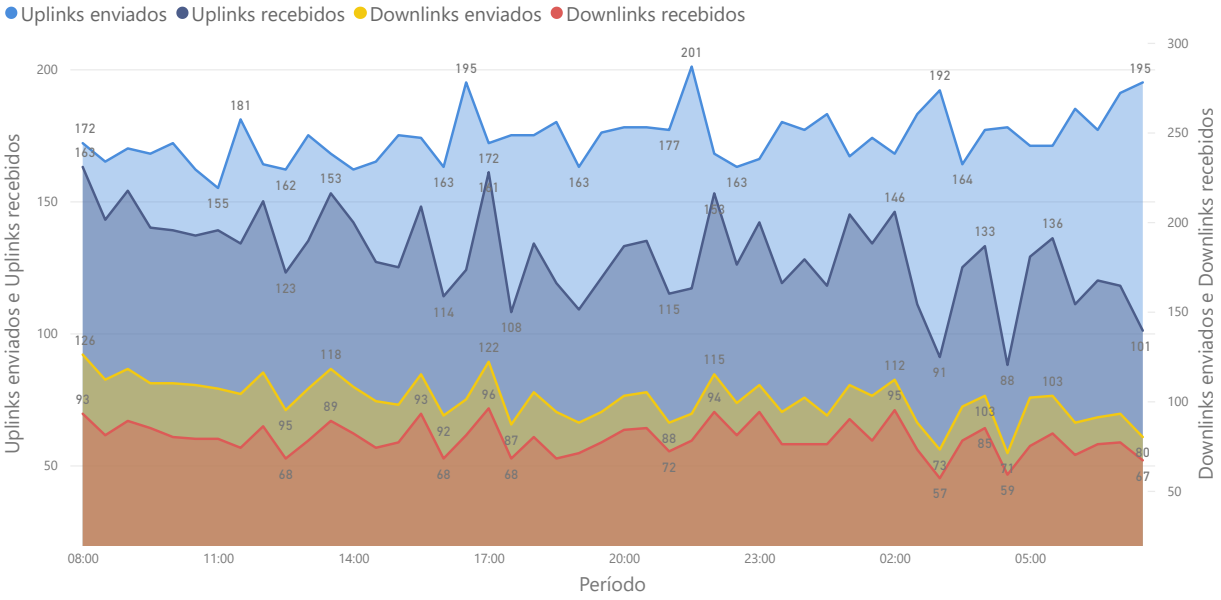
### 4.1.2 Rede LoRaWAN

O sucesso na transmissão dos pacotes IPv6/UDP está diretamente atrelado à qualidade da rede LoRaWAN, de modo que os fragmentos do protocolo SCHC precisam chegar ao destinatário para o correto funcionamento do sistema. Com o intuito de formar uma base fundamentada no comportamento nos eventos ocorridos ao longo do período considerado, o Gráfico 1 mostra a relação entre a quantidade de mensagens que foram enviadas e a quantidade de mensagens que realmente foram recebidas, tanto no sentido de *uplink* na rede LoRaWAN (dados representados em tons de azul) quanto no sentido contrário (dados representados nas cores amarelo e laranja). O gráfico usa a média do período em intervalos de 30 minutos, de modo a mostrar que os *uplinks* sofreram com momentos de oscilação ao longo das 24 horas, gerando consequências diretas na comunicação do protocolo SCHC. Por outro lado, os *downlinks* se mostraram mais constantes, sendo uma possível causa disso a forma como a transmissão das mensagens é realizada. Como apresentado na Seção 2.1.2, as janelas de *downlink* ocorrem na sucessão de janelas de *uplink*, onde a probabilidade de que a rede LoRaWAN esteja estável é maior justamente pelo fato de que a primeira janela (de *uplink*) obteve êxito, indicando que há comunicação entre os dois pontos. Isso soma-se ao fato de que o dispositivo estará com duas janelas de recebimento sincronizadas com o *gateway* da rede, promovendo assim o recebimento do *downlink*.

Como consequência disso, aproximadamente 23% dos *bytes* transmitidos em *uplink* e 10% dos *bytes* transmitidos em *downlink* foram perdidos, conforme indica o Gráfico 2.

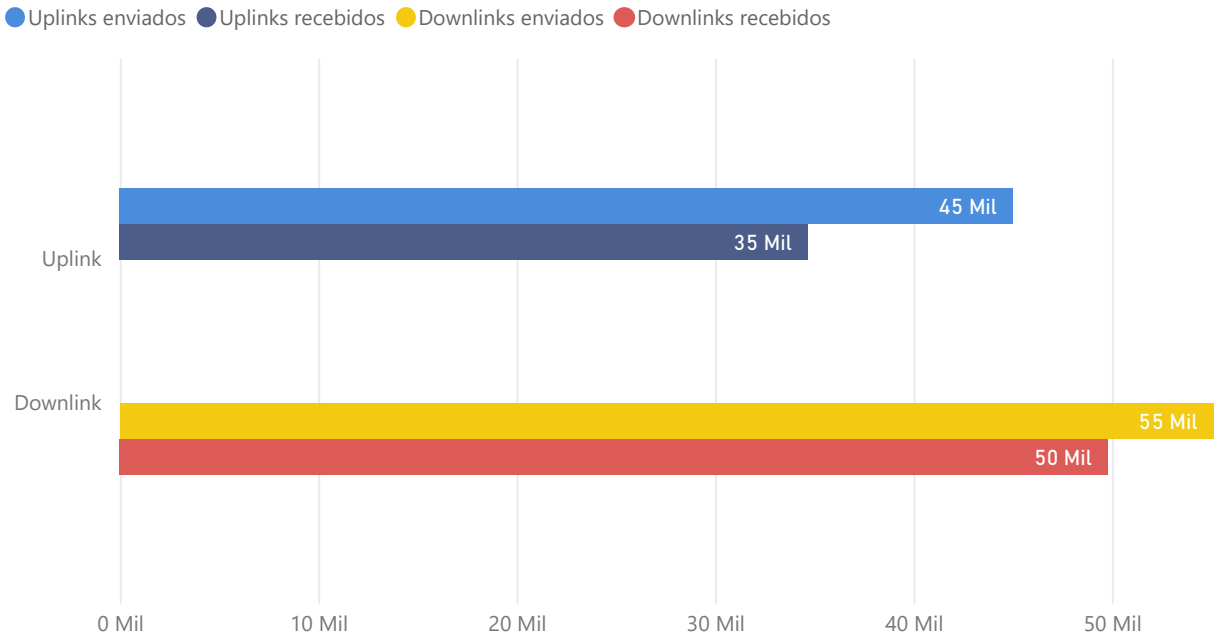
A seguir será analisado o comportamento do SCHC quanto aos sentidos de transmissão *uplink* (ACK *On Error*) e *downlink* (ACK *Always*), bem como a camada de aplicação composta pelo uso do protocolo DLMS/COSEM.

Gráfico 1 – Situação da rede LoRaWAN entre Gateway e dispositivo 1.




Fonte: Autor.

Gráfico 2 – Pacotes enviados e recebidos entre Gateway e dispositivo 1 em *bytes*.



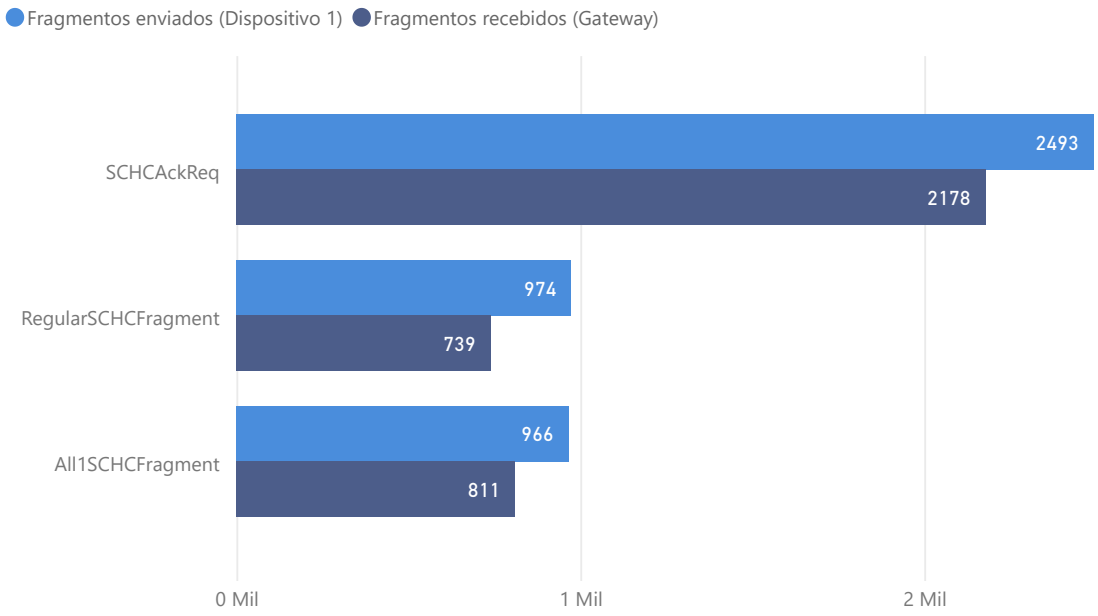

Fonte: Autor.

### 4.1.3 ACK On Error

Conforme apresentado na Seção 3.1.3.2, o modo ACK *On Error* faz o disparo de todos os fragmentos SCHC (primeiro os fragmentos Regulares e por último o All-1), enviando uma requisição de ACK ao final de cada transmissão. No caso das respostas das solicitações com DLMS/COSEM, os pacotes SCHC foram em sua grande maioria partidos

em um fragmento Regular e outro All-1. O Gráfico 3 apresenta a relação destes fragmentos enviados do dispositivo 1 ao *Gateway*, ao passo que compara também a quantidade de fragmentos que foram enviados versus o que realmente foi recebido. A comparação apresentada no Gráfico 1 já indicava a existência de mensagens perdidas na rede LoRaWAN, sendo agora possível confirmar que para cada pacote IPv6/UDP transmitido, existe a necessidade de retransmissão de fragmentos SCHC devido à falhas na comunicação entre o modem e o *gateway* LoRaWAN.

Gráfico 3 – Fragmentos SCHC em *uplink* no modo ACK *On Error* entre dispositivo 1 e Gateway.

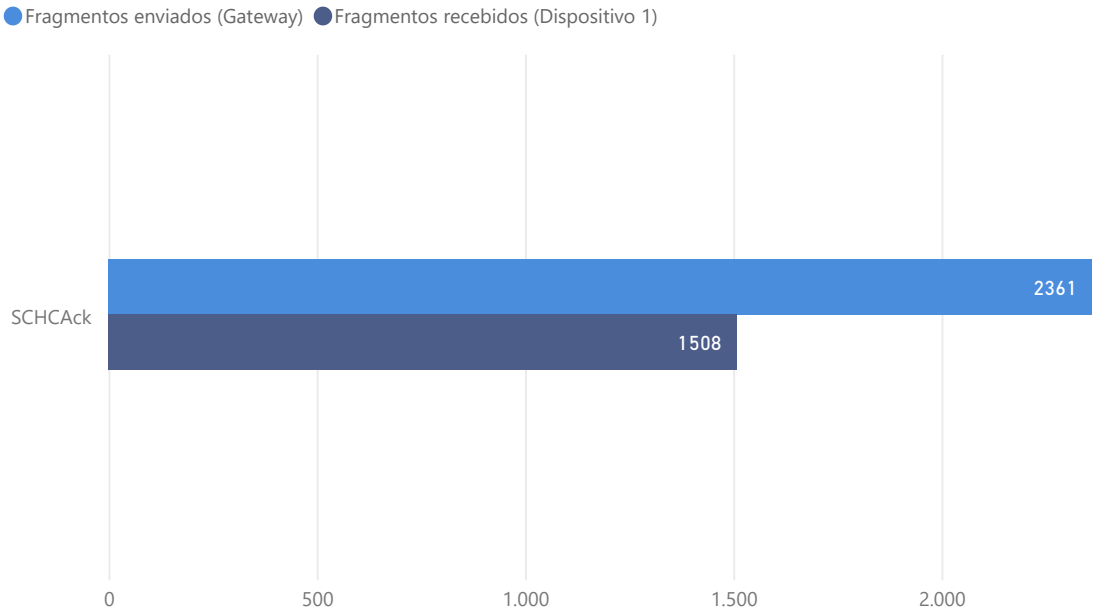



Fonte: Autor.

Um segundo ponto observado no Gráfico 3 é a quantidade de solicitações de ACK enviadas pelo dispositivo. Esse número deveria ser próximo dos fragmentos All-1, visto que a solicitação é realizada após o envio destes. O *gateway* recebe a maior parte destas requisições, entretanto, dois agendamentos de *downlink* na rede LoRaWAN são realizados nesse momento: a confirmação com ACK e o primeiro fragmento da transmissão *downlink* de um novo pacote IPv6, pois a aplicação recebeu a resposta do dispositivo e já enviou uma nova solicitação de informações do medidor. A partir disso, a rede considera apenas o último agendamento e descarta o primeiro. Uma possível solução seria a inserção de um intervalo entre esses dois agendamentos, sendo necessária uma avaliação quanto ao possível aumento de latência provocado por essa alternativa.

Como consequência, o dispositivo terá que enviar ao menos duas requisições de ACK ao *gateway* até realmente receber a confirmação (desconsiderando aquelas perdidas em virtude de falhas na comunicação LoRaWAN), sendo possível visualizar o contraste disso no Gráfico 4. A quantidade de ACKs enviados ao dispositivo é próxima das requisições de ACK recebidas pelo *gateway*, entretanto a disparidade entre os ACKs enviados e recebidos revela que grande parte destes não chega de fato a ser enviado ao dispositivo.

Gráfico 4 – Fragmentos SCHC em *downlink* no modo ACK *On Error* entre dispositivo 1 e Gateway.




Fonte: Autor.

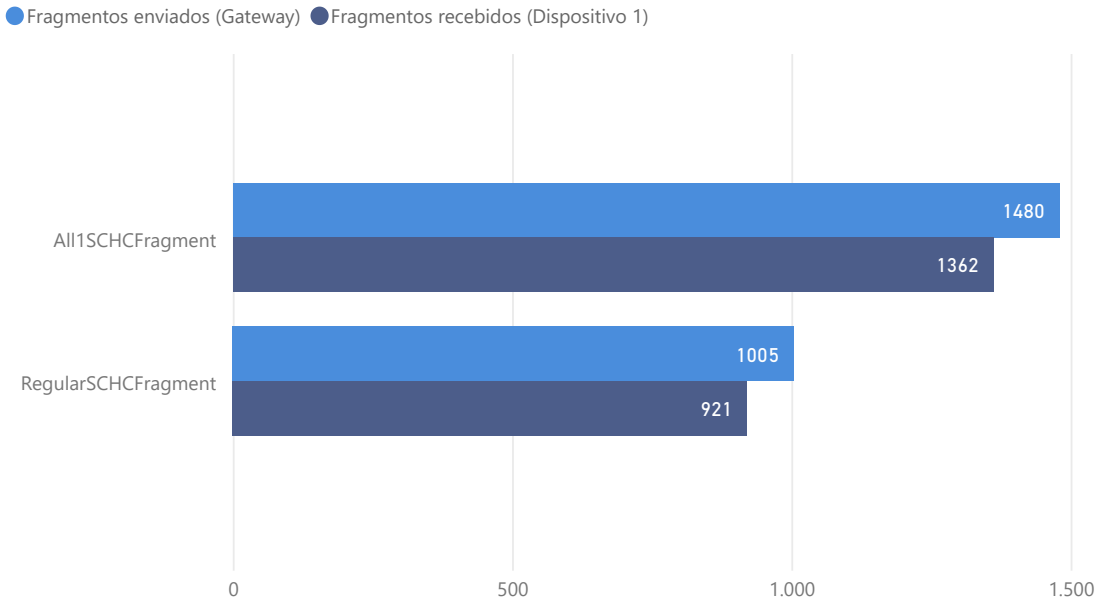
### 4.1.4 ACK Always

No envio de pacotes IPv6/UDP no sentido *downlink*, o transmissor envia cada fragmento utilizando o modo ACK *Always* e aguarda pelo ACK por parte do receptor. Esse processo é utilizado aqui para o envio de solicitações oriundas da aplicação em direção ao medidor de energia elétrica, considerando a requisição de autenticação ou da tensão na rede elétrica conectada ao medidor. Em maior ocorrência, a requisição da tensão requer a transmissão de apenas um fragmento SCHC Regular e outro All-1.

