UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

TECNÓLOGO EM REDES DE COMPUTADORES

ABIMAEL SILAS BARROS

ANÁLISE DE DESEMPENHO DOS PROTOCOLOS COAP, MQTT-SN E HTTP EM SMART HOMES

QUIXADÁ

2022

ABIMAEL SILAS BARROS

ANÁLISE DE DESEMPENHO DOS PROTOCOLOS COAP, MQTT-SN E HTTP EM SMART HOMES

Trabalho de Conclusão de Curso apresentado ao Curso de Redes De Computadores da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Tecnólogo em Redes De Computadores. Área de concentração: Computação.

Orientador: Prof. Me. Marcos Dantas Ortiz

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B273a Barros, Abimael Silas.
Análise de desempenho dos protocolos coap, mqtt-sn e http em redes IoT / Abimael Silas Barros. – 2022.
52 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2022.
Orientação: Prof. Me. Marcos Dantas Ortiz.

1. Desempenho - Análise. 2. Internet das coisas. I. Título.

CDD 004.6

ABIMAEL SILAS BARROS

ANÁLISE DE DESEMPENHO DOS PROTOCOLOS COAP, MQTT-SN E HTTP EM SMART HOMES

Trabalho de Conclusão de Curso apresentado ao Curso de Redes De Computadores da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Tecnólogo em Redes De Computadores. Área de concentração: Computação.

Aprovado em: ___/___/____.

BANCA EXAMINADORA

Prof. Me. Marcos Dantas Ortiz (Orientador)
Universidade Federal do Ceará – UFC

Prof. Dr. Michel Sales Bonfim
Universidade Federal do Ceará - UFC

Prof. Dr. Paulo Antonio Leal Rego
Universidade Federal do Ceará - UFC

RESUMO

Os crescentes avanços na área da tecnologia mudaram a forma como lidamos com os objetos que utilizamos em nosso dia-a-dia. O surgimento de objetos inteligentes, munidos de recursos computacionais e de comunicação, nos levam a uma nova realidade: a revolução da Internet das Coisas, do inglês, *Internet of Things* (IoT). Podemos definir a IoT como uma expansão da Internet atual, em que objetos físicos como sensores, aparelhos eletrônicos, entre outros, estão interligados através das redes de computadores e interagem entre si. Com o advento da IoT, uma série de novos problemas e desafios surgiram, principalmente no que se refere à comunicação, devido às limitações de seus dispositivos. Novos protocolos surgiram com o intuito de sanar tais deficiências, entretanto, surge também a questão: Qual protocolo da camada de aplicação deve ser usado? A partir disso, o objetivo central deste trabalho é comparar os protocolos da camada de aplicação CoAP, MQTT-SN e HTTP, em cenários que simulam *Smart homes* e desse modo identificar o mais eficiente energeticamente, com menor índice de perda de pacotes e menor latência. Após a criação e execução dos cenários, e com as métricas de consumo energético latência e perda de pacotes devidamente coletadas, identificou-se que em todos os gráficos o protocolo CoAP se saiu melhor em comparação ao MQTT-SN e ao HTTP. Este trabalho destina-se a profissionais da área de Tecnologia da Informação e Comunicação (TIC) que queiram desenvolver e/ou aprimorar soluções para a IoT.

Palavras-chave: Análise de desempenho. Internet das coisas.

ABSTRACT

Growing advances in technology have changed the way we deal with the objects we use in our daily lives. The emergence of intelligent objects, equipped with computational and communication resources, takes us to a new reality: the revolution of the Internet of Things. We can define the IoT as an expansion of the current Internet, in which physical objects such as sensors, electronic devices, among others, are interconnected through computer networks and interact with each other. With the advent of IoT, a series of new problems and challenges arose, mainly with regard to communication, due to the limitations of its devices. New protocols have emerged in order to remedy such deficiencies, however, the question also arises: Which application layer protocol to be used? From this, the main objective of this work is to compare the CoAP, MQTT-SN and HTTP application layer protocols, in scenarios that simulate *Smart homes* and thus identify the most energy efficient, with the lowest rate of packet loss and lower latency. After the creation and execution of the scenarios, and with the energy consumption, latency and packet loss metrics duly collected, it was identified that in all graphs the CoAP protocol did better compared to MQTT-SN and HTTP. This work is aimed at professionals in the field of Information and Communication Technology (ICT) who want to develop and/or improve IoT solutions.

Keywords: Performance analysis. Internet of things.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

6LoWPAN	*IPv6 over Low power Wireless Personal Area Networks*
ACK	*Aknowledgement*
AMQP	*Advanced Message Queuing Protocol*
CON	*Confirmable*
CPU	*Central Processing Unit*
CTP	*Collection Tree Protocol*
DAG	*Direct Acyclic Graphs*
DAO	*Destination Advertisement Object*
DAO-ACK	*Destination Advertisement Object Acknowledgment*
DIO	*DODAG Information Object*
DIS	*DODAG Information Solicitation*
DODAG	*Destination Oriented Acyclic Graph*
DUP	*Duplicate delivery*
ETX	*Expected Transmission*
HTML	*Hipertext Markap Language*
ICMPv6	*Internet Control Message Protocol Version 6*
IETF ROLL	*Internet Engineering Task Force Routing Over Lowpower and Lossy networks*
IP	*Internet Protocol*
IPv4	*Internet Protocol version 4*
IPv6	*Internet Protocol version 6*
JNI	*Java Native Interface*

LPM	*Low Power Mode*
M2M	*Machine-to-Machine*
MQTT	*Message Queuing Telemetry Transport*
MQTT-SN	*MQTT for Sensor networks*
NON	*Non-confirmable*
QoS	*Quality of Service*
RAM	*Random Access Memory*
RPL	*IPv6 Routing Protocol for Low Power and Lossy Networks*
RSSF	*Redes de Sensores Sem Fio*
RST	*Reset*
RTT	*Round Trip Time*
TCP	*Transmission Control Protocol*
TIC	*Tecnologia da Informação e Comunicação*
UDP	*User Datagram Protocol*
URI	*Uniform Resource Identifier*

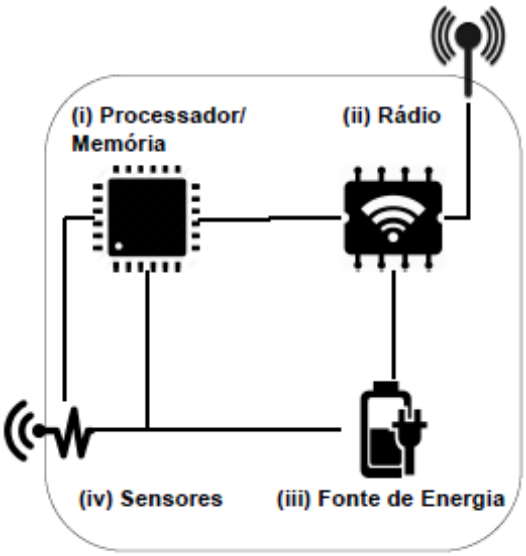
SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais a tecnologia e a Internet tornaram-se elementos bem presentes em nossas vidas, além disso, os crescentes avanços na área da tecnologia mudaram a forma como lidamos com os objetos que utilizamos em nosso dia-a-dia. O surgimento de objetos inteligentes, munidos de recursos computacionais e de comunicação, nos levam a uma nova realidade: a revolução da Internet das Coisas, do inglês, *Internet of Things* (IoT). O termo IoT foi utilizado pela primeira vez pelo especialista Kevin Ashton, em 1999, que utilizou a frase para descrever a evolução da Internet e uma nova revolução no mundo da tecnologia da informação (ASHTON, 2009).

Podemos definir a IoT como uma expansão da Internet atual, em que objetos físicos como sensores, aparelhos eletrônicos, entre outros, estão interligados através das redes de computadores e interagem entre si, proporcionando aos mesmos a capacidade de enviar e/ou disponibilizar informações sobre o ambiente. Estes objetos são denominados de *Appliances* e em sua grande maioria possuem a mesma arquitetura básica: 1 unidade de processamento, 1 unidade de armazenamento interno, 1 fonte de alimentação, 1 unidade de comunicação e n sensores/atuadores, como mostra a Figura 1.

Figura 1 – Arquitetura básica *Appliances*



Fonte: SANTOS et al. (2016)

As *Appliances* podem ser incorporadas em diversas áreas e ambientes como, por exemplo, em prédios, na agricultura, na indústria, no transporte, na educação, na saúde, no lar,

etc., desta forma, as vantagens e benefícios que a IoT pode nos proporcionar, dependendo da forma que for empregada, são imensas. A incorporação de tais aparelhos nas áreas supracitadas trouxeram consigo a criação de novos conceitos, dos quais podemos destacar: *Smart Building*, *Smart City* e *Smart Home*.

Segundo SINOPOLI (2009) o conceito de *Smart Building* (prédio inteligente) surgiu na década de 1980. Conforme o autor, no ano de 1984 foi publicado um artigo no *The New York Times* que dizia que construtores estavam criando uma nova geração de edifícios: prédios inteligentes, com a habilidade de pensar por si mesmos. A disseminação dos microprocessadores na década de 1960 e o desenvolvimento de novos chips na década de 1970 possibilitaram a automatização de sistemas como os de ar-condicionado, ventilação e aquecimento, que por sua vez impulsionou a busca por atribuir inteligência a edifícios.

Uma *Smart City* (ou cidade inteligente), por sua vez, pode ser definida como um centro urbano munido de dispositivos inteligentes que automatizam serviços, beneficiando os locais em que são aplicados e melhorando a qualidade de vida de seus moradores. Para RIZZON et al. (2017) uma *Smart City* é uma cidade que usa Tecnologia da Informação e Comunicação (TIC) para melhorar a qualidade de vida de seus habitantes, contribuindo dessa forma para um desenvolvimento sustentável.

Uma *Smart Home* trata-se de uma casa automatizada (ou casa inteligente), em que existe uma série de dispositivos que a integram e realizam determinadas funções para facilitar a vida das pessoas que nela habitam. A primeira vez que a ideia de casa inteligente foi utilizada foi no ano de 1950, no conto Chuvas Leves Virão, de Ray Bradbury, onde o autor retrata uma casa inteligente que, mesmo depois da morte de seus habitantes, continua realizando as tarefas domésticas e outras atividades específicas rotineiras a seus residentes (BRADBURY, 2013). Como é abrangente o crescimento de edifícios residencias conectados à Internet, possuindo assim uma infra-estrutura de rede capaz de comportar dispositivos de TIC, o cenário escolhido para realização deste trabalho foi o de *Smart Homes*.

Contudo, devido as limitações de *hardware* das *Appliances* os protocolos empregados na Internet convencional não são adequados para uso em tais dispositivos, surge então um novo desafio: o emprego de mecanismos que regulem a transmissão de dados entre estes dispositivos, garantindo assim, comunicação de qualidade em tempo real e economia de recursos. Para tal, foram desenvolvidos protocolos máquina-a-máquina (*Machine-to-Machine* – M2M), que possibilitam a dispositivos remotos distintos permutar dados através da rede sem a necessidade

de intervenção humana (LIMA, 2018).

Na IoT, eficiência energética, adaptabilidade, escalabilidade, e segurança são requisitos fundamentais, ademais, é imprescindível que tais protocolos sejam compatíveis com as limitações dos dispositivos, de modo a proporcionar um consumo eficiente da energia e dos recursos de processamento e armazenamento. Há vários protocolos propostos para comunicação M2M focados em ambientes limitados, podemos citar como exemplo os protocolos:

- *Constrained Application Protocol* (CoAP).
- *Advanced Message Queuing Protocol* (AMQP).
- *Message Queuing Telemetry Transport* (MQTT).
- *IPv6 over Low power Wireless Personal Area Networks* (6LoWPAN).

No entanto, não há um protocolo definido como padrão, capaz de atender aos requisitos de todos os diferentes dispositivos existentes ou pelo menos a grande maioria (MAZZER; FRIGIERI; PARREIRA, 2013) . Desta forma, qual o protocolo certo ou que protocolo escolher continuam como perguntas sem uma resposta exata e satisfatória.

A IoT, por outro lado, traz uma série de novos desafios no que se refere a comunicação, devido às limitações de seus dispositivos. Novos protocolos surgiram com o intuito de sanar os desafios advindos das limitações dos dispositivos IoT, a fim de assegurar que tais dispositivos, mesmo possuindo baixa capacidade de memória e processamento, assegurassem seus requisitos fundamentais (eficiência energética, adaptabilidade, escalabilidade e segurança). Surge a questão: qual protocolo da camada de aplicação a ser usado por estes dispositivos? Portanto, o presente trabalho tem como objetivo realizar uma análise de desempenho entre os protocolos da camada de aplicação: CoAP, MQTT-SN e HTTP, a fim de identificar qual destes protocolos apresentam a melhor eficiência energética, o menor índice de perda e a menor latência.

Este trabalho destina-se a profissionais da área de TIC que visam desenvolver e/ou aprimorar soluções para a IoT. A análise de desempenho de protocolos em cenários típicos da IoT permite verificar se os protocolos submetidos a experimentação atendem aos requisitos das *Appliances* (FERNANDES; BRITO, 2020). Neste contexto, com base nas informações expostas até o momento, podemos dizer que esta pesquisa se justifica pela necessidade da implementação de protocolos confiáveis, seguros e econômicos.

