UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

SIMULAÇÃO DE REDES DE SENSORES SEM FIO UTILIZANDO PROTOCOLOS 6LOWPAN, RPL, MQTT E COAP EM SMART CITIES

João Cândido de Lima

Passo Fundo

2018

UNIVERSIDADE DE PASSO FUNDO
INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

SIMULAÇÃO DE APLICAÇÕES UTILIZANDO PROTOCOLOS 6LOWPAN, RPL, MQTT E COAP EM SMART CITIES

João Cândido de Lima

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada na Universidade de Passo Fundo.
Orientador: Dr. Marco Antônio Sandini Trentin

Passo Fundo
2018

CIP – Catalogação na Publicação

L732s Lima, João Cândido de
Simulação de aplicações utilizando protocolos 6LOWPAN, RPL, MQTT e COAP em Smart Cities / João Cândido de Lima. – 2018.
56 f. : il. color. ; 30 cm.

Orientador: Dr. Marco Antônio Sandini Trentin.

Dissertação (Mestrado em Computação Aplicada) – Universidade de Passo Fundo, 2018.

1. Softwares. 2. Programas de computador. 3. Redes de computação – Protocolos. 4. Smarts Cities. I. Trentin, Marco Antônio, orientador. II. Título.

CDU: 004.41

Catalogação: Bibliotecária Marciéli de Oliveira - CRB 10/2113

UPF Universidade de Passo Fundo | PPGCA Programa de Pós-Graduação em Computação Aplicada
Instituto de Ciências Exatas e Geociências - ICEG

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ACADÊMICO

JOÃO CÂNDIDO DE LIMA

Aos sete dias do mês de agosto do ano de dois mil e dezoito, às 14 horas, realizou-se, no Instituto de Ciências Exatas e Geociências, prédio B5, da Universidade de Passo Fundo, a sessão pública de defesa do Trabalho de Conclusão de Curso **"Simulação de redes de sensores sem fio utilizando protocolos 6lowpan, rpl, mqtt e coap em Smart Cities"**, de autoria de João Cândido de Lima, acadêmico do Curso de Mestrado em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada – PPGCA/UPF. Segundo as informações prestadas pelo Conselho de Pós-Graduação e constantes nos arquivos da Secretaria do PPGCA, o aluno preencheu os requisitos necessários para submeter seu trabalho à avaliação. A banca examinadora foi composta pelos doutores Marco Antônio Sandini Trentin, Marcelo Trindade Rebonatto e Gerson Antunes Soares. Concluídos os trabalhos de apresentação e arguição, a banca examinadora considerou o candidato APROVADO. Foi concedido o prazo de até quarenta e cinco (45) dias, conforme Regimento do PPGCA, para o acadêmico apresentar ao Conselho de Pós-Graduação o trabalho em sua redação definitiva, a fim de que sejam feitos os encaminhamentos necessários à emissão do Diploma de Mestre em Computação Aplicada. Para constar, foi lavrada a presente ata, que vai assinada pelos membros da banca examinadora e pela Coordenação do PPGCA.

Prof. Dr. Marco Antônio Sandini Trentin - UPF
Presidente da Banca Examinadora
(Orientador)

Prof. Dr. Marcelo Trindade Rebonatto - UPF
(Avaliador Interno)

Prof. Dr. Gerson Antunes Soares - Ulbra
(Avaliador Externo)

Prof. Dr. Rafael Rieder
Coordenador do PPGCA

RESUMO

Este trabalho apresenta a simulação dos protocolos de rede IoT, através da instalação do simulador COOJA no Ubuntu 12.04 LTS, emulando o sistema operacional contiki. O ambiente simulado consiste na composição de quadras de ruas urbanas com postes inteligentes que sejam capazes de transmitir dados de uma rede de sensores sem fio com temperatura, umidade e luminosidade, usando para isto os protocolos 6LoWPAN, RPL, MQTT e COAP, representando endereçamento da rede, composição da estrutura lógica da rede e transporte de dados em um ambiente de cidades inteligentes. Com o intuito de propor soluções para as cidades surgiu o conceito de *Smart Cities*, que utiliza as tecnologias da Internet das Coisas, ou como é conhecida mundialmente, *Internet of Things*. A IoT conecta uma gama de diferentes dispositivos, sensores, atuadores, softwares e objetos de forma que os mesmos conseguem interagir entre si nas mais diferentes regiões do planeta com o uso da rede mundial de computadores. Com a possibilidade de utilizar a comunicação com os mais variados mecanismos, o conceito de Machine to Machine (M2M) visa a comunicação de máquina a máquina e ganha força no mercado com a tendência de ser um padrão em construção de conceitos para *Smart Cities*. Os esforços gerados para o desenvolvimento de *Smart Cities* estão concentrados em aplicações para smartphones, e softwares para equipamentos em geral, mas pouco se tem feito em relação à comunicação entre os mais diversos objetos, uma vez que ao utilizar este conceito a quantidade de "coisas" conectadas aumenta e muito. Já é de conhecimento que o Protocolo IPv4 está há muito tempo com os blocos de endereçamento esgotados e a solução é migrar para o IPv6, porém quando o assunto é conectar coisas, tem-se que pensar na administração destes parâmetros de forma diferente, pois os dispositivos, em sua grande maioria, são alimentados por bateria e colocados em locais remotos, tornando, assim, a segurança mais frágil e a eficiência energética delicada. Logo, é de extrema importância estudar protocolos de comunicação para que se tenha sucesso com este tipo de conexão inovadora que o planeta está tendo que conviver em tempos exponenciais. Os resultados obtidos no presente estudo representam o tempo de composição da rede, tempo de endereçamento e características do transporte de dados na rede de sensores sem fio, nos quais os dados quantitativos foram tratados de forma estatística qualitativa e os dados de transporte foram tratados de forma qualitativa.

Palavras-chave: Internet das Coisas. Machine to Machine. Protocolos. Smart Cities.

ABSTRACT

This work presents the simulation of IoT network protocols, through the installation of the COOJA simulator in Ubuntu 12.04 LTS, emulating the contiki operating system. The simulated environment consists of the composition of urban street blocks with intelligent poles that are capable of transmitting data from a sensor network without temperature, humidity and luminosity using the 6LoWPAN, RPL, MQTT and COAP protocols, representing network addressing, composition of the logical structure of the network and transport of data in an environment of intelligent cities. With the objective of proposing solutions to the cities, the concept of Smart Cities emerged, using Internet technologies of Things or as Internet of Things is known worldwide. The IoT connects a range of different devices, sensors, actuators, software and objects in a way that can interact with each other in the most different regions of the planet using the world wide web. Regarding the possibility of using communication with the most varied mechanisms, the concept of Machine to Machine (M2M) aims at the communication of machine to machine and gains force in the market with the tendency to be a standard in construction of concepts for Smart Cities. The efforts generated for the development of Smart Cities are concentrated in applications for smartphones, and software for equipment in general, but little has been done in relation to the communication between the most diverse objects. Since in using this concept the amount of connected "things" increases and much. It is known that the IPv4 Protocol has long been with the address blocks depleted and the solution is to migrate to IPv6, but when it comes to connecting things, one has to think about the administration of these parameters differently, since the majority of devices are battery powered and placed in remote locations, thus making the security more fragile and the energy efficiency delicate, so it is of utmost importance to study communication protocols to be successful with this type of innovative connection that the planet is having to live in exponential times. The results obtained represent the network composition time, addressing time and data transport characteristics in the wireless sensor network, where quantitative data were treated in qualitative statistical form and the transport data were treated in a qualitative way.

Keywords: Internet of Things. Machine to Machine. Protocols. Smart Cities.

LISTA DE FIGURAS

Figura 1. Três Núcleos de funções de uma *Smart City* 15
Figura 2. Comunicação M2M e IoT 16
Figura 3. Foco da pesquisa 18
Figura 4. Infraestrutura de uma rede 6LoWPAN 20
Figura 5. Encadeamento de cabeçalhos 6LoWPAN 21
Figura 6. Cabeçalho IPv6 comprimido (HC1) 22
Figura 7. DAG DODAG 24
Figura 8. Tipos de respostas do protocolo CoAP 25
Figura 9. Formato de uma mensgem CoAP 25
Figura 10. Intersecção de ruas e nós de redes simulados 34
Figura 11. RSSF Simulada 36
Figura 12. Tmote Sky simulado no Cooja 37
Figura 13. Funcionamento do Roteador RPL 38
Figura 14. Estrutura da Simulação 40
Figura 15. Recorte da simulação das duas quadras. 42
Figura 16.Box Plot Enderecamento 43
Figura 17. DODAG Simulada 44
Figura 18. Tempo de construção da DODAG 45
Figura 19. 30 Simulações taxa de entrega com RPL 46
Figura 20. Boxplot Neighbor 48
Figura 21. Dados do sensor de luminosidade 49
Figura 22. Dados do sensor de temperatura 50
Figura 23. Dados do sensor de umidade 50
Figura 24. Média de consumo por nó 51
Figura 25. Tela de manipulação do CoAP 52
Figura 26. Informação do sensor de luminosidade 53

LISTA DE TABELAS

Tabela 1. Cabeçalho de tamanho fixo MQTT 28
Tabela 2. Tipos de Mensagens MQTT 28
Tabela 3. Comparativo entre os principais simuladores/emuladores para IoT 30
Tabela 4. Configuração do equipamento utilizado nas configurações 34
Tabela 5. Parâmetros do Simulador..... 38
Tabela 6. Tratamento de dados 6LoWPAN..... 41
Tabela 7. Tratamento de dados tempo DODAG 44
Tabela 9. Tratamento de dados para Neighbor..... 47

LISTA DE SIGLAS

6LoWPAN - IPv6 over Low power Wireless Personal Area Networks

BR - Border Router

COAP - Constrained Application Protocol

COOJA - Contiki OS Java

DAG - Direct Acyclic Graphs

DAO - Destination Advertisement Object

DODAG - Destination Oriented Directed Acyclic Graph

DODAG - Information Object (DIO)

DOGAG - Information Solicitation (DIS)

EUA -Estados Unidos da América

FTDI - Future Technology Devices International

HC - Header Compression

IoT - Internet of Things

LLN - Low Power and Lossy Network.

LSB - Bytes Menos Significativos

M2M - Machine-to-Machine

MQTT - Message Queuing Telemetry Transport

MSB - Bytes Mais Significativos

MTU - Max Transfer Unit

OF - Funções de Objetivo

OSI - Open System Interconnection

REST - Representational State Transfer

RFC - Request for Comments.