O resultado esperado a partir disso era que os números de envios do *gateway* ao dispositivo referentes aos dois tipos de fragmentos fosse semelhante, entretanto, o Gráfico 5 revela o comportamento oposto, sendo o envio de fragmentos All-1 aproximadamente 47% maior que fragmentos Regulares. Isso se deve ao fato de que, na atual implementação do *downlink* com o modo ACK *Always* juntamente com uma imposição da TTN para a rede LoRaWAN, cada fragmento é enviado pelo menos duas vezes ao dispositivo, de modo que a FSM do transmissor não consegue avançar antes da confirmação com o recebimento do ACK correto. A perda de pacotes na rede LoRaWAN aumenta as proporções desse problema, tendo como resultado o reenvio dos fragmentos. Essa problemática será melhor explorada neste capítulo na apresentação e discussão dos resultados referentes ao segundo dispositivo.

A partir de cada recebimento, o receptor envia uma confirmação ACK referente ao fragmento recebido, permitindo com isso que o *gateway* prossiga com o envio dos próximos fragmentos. Em algumas situações, o transmissor não recebe esse ACK devido à perda de pacotes na rede LoRaWAN, podendo causar atrasos na transmissão do pacote

Gráfico 5 – Fragmentos SCHC em *uplink* no modo ACK *Always* entre dispositivo 1 e Gateway.




Fonte: Autor.

IPv6. Entretanto, as normas que padronizam o funcionamento do SCHC prevêm a implementação de um temporizador nestas situações, de modo que caso o dispositivo envie um ACK e não receba novos fragmentos em um intervalo de tempo futuro (caso a transmissão ainda não tenha sido finalizada), o dispositivo reenvia o último ACK com o intuito de gerar uma nova oportunidade de continuidade na transmissão.

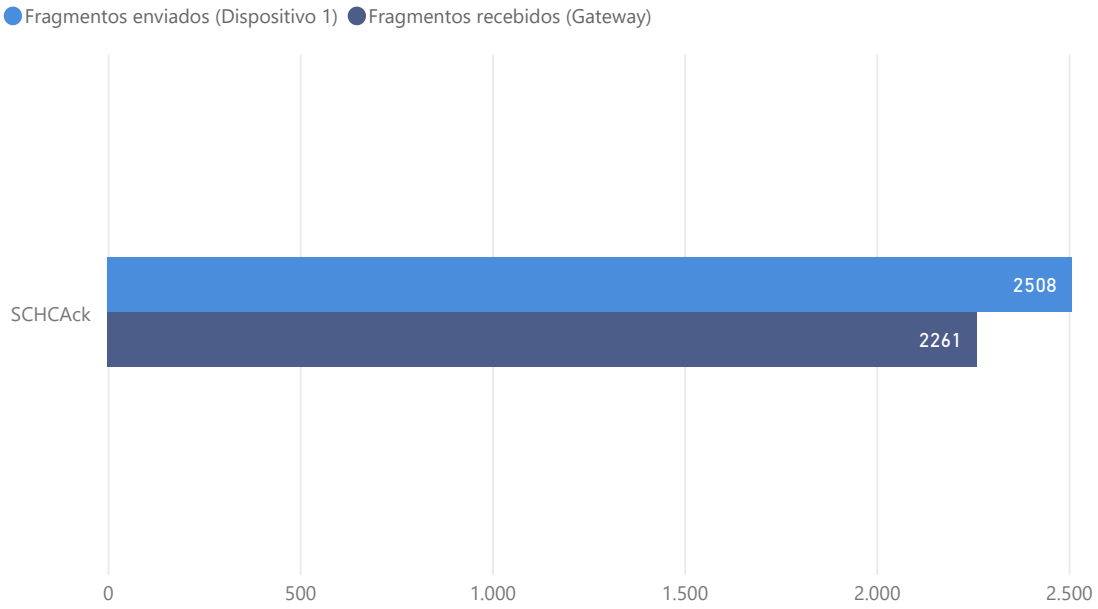
O Gráfico 6 mostra que o número de ACKs enviados contempla o número de fragmentos Regulares e All-1 entregues ao receptor, somados com o volume de ACKs que foram reenviados por conta do limite de tempo estipulado em 3 minutos onde nenhum fragmento novo foi recebido pelo dispositivo. Ressalta-se que este fato não possui relação com o envio duplicado de fragmentos comentado acima, de modo que aqui o reenvio de ACKs se deve às perdas nas transmissões da rede LoRaWAN.

A seção a seguir tratará das camadas de rede, transporte e aplicação, tendo como foco a análise da comunicação entre a aplicação e o medidor de energia elétrica por meio do envio de solicitações e recebimento de informações com o protocolo DLMS/COSEM.

### 4.1.5 Análise da aplicação

Todos os processos orquestrados pelo SCHC resultam em pacotes IPv6/UDP, que transportam pacotes DLMS/COSEM com requisições e respostas entre aplicação e medidor de energia elétrica. A confiabilidade nessa comunicação depende do sucesso no tráfego de pacotes de *uplink* e *downlink*, nos quais o trabalho se mostrou estável quando analisadas as taxas de entrega. A partir do Gráfico 7 (com somas em intervalos de 30

Gráfico 6 – Fragmentos SCHC em *downlink* no modo ACK *Always* entre Gateway e dispositivo 1.



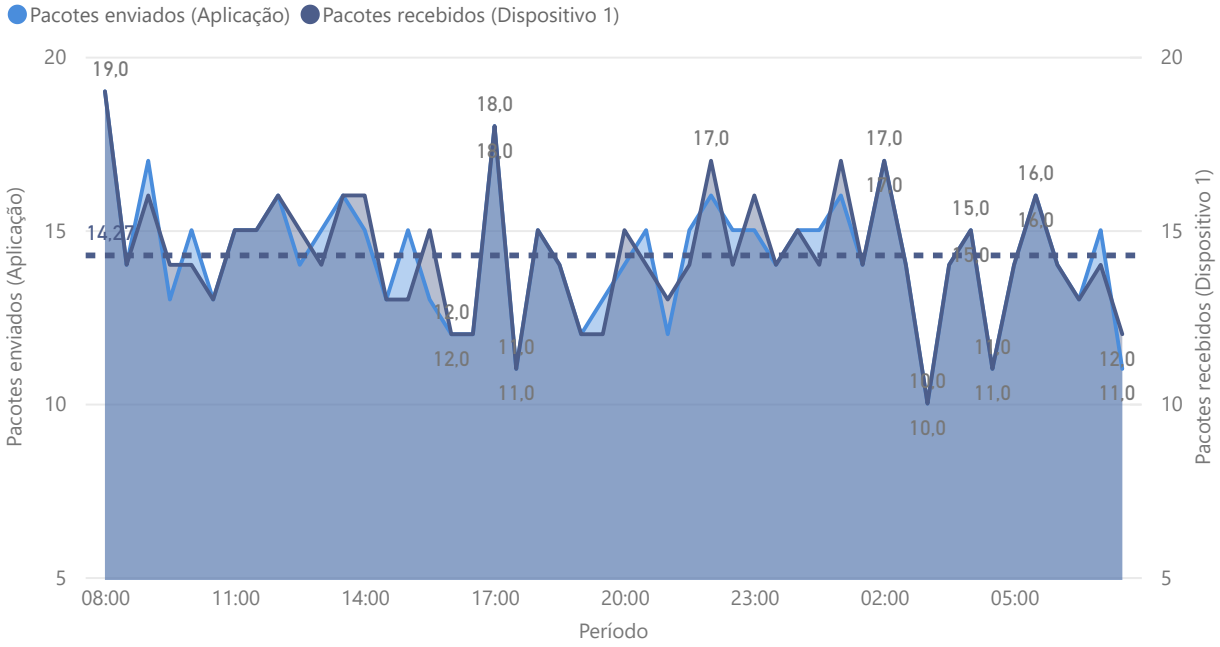

Fonte: Autor.

minutos), é possível perceber que a maioria dos pacotes IPv6/UDP enviados pela aplicação foram recebidos pelo dispositivo, mostrando que, apesar das oscilações ao longo do intervalo provocadas por falhas na rede LoRaWAN, o processo de transmissão de pacotes de *downlink* do SCHC mostrou-se confiável. Isso ocorre pois, mesmo diante das perdas de fragmentos na rede, o SCHC foi capaz de identificar as falhas e solicitar as devidas retransmissões. Além disso, observa-se no gráfico que o recebimento de pacotes é maior que o envio em determinados momentos. Isso se deve ao agrupamento do intervalo em períodos de 30 minutos para exibição no gráfico, que pode gerar essas divergências pelo fato do envio e o recebimento de alguns pacotes estarem em agrupamentos diferentes.

Já para o fluxo de *uplink* de pacotes IPv6, houveram maiores perdas entre os pacotes enviados pelo dispositivo e os recebidos pela aplicação, conforme observa-se no Gráfico 8 (também com somas em intervalos de 30 minutos). Embora a taxa de sucesso ainda seja alta, perdas de pacote ocorreram devido à estratégia da resposta positiva para solicitações de ACK sem um "contexto conhecido" (apresentada na Seção 3.1.3.1), em conjunto com a quantidade pequena de fragmentos desses pacotes devido aos seus tamanhos. Nestes casos, apesar dos mecanismos de confiabilidade implementados no SCHC, a detecção da perda e o controle destes pacotes fica a cargo da camada de aplicação.

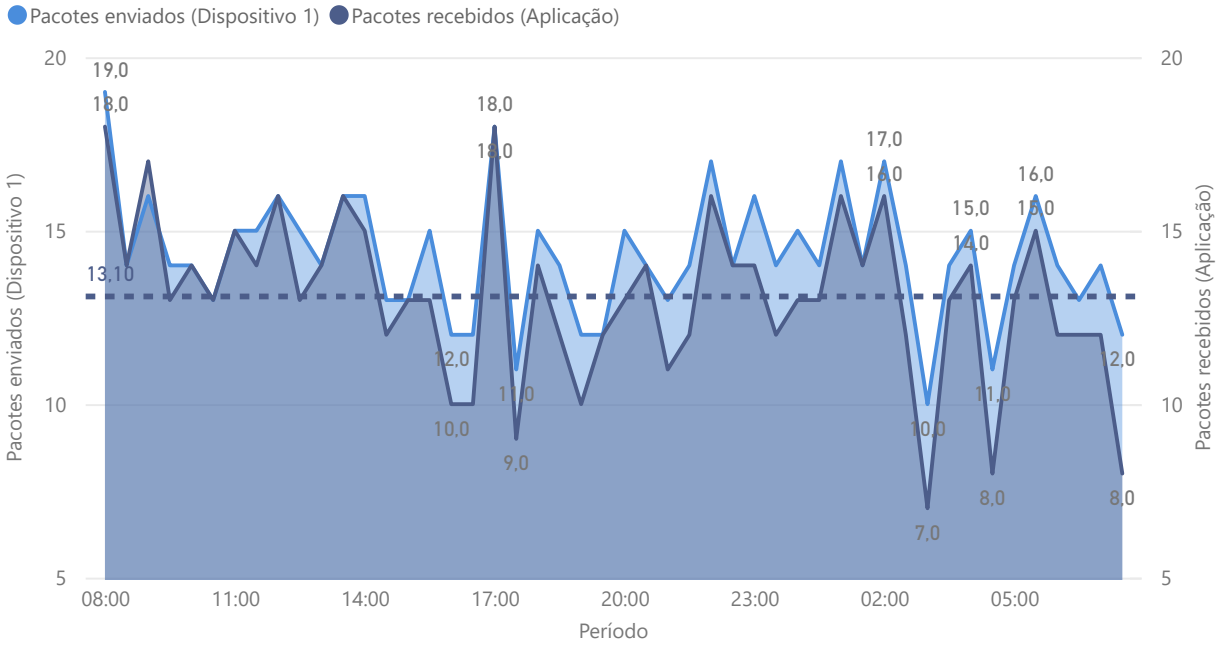
Em termos práticos, o Gráfico 9 compara as solicitações realizadas com as respostas recebidas, estando relacionadas aos pacotes que foram enviados ao medidor inteligente de energia elétrica e recebidos deste. Nota-se que grande parte das requisições realizadas pela aplicação receberam suas respectivas respostas, considerando que algumas estavam fora do padrão esperado. As mensagens do tipo "Error" são categorizadas como respostas que o medidor enviou fora do padrão DLMS/COSEM, sendo sua origem

Gráfico 7 – Relação de *downlinks* entre Aplicação e dispositivo 1.




Fonte: Autor.

Gráfico 8 – Relação de *uplinks* entre Aplicação e dispositivo 1.

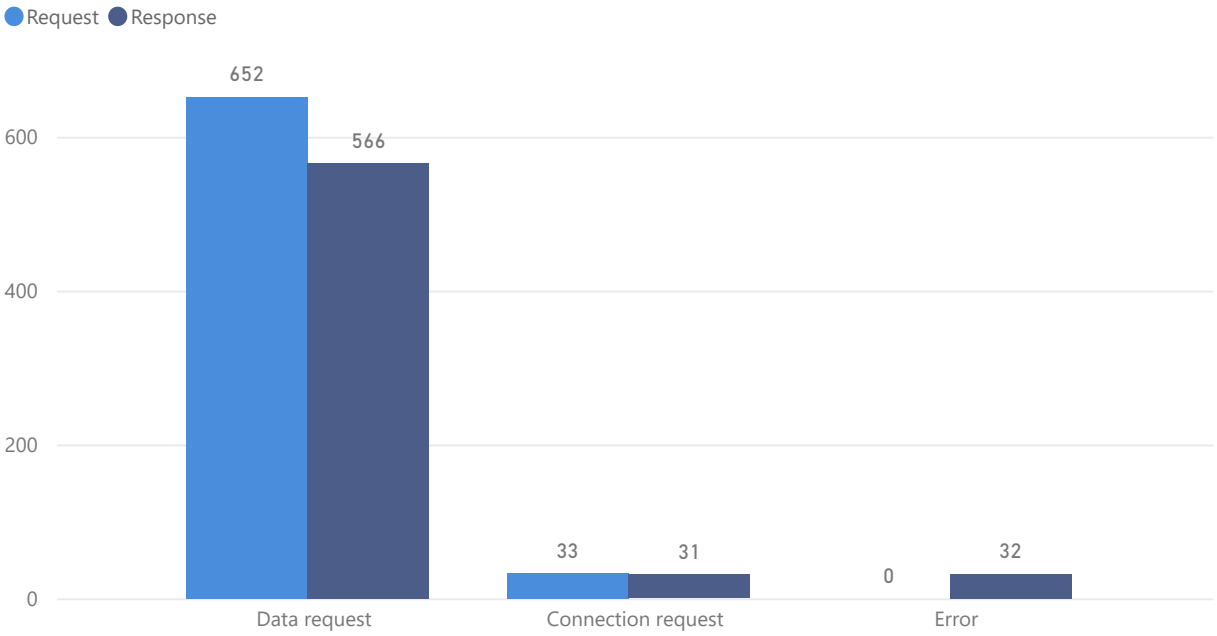



Fonte: Autor.

possivelmente atrelada a problemas pontuais na comunicação serial entre o medidor e o Raspberry. Ainda assim, o número é pequeno, de modo que 687 solicitações foram enviadas e 632 respostas foram recebidas, representando um sucesso de 92%.