O presente trabalho mantém a seguinte organização: na seção 1.1 são explicitados o objetivo geral (Subseção 1.1.1) e os objetivos específicos (subseção 1.1.2). O objetivo geral traz a

delimitação do trabalho com a questão central, enquanto os objetivos específicos listam os pontos chave que ao serem descobertos ajudam a responder a questão central delimitada no objetivo geral. O Capítulo 2 fala dos trabalhos relacionados, fazendo um condensado de trabalhos que abordam total ou em parte os temas centrais deste trabalho, relatando pontos de similaridade e divergência em relação ao presente trabalho. O Capítulo 3 apresenta a fundamentação teórica, sendo vistos os protocolos M2M utilizados neste trabalho: CoAP; MQTT; HTTP, 6LoWPAN e RPL. Ainda na fundamentação teórica discorre-se sobre o simulador/emulador Cooja. Em seguida, o Capítulo 4 aborda a metodologia apresentando as etapas cruciais que foram realizadas com a finalidade de resolver a questão de pesquisa. O Capítulo 5 exibe os gráficos referentes ao consumo energético, latência e a perda de pacotes, e discorre sobre os resultados obtidos. Por fim, o Capítulo 6 apresenta as considerações finais do trabalho.

1.1 Objetivos

O objetivo central deste trabalho é exposto na subseção 1.1.1. Esta subseção tem a função de descrever os limites propostos para o trabalho (escopo do trabalho). Em seguida, os objetivos específicos são apresentados na subseção 1.1.2, os objetivos específicos mostram os pontos chave que quando esclarecidos ajudam a resolver o objetivo central visto na subseção 1.1.1.

1.1.1 Objetivo geral

O presente trabalho tem como objetivo comparar os protocolos da camada de aplicação CoAP, MQTT-SN e HTTP, em um ambiente simulado da IoT voltado para *Smart Homes*, e identificar o mais eficiente energeticamente, com menor índice de perda de pacotes e menor latência. O HTTP não é um protocolo desenvolvido para IoT, mas foi inserido neste trabalho para analisar como protocolos desenvolvidos para dispositivos mais robustos se comportam em cenários IoT.

1.1.2 Objetivos específicos

Os seguintes objetivos específicos foram considerados para se alcançar o objetivo geral:

- Planejar e executar os experimentos.

- Identificar qual protocolo da camada de aplicação apresenta melhor eficiência energética.
- Identificar qual protocolo da camada de aplicação apresenta menor taxa de descarte.
- Identificar qual protocolo da camada de aplicação apresenta menor latência.

2 TRABALHOS RELACIONADOS

Com o advento da IoT, surgiram novos problemas e desafios, principalmente no que se refere a comunicação, em razão das limitações de seus dispositivos. Novos protocolos foram desenvolvidos com o objetivo de sanar tais deficiências, contudo, das diversas questões que surgiram podemos ressaltar a seguinte: qual o melhor protocolo da camada de aplicação? Com isso, alguns autores realizaram testes em ambientes reais ou simulados para responder tal questão. Este Capítulo faz um condensado dos trabalhos relacionados e, a partir disso, relata os pontos de igualdade e os pontos que diferem do atual trabalho.

No trabalho de Lima (2018), são realizados experimentos com os protocolos CoAP e MQTT-SN no simulador Cooja com foco em ambientes urbanos, ou seja, *Smart Cities*. Para realização dos testes foi implementada a planta da cidade de Carazinho, Rio Grande do Sul, de forma que 30 nós foram dispostos em 2 quadras, cada nó representando um poste inteligente, e 1 nó coletor de informações nas proximidades. O 6LoWPAN foi utilizado para endereçamento e o RPL para a construção das rotas. O autor teve como objetivo principal avaliar o comportamento dos protocolos da camada de aplicação CoAP e MQTT-SN em uma Rede de Sensores Sem Fio (RSSF). O trabalho foi aplicado ao contexto de *Smart Cities*, com cenário montado em cima de postes onde cada um deles continha um sensor distinto, podendo ele ser voltado a aferição da temperatura, umidade ou luminosidade. Este trabalho difere do proposto por Lima (2018), por ser contextualizado em um ambiente de *Smart Homes*. Além disso, o tipo de dispositivos IoT simulados é diferente, neste trabalho foram utilizados motes Z1 enquanto no de Lima (2018) foram utilizados motes do tipo *Sky*. Assim como o trabalho de (LIMA, 2018), este trabalho utiliza o Cooja como plataforma de simulação e também aplicam-se os protocolos MQTT-SN e CoAP, porém realiza-se o comparativo com o protocolo HTTP, e utilizam-se as métricas de eficiência energética, perda de pacotes e latência, dessa forma, diferenciando-se do trabalho de Lima (2018).

A pesquisa de Bahia e Campista (2017), tem como foco o CoAP em um cenário Industrial e propõe um mecanismo de controle que visa aumentar o desempenho e a escalabilidade de servidores CoAP no provimento de serviços em um cenário industrial típico de IoT. No cenário base a rede é composta por um nó atuando como roteador de borda executando o protocolo RPL, um nó como servidor CoAP e um número variável de clientes CoAP. A comunicação entre os clientes e o servidor sempre é intermediada pelo roteador de borda, de forma que a origem do pedido é irrelevante para o servidor, seja de um cliente interno ou externo. Para cada

configuração, a simulação teve uma duração de 10 minutos. Este trabalho, assim como o de Bahia e Campista (2017), utiliza de um nó que atua como roteador de borda da rede executando o protocolo RPL. Porém, este trabalho se distingue do realizado pelo autor por estar em um cenário base diferente, e comparar três protocolos distintos: CoAP, MQTT-SN e HTTP, enquanto Bahia e Campista (2017), analisam somente o protocolo CoAP, além do mais, tem-se como métricas para o atual trabalho: a eficiência energética, a taxa de perda de pacotes e a latência.

No trabalho de Barrera (2017), o objetivo principal é a criação de um RSSF que possua um algoritmo para construção de topologia, detecção de nós mortos e autorrecuperação. Foram realizados testes em ambiente real no Departamento de Engenharia Eletrônica da Faculdade de Engenharia da Pontificia Universidad Javeriana (*Department of electronics engineering on the faculty of Engineering at the Pontificia Universidad Javeriana*) e simulações no simulador Cooja, com os testes possuindo um número variado de nós. O Cooja foi utilizado para verificar o desempenho em termos de eficiência energética, em um cenário de rede com 1 nó raiz e 24 outros nós normais. Cada nó foi selecionado manualmente para verificar métricas do RPL, pacotes recebidos e outras informações usando a ferramenta de visualização de coleta do Cooja. Igualmente ao trabalho de Barrera (2017), o vigente trabalho usa o simulador Cooja como plataforma de execução dos cenários, também faz uso de um roteador de borda central, executando o protocolo RPL, responsável por fazer a ponte de comunicação entre a rede de nós IoT dos cenários com a rede externa. Em contra partida, este trabalho não foca nas métricas do RPL, mas sim nas métricas de interesse deste trabalho, para verificar quais dos protocolos da camada de aplicação CoAP, MQTT-SN e HTTP em cenário de *Smart Home* aponta a melhor eficiência energética, o menor índice de perda e a menor latência.

Quadro 1 – Comparação entre os trabalhos relacionados e o vigente trabalho

	Protocolos	**Simulador**	**Métricas**	**Cenário**	**Ambiente**
Lima (2018)	CoAP, MQTT-SN, 6LoWPAN e RPL	COOJA	Latência, taxa de entrega, sobrecarga, taxa de transferência e uso de sensores	*Smart Cities*	Simulado
Bahia e Campista (2017)	CoAP, 6LoWPAN e RPL	COOJA	Quantidade de pacotes CoAP gerados, perdas, vazão de dados e consumo de energia	Indústria	Simulado
Barrera (2017)	CoAP, MQTT, 6LoWPAN e RPL	COOJA	Consumo de Energia	Laboratório	Real/Simulado
Próprio Autor (2022)	CoAP, MQTT-SN, HTTP, 6LoWPAN e RPL	COOJA	Consumo de energia, índice de perda de pacotes e latência	*Smart Homes*	Simulado

Fonte: Próprio Autor (2022)

O Quadro 1 mostra as diferenças e semelhanças entre o presente trabalho e os trabalhos relacionados supracitados. O diferencial entre este trabalho e o elaborado pelos autores são em suma: Cenário Base, Ambiente de Execução, as Métricas e os Protocolos utilizados. Barrera (2017) realiza os testes tanto no Cooja quanto em ambiente Real; os trabalhos desenvolvidos pelos autores mencionados utilizam no máximo duas das métricas utilizadas neste trabalho; E por fim, diferente dos demais, este trabalho investiga três diferentes protocolos da camada de aplicação, que são o CoAP, MQTT-SN e HTTP.

3 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo serão abordados os principais conceitos utilizados no decorrer deste trabalho. Na seção 3.1 serão apresentados os protocolos M2M utilizados neste trabalho, sendo que estão dispostos em subseções da Seção 3.1 Protocolos, na seguinte ordem: CoAP (3.1.1), MQTT (3.1.2), HTTP (3.1.3), 6LoWPAN (3.1.4), RPL (3.1.5) e CTP (3.1.6). A Seção 3.2, por sua vez, discorre sobre o simulador Cooja.

3.1 Protocolos

Nas seguintes subseções serão abordados os diferentes protocolos utilizados neste trabalho. Segundo Kurose, Ross e Zucchi (2007) um protocolo é o que define o formato e a ordem em que as mensagens são trocadas entre duas ou mais entidades comunicantes, além disso, define as ações a serem realizadas na transmissão e/ou no recebimento de uma mensagem ou no decorrer de outro evento.

3.1.1 CoAP

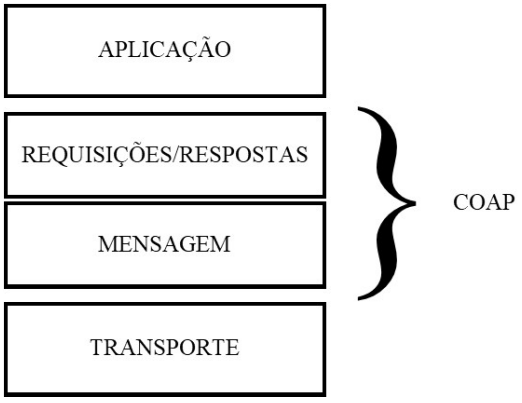
O *Constrained Aplication Protocol* (CoAP) é um protocolo M2M especializado em transferência de dados WEB para uso com nós restritos em redes IoT, algumas de suas características principais são: troca de mensagens assíncronas, suporte a *Uniform Resource Identifier* (URI), baixa sobrecarga de cabeçalho e baixa complexidade de análise, proxy e armazenamento em *cache* simples, entre outros, e um de seus principais objetivos é projetar um protocolo web genérico para os requisitos especiais de ambientes restritos (SHELBY et al., 2014). Vale ressaltar que o CoAP foi projetado para interagir com o *Hypertext Transfer Protocol* (HTTP) e assim prover integração com a WEB (MARTINS; ZEM, 2016).

Pode-se dizer que no modelo TCP/IP o protocolo CoAP encontra-se entre as camadas de Aplicação e Transporte, gerando duas subcamadas, a saber, as camadas Requisições/Respostas e Mensagem, como mostra a Figura 2. Estas duas subcamadas, por sua vez, são integradas no cabeçalho das mensagens CoAP (ANSARI; REHMAN; ALI, 2018).

O CoAP possui um modelo de interação que assemelha-se ao modelo cliente/servidor do HTTP, entretanto, as interações M2M em sua maioria permitem aos dispositivos atuarem como cliente e servidor simultaneamente. Diferente do HTTP, que é um protocolo da camada de aplicação para sistemas de informações hipermídia distribuídos e colaborativos que geralmente

utiliza o protocolo TCP para comunicação (FIELDING et al., 1999), o CoAP lida com os pedidos e requisições de forma assíncrona utilizando o protocolo orientado a datagrama UDP.

Figura 2 – Posição do CoAP na Arquitetura TCP/IP



Fonte: Adaptado de Ansari, Rehman e Ali (2018)

O CoAP utiliza um pequeno cabeçalho de 32 *bits* que pode ser seguido de opções binárias compactas e uma carga útil e o mesmo possui quatro tipos de mensagens, são elas: CON (*Confirmable*), NON (*Non-confirmable*), ACK (*Acknowledgement*) e RST (*Reset*). As requisições são realizadas por meio das mensagens CON (Com Confirmação) e NON (Sem Confirmação) e as respostas são enviadas como ACK (Confirmação), contendo a informação solicitada pelo cliente.

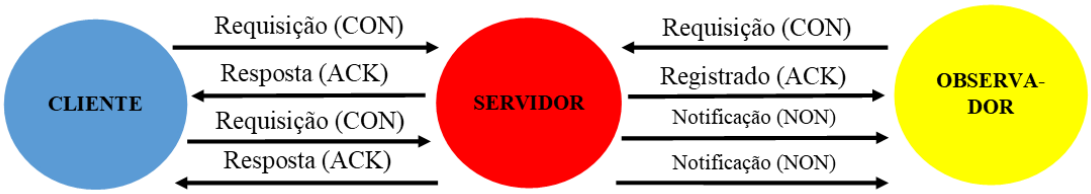
Quando o destinatário recebe uma mensagem CON mas não consegue processá-la o mesmo envia uma mensagem RST ao invés de uma ACK, desta forma, as mensagens RST atuam como um pedido de retransmissão. No entanto, caso o pedido seja processado mas o recipiente não puder responder imediatamente, um ACK sem conteúdo é enviado ao emissor para que este não continue retransmitindo.

O CoAP implementa um identificador em cada mensagem para permitir a detecção de duplicatas e para prover confiabilidade. As mensagens CON utilizam um tempo limite padrão e *backoff* exponencial para realizar retransmissões. O *backoff* é um algoritmo usado quando há detecção de falhas, de modo que este incrementa exponencialmente o tempo limite de retransmissão. Caso a comunicação exija transmissão confiável, as requisições devem ser enviadas através de mensagens CON, senão, podem ser enviadas por meio de mensagens NON.