RLP - Recursive Length Prefix

ROLL - Routing Over Low Power and Lossy Networks

RPL - Routing Protocol Low Power and Lossy Networks

RSMB - Really Small Message Broker

RSSF - Rede de Sensores Sem Fio

TIC - Tecnologias de Informação e Comunicação

URI - Identificador Uniforme de Recurso

SUMÁRIO

1. INTRODUÇÃO 12

2. FUNDAMENTAÇÃO TEÓRICA 15

2.1. SMART CITIES 15

2.2. IOT E M2M 17

2.3. IMPORTÂNCIA DOS PROTOCOLOS USADOS EM IOT/M2M 18

2.3.1. Protocolo 6LoWPAN 19

2.3.2. Protocolo RPL 22

2.3.3. Protocolo CoAP 24

2.3.4. Protocolo MQTT 26

2.3.4.1. Cabeçalho fixo 27

2.3.4.2. Cabeçalho variável 29

2.4. O PROCESSO DE SIMULAÇÃO 30

2.4.1. Modelos de simulação 31

2.4.2. Sistema Operacional Contiki e o emulador / simulador COOJA 31

3. MÉTODOS 33

3.1. AMBIENTE DE SIMULAÇÃO 33

3.1.1. Ambiente Computacional 33

3.1.2. Cenário da simulação 34

3.1.3. Estrutura do Simulada 35

3.1.3.1. Postes 35

3.1.3.2. Tmote Sky 36

3.1.3.3. Roteador RPL Border Router 37

3.1.3.4. Parâmetros do simulador 38

3.1.3.5. Cálculo de confiança para os resultados apresentados 39

4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS 40

4.1. RESULTADOS DA COMPOSIÇÃO DA REDE 40

4.1.1. Endereçamento da rede 41

4.1.1.1. Discussão do resultado de endereçamento 6LoWPAN 42

4.1.2. Roteamento RPL 43

4.1.2.1. Discussão dos resultados – Tempo de formação DODAG 44

4.1.2.2. Taxa de Entrega RPL 45

4.1.2.3. Contagem de Neighbor 46

4.2. DADOS SIMULADOS NOS SENSORES 48

4.3. RESULTADOS COAP E MQTT 51
5. CONCLUSÃO.....53
REFERÊNCIAS 55

1. INTRODUÇÃO

Nos últimos anos as Redes de Sensores Sem Fio (RSSF) e outros sistemas pervasivos se tornaram muito populares, e com essa popularidade surgiu uma demanda significativa por comunicação de qualidade e em tempo real entre máquinas. Neste escopo são desenvolvidas tecnologias do tipo *Machine-to-Machine* (M2M), que possibilitam a transmissão de dados através de dispositivos remotos distintos sem a necessidade de intervenção humana, e estes, por sua vez, utilizam a captura de eventos registrados por sensores ou medidores, enviando o status de suas aferições para um sistema ou aplicação. Isto está sendo muito utilizado em diversos monitoramentos nas *Smart Cities* [36].

A publicação de dados para Cidades Inteligentes está alimentando um crescente ecossistema de serviços de valor para a sociedade envolvida. A colaboração aberta entre os atores do ecossistema (cidadãos, empresas, organizações e gerentes de cidade) permite o desenvolvimento de novos serviços baseados em dados os quais utilizam de forma significativa a Internet e os mais variados protocolos.

Esta integração ocorre com a *Internet of Things* (IoT), que possibilita a interação entre uma rede de objetos físicos bem como a transmissão de dados entre eles através da Internet [1]. Segundo [2], as propostas de uso da tecnologia de IoT são normalmente baseadas em infraestrutura centralizada e focadas na nuvem. Já as aplicações M2M são caracterizadas por um escopo limitado de tempo e espaço, onde os dados precisam ser processados somente quando e onde gerados, pois requerem um simples ciclo de informação, e muitas vezes interações altamente sensíveis ao tempo.

Avanços tecnológicos transformam os objetos que usamos em nossas vidas diárias em objetos inteligentes, ou seja, objetos regulares capacitados com recursos computacionais e de comunicação. As soluções inteligentes de automação residencial como termostatos inteligentes, luzes inteligentes, produtos de saúde inteligentes e sensores vestíveis que são comercializados hoje, são apenas alguns exemplos que representam o surgimento da revolução da IoT [37].

Um automóvel trafegando em uma rodovia pouco movimentada, pode se comunicar com um semáforo inteligente e informar em quanto tempo estará passando por ele, e desde que existam possibilidades físicas de não colisão, o sinal verde estará disponível no momento da passagem do veículo. O que foi apresentado neste caso foi uma comunicação máquina a máquina ou M2M usando tecnologia de IoT com objetos conectados.

Uma vez que vários objetos tais como Aplicações, Sensores, Atuadores e Dispositivos Vestíveis, entre muitos outros estão conectados, muitas vezes não temos nenhum padrão de interoperabilidade ao longo das plataformas de IoT, o que acaba gerando uma fragmentação de informação com altos custos de desenvolvimento e riscos maiores ainda para fornecedores e usuários, limitando a abrangência do mercado. Isso também ocorre devido ao fato de os protocolos de rede de IoT não serem padronizados, uma vez que muitas são as tecnologias desenvolvidas sem se levar em consideração a comunicação M2M. Hoje, o foco em IoT é no desenvolvimento de aplicações e não na interconexão, o que pode apresentar resultados onde a amplitude das oportunidades de desenvolvimento tecnológico fica limitada, especialmente em *Smart Cities*.

Segundo [19], um considerável esforço e pesquisas a respeito do protocolo IPv6 têm sido realizados no intuito de melhorar a intercomunicação de RSSF. Porém, até o momento vários dispositivos não sabem como trabalhar com os protocolos *Recursive Length Prefix* (RLP), IPv6 *over Low power Wireless Personal Area Networks* (6LoWPAN), *Constrained Application Protocol* (CoAP)*, Message Queuing Telemetry Transport* (MQTT), pois os esforços são concentrados no desenvolvimento de software e pouco tem sido feito no que diz respeito à comunicação de dados. Então, investigar a eficiência e a comunicação destes protocolos é extremamente justificável neste momento.

Neste contexto, esta pesquisa pretende responder a seguinte questão: como seria o comportamento de uma RSSF no que tange: a composição do endereçamento da rede, taxa de entrega baseada no protocolo de roteamento, o comportamento da transmissão de dados de seus sensores e como seria o comportamento dos protocolos mais conhecidos para este tipo de rede? Para isto o presente trabalho propõe uma simulação e extração de dados de uma rede RSSF.

O objetivo geral desse trabalho é avaliar o comportamento de uma rede de sensores montada em cima de postes de uma cidade, através de simulações, sendo que cada um deles conterá um dispositivo com sensores voltados para medir a temperatura, umidade e luminosidade.

Os objetivos específicos são: configurar o ambiente de simulação para os protocolos RPL, 6LoWPAN, CoAP e MQTT, considerando o transporte de dados de 3 sensores; averiguar o comportamento da simulação de trocas de mensagens com protocolo CoAP e a troca de mensagens do protocolo MQTT, ambos utilizando roteamento RPL e endereçamento 6LoWPAN; realizar experimentos com a abordagem proposta a partir de ensaios; e analisar os resultados e compará-los através métricas estabelecidas para o RPL. Os protocolos RPL e 6LowPAN, serão utilizados de forma operacional tendo como resultado a coleta de dados

quantitativos para tratamento. Já os protocolos CoAP e MQTT serão simulados de forma interativa com a finalidade de coleta de dados qualitativos para exposição.

A presente dissertação está estruturada da seguinte forma: serão 5 capítulos, no qual o capítulo 1 é apresentada a contextualização, motivação e objetivos. No capítulo 2 são apresentados os principais conceitos que fundamentam a dissertação, bem como a revisão literária, onde serão descritas as principais características dos protocolos utilizados na composição deste trabalho e a importância da simulação destes em larga escala. Também serão apresentados os principais simuladores de comunicação em rede suas funcionalidades mais relevantes, de forma a especificar cada um. No capítulo 3 serão apresentados os procedimentos metodológicos para a atingir os objetivos propostos no capítulo 1, tais como: ambiente de desenvolvimento da simulação, metodologia para a realização de experimentos e cenário da simulação. No capítulo 4 serão expostos os resultados obtidos bem como as discussões a partir deles. Por fim, no capítulo 5 será apresentará a conclusão, principais contribuições e trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem por finalidade apresentar os principais conceitos sobre os protocolos e simuladores pesquisados, bem como o entorno no qual essa tecnologia pesquisada se insere. Também visa obter um maior conhecimento acerca dos detalhes das tecnologias que serão empregadas na simulação, a fim de que se faça um uso adequado e correto das mesmas.

2.1. SMART CITIES

Conforme [10], uma *Smart City* deve coletar informações sobre si através de sensores, sistemas e todo dispositivo existente que proporcione informações úteis. Em seguida, os dados devem ser comunicados através do uso de redes com ou sem fio e, por fim, os dados devem ser analisados e interpretados para garantir benefícios aos habitantes.

Então [10], define os "Três Núcleos de funções de uma *Smart City*", conforme a Figura 1.

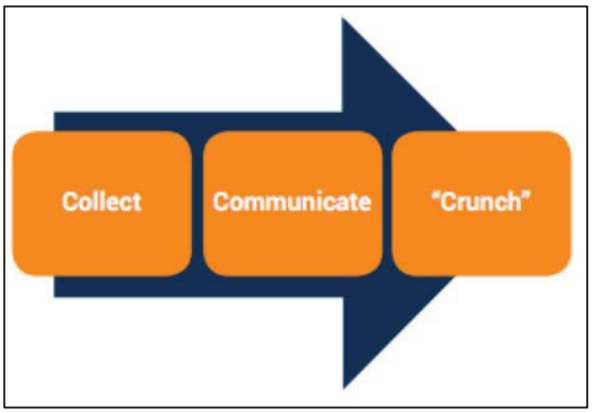

Figura 1. Três Núcleos de funções de uma *Smart City* [10]

Estes Três Núcleos são baseados na Coleta, Comunicação, e Fragmentação sendo que estas etapas consistem em organizar a construção de tecnologias que serão utilizadas pelas Cidades Inteligentes.

Segundo [10]:

- **COLETA:** são informações sobre a atual situação da cidade, como: energia, água, tráfego, tempo, edifícios etc...

- **COMUNICAÇÃO:** consiste em informações para outros dispositivos (M2M), para centros de controle e, muitas vezes, com a execução de softwares poderosos com o uso de protocolos.
- **CRUNCH:** é a análise de dados de forma responsável, apresentando e otimizando a informação de forma perfeita e prevendo o que pode acontecer em seguida.

Na contextualização de [5], uma *Smart City* tem que ser inovadora e usufruir das Tecnologias de Informação e Comunicação (TIC) e outros meios para a tomada de decisão como a eficiência de operações, a prestação de serviços enquanto garante a sustentabilidade e preservação da natureza tornando-se competitiva em aspectos econômicos. Um projeto completo em *Smart Cities* contém aspectos humanos, sociais e ambientais com a finalidade de melhorar a vida das pessoas que habitam centros urbanos.