Os pacotes enviados pela aplicação através de um *Socket* IPv6/UDP ao *gateway*

Gráfico 9 – Relação de solicitações e respostas entre Aplicação e medidor.

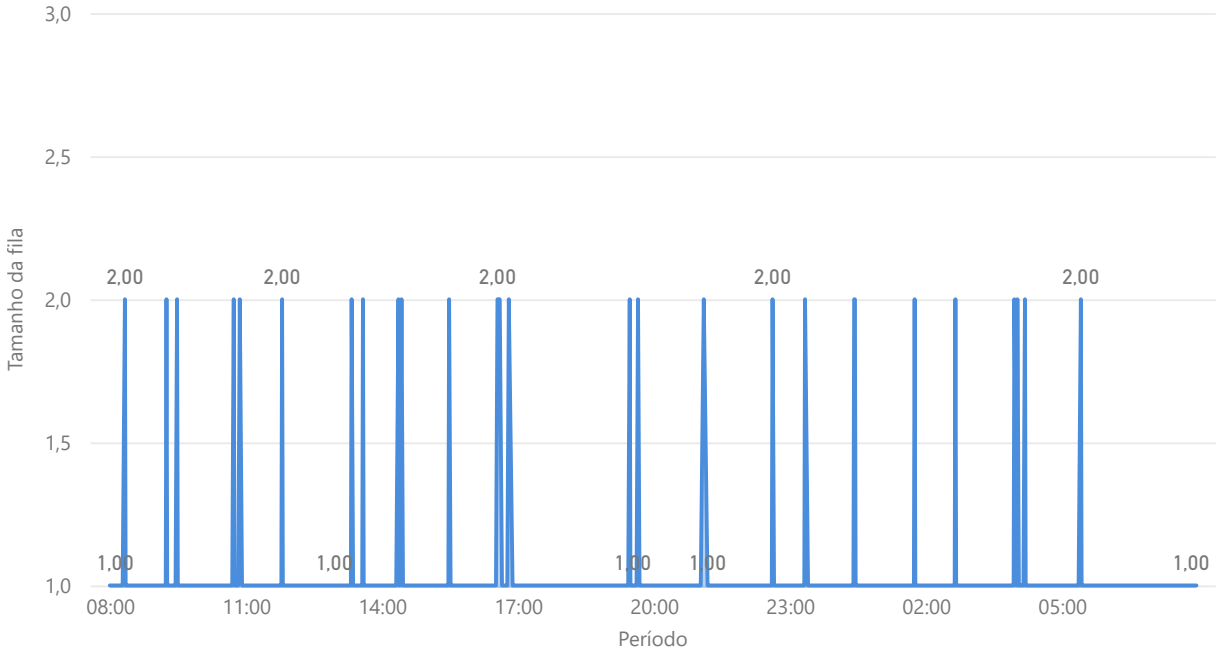



Fonte: Autor.

são armazenados em uma fila FIFO (*First in first out*), de modo que estes são encaminhados para o *handler* processar a transmissão na respectiva FSM. Neste caso, o *Socket* da aplicação conta com um *timeout* de 5 minutos, tendo como intuito atender o tempo de solicitação e resposta (*downlink* e *uplink* respectivamente), que em alguns casos pode ter a soma perto deste limite estipulado. Entretanto esse intervalo pode não ser satisfeito em virtude de problemas na rede LoRaWAN, que acabam atrasando o tráfego dos pacotes. Como visto anteriormente, poucos pacotes acabam perdidos, apesar de que atrasos são possíveis. O Gráfico 10 apresenta a relação da fila ao longo do tempo, sendo o seu valor ideal igual a 1. Valores maiores que este indicam que a rede não está sendo capaz de transmitir os pacotes IPv6/UDP dentro do limite de tempo estipulado, de modo que novas solicitações são enviadas pelas aplicações ao passo que as respostas referentes às solicitações anteriores não são recebidas em tempo. É possível perceber que o tamanho da fila apresenta picos ao longo do tempo, retornando posteriormente ao seu valor ideal.

Além disso, um segundo ponto que colabora com o estouro do *timeout* é a perda de pacotes em *uplink* comentada anteriormente. Esse evento não influencia no valor da fila pois o pacote de *downlink* foi enviado pelo transmissor e recebido com sucesso pelo receptor, porém a resposta referente a esse pacote não chegará à aplicação, provocando o envio de uma nova solicitação após o período estipulado de 5 minutos.

Gráfico 10 – Fila de *downlink* entre *gateway* e dispositivo 1.




Fonte: Autor.

## 4.2 DISPOSITIVO PRODUTOR DE TRÁFEGO ALEATÓRIO

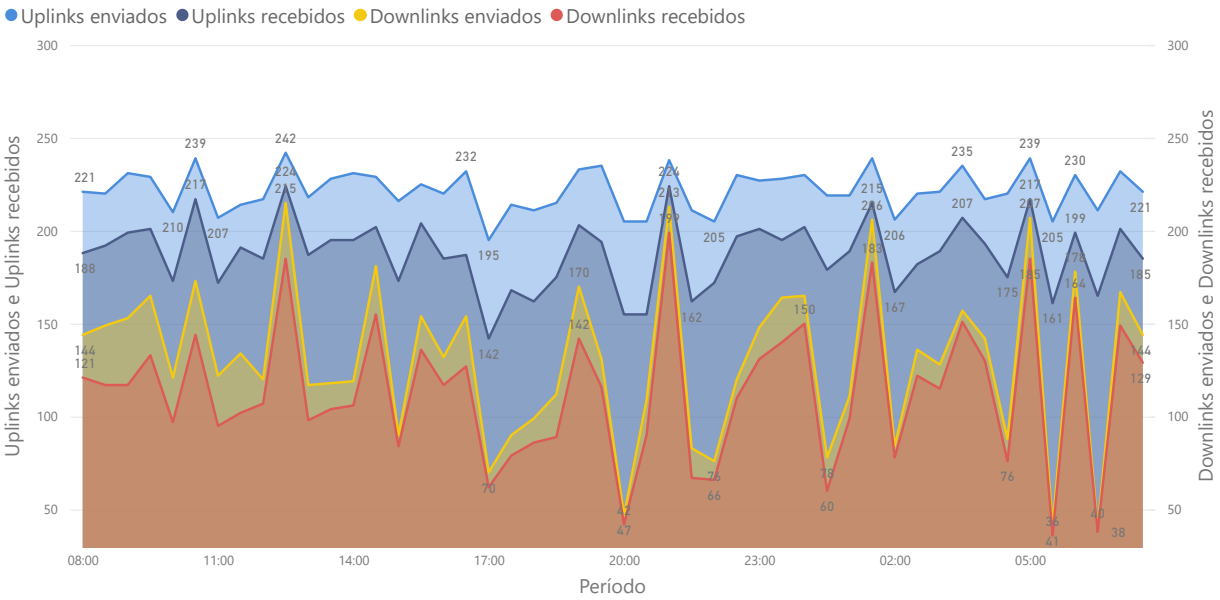
O primeiro dispositivo avaliado possui pouca variação no tamanho e no conteúdo dos *payloads*, não permitindo análises confiáveis da rede justamente devido à alta homogeneidade dos pacotes. Dessa forma optou-se pela condução de um segundo dispositivo simultâneo ao primeiro, com a diferença de que tanto os pacotes de *uplink* quanto os de *downlink* possuem *payload* aleatórios compostos por letras e números, considerando um intervalo de 1 a 512 caracteres (diferente do primeiro dispositivo que possui pequenas quantidades). Além da possibilidade de análises mais fidedignas, isso permite também testar o sistema com mais de um dispositivo rodando simultaneamente. O mecanismo ADR da rede LoRaWAN continuou desligado para os resultados apresentados a seguir, sendo este habilitado nas últimas análises como parâmetro de comparação.

### 4.2.1 Rede LoRaWAN

Novamente com o intuito de apresentar os eventos da rede LoRaWAN ao longo do período com o intuito de melhor embasar a análise do desempenho do SCHC, o Gráfico 11 (com somas em intervalos de 30 minutos) revela maior oscilação da rede em relação ao primeiro dispositivo, sendo que agora as mensagens de *downlink* passaram por maiores períodos de falhas na transmissão. Sugere-se que esse comportamento ocorre devido ao maior tamanho dos pacotes se comparado com o primeiro caso, possibilitando a maior

ocorrência de colisões na faixa de frequências utilizada. De acordo com Aguilar e colaboradores, pacotes menores possuem melhor desempenho quando considerado o tamanho das janelas SCHC. Ao passo que estas aumentam, fazem-se necessárias quantidades maiores de fragmentos por pacote, de modo que a eficiência regride proporcionalmente ao tamanho destes fragmentos (AGUILAR et al., 2020).

Gráfico 11 – Situação da rede LoRaWAN entre Gateway e dispositivo 2.




Fonte: Autor.

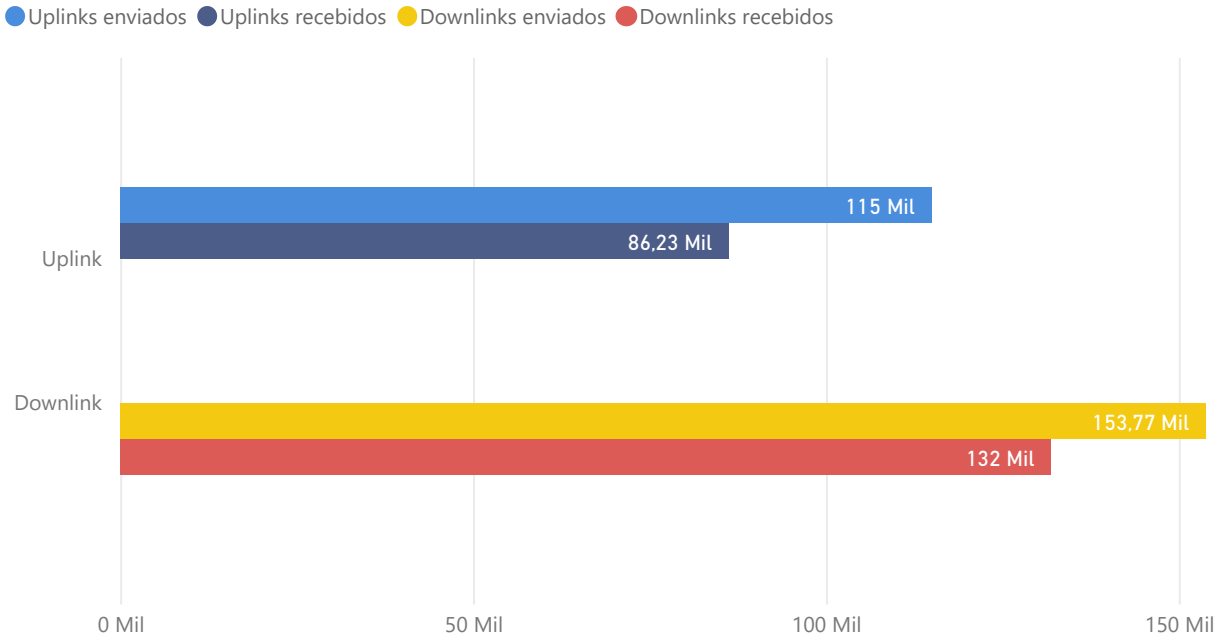
O Gráfico 12 mostra a relação da quantidade de *bytes* transmitidos juntamente com o que foi recebido tanto no dispositivo quanto no *gateway*, tendo uma taxa de falha de 25% para as transmissões de *uplink* e 14% para *downlink*.

### 4.2.2 ACK On Error

O uso de pacotes maiores altera a proporção entre a quantidade de fragmentos Regulares e All-1 trafegados na rede. Nessa situação, praticamente todo o pacote é transmitido por meio de fragmentos Regulares, sendo apenas o último fragmento do tipo All-1 para sinalizar o final da transmissão. O Gráfico 13 mostra justamente essa proporção inversa ao que foi apresentado no primeiro dispositivo no sentido de *uplink*, sendo os fragmentos Regulares representando 70% do volume total enviado. Isso revela o comportamento de uma rede mais próxima da realidade, onde mesmo com tamanhos variáveis, os pacotes maiores possuem boa representatividade.

Além disso, o Gráfico 13 mostra um comportamento diferente quanto à relação entre a quantidade de fragmentos All-1 e de requisições de ACK. No primeiro dispositivo

Gráfico 12 – Pacotes enviados e recebidos entre Gateway e dispositivo 2 em *bytes*.




Fonte: Autor.

essa relação era superior a um para dois, devido à questão da solicitação de *downlink* sobrescrita no *gateway* da rede LoRaWAN. A diferença agora está no fato de que com pacotes maiores, a quantidade de fragmentos enviados é maior e consequentemente as chances de perdas de mensagens também aumenta. O Gráfico 12 mostrou que uma parte considerável das informações enviadas foram perdidas em função de problemas na rede LoRaWAN.

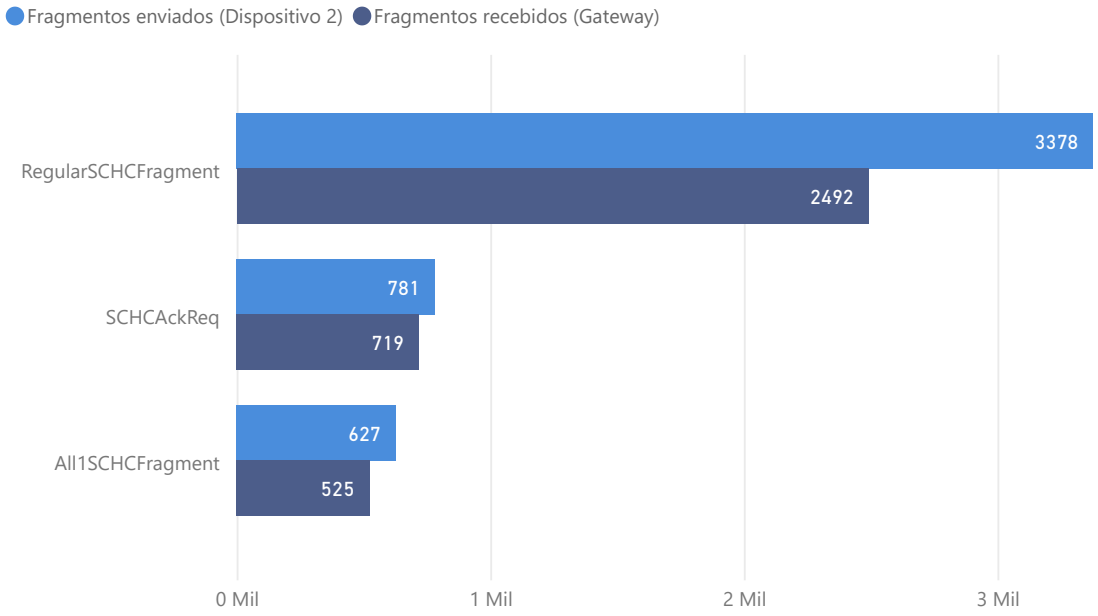
Desta forma, quando o *gateway* recebia solicitações de ACK, em boa parte dos casos ele retornava o *bitmap* dos fragmentos recebidos devido às necessidades de retransmissões, fazendo com que a questão da sobrescrita nos agendamentos de *downlinks* não ocorresse. Além disso, o não recebimento das respostas (enviadas pelo dispositivo) na aplicação provoca o reenvio destas por parte do *gateway*, que no final acaba ficando redundante quanto ao *feedback* que deve ser enviado ao dispositivo.

A confirmação deste fato está na quantidade de ACKs enviados pelo *gateway*, conforme mostra o Gráfico 14, sendo que aproximadamente 12% destes não são de fato recebidos pelo dispositivo.