Uma *Appliance* geralmente envolve a transmissão de informações sobre seus sensores para outros dispositivos, registrando-se em uma lista, vale ressaltar que esta possui um tamanho limitado à capacidade de armazenamento do dispositivo e cada cliente só pode registrar-se uma

única vez em uma lista. Segundo HARTKE (2015) o CoAP possui uma função denominada Observar, que permite a um dispositivo monitorar os recursos de outro. Tal função permite aos clientes observar as mudanças nos recursos de seu interesse sem gerar novos pedidos, proporcionando decréscimo na sobrecarga de rede, redução do uso de banda, menor latência e aumentando a confiabilidade em relação ao HTTP.

Figura 3 – Funcionamento do protocolo CoAP



Fonte: Adaptado de HARTKE (2015)

Esse recurso de Observação presente no CoAP é muito importante para a escalabilidade e para o controle do consumo de energia, pois garante um maior tempo de disponibilidade do servidor. O funcionamento do protocolo CoAP é exemplificado na Figura 3, onde podemos observar interações sob demanda entre cliente e servidor e comunicação por notificação entre o servidor e um cliente observador.

3.1.2 MQTT

O *Message Queue Telemetry Transport* (MQTT) é um protocolo desenvolvido por Andy Stanford Clark e Arlen Nipper, em meados de 1999, e é projetado para conectar dispositivos remotos com limitações de memória ou cuja largura de banda da rede é restrita (BARRERA, 2017). Diferente do HTTP, que utiliza o modelo requisição/resposta, o MQTT implementa uma arquitetura publicação/assinatura e possui apenas 3 tipos de entidades: *publisher*, *broker* e *subscriber*.

A maior diferença em relação ao HTTP é o fato de no MQTT um cliente não precisar solicitar as informações necessárias, ele simplesmente realiza a assinatura em um recurso de seu interesse e, a partir de então, os *brokers* passam a enviar as informações para os *subscribers* caso existam novas informações dos *publishers*. O *Broker* (Servidor) é o dispositivo central no protocolo MQTT pois tem como principais responsabilidades intermediar a comunicação entre os *subscribers* (Assinantes) e *publishers* (Editores) e realizar a entrega das mensagens entre eles com base em seus tópicos interessados (YASSEIN et al., 2017).

Cada uma das mensagens MQTT possui um cabeçalho fixo de 2 *bytes*, onde o primeiro *byte* contém cinco campos: o tipo da mensagem, *Duplicate delivery* (DUP), *Quality of Service* (QoS) e RETAIN e o segundo *byte* contêm o campo Comprimento Restante. Todos os valores de dados estão em ordem *big-endian*, ou seja, os *bytes* de ordem superior antecedem os *bytes* de ordem inferior.

Alguns tipos de mensagens MQTT exigem um cabeçalho variável e uma carga útil, sendo que tais componentes estão dispostos da seguinte forma: cabeçalho fixo, cabeçalho variável e carga útil, respectivamente. O Quadro 2 representa o cabeçalho de tamanho fixo de uma mensagem MQTT.

Quadro 2 – Cabeçalho de tamanho fixo MQTT

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Byte 1	Tipo de Mensagem				DUP	QoS		RETAIN
Byte 2	Comprimento Restante							

Fonte: Adaptado de IBM e EUROTECH (2010)

Como mostra o Quadro 2, o Tipo de Mensagem ocupa o espaço dos *bits* 7 ao 4 e está localizado no primeiro *byte* do cabeçalho fixo e é representado como um valor sem sinal de 4 *bits* presente no Quadro 3.

Quadro 3 – Tipos de mensagens MQTT

MNEMÔNICO	**ENUMERAÇÃO**	**DESCRIÇÃO**
Reserved	0	Reservado
CONNECT	1	Solicitação do Cliente para se Conectar ao Servidor
CONNACK	2	Confirmação de Conexão
PUBLISH	3	Publicar Mensagem
PUBACK	4	Confirmação de Publicação
PUBREC	5	Publicação Recebida (Garantia de Entrega Parte 1)
PUBREL	6	Publicação Liberada (Garantia de Entrega Parte 2)
PUBCOMP	7	Publicação Completa (Garantia de Entrega Parte 3)
SUBSCRIBE	8	Pedido de Assinatura do Cliente
SUBACK	9	Confirmação de Assinatura
UNSUBSCRIBE	10	Pedido de Cancelamento de Assinatura do Cliente
UNSUBACK	11	Confirmação Pedido de Cancelamento de Assinatura
PINGREQ	12	Pedido de PING
PINGRESP	13	Resposta PING
DISCONNECT	14	Cliente está Desconectando-se do Servidor
Reserved	15	Reservado

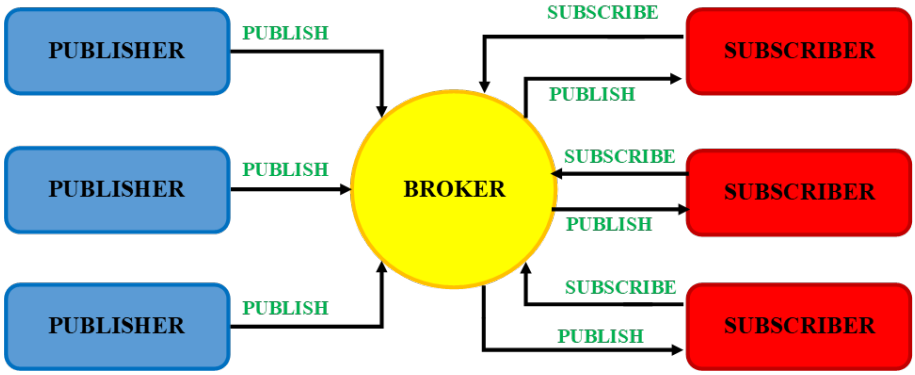
Fonte: IBM e EUROTECH (2010) (Traduzido pelo Autor)

O DUP é um sinalizador relacionado à entrega de mensagens duplicadas e é utilizado quando um cliente ou servidor tenta retransmitir mensagens do tipo PUBLISH, PUBREL, SUBSCRIBE ou UNSUBSCRIBE cujo valor de QoS seja maior que 0 e uma confirmação é necessária. O DUP não deve ser usado para detectar duplicatas, mas como uma dica de que a mensagem já pode ter sido recebida.

O QoS indica o nível de garantia para a entrega de uma mensagem do tipo PUBLISH e possui 3 tipos: QoS 0, QoS 1 e QoS 2. O QoS 0 é o modo de transferência mais rápido, mas não armazena a mensagem localmente, desta forma, caso o cliente esteja indisponível ou ocorra algum erro, ele perderá a mensagem. No QoS 1 cada mensagem é enviada pelo menos uma vez e é armazenada, quando a confirmação é recebida a mensagem é apagada. O QoS 2 trata-se do modo de transferência mais lento, por utilizar um mecanismo de *handshake* de 4 vias para garantir que a mensagem seja entregue somente uma vez, assim como no QoS 1 a mensagem é armazenada e quando o emissor recebe a confirmação de entrega a mesma é descartada.

O marcador RETAIN é usado quando um cliente envia uma mensagem PUBLISH ao servidor e deseja que ela seja mantida no servidor mesmo depois de ser entregue aos assinantes, com isso, quando um novo Assinante definir um tópico como de seu interesse a última mensagem retida para este tópico será enviada para o mesmo caso esta opção esteja habilitada. O Comprimento Restante é usado para representar a quantidade de *bytes* restantes na mensagem. Por fim, na Figura 4 temos um cenário simplificado do protocolo MQTT.

Figura 4 – Cenário de uso do MQTT



Fonte: Adaptado de Barrera (2017)

Neste trabalho foi utilizado a versão MQTT-SN para redes de baixa potência de sensores sem fio (COSMI; MOTA, 2019). A versão escolhida deu-se pelo motivo do Cooja não disponibilizar nós com a versão MQTT padrão. De acordo com Cosmi e Mota (2019), o MQTT-SN foi criado para permitir o uso do MQTT em redes não TCP/IP, assim como *Zigbee*

ou UDP/IP. Para isto, o MQTT-SN implementa seu próprio controle de fluxo e retransmissão, além de possuir algumas características interessantes como a criação de um mecanismo de identificadores de tópicos, em que cada cliente pode pedir ao *Broker* que gere um número para o tópico. Com isso, ao invés de utilizar o nome do tópico em UTF-8 para a troca de mensagens, é utilizado o identificador, dessa forma, consumindo menos dados e consequentemente menos energia.

3.1.3 HTTP

De acordo com Berners-Lee, Fielding e Frystyk (1996), Point (2021), o protocolo *Hypertext Transfer Protocol* (HTTP) é um protocolo do nível de aplicação, para sistemas de informação de hipermídia distribuídos e colaborativos. Sendo a base para a comunicação de dados para a *World Wide Web*. Basicamente o HTTP é um protocolo de comunicação construído no modelo TCP/IP, utilizado para a entrega de dados como arquivos HTML, imagens e *strings* de busca na *World Wide Web*. Tendo como porta padrão 80 do TCP, todavia outras portas podem ser usadas. O HTTP fornece uma maneira padronizada para a comunicação entre dois *hosts*. Em sua especificação o HTTP estrutura como os dados de solicitação serão construídos e enviados ao servidor e como os servidores tratam e respondem essas solicitações (BERNERS-LEE; FIELDING; FRYSTYK, 1996).

O protocolo HTTP é baseado no paradigma Requisição/Resposta, quando um cliente estabelece uma conexão com um servidor, e a partir disso envia uma requisição para o servidor na forma de URI, passando a versão do protocolo com todo o corpo pertinente e os cabeçalhos necessários o servidor trata e responde com o *status* da requisição (se foi tratada corretamente ou não) e outros informações assim como metadados que podem ser melhor estudados em (BERNERS-LEE; FIELDING; FRYSTYK, 1996). Dentre os vários métodos definidos pelo HTTP existem quatro que são os principais segundo Rodriguez (2008), são eles: POST; GET; PUT e DELETE. O método POST é utilizado para a criação de recursos dentro do servidor. Já o método GET é utilizado para recuperar um recurso que já foi posteriormente salvo no servidor. O método PUT é usado para mudar o estado de um recurso ou ainda atualiza-lo. E por fim, o método DELETE é usado para remover ou deletar um recurso no servidor. Neste trabalho o protocolo HTTP foi utilizado na experimentação, sendo o mesmo comparado com outros dois protocolos de aplicação, o MQTT-SN e o CoAP.

3.1.4 6LoWPAN

A Internet atualmente é baseada principalmente no protocolo IPv4 e usa endereços de 32 *bits*, limitando a quantidade de endereços disponíveis para 4.294.967.296 endereços únicos, no entanto, com a crescente difusão da Internet e o aumento do número de dispositivos conectados a mesma, em 3 de fevereiro de 2011 o IPv4 chegou a exaustão (OLSSON, 2014). Essa limitação levou ao desenvolvimento de um novo protocolo: o IPv6, que diferente do IPv4 usa endereçamento de 128 *bits* e dispõe de $3,4 \cdot 10^{38}$ (340 undecilhões) de endereços IP.

Entretanto, o IPv6 possui desvantagens no âmbito da IoT, pois apresenta consumo de memória elevado e os nós presentes em redes IoT, em sua maioria, são caracterizados por portar limitações de *hardware* e de processamento. Desta forma, torna-se imprescindível a criação de um protocolo capaz de suprir as necessidades de um sistema IoT. Tal circunstância deu origem ao *IPv6 over Low power Wireless Personal Area Networks* (6LoWPAN), que significa IPv6 em Redes pessoais sem fio de baixa potência.

Para SANTOS (2014) o 6LoWPAN nada mais é que a junção do protocolo IPv6 com o padrão IEEE 802.15.4. Segundo Mulligan (2007) o conceito de 6LoWPAN surgiu da ideia de que até mesmo o menor dos dispositivos poderia e deveria aplicar o protocolo IP. Ele utiliza os conceitos usados no IPv6 de forma simplificada e possui cabeçalhos muito compactos, sendo que a taxa de compressão é de 1/10. A técnica de compressão usada no 6LoWPAN é denominada *stateless* e pode chegar à comprimir 40 *bytes* do IPv6 em 2 *bytes*.

Uma rede que implementa 6LoWPAN requer um *Gateway* para integração com a Internet, e que é responsável por comprimir e descomprimir o protocolo IPv6. Ao contrário do IPv4 que define um único cabeçalho, assim como o IPv6 o 6LoWPAN usa cabeçalhos empilhados, os quais detêm quatro tipos:

- Cabeçalho de envio;
- Cabeçalho de compressão de cabeçalho IPv6;
- Cabeçalho de fragmentação;
- Cabeçalho de Mesh.

O 6LoWPAN tem se tornado o protocolo padrão no que se refere a endereçamento das *Appliances* em geral devido às suas vantagens, das quais podemos citar: compressão do cabeçalho TCP/IP e das mensagens, fragmentação e reagrupamento de pacotes, entre outras. A Zigbee Alliance, que corresponde a um consórcio de empresas que trabalha com aplicações de

automação e redes de sensores e que determina padrões amplamente utilizados, anunciou que abriria mão de seu padrão e adotaria um padrão aberto baseado em IPv6 e 6LoWPAN como alternativa de endereçamento [1]. Assim, dadas suas vantagens e relevância, o mesmo será usado neste trabalho para endereçamento dos dispositivos simulados/emulados.

3.1.5 RPL

O *Routing Protocol Low Power and Lossy Networks* (RPL) é um protocolo criado pelo grupo IETF ROLL (*Internet Engineering Task Force Routing Over Low power and Lossy networks*) (GAO; ZHENG; HUO, 2018) e, assim como o CTP, é um protocolo de roteamento de vetor de distância e implementa uma arquitetura em árvore. A estrutura do RPL é conhecida como DODAG (acrônimo para *Destination Oriented Directed Acyclic Graph*), nela um dos componentes da rede age como nó raiz concentrando o recebimento de dados dos outros nós (LIMA, 2018). Em contrapartida, se distingue dos protocolos com estrutura em árvore tradicionais por seus nós serem capazes de associar-se a vários nós. O RPL será utilizado para mapeamento da rede neste trabalho.