Como exemplo, a Figura 2 mostra os *Intelligent Streetlights* ou postes de rua inteligentes, que auxiliam na sustentabilidade ambiental. O Departamento de Energia dos Estados Unidos da América (EUA) prevê um aumento de 40% no consumo de iluminação até 2035, sendo o maior consumidor de energia das cidades.

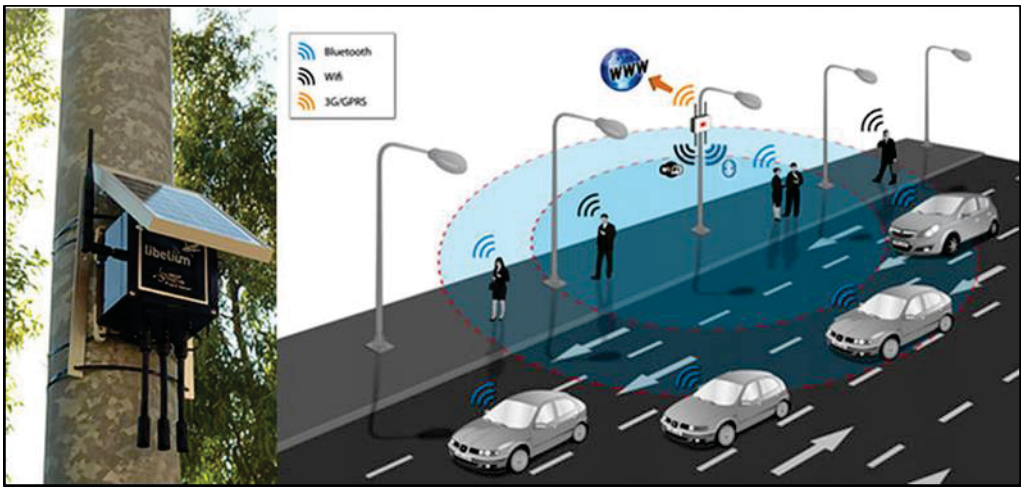

Figura 2. Comunicação M2M e IoT [23].

Do mesmo modo que este ecossistema será o maior consumidor de energia das cidades, outras fontes como [34] apontam que o mesmo terá a maior comunicação de dados das cidades juntamente com a maior interferência de sinais *wireless*, podendo ocasionar problemas de latência, entrega, sobrecargas e transferência na interconexão de dispositivos de IoT.

2.2. IOT E M2M

A Internet das Coisas (IoT) surgiu da necessidade de interligar vários dispositivos diferentes, tais como sensores, atuadores, celulares, dispositivos vestíveis e muitos outros. Para [31], estes objetos conectados podem ganhar o título de *Smart Objects*, desde que consigam perceber, raciocinar e agir, executando atividades que associamos ao pensamento humano, como a tomada de decisão, resolução de problemas e aprendizado. Conforme [17], este conceito tem uma abordagem semelhante ao de *Machine to Machine* (M2M), que é específico para a comunicação de máquina a máquina, não existindo nenhuma interação humana e envolvendo processos como tratar, registrar e manipular os dados gerados e transmitidos por estes objetos.

Sobre as aplicações voltadas à M2M, [17] dizem que:

> As aplicações voltadas à M2M têm potencial para se tornar uma tendência no desenvolvimento de softwares nos próximos anos tendo em vista a variedade de áreas, tanto do setor da indústria, quanto de utilidade doméstica que poderão fazer uso de uma solução automatizada que possa integrar os dispositivos que compõem seu ambiente, além de também trazer à tona a realidade da Internet das Coisas. Quando se pensa no porquê de estudar a comunicação em aplicações M2M, pode-se enumerar que:
>
> - M2M é um assunto em expansão;
> - A computação está ficando a cada dia mais ubíqua, ou seja, em toda parte. Essa imensa disponibilidade de conectividade é um cenário ideal para a exploração de tecnologias M2M;
> - Esta ubiquidade na computação traz também um momento propício para investir em tecnologias que coloquem não apenas supercomputadores na rede, mas também dispositivos restritos, de pequeno porte, com baixo consumo de energia elétrica e baixa capacidade computacional (como memória e processamento);
> - Com o advento do IPV6 (Internet Protocol Versão 6), a capacidade para endereçamento de nós na rede irá possibilitar um número imenso de dispositivos conectados (p. 2).

Como observa-se, a IoT, pode ser implementada em várias escalas. Pode ser com objetos comuns que precisem de acesso à Internet, pode ser máquinas que precisem se comunicar com outras máquinas ou simplesmente por humanos que desejam interagir com objetos que lhes proporcionem informação. Assim, em uma ponta tem-se a comunicação M2M e na outra a interação por IoT. E no meio destes dois conceitos existem 4 camadas que fazem o "caminho" de uma a outra, que são: Aplicações, Comunicações, Sensores/Atuadores e Processamento.

Graças a esta arquitetura é possível integrar a Internet das Coisas com a Comunicação Máquina a Máquina.

IoT e M2M são tecnologias distintas, sendo que para uma melhor performance de comunicação entre máquinas, que serão utilizadas em *Smart Cities*, a Internet torna-se extremamente necessária, criando a tríade de dependência sequencial: *Smart Cities, Machine to Machine e Internet of Things.*

2.3. IMPORTÂNCIA DOS PROTOCOLOS USADOS EM IOT/M2M

Conforme [18], a padronização de protocolos de comunicação facilita interconexão entre os componentes de software e garantem a interoperabilidade entre os diferentes objetos. Ele ainda aponta a importância da compactação 6LoWPAN e o RPL como roteamento e destaca o MQTT e CoAP como uma solução para a troca de mensagens em redes RSSF. Muitas organizações como *International Electrotechnical Commission* (IEC) e também *Institute of Electrical and Electronic Engineers* (IEEE) estão introduzindo seus protocolos de comunicação em redes de energia inteligentes, visando garantir essa padronização. Hoje existem vários medidores de energia inteligentes usando estes protocolos, sendo que estes padrões incluem IEEE 802, IEEE 1815, IEEE 1901, IEC 62056, IEC 60870-5-104, IEC 61850 entre outros.

Segundo [2], o futuro das *Smart Cities* está na padronização de protocolos de IoT e sua interoperabilidade com M2M, e assim como [18], destaca os protocolos 6LoWPAN, RPL, CoAP e MQTT. Estudar e entender o funcionamento dos protocolos é extremamente necessário para o contexto deste trabalho conforme observa-se na Figura 3.

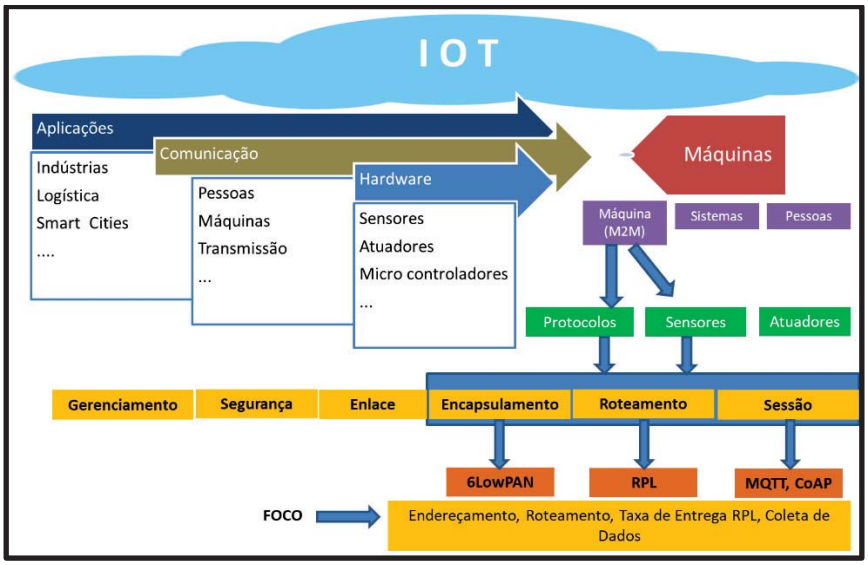

Figura 3. Foco da pesquisa

Nos próximos tópicos serão descritos os protocolos de comunicação que serão simulados neste trabalho.

2.3.1. Protocolo 6LoWPAN

O *IPv6 over Low power Wireless Personal Area Networks* (6LoWPAN) trabalha com sensores de baixa potência, e foi desenvolvido para conexões com a Internet através de redes sem fio tendo a funcionalidade do IPv6 simplificada com cabeçalhos muito compactos.

Este protocolo cria uma camada de interação com o padrão IEE 802.15.4, possibilitando uma baixa largura de banda e menor consumo de energia com *Max Transfer Unit* (MTU) de 127 Bytes, sendo que no IPv6 são de 1280 Bytes, ou seja, este protocolo tem uma compressão de 1/10. [6]. Existem várias RFCs para o 6LoWPAN sendo as principais *Request for Comments*: RFC 4919 [30], RFC 4944 [20] e RFC 6575 [5].

O grupo 6LoWPAN, descreveu algumas metas para o padrão IEE 802.15.4, e estas foram descritas na RFC 4919, onde pode-se observar: a necessidade do protocolo trabalhar com dispositivos que possuem restrições de processamento e capacidade de memória; como possui um pequeno MTU de rede, a compressão e fragmentação de pacotes é necessária na camada 2; suportar topologias *Mesh*, com protocolos de roteamento simples que possam ser suportados pelos dispositivos e Autoconfiguração devido à grande quantidade de *hosts* presentes nas redes [38].

A estrutura de rede do 6LoWPAN, requer um *Gateway* para integração com a Internet. Este dispositivo serve para realizar a compressão e descompressão do IPv6, conforme pode-se observar na Figura 4.