### 4.2.3 ACK Always

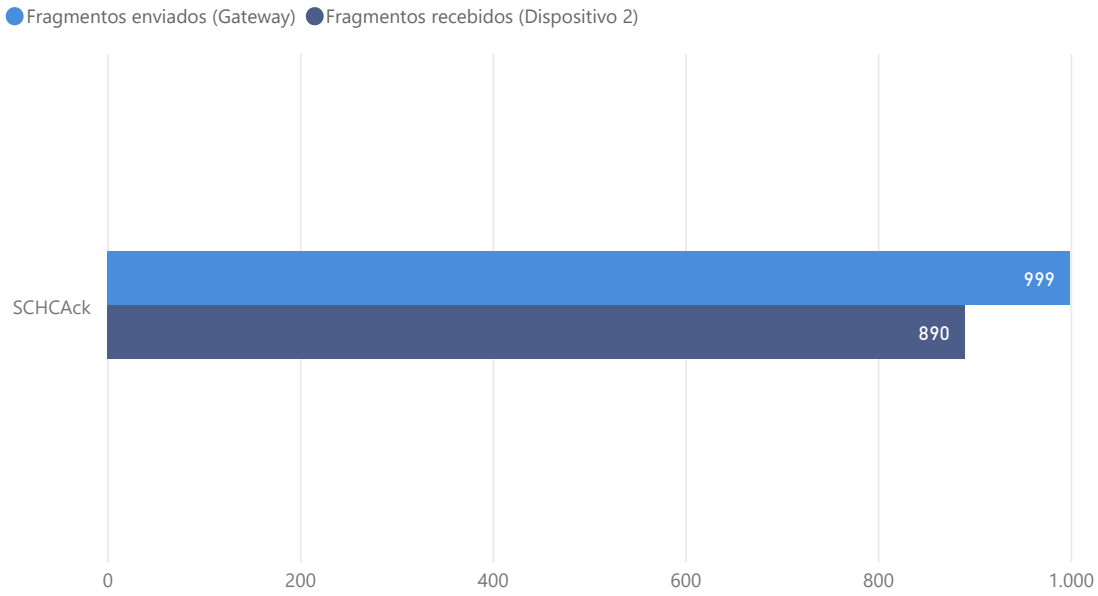
O envio de mensagens maiores em *downlink* provocou também a maior ocorrência de fragmentos Regulares, de modo a dominar as transmissões em relação aos fragmentos All-1, como mostra o Gráfico 15. Em virtude disso, o comportamento percebido no

Gráfico 13 – Fragmentos SCHC em *uplink* no modo ACK *On Error* entre dispositivo 2 e Gateway.




Fonte: Autor.

Gráfico 14 – Fragmentos SCHC em *downlink* no modo ACK *On Error* entre dispositivo 2 e Gateway.

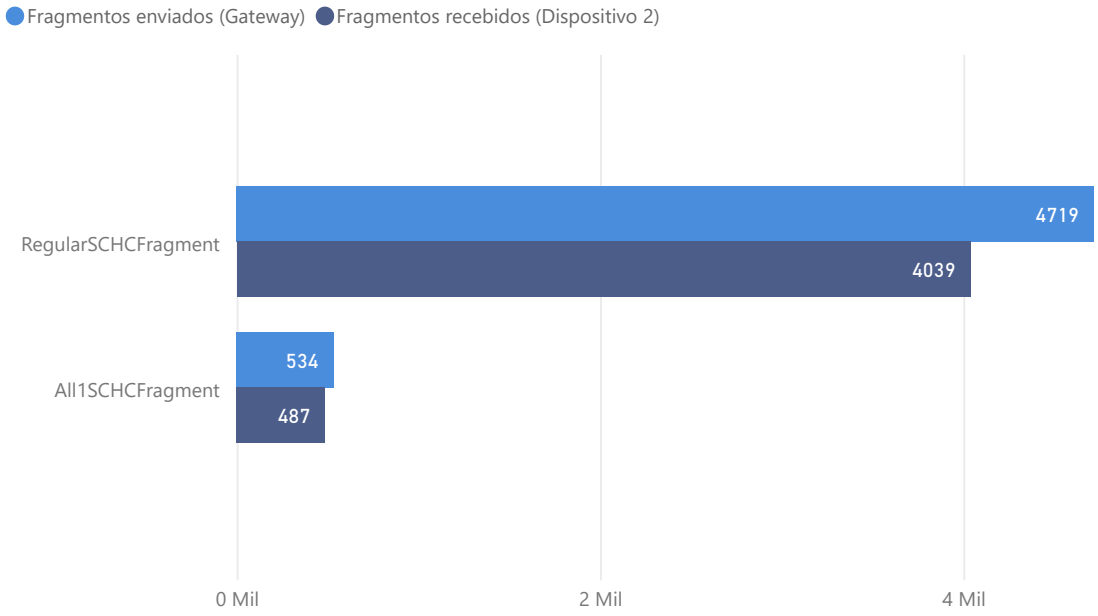



Fonte: Autor.

primeiro dispositivo com uma quantidade maior de fragmentos All-1 não se repetiu justamente devido a sua baixa proporção, reduzindo também os casos de falha para esse tipo de fragmentos.

A soma de fragmentos enviados pelo *gateway* e recebidos pelo dispositivo é 4553, sendo que cada evento destes gera o disparo de um ACK por parte do dispositivo. Em contrapartida, o Gráfico 16 mostra que foram enviados 5762 ACKs, representando 26% a mais do que o mínimo necessário. Boa parte destas confirmações não chegaram ao *gateway*, fazendo com que o dispositivo as reenviasse, provocando com isso atrasos na

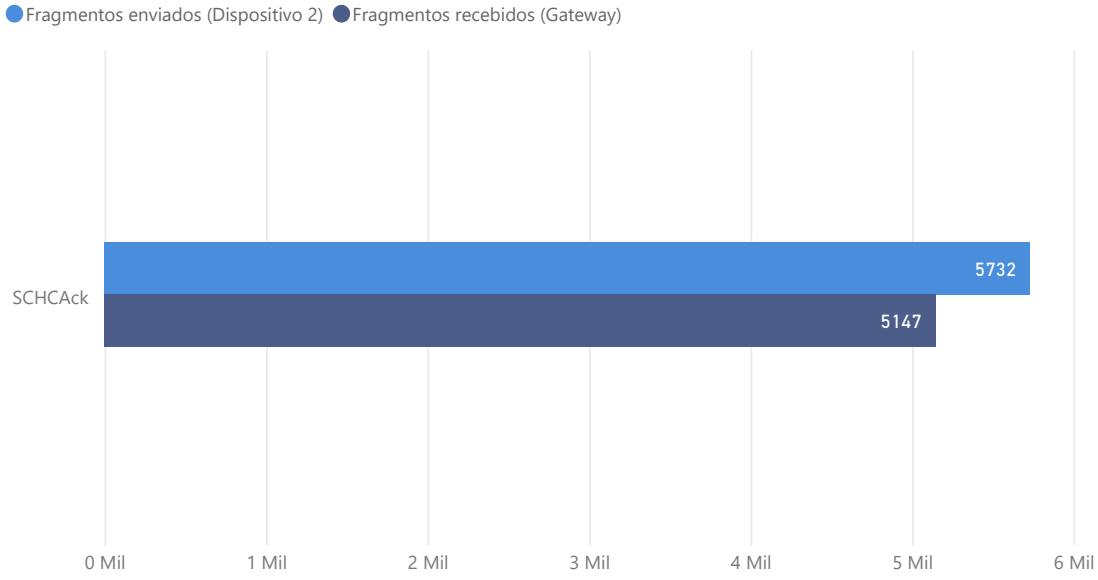
Gráfico 15 – Fragmentos SCHC em *uplink* no modo ACK *Always* entre dispositivo 2 e Gateway.




Fonte: Autor.

transmissão dos pacotes.

Gráfico 16 – Fragmentos SCHC em *downlink* no modo ACK *Always* entre Gateway e dispositivo 2.

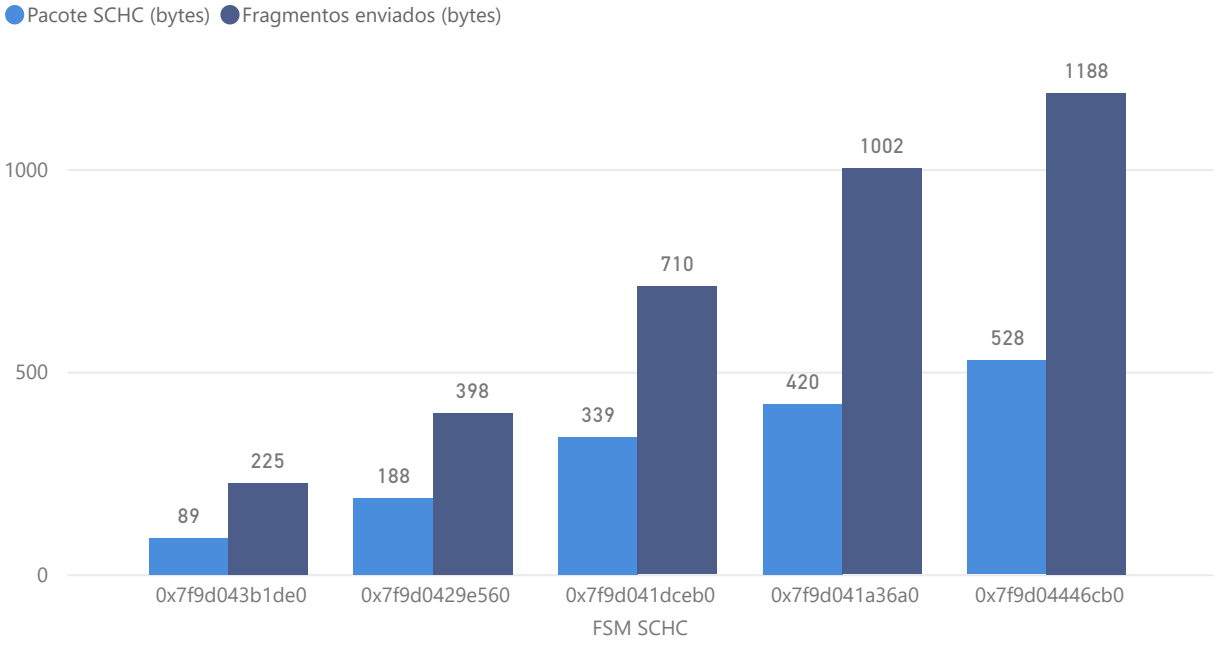



Fonte: Autor.

Não obstante, o uso de pacotes IPv6/UDP com *payloads* maiores e variáveis permitiu uma melhor análise quanto a um evento comentado anteriormente: o envio duplicado de fragmentos no modo ACK *Always*. Para demonstrar isso, foram escolhidas dentro do intervalo de testes, algumas máquinas de estado utilizadas para o envio de pacotes em *downlink*. A escolha destas foi baseada nos valores obtidos nos dados dos testes, de modo a considerar amostras que representassem a evolução entre os valores mínimos

e máximos referentes aos tamanhos dos pacotes em *bytes*. Representadas pelo endereço alocado em memória pelo *gateway*, as máquinas de estado podem ser observadas no Gráfico 17 juntamente com a comparação entre o tamanho original do pacote SCHC e o volume de *bytes* transmitidos na rede LoRaWAN. Optou-se pela representação relacionada ao endereço de memória da máquina pois um mesmo pacote conta com mais de um fragmento enviado na rede LoRaWAN, estando sempre relacionado à mesma FSM. Ainda no gráfico é possível observar que, para cada pacote, pelo menos o dobro do seu tamanho em *bytes* foi trafegado na rede. O restante da diferença pode estar atrelado às retransmissões que foram realizadas em virtude de perda de mensagens.

Gráfico 17 – Comparação de *bytes* entre pacotes SCHC selecionados e seus volumes trafegados em *downlink* entre *gateway* e dispositivo 2.



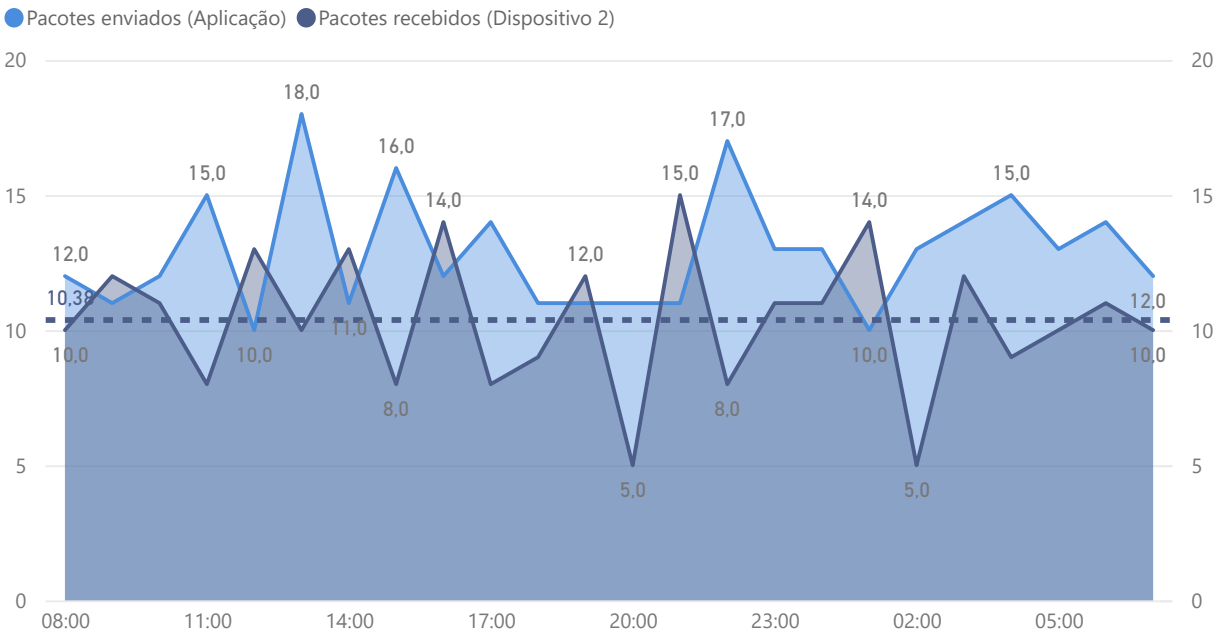

Fonte: Autor.

Esse problema ocorre devido à relação entre a necessidade de confirmações constantes do modo ACK *Always* e o agendamento dos mensagens de *downlink* na rede LoRaWAN. Assim que o receptor valida o fragmento recebido, este envia um ACK ao *gateway* e na sequência abre uma janela de *downlink*. O *gateway* realiza o agendamento da transmissão de um novo fragmento nesse intervalo, porém ele não é recebido pelo receptor nessa janela que foi aberta. Assim, o dispositivo precisa abrir uma nova janela de *uplink* (enviando novamente o ACK) para receber o novo fragmento, de modo que o *gateway* realizará novamente o agendamento daquele mesmo fragmento e com isso sempre duplicará os seus envios. Esse problema não apenas gera tráfego desnecessário na rede LoRaWAN como também aumenta o tempo de envio de *downlink* dos pacotes IPv6/UDP.

### 4.2.4 Análise da aplicação

As oscilações na rede LoRaWAN apresentadas no início da seção provocaram o atraso na comunicação entre a aplicação e o dispositivo, de modo a interferir também na entrega tanto dos pacotes de *uplink* quanto de *downlink*. O Gráfico 18 (com somas em intervalos de 30 minutos) mostra que a oscilação foi maior se comparada ao primeiro dispositivo, tendo o dispositivo recebido menos pacotes do que o enviado durante o período analisado. Essa diferença é de uma média de 14 pacotes no primeiro dispositivo contra 10 por hora no segundo considerando o período inteiro.

Gráfico 18 – Relação de *downlinks* entre Aplicação e dispositivo 2.