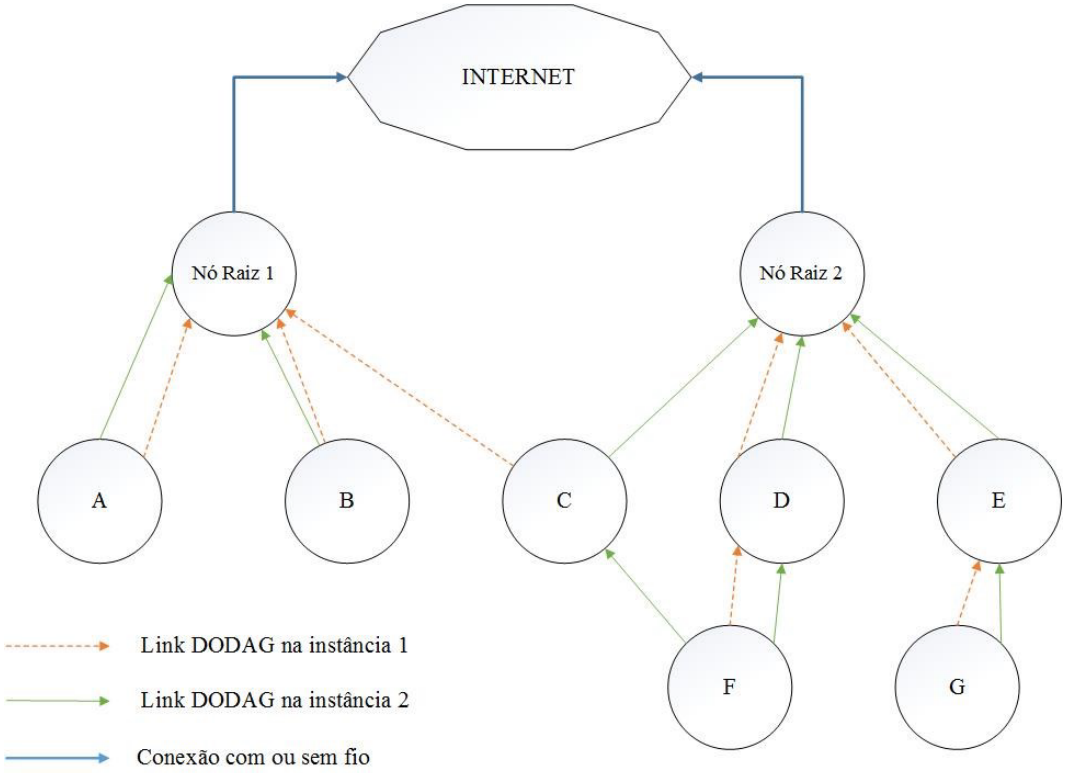
Uma rede RPL é constituída por uma ou mais DODAGs e que formam uma instância, a qual recebe um identificador único, denominado *RPLInstanceID*. Geralmente várias instâncias do RPL são implementadas para dar suporte a diferentes aplicações, entretanto, vale salientar que em uma instância específica, um nó só pode pertencer a uma DODAG. A Figura 5 mostra uma rede RPL em que há 2 DODAGs e 2 instâncias.

O RPL utiliza quatro tipos de mensagens ICMPv6 com o intuito de manter e atualizar as rotas da rede, que serão descritas a seguir:

- DODAG *Information Object* (DIO): Contém informações sobre o DODAG, como, *RPLInstanceID*, parâmetros de configuração e métricas utilizadas para geração de rotas. Mensagens DIO são enviadas por *broadcast* através da rede para construir e manter o DODAG e somente o nó raiz é o único que pode começar a disseminar este tipo de mensagem.
- DOGAG *Information Solicitation* (DIS): Possibilita a descoberta de vizinhos, pois solicita aos nós da DODAG informações sobre os objetos da rede, recebendo como resposta um pacote DIO.
- *Destination Advertisement Object* (DAO): Realiza a propagação das informações

[1] Disponível em: http://ipv6.br/post/zigbee-usa-agora-6lowpan-sua-proxima-lampada-tera-ipv6/

Figura 5 – Rede RPL com 2 DODAGs e 2 instâncias



Fonte: Braga et al. (2018)

de roteamento de forma ascendente, ou seja, dos nós para a raiz possibilitando a cada nó montar uma tabela de roteamento mais completa e eficaz.

- *Destination Advertisement Object Acknowledgment* (DAO-ACK): Esta mensagem nada mais é que a confirmação de que o destinatário DAO recebeu o pacote a ele enviado.

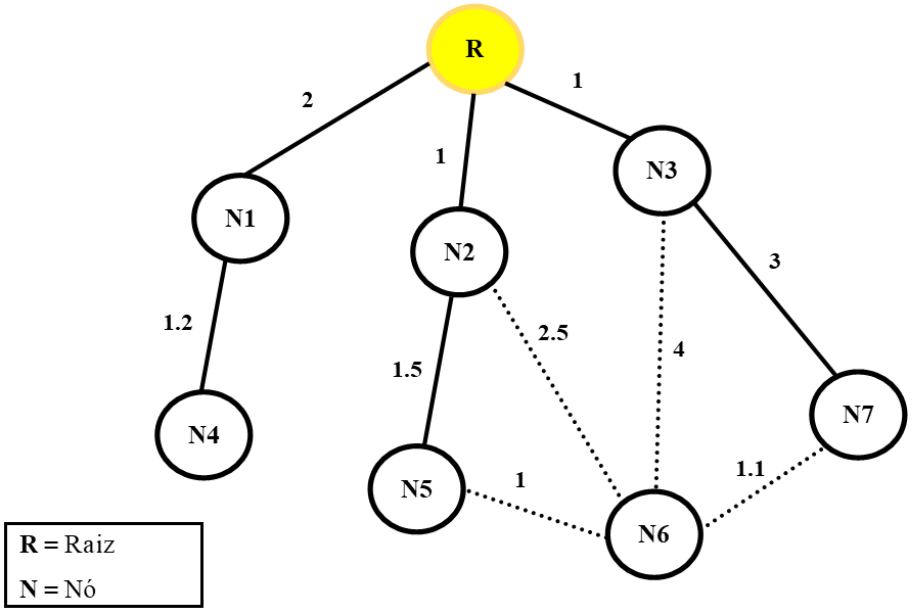
3.1.6 CTP

O *Collection Tree Protocol* (CTP) é um protocolo de roteamento de vetor de distância, projetado para Redes de Sensores Sem Fio (RSSF), que organiza a rede em uma estrutura semelhante a uma árvore e tem como objetivo principal designar o melhor caminho dos nós fonte ao nó raiz (BRAGA et al., 2018). Entretanto, neste trabalho o CTP não será utilizado para mapeamento dos dispositivos presentes na rede, pois sua implementação no Cooja não foi realizada com êxito, pelo fato de não ter exemplos factuais de execução no Cooja. No CTP os nós fonte são os sensores e o nó raiz é o nó responsável pela coleta de informações e está situado no início da árvore.

Na arquitetura em árvore do CTP os sensores atuam como as folhas da árvore e

coletam dados do ambiente externo e os transmitem para o próximo sensor até que a informação chegue ao nó raiz. Entretanto os nós não escolhem o próximo salto de forma particular, mas utilizam um gradiente de rotas denominado *Expected Transmission* (ETX) .

Figura 6 – Seleção de parentes CTP



Fonte: Adaptado de Braga et al. (2018)

Segundo Braga et al. (2018) o ETX é um indicador de qualidade do link cujo valor é calculado como o número de transmissões necessárias para que um nó envie um pacote *unicast* (pacote endereçado a um único destino) a seu vizinho e este seja recebido com sucesso. Cabe lembrar que cada link possui um ETX próprio. No início da configuração da rede os nós raiz enviam um alerta de identificação deles mesmos na rede, a partir desse alerta os nós fonte definem o ETX de seus links e utilizam o mesmo para escolher o próximo salto em direção ao nó raiz, vale destacar que o nó raiz possui ETX igual a 0.

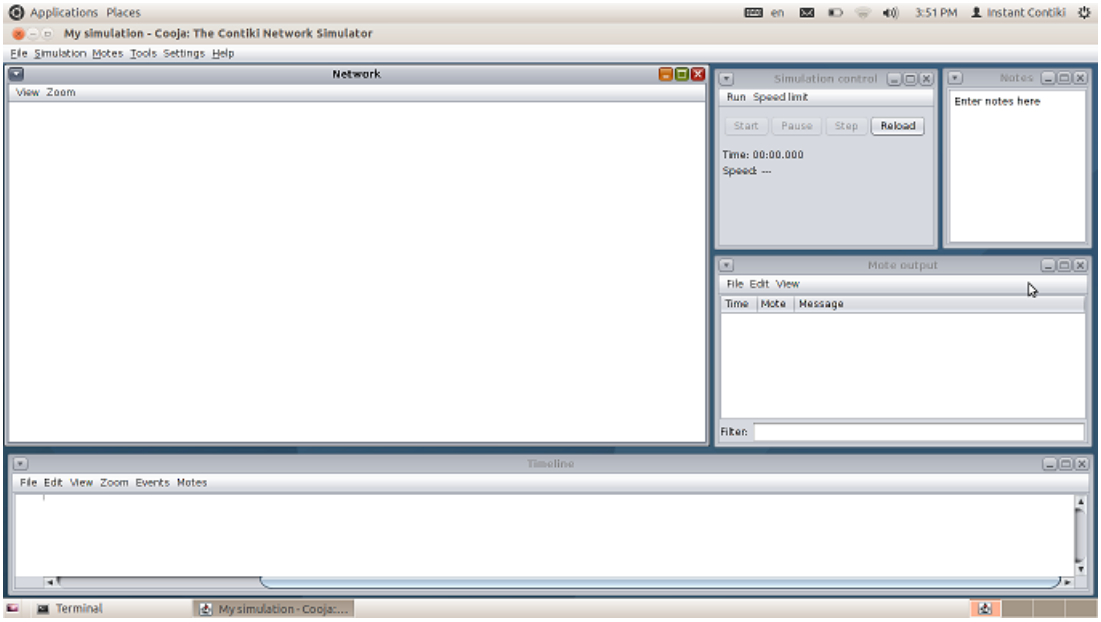
A Figura 6 mostra as diferentes rotas do nó 6 até a raiz. A partir dela pode-se observar que o nó 6 é vizinho e está em comunicação com os nós 2,3,5 e 7 e possuem os seguintes valores ETX: ETX_2=2.5, ETX_3=4, ETX_5=1 e ETX_7=1.1. Como já mencionado no decorrer do texto um nó sempre escolhe o próximo salto através do menor valor de ETX, consequentemente, o nó 6 escolherá o nó 5 como seu parente. Experimentos com o CTP influenciaram o desenvolvimento do Protocolo RPL (GNAWALI et al., 2009).

3.2 Contiki OS/Cooja

O Contiki OS é um sistema operacional de código fonte aberto destinado a IoT, que conecta pequenos microcontroladores de baixo custo e de baixo consumo à Internet (CONTIKI, 2019). É um sistema baseado na linguagem C e possui uma série de ferramentas para a construção de sistemas sem fio complexos, o mesmo é projetado para sistemas de baixíssima potência que podem precisar operar durante anos com um par de baterias AA.

O Cooja (Contiki OS Java) é um simulador/emulador baseado em Java presente no Contiki OS que permite aos desenvolvedores trabalhar com mais facilidade, visto que o ambiente de simulação da ferramenta possibilita aos mesmos ver suas aplicações sendo executados em redes de larga escala e com detalhes extremos em dispositivos de *hardware* totalmente emulados (CONTIKI, 2019). A emulação proporciona a um sistema de computador (denominado *host*) a habilidade de se comportar como um outro sistema de computador (denominado *guest*) (SANTOS et al., 2016).

Figura 7 – Simulador/Emulador Cooja



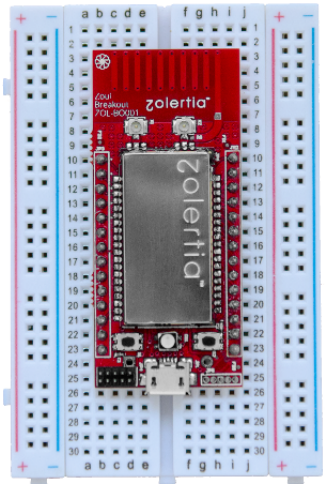
Fonte: Próprio Autor (2022)

As simulações e emulações são muito importantes na fase de avaliação de arquiteturas e protocolos de redes, pela capacidade de trazer situações do mundo real que não seriam possíveis quantificar ou qualificar de forma fidedigna, ademais, o processo de simulação possui inúmeras vantagens: menor custo; capacidade de previsão de desempenho; adaptabilidade do modelo às

variações de condições do teste e operação, entre outros (LIMA, 2018).

Por mais que seja implementado na linguagem Java o Cooja (o simulador/emulador Cooja pode ser visto na Figura 7) propicia o uso da linguagem de programação C no *software* de seus nós, além disso, uma série de bibliotecas Contiki diferentes podem ser carregadas simultaneamente em uma única simulação, devido a biblioteca *Java Native Interface (JNI)*, nativa do próprio simulador (GOIS; LIMA, 2014). No simulador cada nó possui um tipo, uma memória de dados e n periféricos e diversos relatórios são gerados a respeito do *hardware* e da comunicação, podendo ser visualizados em tempo real e salvos para análise posterior. A biblioteca JNI outorga ao simulador controle total sobre a memória dos nós, o que concede ao simulador visualizar e/ou alterar as variáveis da simulação.