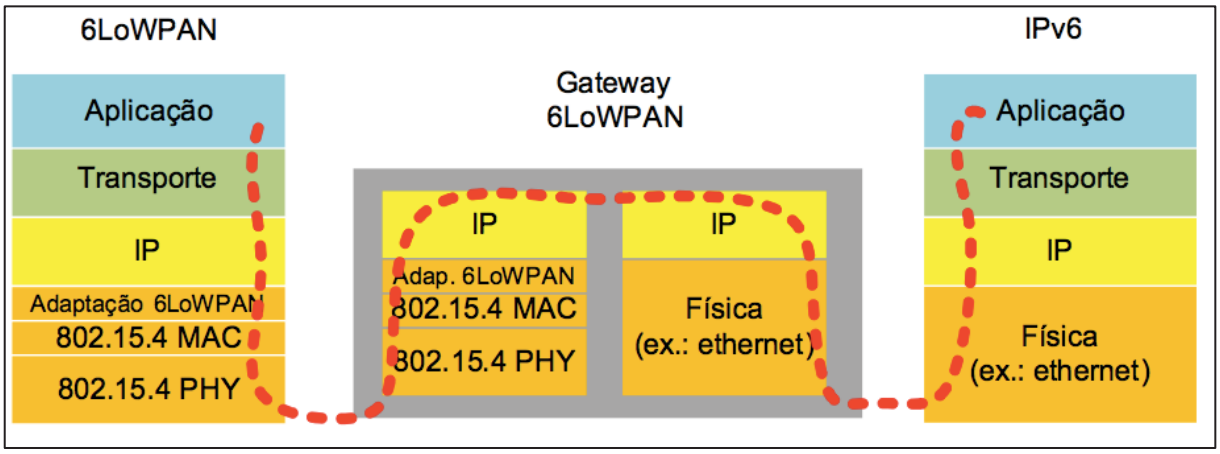

Figura 4. Infraestrutura de uma rede 6LoWPAN [38]

Para atingir as metas descritas anteriormente, o grupo de trabalho IETF 6LoWPAN foi criado e provê suporte do IPv6 sobre IEEE 802.15.4. Este grupo centrou-se principalmente nos seguintes itens:

- definir limites e extensões ao protocolo de *Neighbor Discovery* IPv6 mais adaptado para WSN;
- descrever mecanismos para compactar cabeçalhos 6LoWPAN;
- Definir abordagens e protocolos de roteamento 6LoWPAN.

Em vez de definir um único cabeçalho, como o IPv4, o 6LoWPAN usa cabeçalhos empilhados como o protocolo IPv6 original. Nesse caso, ele não precisa usar campos de cabeçalho desnecessários para fragmentação e usa somente os cabeçalhos mínimos necessários.

O padrão 6LoWPAN define quatro tipos de cabeçalho:

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

O protocolo de camada de rede deve obedecer às restrições impostas pelo protocolo de camada inferior em uso. Na verdade, os requisitos do protocolo IPv6 não coincidem totalmente com as restrições IEEE 802.15.4, como o MTU. Além dessa incompatibilidade, o uso de cabeçalhos IPv6 padrão resultaria em carga extremamente pequena para protocolos das camadas superiores como a de Aplicação.

Segundo [38], foi criada uma camada de adaptação para suportar as redes IP no padrão IEEE 802.15.4, a qual também pode ser observada na Figura 4, e sendo nesta camada onde ocorre a fragmentação e desfragmentação dos pacotes, já que o MTU da camada física é menor do que o mínimo especificado no IPv6. Neste momento também ocorre a emulação do *broadcast* e o suporte aos protocolos de roteamento da camada 2 para redes *mesh*. Na Figura 5 pode-se observar os cabeçalhos que dão suporte às mais variadas funções da camada de adaptação e seus encadeamentos. O HC1 é o cabeçalho IP comprimido. O *mesh* é utilizado para roteamento na camada 2.

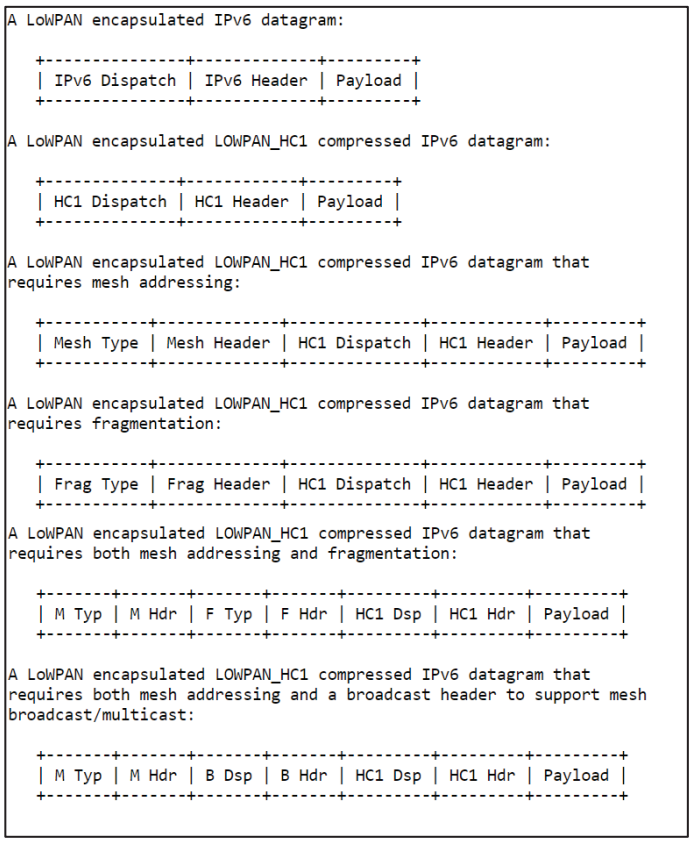

Figura 5. Encadeamento de cabeçalhos 6LoWPAN [24].

Observa-se na Figura 6, a técnica de compressão *stateless*, que é usada no 6LoWPAN, onde pode-se chegar à compressão de 40 bytes do IPv6 em 2 bytes, destacados nos campos em verde. Esta técnica pode ser observada de forma mais detalhada na RFC 4944. Os campos em verde são comprimidos e apenas o limite de hops é enviado integralmente.

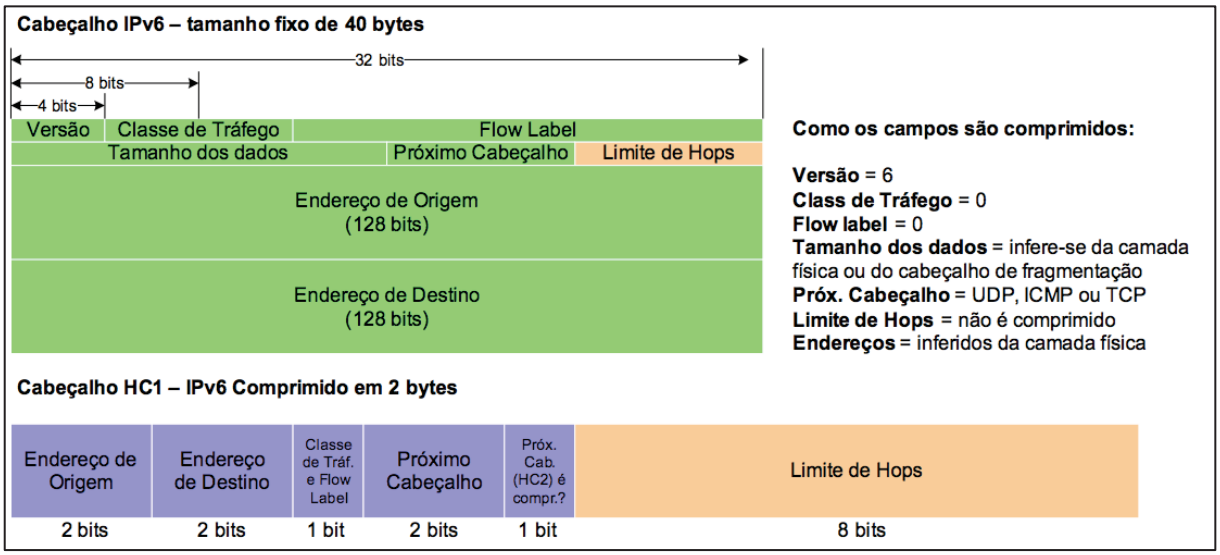

Figura 6. Cabeçalho IPv6 comprimido (HC1) [38]

O 6LoWPAN leva consigo a capacidade de endereçamento do IPv6 com custo baixo de processamento, armazenamento em memória e roteamento, sendo atualmente uma excelente opção para a composição de RSSF, pois a mesma opera com dispositivos que costuma ter estas características.

Segundo [27], *Header Compression* (HC) é o nome dado no 6LoWPAN para o cabeçalho IPv6 comprimido, sendo HC1 quando for referência para o Cabeçalho do IPv6 e HC2 para Cabeçalho UDP.

2.3.2. Protocolo RPL

O *Routing Protocol Low Power and Lossy Networks* (RPL) surgiu da visão do grupo *Routing Over Low Power and Lossy Networks* (ROLL), criado pela IETF para estudar possíveis aplicações em IoT em diversas áreas como redes urbanas com a automação industrial e residencial por exemplo [35].

Sua formação começa por um nó Raiz ou *Root* que determina a estrutura da rede, criando um pacote de mensagem ICMPv6 para o próprio RPL levando os dados da estrutura em *neighboring peer*, e cada nó mantém um conjunto de *neighboring* que retornam as mensagens para o *Root* construindo, assim, um roteamento [35].

Conforme [15], O RPL é um protocolo de roteamento que define suas rotas através de um vetor de distâncias, e foi construído pensando em redes de baixa potência e com perdas também conhecidas como *Low Power and Lossy Network* (LLN).

Muitos protocolos de roteamento são baseados em tabelas, mas o RPL usa uma árvore de roteamento em uma estrutura de grafos acíclicos diretos conhecidos como *Direct Acyclic Graphs* (DAG). O DAG consiste na associação de um nó em mais de um nó, o que não é visto nos protocolos tradicionais de estrutura em árvores, e assim o RPL organiza os nós com destinos orientados. Assim, um dos componentes da rede funciona como nó raiz e centraliza o recebimento de dados dos outros nós, sendo esta estrutura conhecida por *Destination Oriented Directed Acyclic Graph* (DODAG). Uma rede de roteamento é formada por uma ou mais DODAGs, o que forma uma instância conhecida por RPL*Instance*ID em que um nó pode participar de várias instâncias, mas de apenas um DODAG [9].

Conforme descrito por na RFC 6550 [32], as DODAGs são construídas pela troca de mensagens ICMPv6 que serão descritas a seguir:

- DOGAG *Information Solicitation* (DIS): responsável por solicitar aos nós da DODAG informações sobre os objetos da rede, tornando possível a descoberta de vizinhos;
- DODAG *Information Object* (DIO): permite a descoberta de instâncias IPv6, bem como permite que um nó aprenda as configurações da DODAG;
- *Destination Advertisement Object* (DAO): Responsável pela propagação de roteamento de forma ascendente, permitindo que cada nó monte uma tabela de roteamento mais eficaz.

A padronização do RPL não define métricas e nem restrições. Logo, a forma como as definições de associação de elementos para a composição da rede também não existe, mas existe na RFC 6550 as Funções de Objetivo (OF), que é definido como os nós selecionam seus pais e como otimizam as rotas em uma instância RPL.

A DODAG é sempre atualizada com mensagens DIO frequentemente enviadas pelos nós, logo conforme as alterações nas redes se tornam menores, o tempo entre as mensagens tende a aumentar exponencialmente.