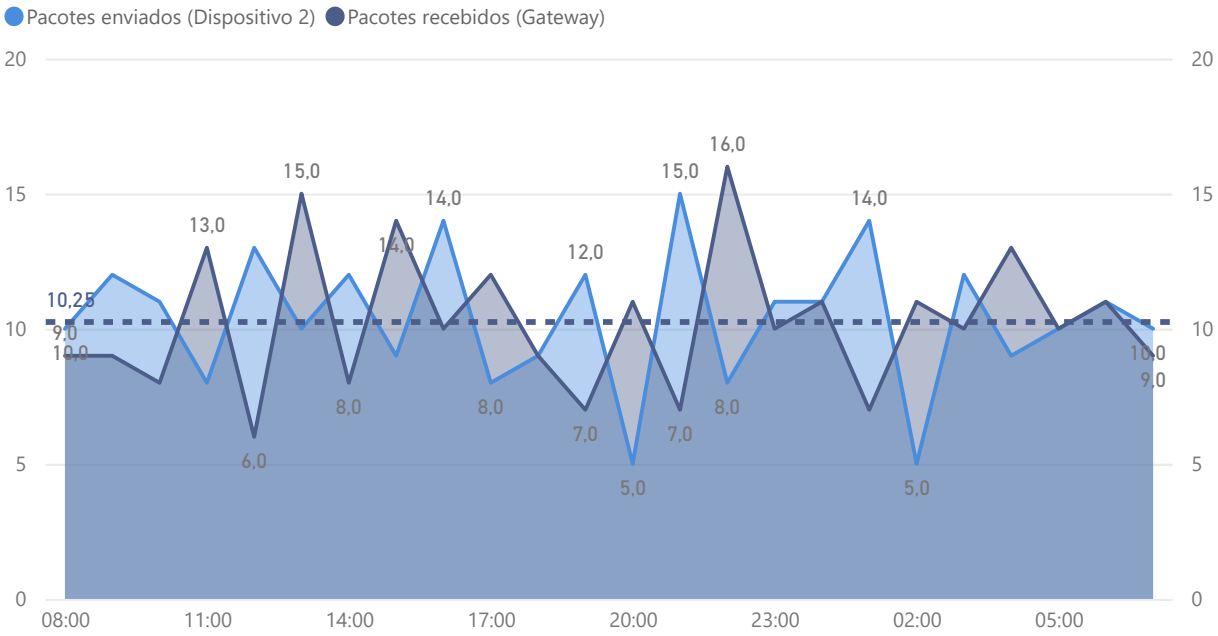



Fonte: Autor.

O Gráfico 19 (também com somas em intervalos de 30 minutos) mostra que as as perdas de mensagens LoRa também trouxeram resultados negativos para os pacotes de *uplink*, mostrando que a dominância nas quantidades oscilava ao longo do tempo entre os pacotes enviados e os recebidos. Em uma situação normal, é esperado que o primeiro número seja sempre superior ao segundo, considerando nisso possíveis perdas. Entretanto o comportamento apresentado mostra que o dispositivo estava enviando pacotes com respostas atrasadas, referentes a solicitações que muito provavelmente já estavam com o *timeout* de 10 minutos atingido na aplicação, que consequentemente enviava uma nova solicitação na sequência.

Ao analisar essa situação nos dados gerados pelo *gateway*, é possível observar a partir do Gráfico 20 que a fila de *downlinks* foi crescendo ao longo do tempo, tendo dificuldades para normalizar embora houvessem pontos de queda. A combinação das perdas de mensagens na rede LoRaWAN juntamente com o envio de pacotes grandes nos dois

Gráfico 19 – Relação de *uplinks* entre Aplicação e dispositivo 2.

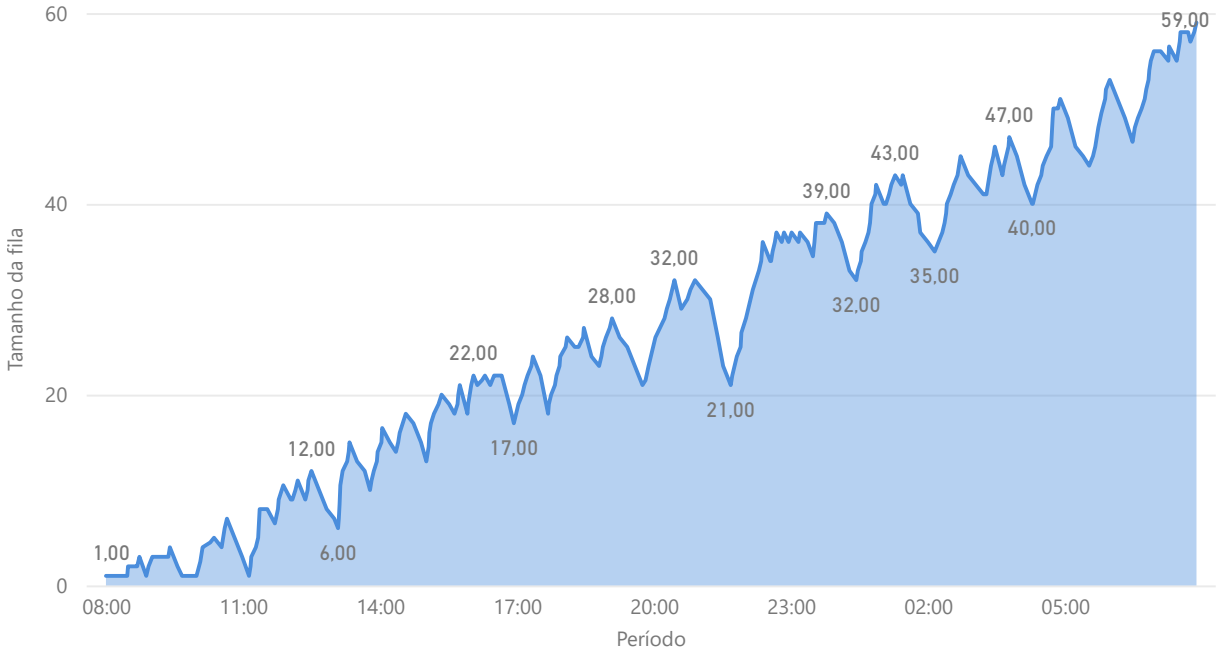



Fonte: Autor.

sentidos da comunicação fez com que o sistema não vencesse a demanda, criando com isso um constante acúmulo de requisições que provocou no aumento também constante da fila. Um segundo ponto observado foi o fato de que a atual implementação concedeu a preferência para os pacotes IPv6/UDP de *downlink*, de modo que ao final da transmissão de um, a transmissão do próximo pacote da fila já era inicializada logo na sequência. Isso se deve ao fato de que, ao final de uma transmissão de *downlink*, o *gateway* já pode agendar o envio de um novo pacote caso haja algum aguardando. Como consequência, os pacotes de *uplink* enviados pelo dispositivo não possuíam oportunidade na rede, acarretando no seu acúmulo para envio e também no estouro do *timeout* na aplicação por conta da ausência de respostas no *Socket*. As possibilidades de andamento da fila de *downlinks* aconteciam nos momentos em que os níveis de transmissão da rede LoRaWAN apresentavam melhoras, juntamente com o agendamento de pacotes com tamanhos menores que consequentemente necessitavam de menor tempo para envio. Entretanto essas oportunidades não foram suficientes para contribuir no andamento da fila.

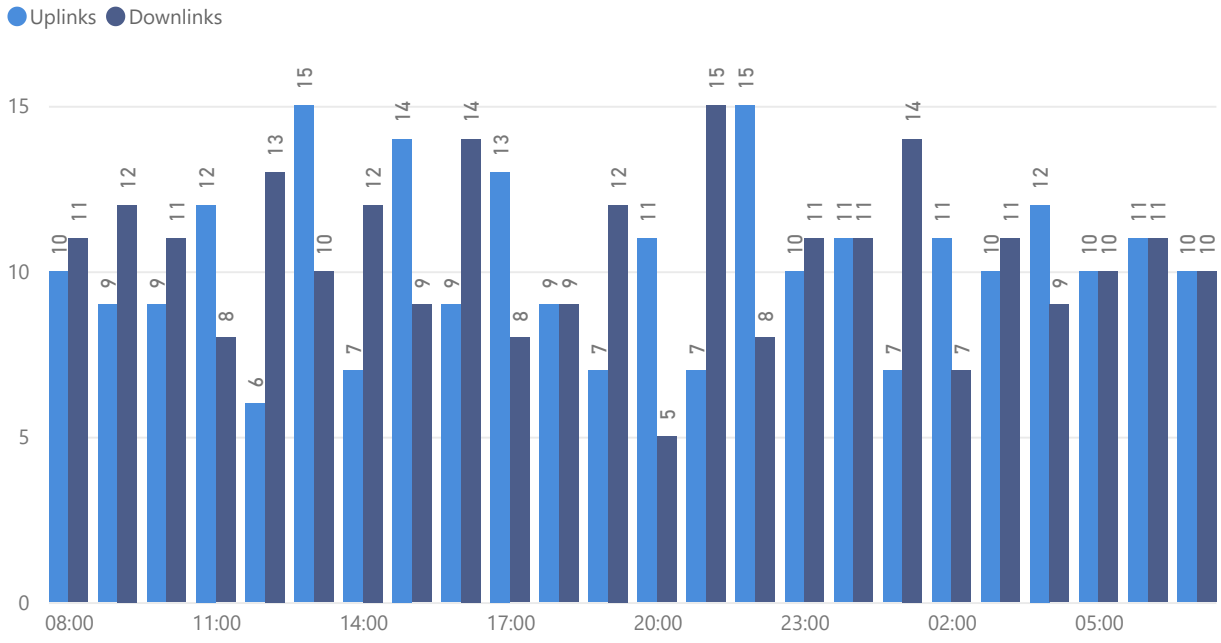
Como resultado disso, o Gráfico 21 (com somas em intervalos de 1 hora) apresenta uma segunda visão da relação de *uplinks* recebidos pelo *gateway* e *downlinks* recebidos pelo dispositivo, referentes a pacotes IPv6/UDP ao longo do período considerado. O gráfico está dividido em intervalos de uma hora, sendo que as discrepâncias entre um intervalo e outro dentro dos dois sentidos de comunicação confirma os atrasos ocorridos devido aos problemas apresentados.

Gráfico 20 – Fila de *downlink* entre *gateway* e dispositivo 2.




Fonte: Autor.

Gráfico 21 – Resumo da comunicação entre aplicação e dispositivo 2 ao longo do período.



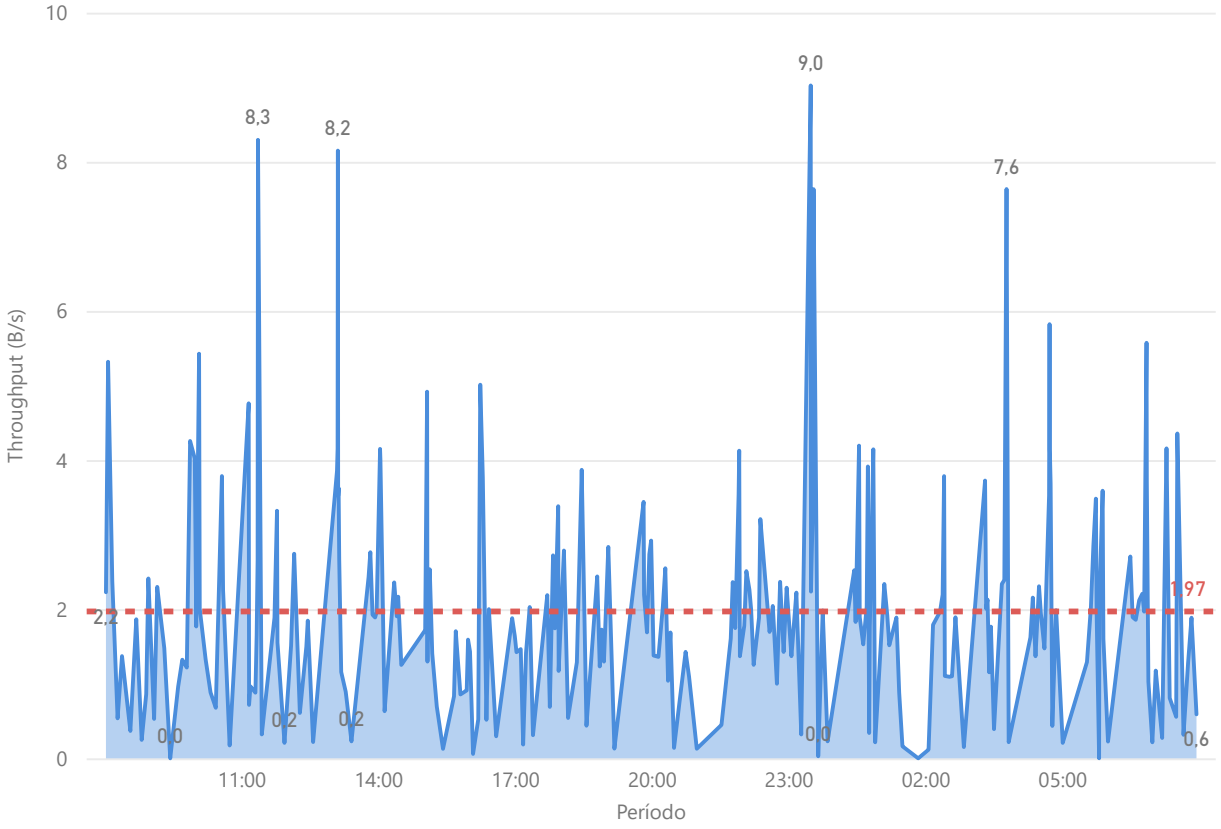

Fonte: Autor.

### 4.2.5 Desempenho da rede

A implementação do segundo dispositivo possibilita também uma análise mais profunda do trabalho com um ponto de vista da rede como um todo, de modo que a maior variabilidade nos pacotes transmitidos permite a observação de comportamentos que se

apresentam como consequência dos processamentos dos pacotes ao longo do sistema. A partir de informações como os momentos em que os pacotes são enviados pela aplicação e os momentos de suas chegadas ao dispositivo (já passado pelo processo de remontagem do SCHC), torna-se possível o cálculo do *Throughput* e suas variações ao longo do tempo. O primeiro resumo a ser considerado é o *Throughput* no sentido *uplink*, apresentado pelo Gráfico 22. O cálculo considera a média das divisões dos tamanhos totais dos pacotes IPv6/UDP em *bytes* pelos tempos gastos nas transmissões, revelando que houveram várias oscilações ao longo do período. Considerando o período de 24 horas com pontos de intervalo de 1 segundo, tem-se que o *Throughput* médio foi de 1.97 B/s, com picos chegando em torno de 9 B/s.

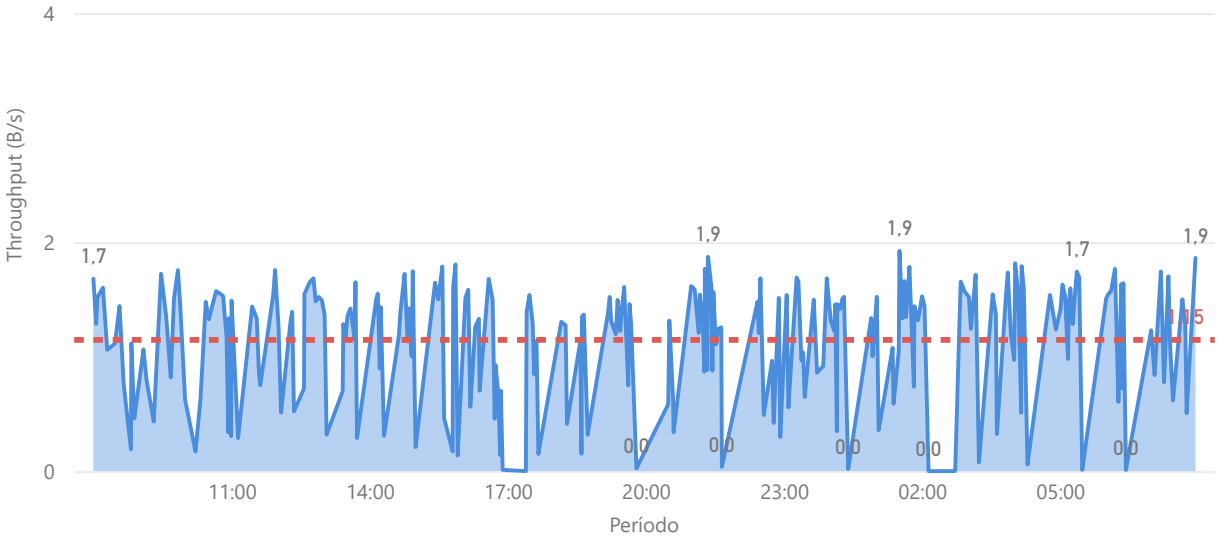
Gráfico 22 – Throughput *uplink* do dispositivo 2.