Figura 8 – Zolertia Z1



Fonte: Zolertia (2010)

Uma rede de sensores sem fio (RSSF) é uma rede sem fio que consiste em dispositivos autônomos distribuídos espacialmente usando sensores para monitorar condições físicas ou ambientais. Estes dispositivos podem se comunicar entre si por meio de uma rede ad-hoc sem fio, sem a necessidade de uma estação base, ou diretamente a um gateway. Neste trabalho escolheu-se usar *Appliances* do tipo Z1, utilizados em sensores de gás e de incêndio, temperatura e umidade, luminosidade e movimento (ZOLERTIA, 2022), por exemplo (Figura 8) em redes RSSF em *Smart Home*. O Z1 apresenta melhor desempenho do que os nós do tipo *Sky* mote, de acordo com o trabalho (SHAH; JINWALA, 2018), pois possui memória RAM de 8 KB e memória *flash* de 92 KB, segundo Musaddiq et al. (2020), Zolertia (2010). O *Sky* mote, bastante utilizado pela comunidade, possui uma memória RAM de 10 KB e memória flash de 48 KB, no entanto,

mesmo dispondo de mais RAM o Z1 se sobressai a ele por utilizar menos ciclos de CPU e usar menos memória RAM e flash, segundo Shah e Jinwala (2018). Assim, o Zolertia Z1 é menos suscetível a erros de alocação de memória para os cenários do que o *Sky* mote.

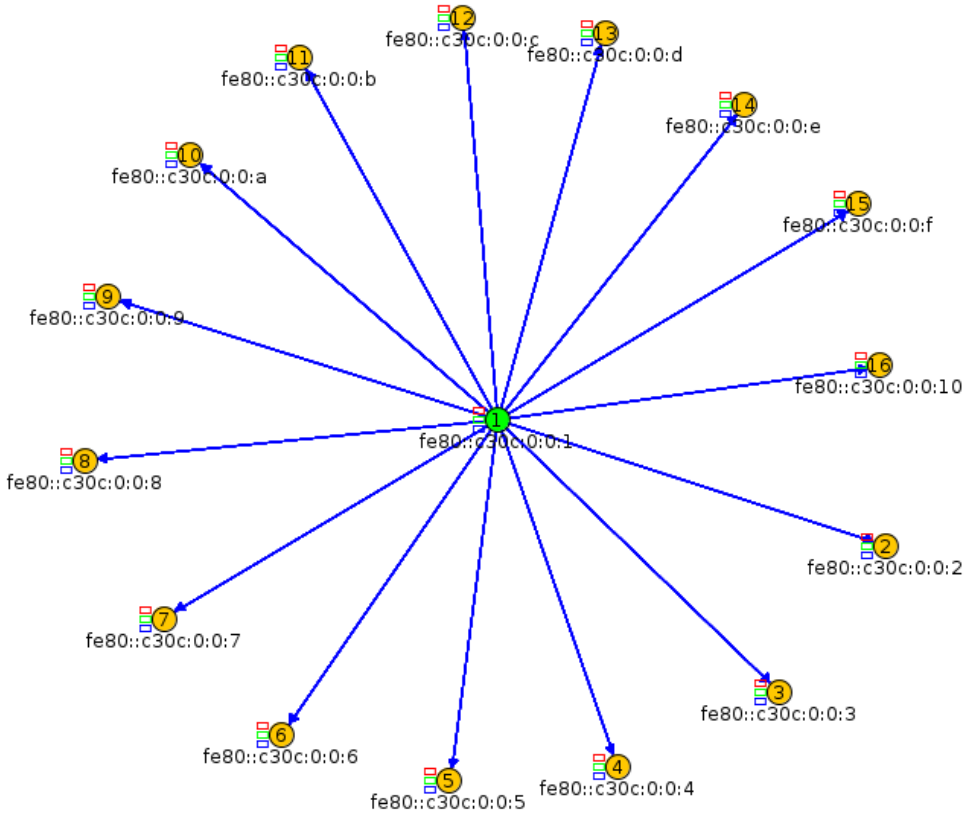
4 METODOLOGIA

Este Capítulo aborda as etapas cruciais que foram realizadas com a finalidade de responder a questão de pesquisa, as etapas se referem a organização dos cenários (Seção 4.1), a definição dos fatores/parâmetros e níveis (Seção 4.2), às configurações da máquina de execução dos cenários (Seção 4.3) e por último ao cálculo do intervalo de confiança e do consumo energético (Seção 4.4).

4.1 Organização dos cenários

Este trabalho consiste na comparação de três protocolos de aplicação, CoAP, MQTT-SN e HTTP, em ambiente simulado de IoT, direcionado para *Smart Homes*, com o propósito de observar qual dos protocolos de aplicação supramencionados apontam a melhor eficiência energética, o menor índice de perda de pacotes e a menor latência. Foi selecionada a topologia estrela tomando como base o trabalho de Stolojescu-Crisan, Crisan e Butunoi (2021).

Figura 9 – Modelo dos cenários



Fonte: Adaptado de Stolojescu-Crisan, Crisan e Butunoi (2021)

A escolha da topologia estrela deu-se por ser a estrutura mais comum em redes

domésticas. Nela existe um dispositivo central, encarregado de fazer a ponte de comunicação entre a rede local (LAN) e a Internet, sendo capaz de conectar vários outros dispositivos.

A topologia estrela deste trabalho é composta por um nó central executando o protocolo RPL, ficando responsável por interligar a rede interna composta por cenários com 5, 15 e 30 nós à rede externa. Possibilizando assim, a troca de informações entre a rede externa e os nós simulados. A estrutura base pode ser visualizada na Figura 9, em que o nó 1, em cor verde, é o roteador de borda da rede simulada, executando o protocolo RPL para realizar o tunelamento entre rede interna e externa. Os nós de 2 à 16, em cor amarela, simulam as máquinas em uma *Smart Home*.

Quadro 4 – Exemplo do mapeamento de endereços da rede interna para a rede externa

Endereço rede interna	**Endereço rede externa**	**Número do nó**
fe80::c30c:0:0:1	aaaa::c30c:0:0:1	1
fe80::c30c:0:0:2	aaaa::c30c:0:0:2	2
fe80::c30c:0:0:3	aaaa::c30c:0:0:3	3
fe80::c30c:0:0:4	aaaa::c30c:0:0:4	4
fe80::c30c:0:0:5	aaaa::c30c:0:0:5	5
fe80::c30c:0:0:6	aaaa::c30c:0:0:6	6
fe80::c30c:0:0:7	aaaa::c30c:0:0:7	7
fe80::c30c:0:0:8	aaaa::c30c:0:0:8	8
fe80::c30c:0:0:9	aaaa::c30c:0:0:9	9
fe80::c30c:0:0:a	aaaa::c30c:0:0:a	10
fe80::c30c:0:0:b	aaaa::c30c:0:0:b	11
fe80::c30c:0:0:c	aaaa::c30c:0:0:c	12
fe80::c30c:0:0:d	aaaa::c30c:0:0:d	13
fe80::c30c:0:0:e	aaaa::c30c:0:0:e	14
fe80::c30c:0:0:f	aaaa::c30c:0:0:f	15
fe80::c30c:0:0:10	aaaa::c30c:0:0:10	16

Fonte: Próprio Autor (2022).

Adicionalmente, a informação de como o roteador de borda RPL realiza o mapeamento foi de suma importância para o envio/recebimento de dados da rede externa para a rede interna. O Quadro 4 mostra como ocorre a correspondência de endereçamento realizado pelo nó RPL. Sendo o nó 1 o roteador RPL, quando algum pedido de requisição ou obtenção de dados chega vindo da rede externa com endereço no padrão aaaa::c30c:0:0:x o roteador faz o mapeamento para o nó com endereço fe80::c30c:0:0:x.

As subseções 4.1.1, 4.1.2 e 4.1.3 detalham como os cenários dos protocolos CoAP,

MQTT-SN e HTTP, foram elaborados, partindo sempre do modelo visto na Figura 9. Vale salientar que todos os *scripts* deste trabalho estão disponíveis no github[1]. Outro ponto é que os cenários propostos nas imagens das subseções relatadas são criados levando em consideração o cenário com 15 nós IoT, mas ao todo são utilizados cenários com 5, 15 e 30 nós IoT nas experimentações. E meio de acesso é uma rede de sensores sem fio (RSSF).

4.1.1 Cenário CoAP

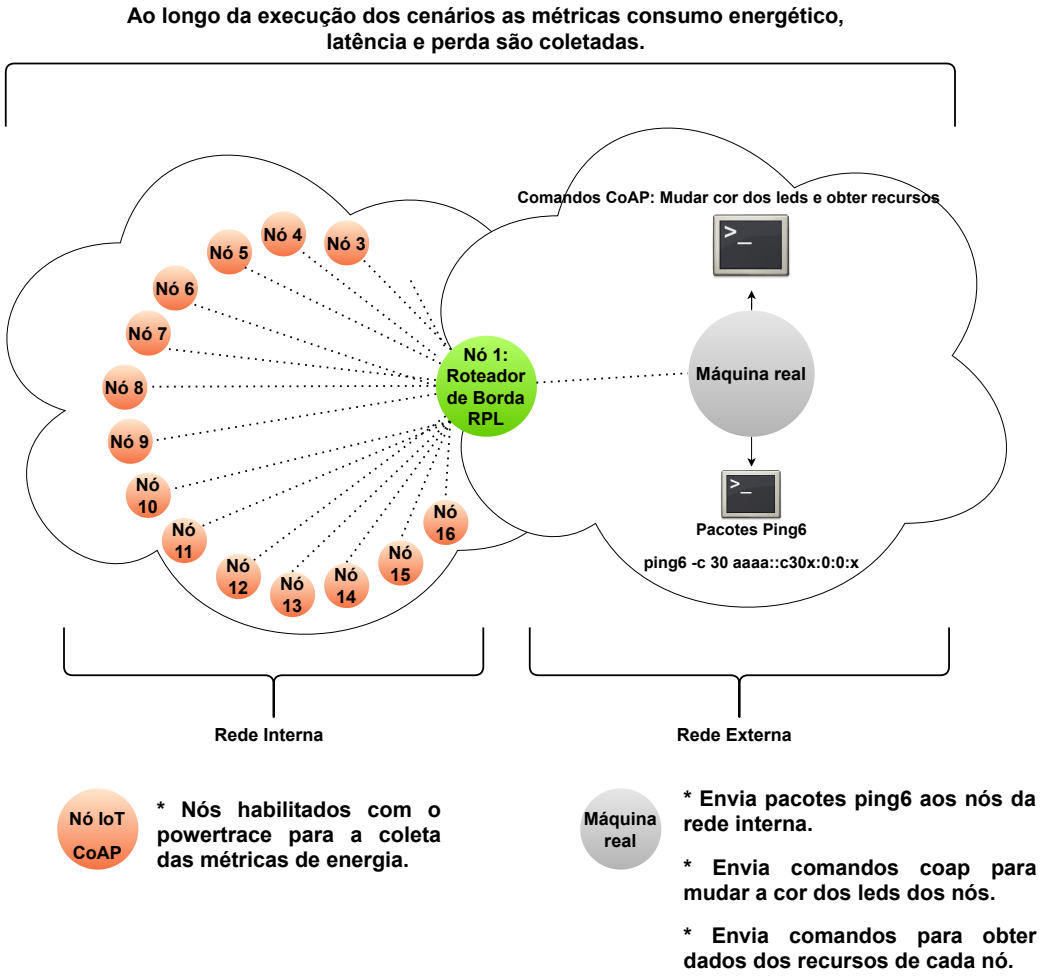
O cenário de execução do CoAP é exibido na Figura 10. Neste exemplo, está sendo utilizado um nó central executando o protocolo RPL, para a interligação com a rede externa em que reside a máquina real.

Na máquina real foram criados dois *scripts*: um destinado ao envio de pacotes *ping6* aos nós da rede interna, a fim de obter dados referentes a latência e a perda de pacotes, e o outro *script* referente a troca de informações entre a máquina real e os nós IoT do tipo CoAP, ou seja, mensagens da camada de aplicação. Ao longo do período de execução de cada cenário esse *script* envia uma quantidade aleatória de pacotes *ping6* aos nós da rede interna, por exemplo: ao longo da simulação o *script* envia um número x de pacotes aleatórios a um determinado nó da rede, escolhido de modo aleatório, para capturar a latência.

No apêndice A a Figura 17 mostra quais campos referentes a saída do comando *ping6* foram usados para a obtenção dos gráficos de perda e latência. Em continuação, o segundo *script* fica com a funcionalidade do envio e recebimentos de dados da rede externa para a interna. No mesmo modelo de envio dos pacotes *ping6*, enviando comandos para acender ou apagar os *leds* de modo aleatório, e após isso envia o comando para obter os dados de todos os recursos presentes no nó. Por fim, os nós CoAP sendo habilitados com o *powertrace* ficam coletando os dados de consumo energético ao longo de todo o experimento.