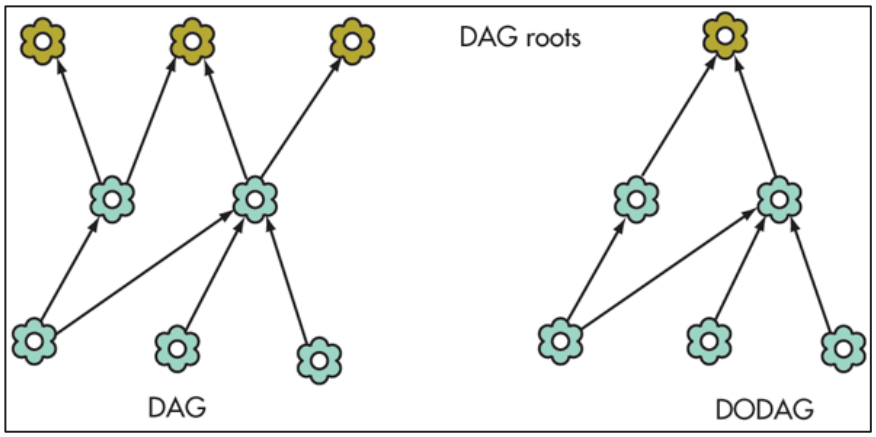

Figura 7. DAG DODAG

Conforme pode-se observar na Figura 7, a DAG possui mais de um nó como raiz, e a DODAG, na qual um nó da rede se transforma em raiz, mas ambas formam gráfico acíclico direto e direcionado de forma ascendente.

2.3.3. Protocolo CoAP

O protocolo Constrained Application Protocol (COAP), é descrito na RFC de número 7252 [28], como sendo um protocolo de aplicação da Internet para dispositivos restritos. O mesmo foi projetado para uso entre nós da mesma rede, entre dispositivos e nós na Internet e entre dispositivos em diferentes redes limitadas. O CoAP também está sendo usado por outros mecanismos, como SMS em redes de comunicação móvel. Ele foi projetado para traduzir em HTTP podendo se integrar de forma simplificada com a web, enquanto também atende requisitos especializados, tais como *multicast*, baixo *overhead*, baixa sobrecarga e simplicidade, o que é extremamente importante para dispositivos IoT/M2M que possuem limitação de memória e alimentação de energia do que outros dispositivos tradicionais da Internet, sendo a sua eficiência o item mais importante.

Uma mensagem CoAP é composta de: um cabeçalho de 32 bits, contendo o código do método de solicitação (status da resposta); um valor de *token* opcional, usado para associar respostas a solicitações; uma sequência de campos de opção; e os dados de carga útil. O protocolo também suporta proxy, permitindo que os aplicativos da Web acessem de forma transparente os recursos hospedados em dispositivos baseados nele [22].

Code	Description	Reference
2.01	Created	[RFC7252]
2.02	Deleted	[RFC7252]
2.03	Valid	[RFC7252]
2.04	Changed	[RFC7252]
2.05	Content	[RFC7252]
4.00	Bad Request	[RFC7252]
4.01	Unauthorized	[RFC7252]
4.02	Bad Option	[RFC7252]
4.03	Forbidden	[RFC7252]
4.04	Not Found	[RFC7252]
4.05	Method Not Allowed	[RFC7252]
4.06	Not Acceptable	[RFC7252]
4.12	Precondition Failed	[RFC7252]
4.13	Request Entity Too Large	[RFC7252]
4.15	Unsupported Content-Format	[RFC7252]
5.00	Internal Server Error	[RFC7252]
5.01	Not Implemented	[RFC7252]
5.02	Bad Gateway	[RFC7252]
5.03	Service Unavailable	[RFC7252]
5.04	Gateway Timeout	[RFC7252]
5.05	Proxying Not Supported	[RFC7252]

Figura 8. Tipos de respostas do protocolo CoAP - [28]

De acordo com [28], o CoAP é um protocolo voltado para redes restritas, ou seja, aquelas que possuem nós com pequena quantidade de memória RAM e cuja taxa de perda de pacotes é grande, sendo também uma ótima opção para M2M.

O CoAP tem uma funcionalidade chamada de Observação, na qual um servidor GET estabelece uma lista na qual cada observador pode se registrar uma vez, e eles recebem as informações do servidor sem a necessidade de gerar pedidos adicionais. Esta função evita excesso de requisições ao servidor por parte do cliente, e que o mesmo não precise manter uma sessão aberta como no caso do HTTP [3], como observa-se na Figura 8.

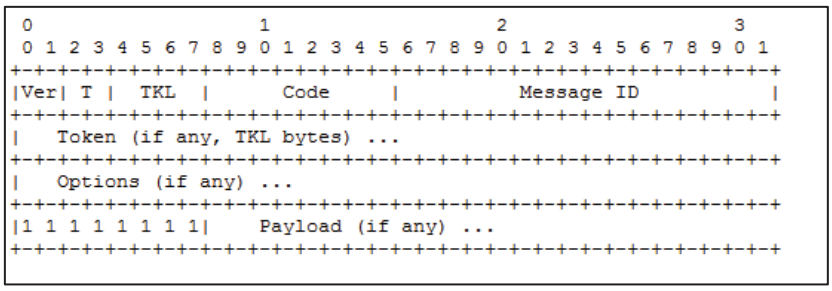

Figura 9. Formato de uma mensgem CoAP [28].

Pode-se observar na Figura 9: **Versão** que usa 2 bits; **Tipo** que identifica Confirmable (0), Non-confirmable (1), Acknowlegment (2) ou Reset (3); **Largura de Token** que identifica o tamanho da varável; **Código** que identifica o tipo da mensagem; **Identificação da Mensagem**

que usa 16 bits para detectar duplicação de mensagens. CoAP é projetado para interagir facilmente com HTTP para integração com a Web, atendendo a requisitos especializados como suporte multicast, sobrecarga muito baixa e simplicidade para ambientes restritos. [17].

O modelo de interação do CoAP é semelhante ao modelo cliente/servidor. Por exemplo, um pedido CoAP equivalente ao do HTTP é enviado por um cliente para solicitar uma ação (usando um código de método) em um Identificador Uniforme de Recurso (URI) em um servidor. O servidor envia uma resposta com um código; essa resposta pode incluir uma representação de recursos.

Após o Cabeçalho, o *token* e as opções (uso opcional), apresenta-se o *payload*, que é um campo opcional e ele é indicado por um marcador de um byte (Payload Marker) que indica o fim das opções e o início da carga. Caso esse marcador não esteja presente, o payload tem largura 0 bytes, mas caso exista este marcador e mesmo assim o payload for de 0 bytes, o pacote deve ser processado como um erro no formato da mensagem [17].

2.3.4. Protocolo MQTT

O MQTT é um protocolo de conectividade M2M/IoT, e foi criado com a ideia de enviar mensagens leves de forma extremamente rápida e eficaz, sendo amplamente utilizado para comunicação com locais remotos e de largura de banda precária. Desenvolvido por Andy Stanford Clark da IBM, e Arlen Nipper da Arcom em 1999, tem como princípio fundamental e funcional a existência de um ponto intermediário chamado *Broker*. Quando um cliente envia ou publica em um determinado dado, o mesmo é registrado neste ponto e múltiplas assinaturas podem ser feitas pelo mesmo cliente para diferentes entradas, bem como diferentes clientes podem subscrevê-la. Isso funciona da mesma forma para os editores dos dados. Assim, os clientes que assinam a entrada X receberão todas as mensagens publicadas pela mesma[4].

Conforme explica [7], O protocolo HTTP convencional é ótimo para fazer um pedido e obter resposta como, por exemplo, quando os clientes querem pedir algumas informações ao servidor e, em seguida, obter a resposta de volta. Mas não é a melhor solução quando a fonte de informações deve enviar comunicação de alteração de contexto para muitos clientes e não tem suporte nativo para a qualidade do serviço do inglês *Quality of Service* (QoS).

Como resultado de não oferecer e de não ser tão eficiente quando a fonte informações necessita de alterações de contexto, como sensores por exemplo, o protocolo HTTP convencional exige mais largura de banda e necessidade de manter o servidor ativo para atender as solicitações de entrada gerando um maior consumo de energia. Logo, HTTP não atenderia

bem aos requisitos M2M para os dispositivos com alimentação por bateria, sendo o *Message Queuing Telemetry Transport* (MQTT) uma solução adequada para este problema.

O MQTT possui três componentes básicos sendo eles: *subscriber*, *Publisher* e o *broker*. Atualmente, o exemplo mais conhecido de implementação é o software Mosquitto que é um intermediário de mensagens de código-fonte aberto que implementa as versões 3.1 e 3.1.1 do protocolo MQTT. O Mosquitto é leve e adequado para uso em todos os dispositivos, desde computadores de mesa única de baixa potência até servidores completos [12].

O protocolo MQTT fornece um método leve de executar mensagens usando um modelo de publicação/assinatura. Isso o torna adequado para mensagens da IoT, com sensores de baixa potência ou dispositivos móveis, como telefones, computadores embutidos ou microcontroladores.

O tamanho do cabeçalho da mensagem para cada comando do MQTT é fixo, e algumas mensagens também exigem um cabeçalho variável e uma *payload* válido. O formato de cada parte do cabeçalho da mensagem é descrito da seguinte maneira:

2.3.4.1. Cabeçalho fixo

O **Byte 1,** contém os campos “tipo da mensagem”, e flags DUP, QoS e Retain, o **Byte 2,** contém o *Remaining Length*, conforme pode ser observado na Tabela 1.

Tabela 1. Cabeçalho de tamanho fixo MQTT [25].

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

Na Tabela 2, representadas como um valor sem sinal de 4 bits, estão as enumerações do protocolo descrevendo seu tipo de mensagem.

Tabela 2. Tipos de Mensagens MQTT [25]

Mnemonic	Enumeration	Description
Reserved	0	Reserved
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Complete (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Reserved	15	Reserved

Os bits restantes do byte 1 contêm os campos DUP, QoS e RETAIN. As posições dos bits são codificadas para representar as *flags*. Segundo [25] as *flags* têm as descrições a seguir:

DUP: Posição: byte 1, bit 3. Esse sinalizador é definido quando o cliente ou servidor tenta entregar novamente uma mensagem *PUBLISH, PUBREL, SUBSCRIBE* ou *UNSUBSCRIBE* . Isso se aplica a mensagens em que o valor de QoS é maior que zero (0) e uma confirmação é necessária. Quando o bit DUP é definido, o cabeçalho da variável inclui um ID da mensagem, o destinatário deve tratar esse sinalizador como uma dica sobre se a mensagem pode ter sido recebida anteriormente. Não deve ser usado para detectar duplicatas.