Fonte: Autor.

Por outro lado, o *Throughput* no sentido *downlink* apresentou oscilações menores, apesar de que sua média foi inferior. O Gráfico 23 mostra que o *Throughput* médio foi de 1.15 B/s, o qual pode ainda se tornar consideravelmente maior com a solução do problema do envio de fragmentos duplicados. Ressalta-se também que em uma rede LoRaWAN, o envio de *downlinks* é mais custoso para os rádios, de modo que este custo relacionado à estrutura acaba influenciando diretamente o custo monetário. Em um caso real, a comunicação em *downlink* pode ser utilizada por exemplo na atualização de *firmware*, o que exige um *link* confiável devido à importância dos pacotes e robusto devido ao tamanho destes.

Gráfico 23 – Throughput *downlink* do dispositivo 2.




Fonte: Autor.

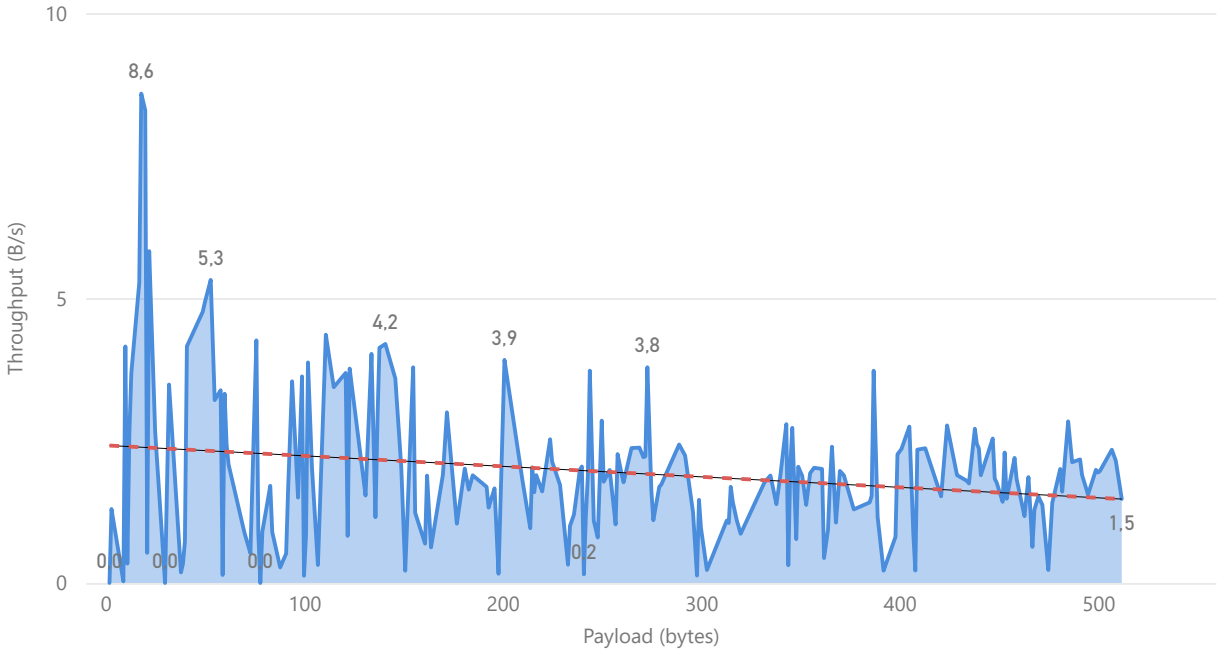
Ao longo destes experimentos, realizou-se também a análise dos *Throughputs* em relação ao tamanho dos pacotes IPv6/UDP em *bytes*. O Gráfico 24 indica que no sentido de *uplink*, o melhor aproveitamento da rede ocorreu com os pacotes menores, indo de encontro com o modo ACK *On Error* utilizado que realiza inicialmente o disparo de todos os fragmentos e por fim solicita o ACK. Conforme mais fragmentos do mesmo pacote são transmitidos nessa rajada, a probabilidade de mensagens perdidas ao longo do caminho aumenta, provocando a necessidade de retransmissão das partes perdidas e consequentemente aumentando o tempo utilizado.

Em contrapartida, o Gráfico 25 aponta que a rede possui tendência de aumento no desempenho do sentido de *downlink* conforme o tamanho dos pacotes IPv6/UDP aumenta. Assim como nos demais experimentos, foi utilizado um MTU de 32 *bytes* com o intuito de gerar vários fragmentos na comunicação com o medidor inteligente de energia elétrica. Acredita-se que caso o MTU máximo da rede LoRaWAN (51 *bytes*) fosse utilizado, o desempenho seria ainda maior.

### 4.2.6 Custos de tráfego

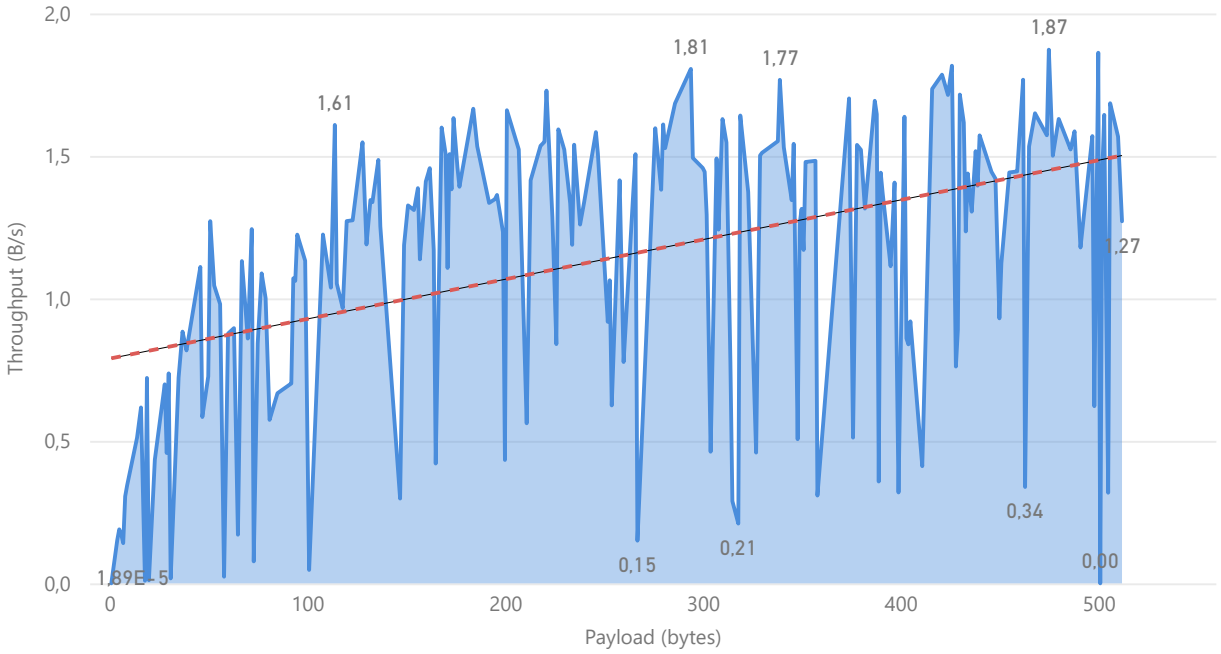
Ao passo que os pacotes IPv6/UDP são transformados em pacotes SCHC e posteriormente transmitidos por fragmentos na rede LoRaWAN, tem-se as relações dos custos envolvidos entre essas etapas. Como visto no início do capítulo, a compressão dos cabeçalhos IPv6/UDP tende a seguir uma tendência constante a partir das definições deste trabalho, sendo que o restante do pacote SCHC acompanha a variação do tamanho do *payload* inserido. Com isso, parte considerável dos custos ficam em torno da implementa-

Gráfico 24 – Comparação entre Throughput *downlink* e tamanhos dos pacotes IPv6/UDP no dispositivo 2.




Fonte: Autor.

Gráfico 25 – Comparação entre Throughput *uplink* e tamanhos dos pacotes IPv6/UDP no dispositivo 2.

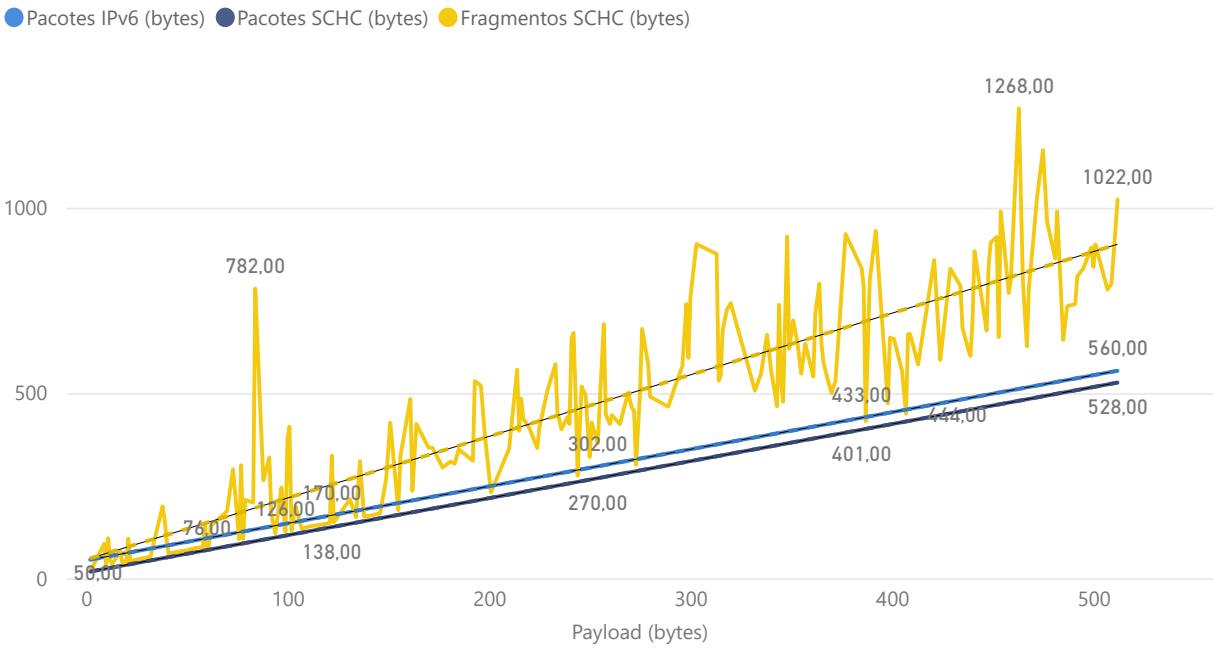



Fonte: Autor.

ção do protocolo SCHC quanto a sua eficiência na condução dos fragmentos, juntamente com a qualidade da rede LoRaWAN no momento das transmissões. O Gráfico 26 mostra que no sentido de *uplink*, a proporção em termos de custo em *bytes* entre os pacotes IPv6/UDP e os pacotes SCHC se mantinha constante conforme o tamanho destes aumen-

tava, já considerando o processo de compressão e consequentemente menos informações trafegadas. Por outro lado, a comparação com a quantidade de *bytes* enviados oscilava conforme os fragmentos chegavam ou não ao seu destino. É possível perceber momentos em que a proporção entre os três parâmetros ficou muito próxima, ocorridas quando a taxa de sucesso da rede LoRaWAN estava alta. Entretanto, situações mais custosas chegaram a exigir a transmissão de volumes superiores ao dobro do pacote SCHC em questão.

Gráfico 26 – Comparações de custos de *uplink* do dispositivo 2.



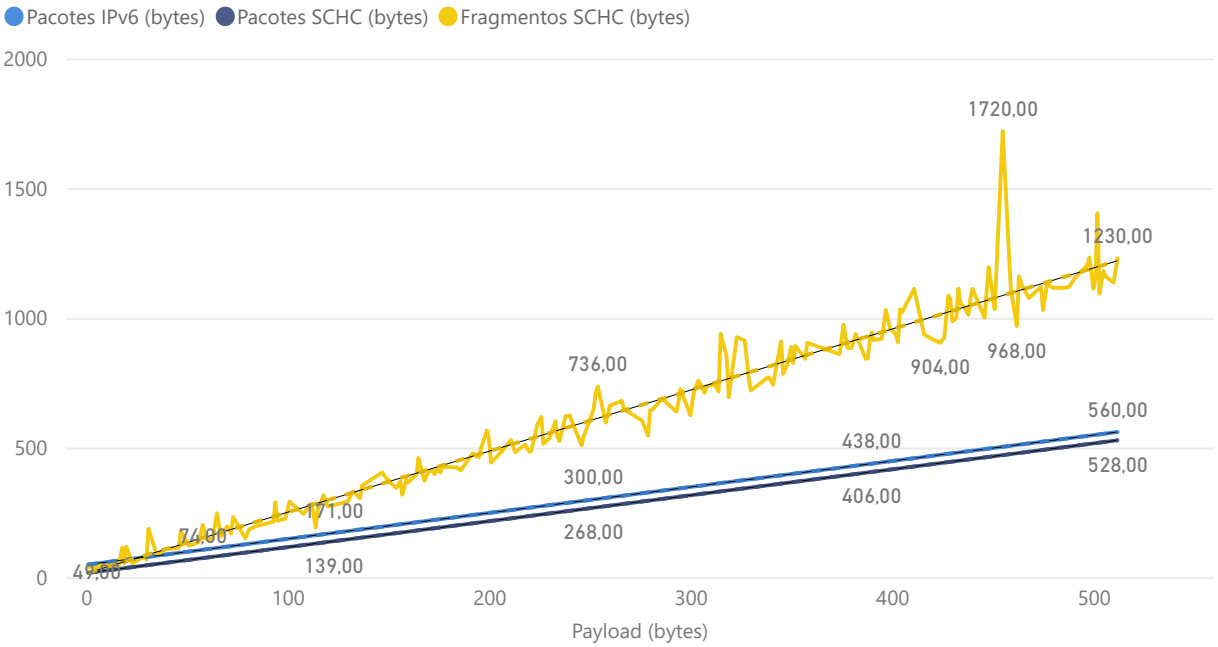

Fonte: Autor.

No sentido oposto da comunicação (*downlink*), já era esperado o dobro do custo de transmissão devido ao envio duplicado dos pacotes. O Gráfico 27 confirma essa hipótese, de modo que as variações ocorridas se devem aos fragmentos que foram eventualmente perdidos. Mais uma vez, percebe-se que a oscilação é menor de forma geral se comparado ao *uplink*, fato que ocorre justamente pelo privilégio que as transmissões de *downlink* possuem na rede LoRaWAN devido às janelas de recepção estarem sincronizadas com o *gateway* da rede, conforme citado anteriormente.