[1] Disponível em: https://github.com/AbimaelSB/TCC2_ABIMAEL

Figura 10 – CoAP



Fonte: Próprio Autor (2022)

4.1.2 Cenário MQTT-SN

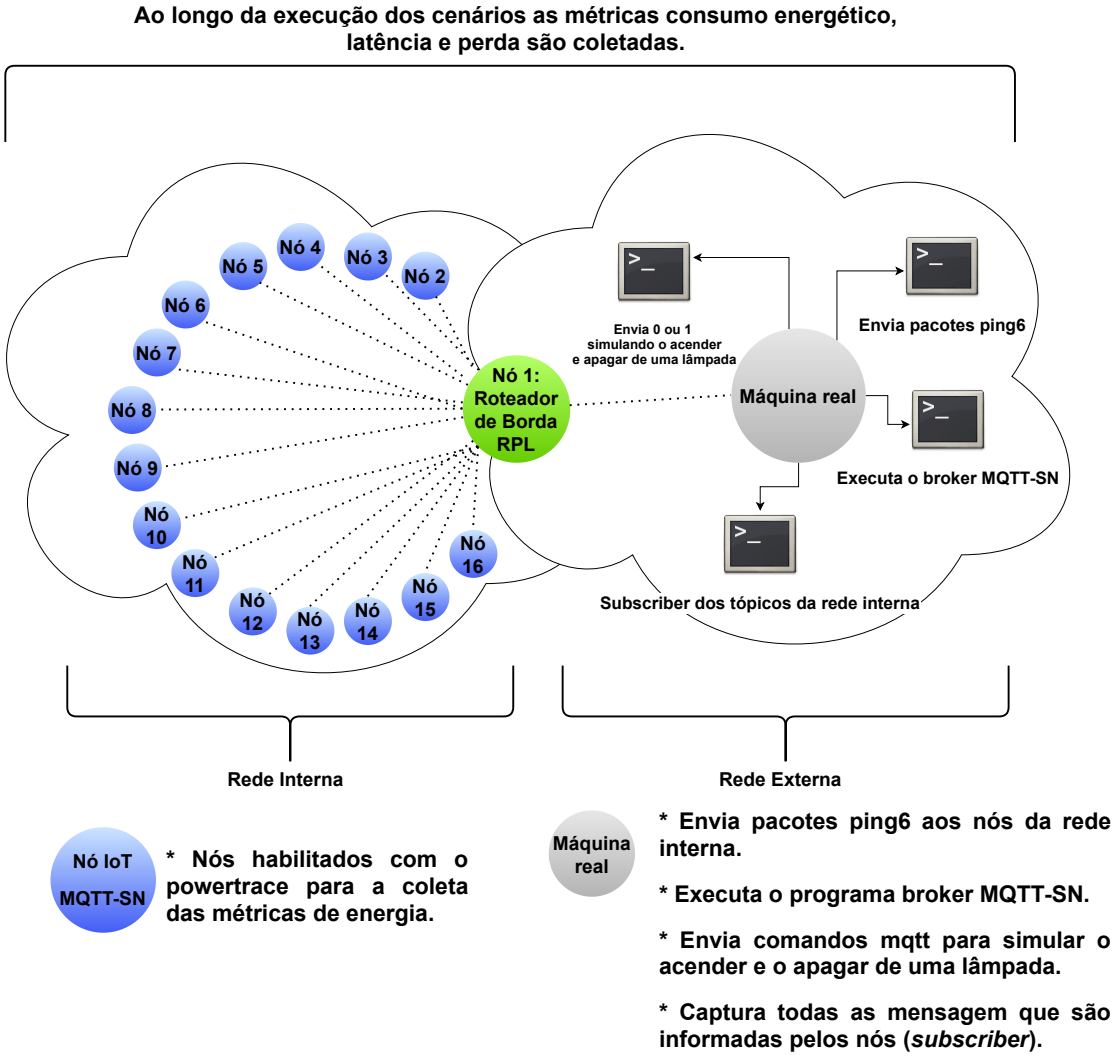
A Figura 11 mostrar como está organizado o cenário do MQTT-SN. Neste exemplo temos nós do tipo MQTT-SN ligados a um roteador de borda RPL. Na rede externa a máquina real fica encarregada de executar o *broker* MQTT-SN que é o responsável por repassar os dados que são publicados para os inscritos nos tópicos de interesse.

Ainda, na máquina real é executado por outro terminal o comando para se inscrever ao tópico de interesse, este tópico tem como publicadores todos os nós MQTT-SN da rede interna. Ademais, em outro terminal é executado o *script* que simula o acender e o apagar de uma lâmpada, a fim de enviar dados aos nós MQTT-SN da rede interna.

Por fim, também é executado o *script* que envia pacotes *ping6* aos nós internos ao longo de todo o experimento. O funcionamento é o mesmo apresentado no cenário do CoAP, fazendo o envio de x pacotes *ping6* (modo aleatório), com o intuito de obter dados referentes a

latência e a perda de pacotes.

Figura 11 – MQTT-SN



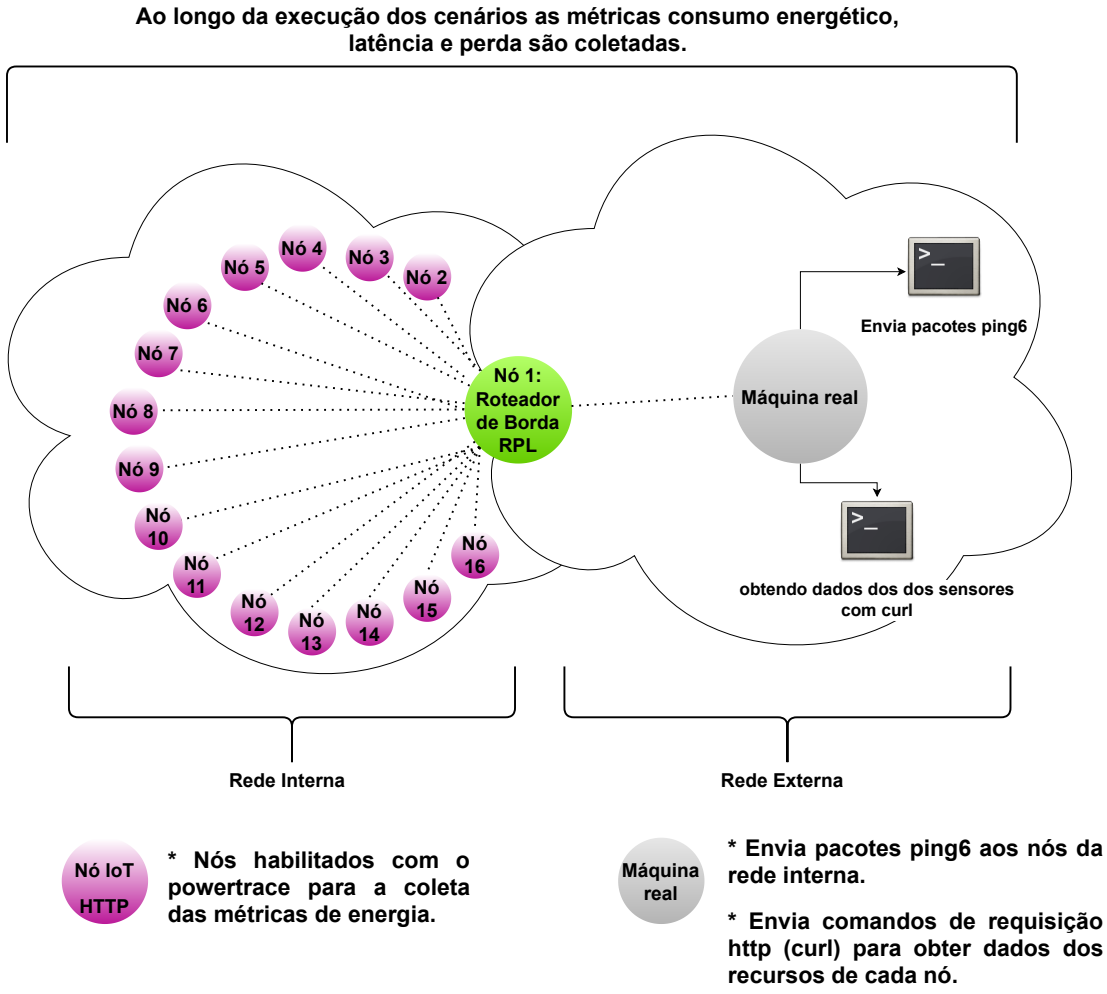
Fonte: Próprio Autor (2022)

4.1.3 Cenário HTTP

O cenário HTTP pode ser visualizado na Figura 12, tem como exemplo o cenário com nós que também são conectados a um roteador de borda RPL, tendo o papel de interligar a rede interna à rede externa. Neste cenário são executados apenas dois *scripts*, o de envio de pacotes *ping6* de maneira aleatória e o *script* que faz requisições HTTP aos nós da rede.

O funcionamento do *script* que faz o envio de pacotes *ping6*, na máquina real, segue o mesmo modelo dos *scripts* anteriores, realizando o envio de x pacotes *ping6*, com aleatoriedade, aos nós da rede interna.

Figura 12 – HTTP



Fonte: Próprio Autor (2022)

Por último, foi criado o *script* para fazer requisições HTTP utilizando do comando *curl*, e dessa maneira conseguir trafegar dados no cenário. Em contrapartida ao MQTT-SN e CoAP o protocolo HTTP não disponibiliza de um mecanismo que automatize o envio e recebimento de pacotes a outros nós. Para tornar o fluxo de dados persistente de modo automatizado, seria preciso criar uma API para ser consumida e assim deixar o fluxo automatizado.

4.2 Fatores e níveis do trabalho

Entende-se como fator/parâmetro as demais configurações que se diferenciam dentro do trabalho de experimentação, já os níveis constituem os valores ou ainda os estados que o fator vai assumir dentro do experimento. Para este trabalho foram determinados os seguintes fatores:

protocolo de aplicação, contendo os níveis, CoAP, MQTT-SN e HTTP; protocolo de transporte tendo o nível RPL; protocolo de rede com o nível 6LoWPAN; tipo de nó IoT possuindo o nível Zolertia; topologia da rede portando o nível estrela; o número de nós IoT apresentando os níveis de 5, 15 e 30; o tipo de tráfego de dados sendo aleatório; O número de execução de cada cenário é 30; e por fim, o tempo de execução de cada cenário foi definido o nível de 3 minutos. O Quadro 5 expõe a organização dos fatores e níveis do vigente trabalho.

Quadro 5 – Fatores e níveis

Fator/Parâmetros	**Nível**
Protocolo de aplicação	CoAP, MQTT-SN e HTTP
Protocolo de roteamento	RPL
Protocolo de rede	6LoWPAN
Tipo de nó IoT	Zolertia
Topologia	Estrela
Número de nós	5, 15 e 30
Tipo de tráfego de dados	Aleatório
Número de execuções	30
Tempo de execução de cada cenário	3 minutos

Fonte: Próprio Autor (2022).

Havendo a prévia definição dos fatores e níveis é possível realizar o cálculo do número total de experimentos que serão feitos no trabalho. Este cálculo é mostrado na equação 4.1.

$$Experimentos = \text{PA} \times \text{PR} \times \text{PRE} \times \text{TN} \times \text{T} \times \text{N} \times \text{TF} \times \text{NE} \times \text{TE} \tag{4.1}$$

Sendo **PA**, **PR**, **PRE**, **TN**, **T**, **N**, **TF** e **NE**, e **TE** a quantidade total de níveis do protocolo de aplicação, protocolo de roteamento, protocolo de rede, tipo de nó IoT, topologia, número de nós, tipo de tráfego de dados, número de execuções e tempo de execução de cada cenário, respectivamente. Realizando o cálculo chega-se ao resultado de cada um dos 9 experimentos que foram executados com *seed* aleatória, isto é a organização dos nós a cada nova execução.

$$Tempo = \text{Experimentos} \times \text{T} \tag{4.2}$$

Para obter o tempo total de execução de todos os cenários é preciso multiplicar o número de **Experimentos** por **T** (tempo de execução de cada cenário) obtido pela multiplicação do número de execuções e pelo tempo de execução de cada cenário, nesse caso com resultado de 90 minutos. A equação 4.2 apresenta esse cálculo. E o resultado do tempo total de execução foi de 13 horas e 30 minutos.

4.3 Configurações da máquina de execução dos cenários

O Quadro 6 informa as principais especificações da máquina utilizada para executar os experimentos, que são: sistema operacional Linux, Ubuntu 16.04 LTS; memória RAM de 4 GB; processador Intel® Celeron(R) CPU N3060 @ 1.60GHz × 2; gráficos Intel® HD Graphics 400 (Braswell); tipo de sistema 64-bit e por fim, o disco tendo 128 *Gygabytes* de armazenamento.

Quadro 6 – Ambiente de trabalho

	Descrição
Sistema operacional	Linux, Ubuntu 16.04 LTS
Memória RAM	4 GB
Processador	Intel® Celeron(R) CPU N3060 @ 1.60GHz × 2
Gráficos	Intel® HD Graphics 400 (Braswell)
Tipo de sistema	64-bit
Disco	128 GB

Fonte: Próprio Autor (2022).

4.4 Cálculo do consumo energético

O Quadro 7 apresenta as especificações de energia do *mote Zolertia* (Z1), que detém uma tensão de 3 Volts (V), corrente TX de 17.4 Miliampere (mA), e corrente RX de 18.8 mA, modo de baixo consumo, do inglês *Low Power Mode* (LPM), com consumo de 0.020 mA e CPU com consumo de 0.0426 mA. O RTIMER_SECOND apresenta o valor de 32768 *ticks* por segundo. E por fim o RUNTIME que é o tempo que o *powertrace* faz a coleta das métricas energéticas é de 2 segundos.