RETAIN Posição: byte 1, bit 0. Este sinalizador é usado apenas em mensagens *PUBLISH*. Quando um cliente envia um *PUBLISH* para um servidor, se o sinalizador de retenção estiver definido (1), o servidor deve manter a mensagem depois de ter sido entregue para os assinantes atuais.

Quando um servidor envia um *PUBLISH* para um cliente como resultado de uma inscrição que já existia quando o *PUBLISH* original chegou, o sinalizador de retenção não deve

ser definido, independentemente do sinalizador de retenção do *PUBLISH* original. Isso permite que um cliente consiga distinguir as mensagens que estão sendo recebidas porque foram retidas e aquelas que estão sendo recebidas "ao vivo".

QoS Posição: byte 1, bits 2-1. Essa sinalização indica o nível de garantia para a entrega de uma mensagem *PUBLISH*.

Remaining Length byte 2. Representa o número de bytes restantes na mensagem atual, incluindo dados no cabeçalho da variável e na carga útil. O esquema de codificação de comprimento variável usa um único byte para mensagens de até 127 bytes de comprimento. Mensagens mais longas são tratadas da seguinte maneira. Sete bits de cada byte codificam os dados do *Remaining Length* e o oitavo bit indica os seguintes bytes na representação. Cada byte codifica 128 valores e um "bit de continuação". Por exemplo, o número 64 decimal é codificado como um único byte, valor decimal 64, hex 0x40. O número 321 decimal (= 65 + 2 * 128) é codificado como dois bytes, menos significativo primeiro. O primeiro byte 65 + 128 = 193. Observe que o bit superior está definido para indicar pelo menos um byte seguinte. O segundo byte é 2. O protocolo limita o número de bytes na representação a um máximo de quatro.

2.3.4.2. Cabeçalho variável

Alguns tipos de mensagens de comando do MQTT também contêm um componente de cabeçalho variável. O campo *Remaining Length* do comprimento variável não faz parte do cabeçalho da variável. Os bytes do campo *Remaining Length* não contribuem para a contagem de bytes do valor do tamanho restante. Esse valor leva em consideração apenas o cabeçalho da variável e a carga útil. O nome do protocolo está presente no cabeçalho da variável de uma mensagem MQTT CONNECT, bem como a versão do protocolo. O campo é um valor sem sinal de 8 bits que representa o nível de revisão do protocolo usado pelo cliente conforme é mostrado na Tabela 1.

2.4. O PROCESSO DE SIMULAÇÃO

Segundo [26], simular sistemas permite elaborar um modelo computacional com base em um sistema real, e realizar experimentos através deste modelo entendendo suas interações e resultados podendo desta forma avaliar sua operação de forma estratégica.

Existem outros métodos de análise de sistemas, porém o processo de simulação apresenta muita vantagem sobre eles, das quais pode-se destacar: capacidade de previsão de desempenho; adaptabilidade do modelo as variações de condições do teste e operação; menor custo; capacidade de reprodução e controle de experimentos; menor consumo de tempo para participação dos resultados e maior simplicidade em relação aos métodos analíticos [11].

Para [29], simulações e emulações são muito importantes na fase de avaliação para arquiteturas e protocolos de redes RFFS, permitindo trazer situações do mundo real que não seriam possíveis quantificar ou qualificar de forma real. Pode-se citar a simulação da comunicação de 13.000 objetos inteligentes, cujo objetivo seria descobrir possíveis falhas ou sucessos antes de colocá-los em produção, evitando erros e desperdício. Um exemplo de simulação pode ser aplicável à evacuação de um estádio de futebol com 50.000 pessoas. É inviável lotar este recinto e testar várias evacuações de modos diferentes. Já com o uso de simuladores e/ou emuladores esta tarefa torna-se mais fácil e muito próximo da realidade.

Existem diferenças realçadas entre simuladores e emuladores, pois um sistema de hardware ou software que permite que um sistema computacional possa se comportar como um outro sistema é considerado um emulador. Já uma situação real ou hipotética representada em um computador é considerada uma simulação. O Cooja emula aplicações do sistema operacional Contiki e também permite simular situações em cenários onde cada nó possui um tipo de memória e um número de interfaces, diferente de outros simuladores conforme descrito por [29] na Tabela 3.

Tabela 3. Comparativo entre os principais simuladores/emuladores para IoT [29]

Nome	Suporte GUI	Licença	Linguagem	Sistema Operacional
Cooja	Sim	Código aberto	C	Linux
ns-2	Não	Código aberto	C++ e oTcl	GNU/Linux, FreeBSD, Mac OS e Windows
ns-3	Limitado	Código aberto	C++ e python	GNU/Linux, FreeBSD, Mac OS e Windows
Tossim	Limitado	Código aberto	nesC e python	Linux ou Cygwin no Windows
OMNet++	Sim	Comercial e código aberto	C++	Linux, Mac OS e Windows
Castalia	Sim	Código aberto	C++	Linux e Windows
Sinalgo	Sim	Código aberto	Java	Linux, Mac OS e Windows

Após pesquisas e testes com os emuladores, chegou-se à conclusão que o CooJA é a melhor opção para a elaboração deste trabalho, pois o mesmo incorpora itens que que auxiliam de forma expressiva a simulação dos protocolos de IoT. O Cooja é um emulador do sistema operacional Contiki que permite simular interações entre dispositivos IoT com o Contiki embarcado, foi desenvolvido em Java e que pode ser executado em um sistema operacional Ubuntu, no entanto os scripts de configuração dos dispositivos simulados devem ser realizados em código da linguagem de programação C, que serão interpretados pelo Cooja.

2.4.1. Modelos de simulação

Conforme [11], as varáveis de estado definem a classificação dos modelos de simulação que podem ser modelos de eventos discretos ou modelo de eventos contínuos. Uma variável de estado representa um conjunto de informações sobre determinado evento que está ocorrendo em algum momento da simulação. Eventos são acontecimentos que podem ser programados ou não e podem provocar uma mudança no estado sistema como, por exemplo, o início de uma comunicação de rede solicitando um recurso. Entidades ou atributos representam os objetos dentro do modelo de simulação como os nós de uma rede. Recursos e filas são entidades estáticas que fornecem auxílio a entidades dinâmicas como, por exemplo, o tratamento de uma fila por *First in First Out* ou *Last In First Out*.

Modelos de Eventos Discretos mantêm o estado da variável durante certo tempo e modificam seus valores em instantes bem definidos, que são conhecidos como tempo de ocorrência do evento, e podendo ter variação discreta ou contínua.

Nos Modelos de Eventos Contínuo, as variáveis podem alterar seus valores continuamente durante toda a simulação, e estes modelos podem ser ainda contínuos ou discretos no período de tempo. Existe ainda a modelagem mista onde as variáveis de estado dependentes podem alterar seus valores continuamente ou discretamente ao longo do tempo.

2.4.2. Sistema Operacional Contiki e o emulador / simulador COOJA

Contiki OS é um sistema operacional para IoT que conecta microcontroladores de baixa potência e com suporte para IPv4 e IPv6, bem como os mais recentes padrões de baixo consumo como RPL COAP e 6LoWPAN [8].

Ele é um dos concorrentes para futuros Sistemas Operacionais da IoT, e foi proposto em 2003. Desde então ele tem sido continuamente desenvolvido e atualizado por profissionais,

acadêmicos e pesquisadores. O Contiki OS suporta módulos para gestão e desenvolvimento de padrões em reconhecimento de energia, carregamento de módulos dinâmicos e muitas plataformas de hardware. As diversas aplicações da IoT, incluindo casas inteligentes, saúde inteligente e cidades inteligentes, exigem conectividade de rede eficiente e demandam ainda protocolos de roteamento inteligentes, que podem lidar com redes heterogêneas, móveis e diversificadas e cuja gerência pode se encontrar no Contiki.

Conforme [29], o Contiki é excelente para sistemas embarcados e permite comunicação eficiente para dispositivos IoT e sistemas inteligentes. Neste sistema operacional, podemos encontrar bibliotecas para a alocação de memória e abstrações para mecanismos de baixa potência. As aplicações no Contiki podem ser desenvolvidas em C, e com ele podemos usar o Cooja, que é um simulador de RSSF. O Contiki foi desenvolvido para ocupar o mínimo de memória, sendo assim ele é extremamente eficaz no gerenciamento de memória. O Contiki OS Java (COOJA) é um simulador desenvolvido em Java, e foi projetado para redes sem fio, sendo ele executado no sistema operacional Contiki. O simulador citado neste capítulo trabalha com a simulação de nós com características distintas entre cada nó, diferindo tanto em hardware como software. Segundo [3], podemos encontrar três partes básicas em um nó simulado pelo COOJA: a memória de dados, o tipo de nó e seus periféricos de hardware.

As interações da simulação e dos nós simulados são realizados através de plug-ins, em que interfaces podem ser adicionadas ao simulador, permitindo que o usuário possa personalizar sua simulação de forma eficiente e organizada.

Embora o Cooja tenha provado ser uma ferramenta ideal para a simulação de em RSSFs, existem problemas em relação à falta de documentação disponível [33].

3. MÉTODOS

Este capítulo descreve como foram definidos os critérios selecionados para o desenvolvimento da pesquisa. A execução deste trabalho foi planejada em cima de pesquisas bibliográficas, estudos dos protocolos relacionados anteriormente, testes com simuladores, criação de *testbed*¸ para melhor compreensão do funcionamento da simulação. Assim como [13], a pesquisa foi realizada de forma quantitativa com o intuito de possibilitar a mensuração de diferentes parâmetros dos protocolos analisados. Foi realizada uma pesquisa experimental na qual classificamos e selecionamos softwares, processos e aplicação de procedimentos que serão descritos a seguir.

3.1. AMBIENTE DE SIMULAÇÃO

A base metodológica concentra-se na simulação dos protocolos CoAP e MQTT, que são as tendências mais pesquisadas para troca de mensagens em aplicações M2M em ambiente de *Smart Cities*. Estes protocolos serão testados com endereçamento 6LoWPAN, por ser o sistema de endereços mais utilizado, inclusive seu maior concorrente, o ZigBEE, que utilizava um padrão proprietário passou a usar o padrão 6LoWPAN, que é aberto [38]. Para a construção de rotas e encaminhamentos através de roteamentos, foi utilizado o protocolo RPL por ser o mais citado e bem avaliado nas pesquisas que serviram de base para elaboração deste trabalho.

Após testes com os simuladores citados na Tabela 3, chega-se à escolha do simulador Contiki/COOJA. Por ser interativo com o ambiente real e não apenas com a simulação, este software faz a emulação dos nós simulados que podem interagir com nós físicos, bem como podem ser acessados por clientes reais. Um exemplo foi o protocolo CoAP, que pode ser acessado pelo navegador web de outra máquina real e enviar mensagens para a rede simulada.