### 4.2.7 Comparações com o uso do mecanismo ADR

Os resultados apresentados aqui contam com o mecanismo ADR da rede LoRaWAN desabilitado, provocando a perda de fragmentos SCHC pois a rede não estava otimizada. Com o intuito de apresentar um comparativo com menores taxas de falhas nas transmissões da rede LoRaWAN, o mecanismo ADR foi habilitado, sendo possível observar a partir

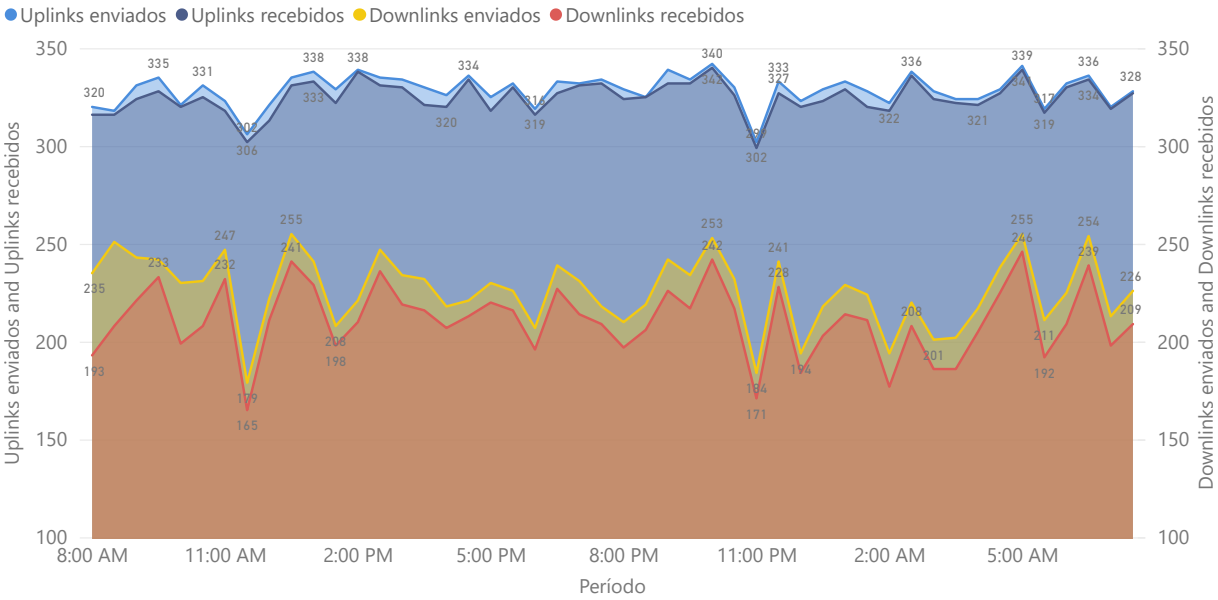
Gráfico 27 – Comparações de custos de *downlink* do dispositivo 2.




Fonte: Autor.

do Gráfico 28 que as perdas na rede foram menores se comparado às situações anteriores.

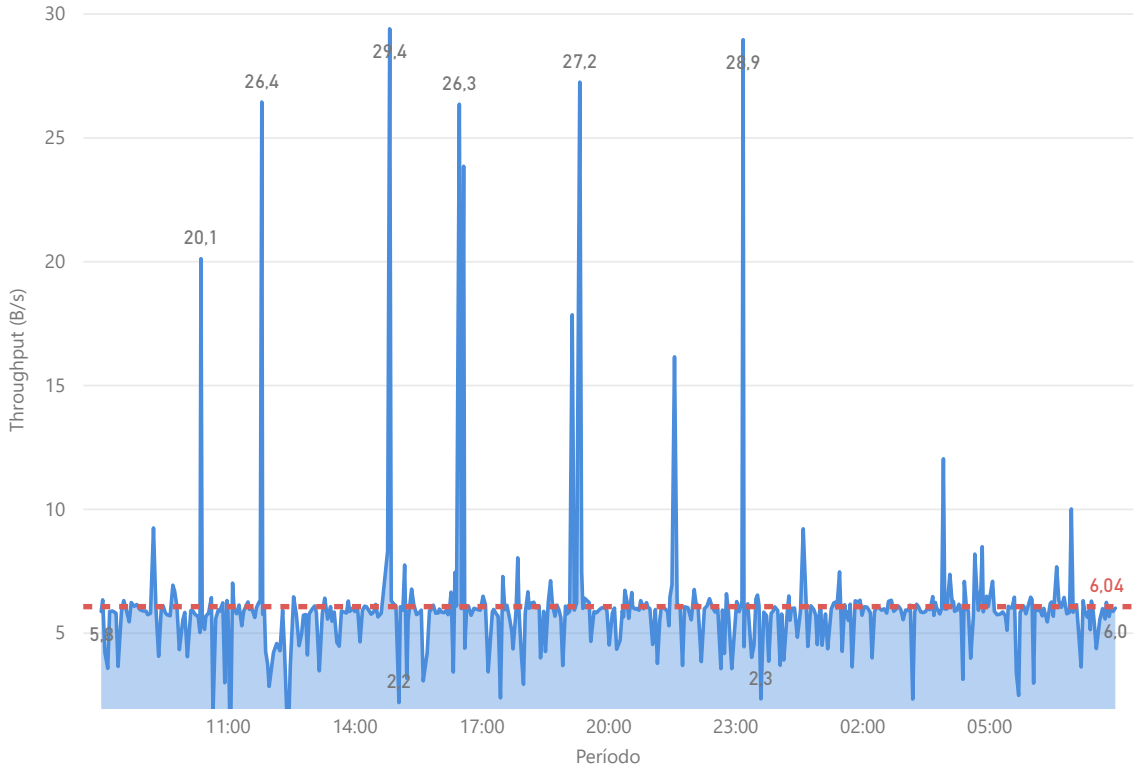
Gráfico 28 – Situação da rede LoRaWAN entre Gateway e dispositivo 2 com ADR.




Fonte: Autor.

O primeiro ponto a ser comparado é o *Throughput* no sentido de *uplink*, que apresentou a média de 6.04 B/s em comparação aos 1.97 B/s anteriores, com picos de mais de 29 B/s conforme indica o Gráfico 29. Com a rede otimizada, menos fragmentos são perdidos e assim menos tempo é necessário para a transmissão dos pacotes IPv6/UDP.

Gráfico 29 – Throughput *uplink* do dispositivo 2 com ADR.




Fonte: Autor.

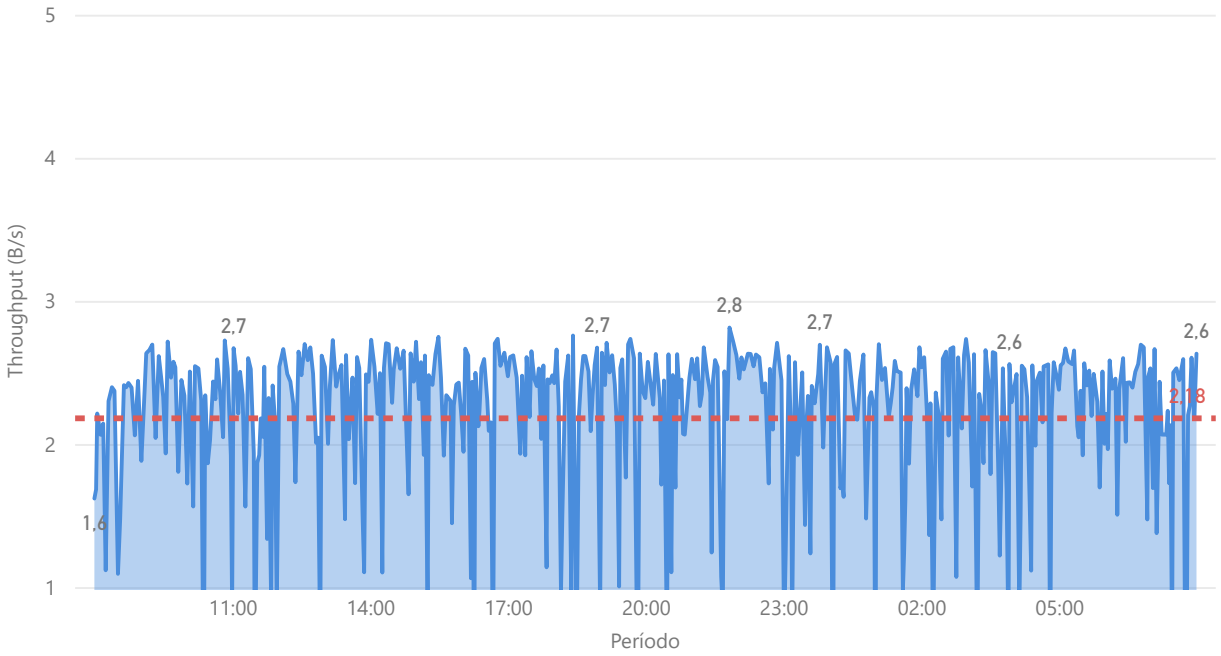
Paralelo a isso, o Gráfico 30 mostra que o *Throughput* médio no sentido de *downlink* foi de 2.18 B/s contra 1.15 B/s no primeiro teste. As oscilações neste caso continuaram pequenas, entretanto percebe-se que os valores ainda são baixos devido à questão do envio duplicado dos fragmentos. Além disso, todos os testes utilizaram um *delay* de 5 segundos na janela de *downlink* do LoRaWAN, sendo sua redução passível de análise.

Um outro ponto interessante a ser avaliado é o custo das transmissões. Diferente das grandes oscilações percebidas anteriormente, agora a quantidade de *bytes* trafegados na rede ficou mais próxima dos tamanhos dos pacotes, tanto para o sentido de *uplink* quanto de *downlink* conforme mostram os Gráficos 31 e 32, respectivamente. No último caso, o volume transmitido continua sendo o dobro do que seria realmente necessário e ideal, contendo algumas oscilações pontuais.

A partir destes dois gráficos, foram coletadas amostras com 6 pontos de cada um para calcular a diferença entre os tamanhos originais dos pacotes IPv6/UDP e o volume que foi trafegado na rede LoRaWAN em *bytes*. As Tabelas 1 e 2 mostram que os custos de transmissão por pacote IPv6/UDP foram proporcionais ao seu tamanho.

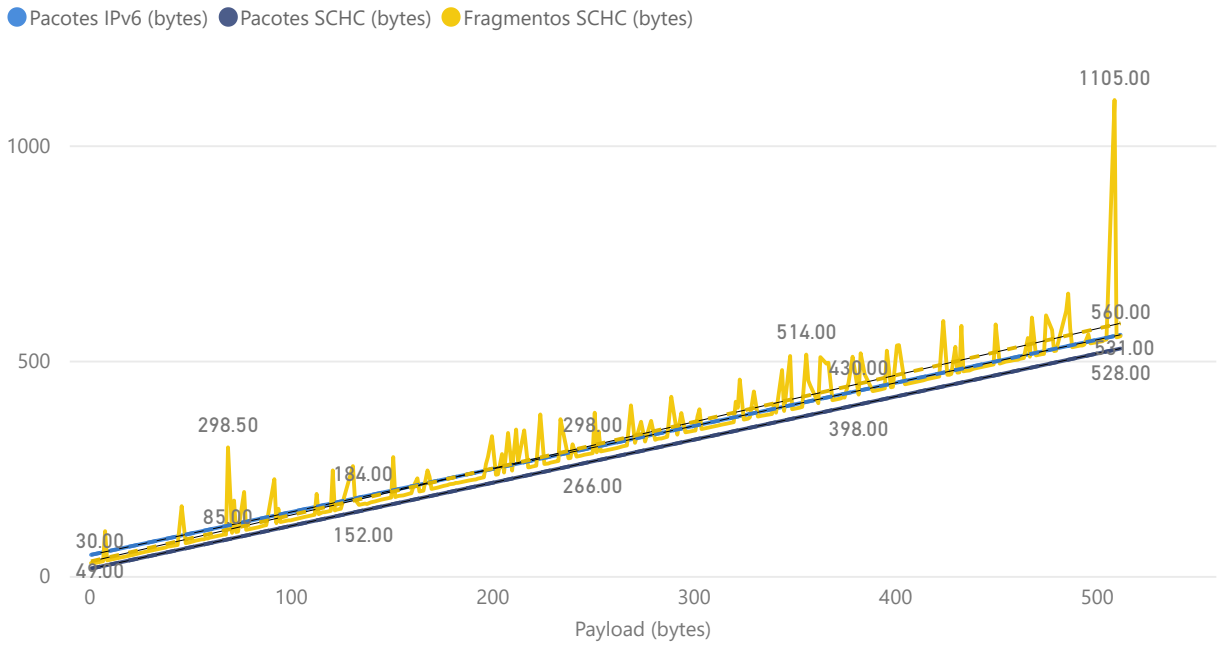
Por fim, percebeu-se que o *gateway* manteve o tamanho da fila de *downlink* no seu valor ideal durante todo o período considerado. Isso indica que, apesar de pequenas oscilações, a rede como um todo foi capaz de processar os pacotes sem grandes atrasos.

Gráfico 30 – Throughput *downlink* do dispositivo 2 com ADR.



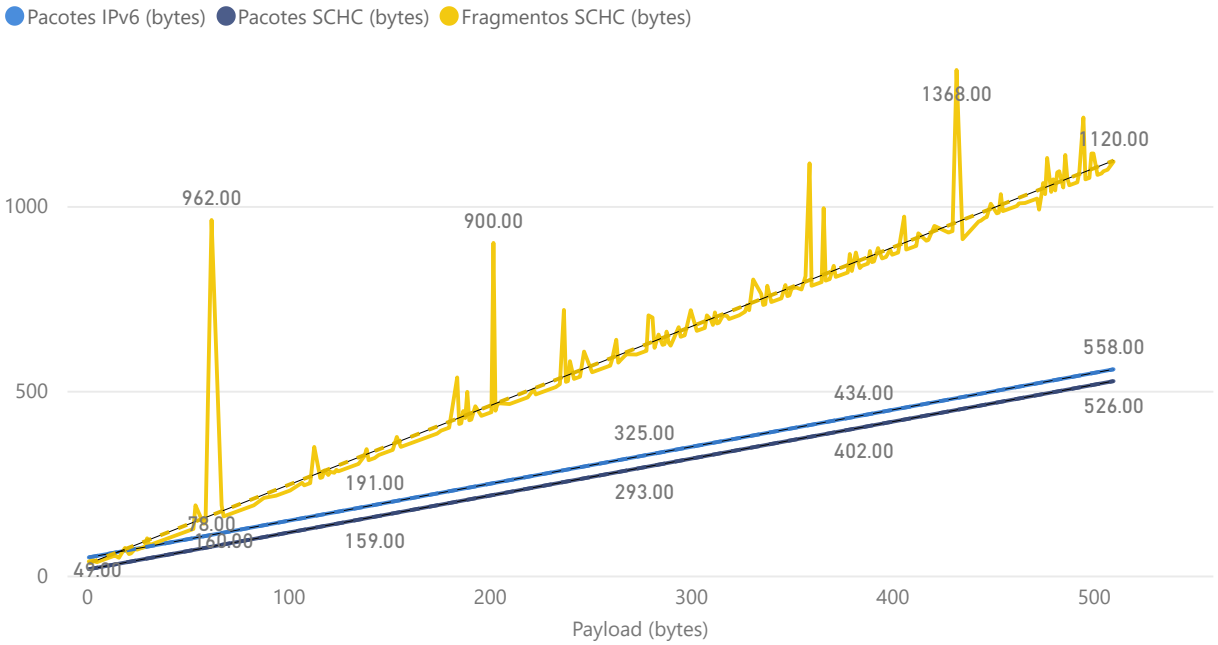

Fonte: Autor.

Gráfico 31 – Comparações de custos de *uplink* do dispositivo 2 com ADR.




Fonte: Autor.

Gráfico 32 – Comparações de custos de *downlink* do dispositivo 2 com ADR.




Fonte: Autor.

Tabela 1 – Exemplos dos custos efetivos do tráfego *uplink* em *bytes*.

| IPv6/UDP | SCHC | LoRaWAN | Variação (%) |
|---|---|---|---|
| 86 | 54 | 66 | -23 |
| 165 | 133 | 148 | -10 |
| 277 | 245 | 263 | -5 |
| 360 | 328 | 349 | -3 |
| 462 | 430 | 454 | -1 |
| 551 | 519 | 545 | -1 |

Fonte: Autor.