Quadro 7 – Especificações do mote Zolertia (Z1)

	Mote Zolertia (Z1)
Voltage	3 Volts
Current TX	17.4 mA
Current RX	18.8 mA
LPM (Low Power Mode)	0.020 mA
CPU	0.0426 mA
RTIMER_SECOND	32768 ticks per second
RUNTIME	2 seconds

Fonte: Adaptado de Musaddiq et al. (2020)

As fórmulas usadas para calcular a eficiência energética são explicitadas nas equações : 4.3, 4.4, 4.5, 4.6, 4.7 e 4.8, que correspondem a fórmula geral do consumo energético (*Power Consumption*), LPM, CPU, TX, RX e Total, de modo respectivo. A equação geral 4.3 é usada como base para as demais fórmulas, sendo que nas outras fórmulas o que realmente varia é o valor *energest_value*, que sempre é definido como o segundo valor na linha temporal de coleta menos o primeiro valor de coleta na linha temporal, como por exemplo, o *energest_value* de LPM em um determinado espaço de tempo corresponde a $(lpm_2 - lpm_1)$. Após obter os demais valores energéticos aplica-se a equação 4.8 para calcular o consumo energético total.

$$\text{Power Consumption} = \frac{\text{energest\_value} \times \text{current} \times \text{voltage}}{\text{RTIMER\_SECOND} \times \text{RUNTIME}} \tag{4.3}$$

$$\text{LPM} = \frac{(lpm_2 - lpm_1) \times 0.020 \times 3}{32768 \times \text{RUNTIME}} \tag{4.4}$$

$$\text{CPU} = \frac{(cpu_2 - cpu_1) \times 0.426 \times 3}{32768 \times \text{RUNTIME}} \tag{4.5}$$

$$\text{TX} = \frac{(tx_2 - tx_1) \times 17.4 \times 3}{32768 \times \text{RUNTIME}} \tag{4.6}$$

$$\text{RX} = \frac{(rx_2 - rx_1) \times 18.8 \times 3}{32768 \times \text{RUNTIME}} \tag{4.7}$$

$$\text{Total} = \text{LPM} + \text{CPU} + \text{TX} + \text{RX} \tag{4.8}$$

A formula é usada para calcular o consumo energético dos nós individualmente. Por fim, um *script* é executado para realizar o cálculo da média amostral da execução atual e em seguida os dados são armazenados. Após todas as execuções é calculada a média destes valores.

No apêndice A a Figura 16 mostra a identificação dos campos CPU, LPM, RX e TX na saída da coleta do *powertrace*.

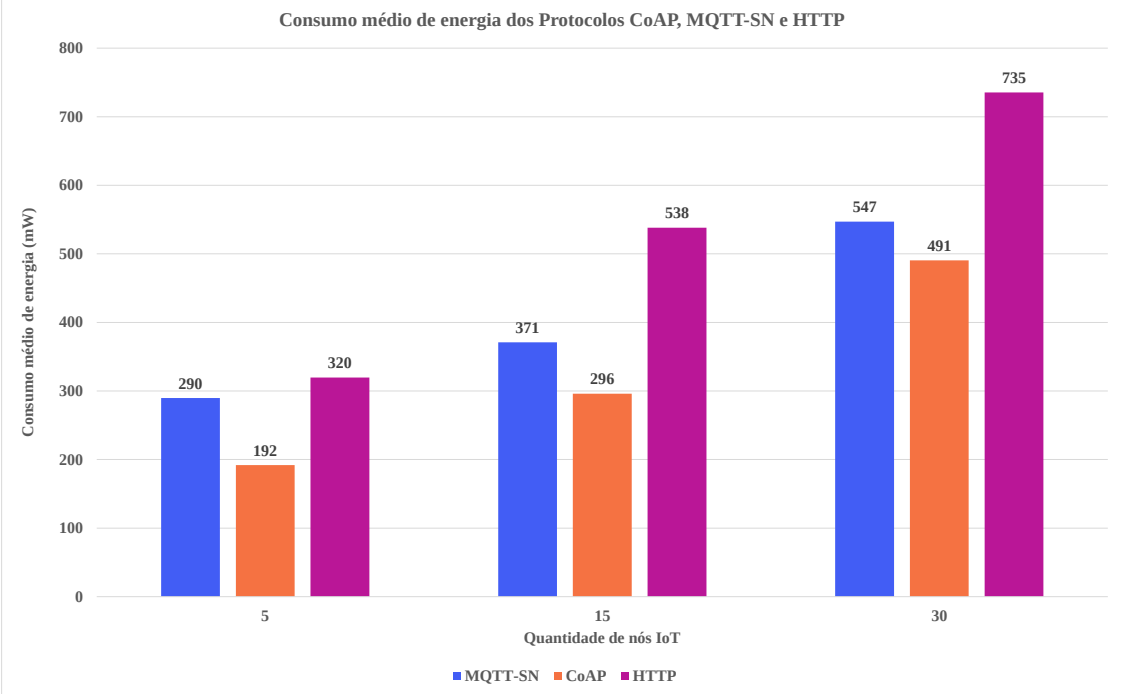
5 RESULTADOS E DISCUSSÃO

Esta Seção tem o objetivo de mostrar os resultados obtidos através de gráficos (ponto e barra), que foram criados utilizando-se do intervalo de confiança de 95%, como relatado no Capítulo 4 da metodologia deste trabalho. A seguir, na subseção 5.1, a descrição dos gráficos de consumo energético, latência e perda de pacotes é vista e discutida.

5.1 Consumo energético, latência e perda de pacotes

O primeiro gráfico da Figura 13 mostra o consumo energético médio com 5, 15 e 30 nós IoT. O consumo energético do protocolo CoAP foi um pouco menor do que os dos outros protocolos, em segundo ficou o protocolo MQTT-SN que teve um consumo levemente menor do que o do protocolo HTTP, que ficou em última posição.

Figura 13 – Consumo energético médio com 5, 15 e 30 nós IoT

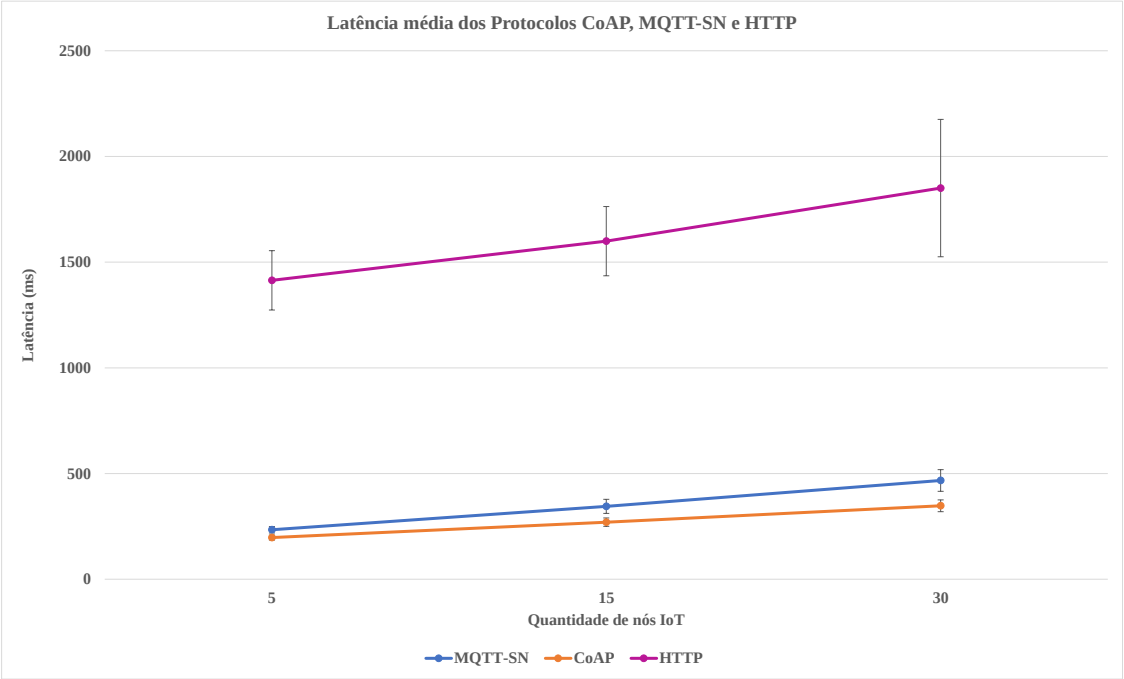


Fonte: Próprio Autor (2022).

O segundo gráfico representado na Figura 14 mostra a latência média com 5, 15 e 30 nós IoT. Nele o protocolo CoAP teve uma latência média levemente menor do que a latência

apresentada pelo protocolo MQTT-SN. Já o MQTT-SN juntamente com o CoAP foram superiores ao HTTP, que apresentou uma latência bem maior.

Figura 14 – Latência média com 5, 15 e 30 nós IoT

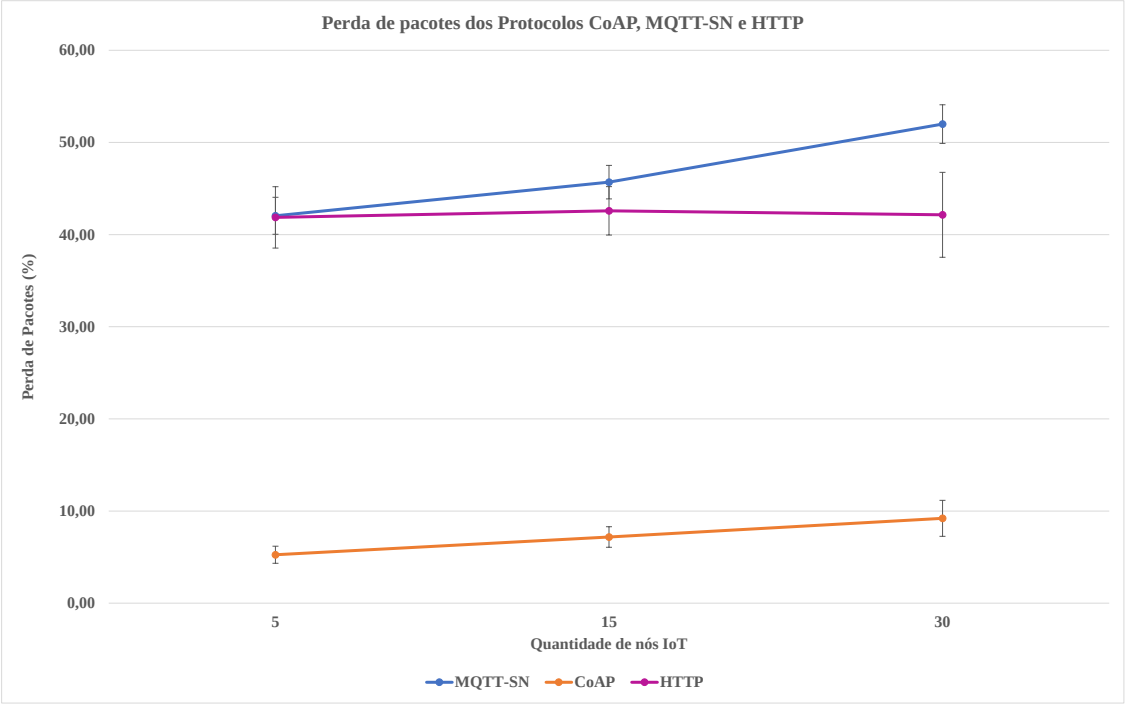


Fonte: Próprio Autor (2022).

O último gráfico representado na Figura 15 mostra o percentual de perda com 5, 15 e 30 nós IoT. Neste cenário podemos observar que o protocolo CoAP apresentou menor perda de pacotes dentre os três protocolos. Os protocolos MQTT-SN e HTTP apresentaram um percentual de perda superior, no entanto o HTTP apresentou menor taxa de perdas em relação ao MQTT-SN.

Em síntese, baseado nos experimentos realizados com os protocolos CoAP, MQTT-SN e HTTP, através das métricas predefinidas de consumo energético, latência e perda de pacotes, pode-se determinar que de maneira geral o protocolo CoAP se destaca, falando-se de todos os cenários, e comparando o resultado geral em relação aos demais protocolos (MQTT-SN e HTTP). Analisando de forma individual, o consumo energético, seguindo o desvio padrão, percebe-se que o CoAP é levemente melhor, mesmo seguido bem de perto pelo restante dos protocolos de aplicação comparados, ficando o MQTT-SN em segundo lugar e o HTTP em terceiro. Já no cenário que compara-se a latência, tem-se o CoAP com o tempo de ida e volta médio menor que

Figura 15 – Percentual de perda de pacotes com 5, 15 e 30 nós IoT



Fonte: Próprio Autor (2022).

o dos demais protocolos, o MQTT-SN segue próximo ao CoAP e no caso do HTTP o mesmo apresenta resultados bem mais distantes, com valor de latência bem alto comparado aos demais, comprovando que não é ideal para redes de sensores sem fio. No último gráfico tem-se a questão da perda de pacotes, em que o CoAP apresentou um menor percentual de perda de pacotes, o HTTP encontra-se em segundo lugar, tendo uma porcentagem de perda de pacotes menor do que o MQTT-SN, mesmo sendo um protocolo voltado para dispositivos mais robustos, e pressupõe-se que o MQTT-SN mesmo sendo uma versão modificada para redes de sensores sem fio, acabou por ter um número de perdas maior por trabalhar com TCP, tendo em vista que o CoAP trabalha com UDP e o HTTP com TCP. O TCP apresenta um número maior de perdas pelo fato de trabalhar com retransmissão de pacotes.

6 CONSIDERAÇÕES FINAIS

Com a popularização da Internet e os avanços tecnológicos, a maneira como lidamos com os objetos do cotidiano mudou, devido ao surgimento de objetos inteligentes, que munidos de recursos computacionais e de comunicação, nos levam a uma nova realidade: a revolução da Internet das Coisas. Novos protocolos surgiram com o intuito de resolver os problemas advindos das limitações dos dispositivos IoT, a fim de assegurar os requisitos fundamentais ao funcionamento destes (eficiência energética, adaptabilidade, escalabilidade e segurança). No entanto, surge também a questão: qual o melhor protocolo da camada de aplicação a ser usado por estes dispositivos?