3.1.1. Ambiente Computacional

Conforme [8], testes foram realizados em máquinas virtuais, porém com resultados de processamento e memória não satisfatórios para o cenário da simulação, dado que os resultados não puderam ser obtidos, pois a *Virtual Machine* travava antes do fim da simulação. Assim, optou-se por realizar a instalação em uma máquina física com as configurações descritas na Tabela 4:

Tabela 4. Configuração do equipamento utilizado nas configurações

Recurso	Capacidade
Processador	Intel(R) Core(TM) i7-5500u CPU @ 2.40GHz
Disco rígido	1 TB
Memória RAM	16,0 GB
Sistema Operacional	Ubuntu 14.04.5 LTS 64 bits
Kernel	4.4.0-128
Software de Simulação	Contiki / Cooja 3.0
Adaptadores de vídeo	AMD Radeon R7 M260 / Intel(R) HD Graphics 5500

3.1.2. Cenário da simulação

Com o intuito de empregar o MQTT e o CoAP dentro das características de IoT, foram criadas aplicações que emulam e simulam processos típicos em ambientes urbanos. Foi implementada a planta da cidade de Carazinho, usando como base a intersecção de cinco ruas: Silva Jardim, Pedro Vargas, Marechal Floriano, Alexandre da Motta e 15 de Novembro, conforme observa-se na Figura 10:

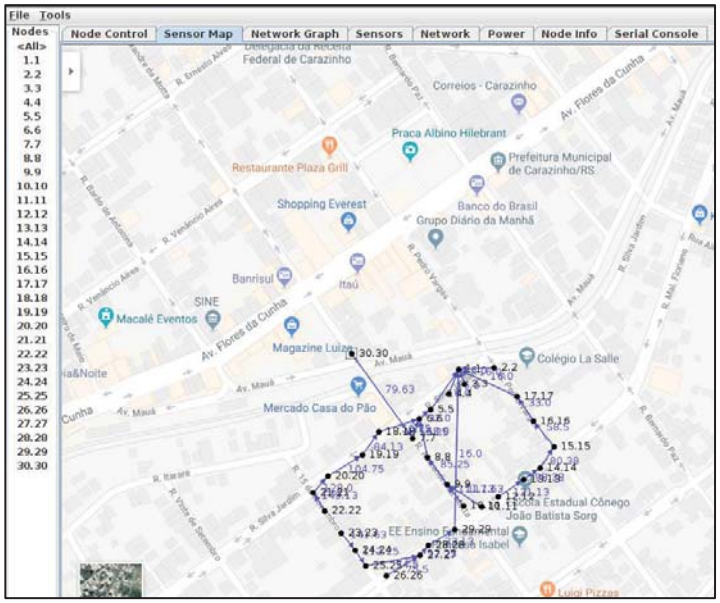

Figura 10. Intersecção de ruas e nós de redes simulados

Para melhor entendimento foi utilizado o mapa da cidade de Carazinho e, como referência, 2 quadras com os postes. Na Figura 10, pode-se observar a presença de 30 nós de

rede que representam os postes nestas duas quadras, sendo que nesta simulação cada poste continha um nó dessa rede. O nó 1.1 é um roteador RPL Collect, com a função de coletar informações sobre os demais nós no que diz respeito aos sensores de: temperatura, umidade, luz e informações de rede. Outra função do nó 1.1 é conectar a rede simulada e emulada na rede real para interações e extração de dados e conhecimento conforme [29], sugere. Os nós estão dispostos da seguinte forma: nó 1.1 está fora da rua, simulando uma base de coleta dentro de um prédio; a sequência do nó 2.2 até o nó 7.7 está localizado na rua Silva Jardim; a sequência do nó 8.8 até o nó 10.10 está na rua Alexandre da Motta; a sequência dos nós 11.11 até 14.14 está na rua Marechal Floriano; a sequência dos nós 15.15 até 17.17 está na rua Pedro Vargas, fechando assim a primeira quadra representada na direita da Figura 10. A quadra da esquerda começa com os nós 18.18 até 20.20 na rua Silva Jardim; do nó 21.21 até o nó 25.25 na rua 15 de Novembro; do nó 26.26 a 29.29 na rua Marechal Floriano. Desta forma, unificamos 2 quadras totalizando uma distância total: 833,35 m2 (2.734,09 pés), no modelo de mobilidade *Manhattan Grid* sugerido por [13], para ambientes urbanos. Não foram considerados interferências físicas, mas sim interferências de transmissão que os próprios rádios podem causar entre eles como a transmissão em um mesmo canal e frequência.

3.1.3. Estrutura do Simulada

Nesta seção é demonstrado como o simulador Cooja foi configurado e a estrutura simulada com a finalidade de representar a situação real de disposição dos nós dessa rede em termos de distância, bem como a implantação dos sensor sem fio de baixa potência, que nestas simulações foram usados os *Tmote Sky* da empresa Advanticsys [16].

3.1.3.1. Postes

O presente trabalho propõe uma simulação e extração de dados de uma rede RSSF de 30 postes inteligentes e 1 coletor de informações (*Broker*) ou seja uma topologia adequada para *Smart Cities* e que seria replicada de quadra em quadra Conforme [26], porém cercados por mais 470 postes de configuração igual e dispostos de forma aleatória apenas com o intuito de ter interferência de sinal causada em um ambiente real de *Smart Cities*. Foram dispostos 500 postes, na distância média real de 30 metros cada um, sendo que esta distância foi medida manualmente. Na Figura 11, temos a representação do simulador em escala de 10 metros. Pode-se observar ainda que os nós simulados possuem 3 *leds* que referenciam a luminosidade do

poste. O nó de número 30 foi posicionado estrategicamente para representar a transmissão em uma outra quadra, a fim de observar o comportamento da rede quando um nó RPL *Collect* falhar, onde espera-se que ele reconfigure a DODAG em outro nó coletor.

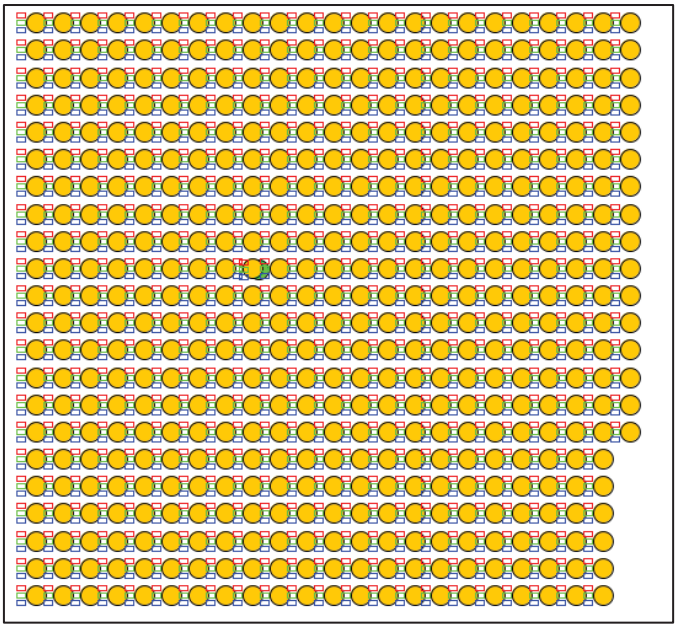

Figura 11. RSSF Simulada

O endereçamento em 6LoWPAN dos postes foi atribuído por Micro IP (µIP), que é uma das menores pilhas TCP/IP que existem e que integra o Contiki. A compressão do 6LoWPAN usa dados de protocolos presentes em outras camadas, como [29] cita o exemplo do 6LoWPAN que pode utilizar parte do endereço MAC do dispositivo para atribuir um endereço IPv6 para o objeto inteligente, sendo exatamente este conceito utilizado neste trabalho para endereçar os postes com DHCP no RPL *Border*, utilizando parte do endereço físico dos hosts que compõem a rede.

3.1.3.2. Tmote Sky

O COOJA permite emular aplicações do Contiki e faz com que estas interajam com dispositivos físicos reais, compondo assim os famosos *testsbeds* híbridos. Neste trabalho utilizamos os motes do tipo *SKY* (para os postes*)*, RPL *Collect* (Coletar Informações sobre os sensores), RPL *Border* (para integrar a rede física com a simulada). O Tmote *SKY* usa uma potência extremamente baixa, vem com alta taxa de dados e possui todos os sensores embutidos, rádio, antena e recursos de programação. A operação de baixa potência do Tmote sky deve-se

ao microcontrolador TIMP430 F1611 de ultra baixa potência. O Tmote SKY usa o controlador USB do *Future Technology Devices International* (FTDI) para transmitir dados ao computador host. Também inclui o chip no rádio CC2420. Este é um rádio compatível com IEEE 802.15.4 e oferece um serviço muito confiável, na Figura 12 temos um modelo de Tmote *Sky*.

Figura 12. Tmote Sky simulado no Cooja

O Tmote *Sky*, é composto por um microcontrolador de 8MHz Texas *Instruments* MSP430 com 10kb RAM, 48Kb de memória interna *flash* e 1MB de memória RAM externa, além de sensores de temperatura, umidade e luz.

3.1.3.3. Roteador RPL Border Router

Para implantar a rede IoT é necessário instalar um *Border Router* (BR). O *Border Router* é uma porta de entrada para essas duas redes, em que os dispositivos 802.15.4 (WPAN) estão em um lado dele, e Ethernet ou Wi-Fi está do outro lado. Um desses roteadores *Border* é um RPL encapsulado pelo 6LoWPAN em uma ponta e IPv6 na outra. É através da emulação deste roteador, que os nós reais podem acessar os virtuais e vice-versa, sendo que na Figura 13 pode-se observar a descrição citada. No simulador é necessário informar que o roteador de borda criou um socket e está escutando na porta local 60001 antes de iniciar a simulação, através de "*Mote tools for sky/Serial socket (SERVER)*". Após iniciar a simulação é necessário criar um túnel criando uma pilha de protocolos no Linux e conectá-los ao roteador de borda na porta 60001com o comando "*make connect-router-cooja TARGET=sky*". Os endereços apresentados serão representados em modo comprimido Por exemplo: [aaaa : ~~0000 : 0000 : 0000~~ : ~~0~~212 : 7401 : ~~000~~1 : ~~0~~101] é o mesmo que [aaaa :: 212 : 7401 : 1 : 101], característica do IPv6 mantida no 6LoWPAN e observável na Figura 13.