Tabela 2 – Exemplos dos custos efetivos do tráfego *downlink* em *bytes*.

| IPv6/UDP | SCHC | LoRaWAN | Variação (%) |
|---|---|---|---|
| 85 | 53 | 94 | +10 |
| 193 | 161 | 326 | +68 |
| 290 | 258 | 532 | +83 |
| 368 | 336 | 696 | +89 |
| 448 | 416 | 868 | +93 |
| 537 | 505 | 1058 | +97 |

Fonte: Autor.

# 5 PERSPECTIVAS

O trabalho desenvolvido abriu caminho para diversas possíveis melhorias que ainda podem ser implementadas em alguns pontos percebidos ao longo da discussão dos resultados. O primeiro deles é a compressão dos pacotes IPv6, que neste trabalho focou na compressão dos endereços IP de origem e destino. O protocolo SCHC possibilita a compressão de outros conteúdos estáticos como as portas do protocolo UDP, além de prever também o cálculo do *checksum* de forma local, excluindo assim a necessidade de inserção do resultado no pacote SCHC. Junto a isso, tem-se que o protocolo DLMS/COSEM também pode ser passível de compressão em seus cabeçalhos estáticos, sendo necessária uma implementação referente a isso no protocolo SCHC.

Um segundo ponto observado foi a questão do envio duplicado de fragmentos ao utilizar o modo ACK *Always*. Apesar do SCHC prever um ACK para cada fragmento enviado, pode-se analisar as condições de funcionamento da plataforma *The Things Network* quanto ao agendamento de um *downlink* no recebimento de um *uplink* via LoRaWAN, possibilitando que o *gateway* do trabalho consiga processar o fragmento SCHC recebido e providenciar uma resposta a tempo de aproveitar a janela em aberto na rede LoRaWAN. Junto a isso, a transição de transmissão entre os modos ACK *On Error* e ACK *Always* pode ser analisada com a inserção de um intervalo na inicialização do segundo modo, impedindo que *downlinks* sejam sobrescritos na fila de envio.

Embora estes pontos afetem o desempenho e a eficiência do uso do protocolo SCHC, o sistema como um todo ainda é passível das perdas de pacotes provocadas por falhas na transmissão das informações na rede LoRaWAN. Atualmente, nenhum parâmetro é alterado diante dessas situações, que poderiam contar com algoritmos para ajustes dinâmicos de configurações como MTU em âmbito do SCHC e *timeout* no contexto da aplicação. O primeiro parâmetro influenciaria no sucesso da entrega de fragmentos de acordo com a qualidade da rede LoRaWAN, enquanto o segundo impediria que a fila de requisições aumentasse expressivamente devido a acúmulos de solicitações não atendidas após um período de instabilidades. Isso vai de encontro à necessidade do controle dos pacotes IPv6/UDP na camada de aplicação, garantindo a confiabilidade das conexões diante das perdas de pacotes, visto que a camada de transporte não desempenhou este papel.

Todos estes pontos foram observados em um ambiente de testes limitado, que contou com apenas dois dispositivos conectados à rede LoRaWAN. Apesar do fato de que o trabalho mostrou a possibilidade de mais de um dispositivo comunicando simultaneamente, o cenário utilizado não possibilitou a visualização de situações extremas, onde uma grande quantidade de dispositivos estariam transportando pacotes ao mesmo tempo. Este ponto gera dúvidas principalmente quanto ao funcionamento do *gateway*, de modo que a atual implementação baseada em máquinas de estados, sessões e *threads* poderia

exigir uma quantidade considerável de recursos, necessitando nestes casos possíveis melhorias de arquitetura e implementação da solução. Estes questionamentos poderiam ser melhor desenvolvidos a partir do uso de simuladores de rede juntamente com analisadores de pacotes, onde grandes quantidades de dispositivos, maiores vazões de dados e erros propositais de transmissão promoveriam o estresse do sistema, possibilitando nisso a visualização de possíveis problemas que neste trabalho não ocorreram devido às limitações de escopo.

Por fim, existe a necessidade de validação do trabalho em ambientes diversos, de modo que a qualidade da comunicação na rede LoRaWAN ficaria passível de interferências, oriundas de componentes como distância e variações climáticas. Em uma aplicação real, seria considerado o uso na comunicação com diversos medidores inteligentes de energia elétrica, sendo que outras questões entrariam em discussão, como a possibilidade de escalonamento do sistema, o desempenho do *gateway* e a segurança relacionada ao controle de acesso.

# 6 CONCLUSÃO

Foi implementado neste trabalho o protocolo SCHC tendo como base a biblioteca "PySCHC", de modo que o trabalho focou também na aplicação deste em um sistema que possibilitou a comunicação entre dois pontos distintos. A implementação atendeu aos critérios estabelecidos pelas RFC atreladas ao protocolo, de forma que alguns procedimentos não previstos em norma foram estabelecidos com o intuito de atender às necessidades que surgiram ao longo do desenvolvimento, acompanhados dos respectivos experimentos.

A integração entre a rede implementada e o medidor inteligente de energia elétrica possibilitou a implantação do sistema em uma situação real de obtenção de dados, que partiu do método de solicitar informações e receber suas respostas ao passo que envolvia o funcionamento de todos os componentes presentes neste trabalho. Os resultados obtidos nos experimentos relacionados a esta comunicação mostraram que o sistema foi capaz de transportar e entregar as informações solicitadas, de modo que o SCHC foi capaz de se adequar diante de situações de instabilidades na rede LoRaWAN.

Além disso, a produção de tráfego com pacotes aleatórios desempenhou a função de possibilitar a visualização do protocolo operando juntamente com a rede LoRaWAN. Ao passo que possíveis melhorias foram identificadas, analisou-se as tendências de desempenho da rede juntamente com os custos atrelados ao transporte de informações, envolvendo os pacotes IPv6/UDP, o protocolo SCHC e a rede LoRaWAN. Este cenário foi comparado com um ambiente mais próximo do ideal (que envolvia poucos pacotes perdidos durante as transmissões).

A realização de todos estes processos ocorreu a partir da implementação de diversos componentes que integraram o sistema, englobando a comunicação deste o dispositivo até a aplicação. Neste cenário, a rede LoRaWAN desempenhou a função de possibilitar a comunicação com os dispositivos conectados a esta rede através da Internet, visto que não há como realizar esta tarefa sem a presença de um interlocutor que neste caso foi o servidor de rede fornecido pela *The Things Network*. Junto a isso, o *gateway* foi responsável pela adaptação de tecnologias entre a rede LoRaWAN e a rede IP, de modo que aplicações conectadas a esta rede através de *Sockets* foram capazes de se comunicar utilizando protocolos já consolidados e amplamente utilizados na Internet, sendo neste trabalho o protocolo IPv6/UDP.

A comunicação dos dispositivos conectados à rede LoRaWAN já era possível a partir da Internet, sendo que serviços como os próprios servidores de rede da TTN já desempenham este trabalho há anos. Entretanto a comunicação ponta-a-ponta entre estes dispositivos e as aplicações por meio de pacotes IPv6/UDP continua não sendo um processo nativo nas redes LPWAN. Desta forma, a implementação e o emprego do protocolo SCHC apresentados neste trabalho proporcionaram esta possibilidade, visto que o

SCHC esteve presente tanto nos dispositivos quanto no *gateway*, justamente para realizar o adaptação dos pacotes IPv6/UDP diante das limitações impostas pela rede LoRaWAN.

Neste cenário, o processo de compressão dos cabeçalhos dos pacotes IPv6/UDP possibilitou a redução dos dados trafegados na rede LoRaWAN, de modo que informações previamente identificadas passaram pelo procedimento de inclusão em regras de compressão. Desta forma, ao invés do SCHC sempre enviar informações estáticas presentes nos cabeçalhos dos pacotes IPv6/UDP, era incluso nos pacotes SCHC o código de identificação referente à regra de compressão em questão, que era conhecida tanto no *gateway* quanto nos dispositivos.

Paralelo a isso, o processo de fragmentação foi responsável pela adaptação dos pacotes SCHC dentro da rede LoRaWAN, juntamente com o controle das transmissões dos fragmentos com o intuito de promover a confiabilidade na comunicação. A biblioteca PySCHC forneceu as implementações iniciais do protocolo, focando principalmente no modo *ACK On Error*. Estes algoritmos não eram suficientes para estabelecer a comunicação bidirecional entre dispositivo e aplicação, além de não garantir a confiabilidade na comunicação devido à falhas de implementação. Desta forma, este trabalho teve também como contribuição a continuidade desta biblioteca, que agora possui tanto o modo *ACK On Error* quanto o modo *ACK Always* implementados, além da correção das falhas identificadas.

Como resultado, percebe-se que o sistema implementado cumpriu com o objetivo de estabelecer a comunicação entre dispositivos e aplicação por meio de pacotes IPv6/UDP, estabelecendo também a comunicação com um medidor inteligente de energia elétrica através do protocolo DLMS/COSEM. Apesar de boa parte dos pacotes IPv6/UDP serem entregues em seus respectivos destinos, foram observados pontos de melhoria que tendem a impactar na eficiência da rede, considerando nisso o tempo e a consequente velocidade de entrega, as taxas de perdas e os custos de envio na rede LoRaWAN que em alguns pontos se mostraram elevados.

Conclui-se que este trabalho cumpriu com o objetivo inicial de estabelecer a comunicação nos moldes estabelecidos, de modo que as implementações referentes ao SCHC foram capazes de promover as adaptações necessárias do protocolo diante de perdas ocorridas na rede LoRaWAN. Os pontos de melhoria e as perspectivas apresentadas visam promover a continuidade nas implementações do sistema, de modo a também analisá-lo em situações mais próximas da realidade com o intuito de torná-lo uma solução viável na comunicação com dispositivos conectados à rede LoRaWAN através de tecnologias já consolidadas na Internet. Desta forma, aumentam-se as possibilidades de uso em soluções voltadas à Internet das Coisas, facilitando a comunicação com os dispositivos nela presentes. Por fim, o trabalho se mostrou flexível e compatível com diferentes possibilidades de uso que vão além do apresentado, que partiu da implementação focada no uso genérico para diferentes combinações de protocolos e aplicações.

## REFERÊNCIAS


ABDELFADEEL, K. Q.; CIONCA, V.; PESCH, D. Dynamic context for static context header compression in lpwans. In: **2018 14th International Conference on Distributed Computing in Sensor Systems (DCOSS)**. [S.l.: s.n.], 2018. p. 35–42.

ABNT. **NBR16968 DE 04/2022**. 2022. Acesso em 11 dez. 2022. Disponível em: <https://www.normas.com.br/visualizar/abnt-nbr-nm/13342/abnt-nbr16968-perfil-dlms-cosem-para-medidores-inteligentes-de-energia-eletrica-requisitos->.

AGUILAR, S. et al. Performance analysis and optimal tuning of ietf lpwan schc ack-on-error mode. **IEEE Sensors Journal**, v. 20, n. 23, p. 14534–14547, 2020.

ANATEL. **Ato n 6506, de 27 de agosto de 2018.** 2018. Acesso em 08 dez. 2022. Disponível em: <https://www.anatel.gov.br/legislacao/atos-de-certificacao-de-produtos/2018/1205-ato-6506>.

BEEHARRY, J.; NOWBUTSING, B. Forecasting ipv4 exhaustion and ipv6 migration. In: **2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)**. [S.l.: s.n.], 2016. p. 336–340.

BORGIA, E. The internet of things vision: Key features, applications and open issues. **Computer Communications**, v. 54, p. 1–31, 2014. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366414003168>.

CAMARGO, E. T. de; SPANHOL, F. A.; SOUZA, . R. Castro e. Deployment of a lorawan network and evaluation of tracking devices in the context of smart cities. **Journal of Internet Services and Applications**, v. 12, n. 8, 2021. ISSN 1869-0238. Disponível em: <https://doi.org/10.1186/s13174-021-00138-7>.

CENTENARO, M. et al. Long-range communications in unlicensed bands: the rising stars in the iot and smart city scenarios. **IEEE Wireless Communications**, v. 23, n. 5, p. 60–67, 2016.

CHAUDHARI, B. S.; ZENNARO, M.; BORKAR, S. Lpwan technologies: Emerging application characteristics, requirements, and design considerations. **Future Internet**, v. 12, n. 3, 2020. ISSN 1999-5903. Disponível em: <https://www.mdpi.com/1999-5903/12/3/46>.

DLMS. **DLMS Overview**. 2022. Acesso em 11 dez. 2022. Disponível em: <https://www.dlms.com/dlms-cosem/overview>.

Francois Sforza. **Communications system**. U.S. Patent Documents, 2013. Acesso em 08 dez. 2022. Disponível em: <https://patents.google.com/patent/US8406275B2/en>.

Gurux. **Gurux for DLMS smart meters**. 2022. Acesso em 02 jan. 2023. Disponível em: <https://www.gurux.fi/GXDLMSDirector>.

J. Postel. **User Datagram Protocol**. 1980. Acesso em 10 dez. 2022. Disponível em: <https://www.rfc-editor.org/rfc/rfc768>.

LoRa Alliance. **LoRa Alliance©**. 2015. Acesso em 08 dez. 2022. Disponível em: <https://lora-alliance.org/>.

M. Rose and K. McCloghrie. **Structure and Identification of Management Information for TCP/IP-based internets**. 1998. Acesso em 09 dez. 2022. Disponível em: <https://www.rfc-editor.org/rfc/rfc1065>.

Margaret Rouse. **LPWAN (Low Power Wide Area Network)**. 2017. Acesso em 08 dez. 2022. Disponível em: <http://internetofthingsagenda.techtarget.com/definition/LPWAN-low-power-wide-area-network>.

MEKKI, K. et al. A comparative study of lpwan technologies for large-scale iot deployment. **ICT Express**, v. 5, n. 1, p. 1–7, 2019. ISSN 2405-9595. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405959517302953>.

MINABURO, A. et al. **SCHC: Generic Framework for Static Context Header Compression and Fragmentation**. 2020. Acesso em 10 dez. 2022. Disponível em: <https://www.rfc-editor.org/rfc/rfc8724.html>.

MUÑOZ, R. et al. Schc over lorawan efficiency: Evaluation and experimental performance of packet fragmentation. **Sensors**, v. 22, n. 4, 2022. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/22/4/1531>.

NIC Chile Research Labs. **GitHub - PySCHC**. 2021. Acesso em 26 dez. 2022. Disponível em: <https://github.com/niclabs/PySCHC>.

O. Gimenez and I. Petrov. **Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN**. 2021. Acesso em 10 dez. 2022. Disponível em: <https://www.rfc-editor.org/rfc/rfc9011.html>.

S. Deering and R. Hinden. **Internet Protocol Version 6 (IPv6) Specification**. 1998. Acesso em 09 dez. 2022. Disponível em: <https://www.rfc-editor.org/rfc/rfc2460>.

S. Farrell. **Low-Power Wide Area Network (LPWAN) Overview**. 2018. Acesso em 08 dez. 2022. Disponível em: <https://www.rfc-editor.org/rfc/rfc8376.html>.

SANCHEZ-GOMEZ, J. et al. Impact of schc compression and fragmentation in lpwan: A case study with lorawan. **Sensors**, v. 20, n. 1, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/1/280>.

Semtech. **LoRa Modulation Basics. AN1200.22**. 2015. Acesso em 08 dez. 2022. Disponível em: <http://wiki.lahoud.fr/lib/exe/fetch.php?media=an1200.22.pdf>.

SEMTECH. **LoRa and LoRaWAN: Technical overview**. 2022. Acesso em 08 dez. 2022. Disponível em: <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/>.

STALLINGS, W. **Data and Computer Communications**. 10. ed. [S.l.]: Pearson, 2014. 907 p.

TANENBAUM, A. S. **Computer Networks**. 5. ed. [S.l.]: Pearson, 2011. 960 p.

WÜLFING, C. A. et al. Evaluation of dlms/cosem data processing setups applied to smart metering. In: **2022 14th Seminar on Power Electronics and Control (SEPOC)**. [S.l.: s.n.], 2022. p. 1–6.