Tendo em vista o paradigma IoT e a questão anteriormente levantada, o objetivo deste trabalho foi o de realizar uma análise comparativa entre três protocolos da camada de aplicação, CoAP, MQTT-SN e HTTP, em um ambiente simulado de IoT focado em *Smart Homes*, com o intuito de identificar o protocolo mais eficiente energeticamente, com menor índice de perda de pacotes e menor latência.

A fim de chegar-se a uma elucidação do objetivo geral, foram definidos os fatores e níveis do trabalho, os cenários foram elaborados e os experimentos executados, com as métricas: consumo energético, latência e perda coletadas ao longo dos experimentos. Após isso foram criados os gráficos de acordo com os dados das métricas, sempre utilizando um intervalo de confiança de 95%.

Por fim, ao analisar-se os gráficos podemos concluir que o protocolo CoAP se destaca em todos os cenários, comparando os seus resultados com os resultados dos demais protocolos (MQTT-SN e HTTP). No gráfico de consumo de energia médio mesmo tendo um empate técnico por conta do intervalo de confiança o CoAP ainda tem um consumo energético suavemente menor, e também no gráfico de latência média, apresentando valor menor do que os outros protocolos. No último gráfico de perda o CoAP foi o melhor em disparada, enquanto o MQTT-SN e o HTTP tiveram bastante perda de pacote, cujo motivo podemos supor que seja a utilização do TCP como protocolo de transporte, enquanto o CoAP utiliza UDP.

6.1 Trabalhos futuros

Ao decorrer da construção deste trabalho foram identificadas algumas alternativas para melhoria do mesmo. Com isso em mente, para trabalhos futuros tem-se como alternativa a

possibilidade de usar outro simulador/emulador que possua suporte a nós IoT, dado que o Cooja possui pouca documentação e algumas limitações. Uma limitação encontrada foi a ausência de exemplos práticos relacionados ao uso do protocolo CTP, protocolo descrito no Capítulo 3 subseção 3.1.6. Como resultado, ao usar outro simulador/emulador pode-se investigar se o mesmo possibiliza a execução de tal protocolo, tornando o uso do CTP uma sugestão para trabalhos posteriores, a fim de compará-lo com o protocolo RPL. Há também a possibilidade de utilizar um cenário maior, como o de um campus universitário, que se encaixa no contexto de *Smart Building* (prédio inteligente), que seria bastante pertinente.

REFERÊNCIAS

ANSARI, D. B.; REHMAN, A.; ALI, R. **Internet of things (iot) protocols**: a brief exploration of mqtt and coap. 2018. [S.l.:s.n]. 9–14 p.

ASHTON, K. **That 'Internet of Things' Thing**. 2009. Artigo publicado no RFID Journal.[s.n]. Disponível em: http://www.rfidjournal.com/articles/view?4986. Acesso em: 14 abr. 2019.

BAHIA, J. G.; CAMPISTA, M. E. M. **Um mecanismo de controle de demanda no provimento de serviços de IoT usando CoAP**. In: SBRC. [S.l.], 2017. v. 22, n. 1/2017. Anais do XXII Workshop de Gerência e Operação de Redes e Serviços (WGRS-SBRC 2017). In: SBRC. [S.l.], 2017. v. 22, n. 1/2017.

BARRERA, H. D. S. **A multilayered algorithm for topology control and self-healing on wireless sensor network**. [S.l.]: Pontificia Universidad Javeriana, 2017.

BERNERS-LEE, T.; FIELDING, R.; FRYSTYK, H. **Hypertext transfer protocol–HTTP/1.0**. 1996. [S.l.:s.n].

BRADBURY, R. **Crônicas Marcianas**. [S.l.]: Globo Livros, 2013. 196-202 p.

BRAGA, A. B. et al. **Análise de desempenho dos protocolos CTP e RPL em ambiente móvel**. 58 p. Monografia (Trabalho de Conclusão de Curso) — Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Patos de Minas, 2018.

CONTIKI. 2019. **Contiki**: The Open Source OS for the Internet of Things. [S.l]. Disponível em: http://www.contiki-os.org. Acesso em: 21 abr. 2019.

COSMI, A. B.; MOTA, V. F. **Uma Análise dos Protocolos de Comunicação para Internet das Coisas**. 2019. In: SBC. Anais do III Workshop de Computação Urbana. [S.l.], 2019. p. 153–166.

FERNANDES, F. L. S.; BRITO, J. M. C. **Análise de desempenho do canal de acesso aleatório para redes NB-IoT com três níveis de cobertura**. 2020. [S.l.:s.n].

FIELDING, R. et al. **Hypertext transfer protocol - HTTP/1.1**. 1999. [S.l.:s.n].

GAO, L.; ZHENG, Z.; HUO, M. **Improvement of RPL Protocol Algorithm for Smart Grid**. 2018. In: 2018 IEEE 18th International Conference on Communication Technology (ICCT). [S.l.: s.n.]. p. 927–930.

GNAWALI, O.; FONSECA, R.; JAMIESON, K.; MOSS, D.; LEVIS, P. **Collection Tree Protocol**. 2009. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09). [S.l.: s.n.].

GOIS, D. A. S.; LIMA, J. P. A. **Simulação de pilhas de protocolos para Internet das Coisas**. DCOMP-Departamento de Computação, Centro de Ciências Exatas e Tecnologia, Universidade Federal de Sergipe, São Cristóvão, 2014.

HARTKE, K. **Observing resources in the constrained application protocol (CoAP)**. 2015. [S.l.:s.n].

IBM; EUROTECH. **MQTT V3.1 Protocol Specification**. 2010. Disponível em: http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html. Acesso em: 11 maio 2019.

KUROSE, J. F.; ROSS, K. W.; ZUCCHI, W. L. **Redes de Computadores e a Internet**: uma abordagem top-down. [S.l.]: Pearson Addison Wesley, 2007.

LIMA, J. C. **Simulação de aplicações utilizando protocolos 6lowpan, rpl, mqtt e coap em smart cities**. 2018. Passo Fundo: Universidade de Passo Fundo, p. 1–56.

MARTINS, I. R.; ZEM, J. L. **Estudo dos protocolos de comunicação mqtt e coap para aplicações machine-to-machine e Internet das coisas**. 2016. Revista Tecnológica da Fatec Americana, v. 3, n. 1, p. 24.

MAZZER, D.; FRIGIERI, E. P.; PARREIRA, L. F. C. G. **Protocolos m2m para ambientes limitados no contexto do iot**: uma comparação de abordagens. [S.l]: Instituto Nacional de Telecomunicações - Inatel, 2013.

MULLIGAN, G. The 6lowpan architecture. In: **Proceedings of the 4th Workshop on Embedded Networked Sensors**. New York, NY, USA: ACM, 2007. (EmNets '07), p. 78–82.

MUSADDIQ, A.; ZIKRIA, Y. B.; KIM, S. W. et al. **Routing protocol for Low-Power and Lossy Networks for heterogeneous traffic network**. 2020. EURASIP Journal on Wireless Communications and Networking, SpringerOpen, v. 2020, n. 1, p. 1–23.

OLSSON, J. 6lowpan demystified. **Texas Instruments**, 2014. [S.l.:s.n], v. 13.

POINT, T. **HTTP - Overview**. 2021. [S.l]. Disponível em: https://www.tutorialspoint.com/http/http_overview.htm. Acesso em: 24 maio. 2020.

RIZZON, F.; BERTELLI, J.; MATTE, J.; GRAEBIN, R. E.; MACKE, J. 2017. Smart City: um conceito em construção. **Revista Metropolitana de Sustentabilidade** (ISSN 2318-3233), v. 7, n. 3, p. 123–142.

RODRIGUEZ, A. **Restful web services**: The basics. 2008. [S.l.:s.n], v. 33, p. 18.

SANTOS, B. P. et al. **Internet das coisas**: da teoria à prática. 2016. Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos, p. 1–50.

SANTOS, R. L. **Internet das Coisas e 6LoWPAN**. Universidade Tecnológica Federal do Paraná, Curitiba, 2014. Monografia (Especialização em Configuração e Gerenciamento de Servidores e Equipamentos de Redes).

SHAH, K.; JINWALA, D. C. **Performance analysis of symmetric key ciphers in linear and grid based sensor networks**. 2018. ArXiv preprint arXiv:1809.06587, [S.l.:s.n].

SHELBY, Z. et al. **The constrained application protocol (CoAP)**. [S.l.], 2014. Request for Comments 7252.

SINOPOLI, J. M. **Smart buildings systems for architects, owners and builders**. [S.l.]: Butterworth-Heinemann, 2009.

STOLOJESCU-CRISAN, C.; CRISAN, C.; BUTUNOI, B.-P. **An IoT-Based Smart Home Automation System**. 2021. Sensors, Multidisciplinary Digital Publishing Institute. [S.l], v. 21, n. 11, p. 3784.

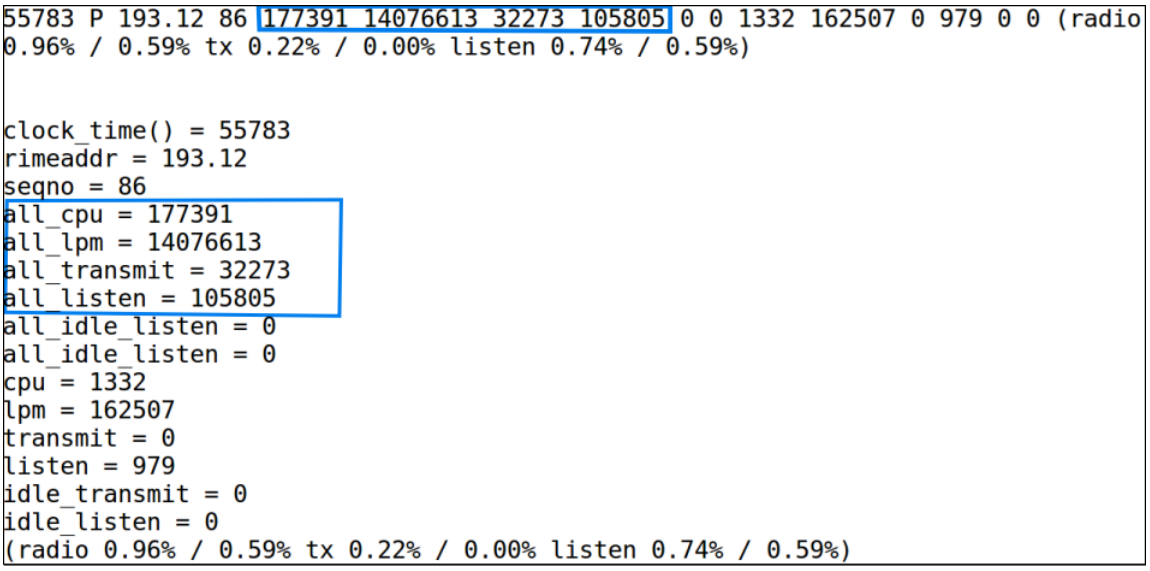
YASSEIN, M. B.; SHATNAWI, M. Q.; ALJWARNEH, S.; AL-HATMI, R. 2017. **Internet of Things**: survey and open issues of mqtt protocol. In: IEEE. 2017 International Conference on Engineering & MIS (ICEMIS). [S.l.]. p. 1–6.
Disponível em: https://ieeexplore.ieee.org/document/8273112. Acesso em: 23 maio. 2019.

ZOLERTIA. **Z1 by Zolertia is a low-power wireless sensor network (WSN)**. 2010. [S.l]. Disponível em: http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf. Acesso em: 22 ago. 2021.

ZOLERTIA. **Professioal sensor pack**. 2022. [S.l]. Disponível em: https://zolertia.io/product/professional-sensor-pack/. Acesso em: 22 maio. 2022.

APÊNDICE A – DADOS COLETADOS NOS EXPERIMENTOS

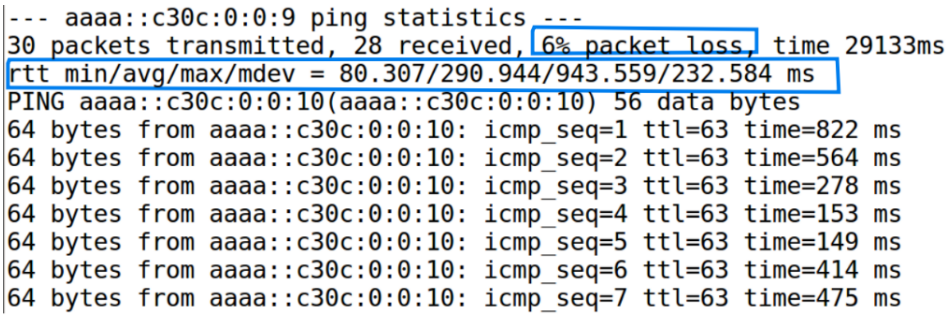
Figura 16 – Dados de saída do *powertrace*



Fonte: Próprio Autor (2022)

O comando da figura 16 é referente aos dados de saída que o *powertrace* coleta ao longo dos experimentos (primeira linha da imagem). Os dados que realmente foram utilizados neste trabalho estão destacados por um retângulo azul, e representam quatro campos do total apresentado. Abaixo da primeira linha foram descritos os nomes dos campos, e os campos que foram utilizados no presente trabalho foram *all_cpu* (CPU), *all_lpm* (LPM), *all_transmit* (TX) e *all_listen* (RX), que são os campos necessários para se calcular a eficiência energética.

Figura 17 – *Ping6* dados utilizados no trabalho



Fonte: Próprio Autor (2022)

A figura 17 mostra os campos que foram usados neste trabalho, ao final do envio de pacotes *ping6* o próprio utilitário do *ping6* disponibiliza os dados referentes para aquela etapa de envios. Os retângulos em azul da figura 17 são a perda de pacotes (6% *packet loss*) e a latência

ou tempo de ida e volta, do inglês *Round Trip Time* (RTT). Para o RTT foi-se utilizado o RTT médio (*avg*).