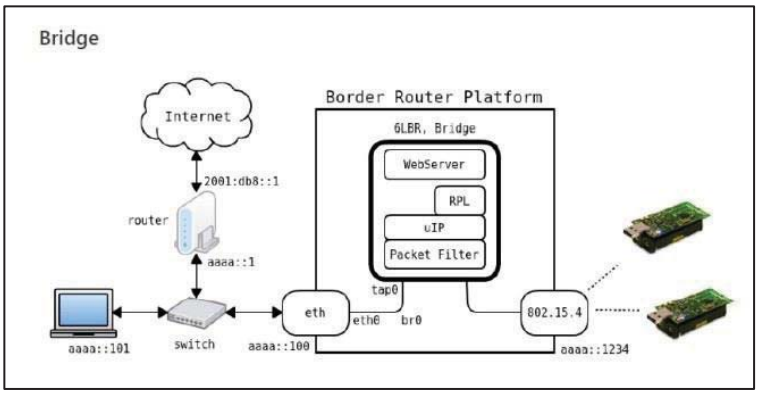

Figura 13. Funcionamento do Roteador RPL

3.1.3.4. Parâmetros do simulador

Os Tmotes, SKY, RPL *collect* e RPL *Border* foram dispostos conforme observa-se na Figura 10. O Tmote 1.1 é um RPL *Collect,* os Tmotes de 2.2 até 30.30 são do tipo *SKY*, o Tmote 31 é do tipo RPL *Border*, classificando assim as funções estabelecidas por software conforme já descrito na seção 3.1.3, mas utilizando a mesma parametrização de comunicação de rede e processamento que serão descritos na Tabela 5:

Tabela 5. Parâmetros do Simulador

Característica	**Parâmetro**
Tipo do canal	Wireless Transceiver
Propagação de sinal	Modelo de 2 Raios
Alcance da antena	70m range TX outdoors
Tipos de MAC	IEEE 802.15 / IEEE 802.15.4 2.4GHz
Quantidade de nós	31
Tipo de tráfego	UDP
Tamanho do pacote de dados	Radio Messages 76 Bytes
Intervalo do envio de pacotes	10s
Tamanho da janela do agente UDP	95 bytes
Protocolos de base para rede	RPL, 6LoWPAN
Protocolos comparados	CoAP e MQTT
Sensores Simulados	Integrated Humidity, Temperature, and Light sensors
Microcontrolador	MSP430 (10k RAM, 48k Flash)

3.1.3.5. Cálculo de confiança para os resultados apresentados

Conforme [14], para se garantir a confiabilidade dos resultados utiliza-se o conceito de intervalo de confiança, no qual calculamos uma média de amostras a partir de um nível de confiança desejado. Neste caso termos 30 simulações para o protocolo CoAP e 30 simulações para o protocolo MQTT, em que serão separadas as médias de cada métrica de cada simulação, aplicando-se a fórmula do intervalo de confiança. Para este cálculo separaremos a média de cada métrica em cada simulação e o desvio padrão que será obtido da raiz quadrada variância padrão, que por sua vez é obtida da média dos desvios da média ao quadrado, e assim como [13] e [14], utilizaremos o nível de confiança de 95% por representar uma proporção aceitável do cálculo de erro, o valor de 95% na distribuição de probabilidade normal representa 0,475, que na tabela Z nos dá o valor de 1,96 como Z crítico, sendo assim nossa fórmula do Intervalo de confiança fica:

$IC = \bar{x}(\pm)1{,}96 * \left(\frac{\sigma}{\sqrt{n}}\right)$, sem fator de correção (N-n)/(N-1) pois usaremos a população como amostra e isto fica abaixo dos 5% do fator de correção.

Onde:

$\bar{\boldsymbol{x}}$ é a média da amostra simulada;

$\mathbf{1,96}$ é o Z crítico que será calculado com soma e subtração;

$\boldsymbol{\sigma}$ é o desvio padrão da amostra;

n é tamanho quantitativo da amostra.

4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Este capítulo visa apresentar os resultados obtidos das validações do simulador COOJA, bem como a análise e comparação das simulações após as caracterizações do ambiente de simulação, dos ambientes simulados e dos protocolos analisados. Então, como seria o comportamento de uma RSSF com aproximadamente 500 nós no que tange: latência, taxa de entrega, sobrecargas e taxas de transferência e o uso de seus sensores? A estrutura de testes pode ser observada pela Figura 14 e Figura 15.

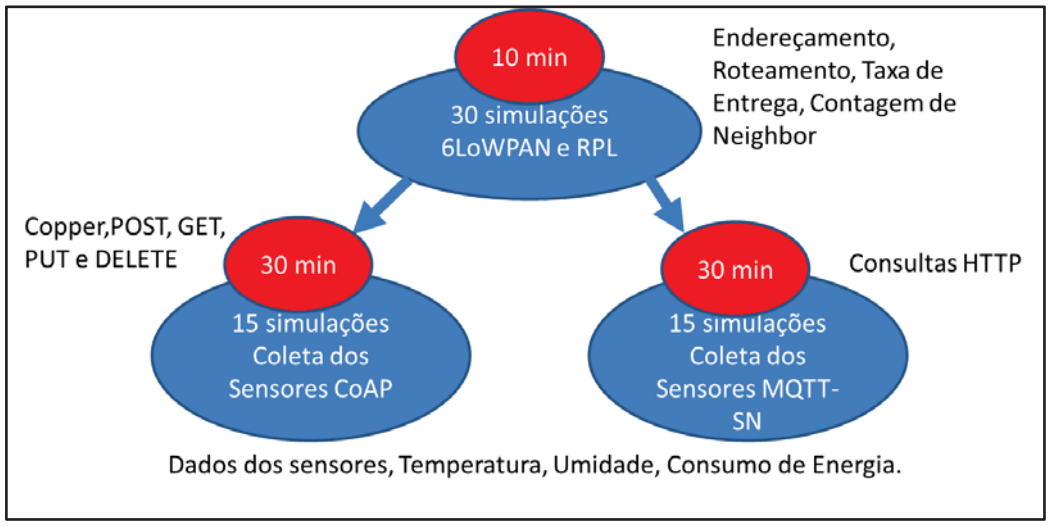

Figura 14. Estrutura da Simulação

4.1. RESULTADOS DA COMPOSIÇÃO DA REDE

Para composição dos resultados foram geradas 30 simulações do endereçamento de rede e do roteamento RPL, bem como a coleta de dados por protocolos CoAP e MQTT. Sendo assim, obteve-se 30 simulações da composição da rede, nas quais ambos tiveram a mesma composição da rede com transporte pelo padrão IEEE 802.15.4, endereçamento de rede IPv6, com compactação 6LoWPAN e roteamento por RPL conforme pode-se observar na Figura 15. Como a composição da rede não muda em ambas as simulações, neste capítulo será apresentado apenas a média dos testes.

4.1.1. Endereçamento da rede

Segundo [29], a proliferação de objetos inteligentes com capacidade de sensoriamento tem aumentado exponencialmente nos últimos tempos. Sendo assim, devemos explorar novos paradigmas de comunicação incluindo questões sobre endereçamento IP e adaptações para interoperar com a Internet. Conforme [38], ZigBee que utilizava um padrão proprietário próprio para a camada de adaptação, usa agora 6LoWPAN. Logo, não existe até o momento a necessidade de simular outro protocolo de endereçamento e adaptação IPv6.

A Figura 15 é composta pelo *background* dos postes das 2 quadras selecionadas, onde pode-se perceber o endereçamento IPv6 com adaptação 6LoWPAN, na qual o endereçamento foi composto com parte de informações de outras camadas como a de enlace conforme descrito na Tabela 6, na qual também são representadas as métricas em segundos.

Tabela 6. Tratamento de dados 6LoWPAN

Recurso	**Configuração**
População	30 simulações
Desvio Padrão	0,58
Nível de Confiança	95%
Intervalo de Confiança	0,42
Limite inferior	6,77
Média	6,98
Limite Superior	7,20
Valor mínimo	6,03
Quartil 1	6.32
Mediana (Quartil 2)	7,12
Quartil 3	7.45
Máximo	7,97

Na Figura 15, o *TX Range* representa o intervalo em que o pacote transmitido pode ser recebido corretamente por qualquer nó dentro diâmetro em verde com alcance de 70 metros, já o *INT Range* é a faixa na qual a transmissão pode ser ouvida, mas o pacote transmitido não pode ser recebido corretamente e é representada pelo diâmetro cinza com alcance de 90 metros.

Como pode-se observar nas simulações estas faixas são de extrema importância para o estabelecimento dos canais dos nós.

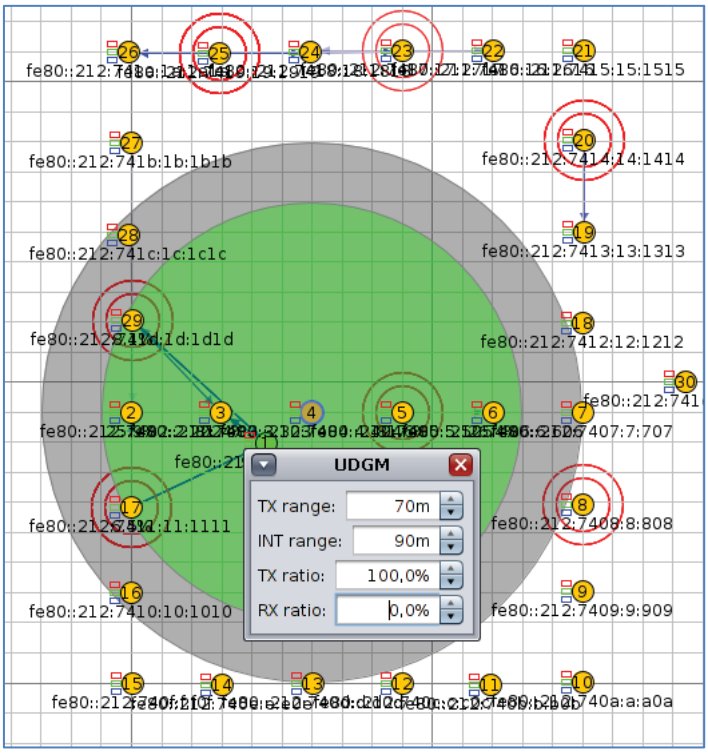

Figura 15. Recorte da simulação das duas quadras.

Para o tratamento dos dados da tabela 6 foram utilizados scripts na linguagem R. A representação da Tabela 6 pode ser melhor compreendida pela análise do gráfico de *Box Plot* na Figura 16, separados em quartis, mediana, limite inferior e superior dos dados coletados.

4.1.1.1. Discussão do resultado de endereçamento 6LoWPAN

Como pode-se observar, o valor mínimo corresponde a 6,03s, que é o valor do menor número da amostra, sendo que o intervalo de 25% de amostras corresponde até o valor de 6,32s que é o Quartil 1. A mediana ou quartil 2 encontra-se no valor de 7,12s e, contando do valor mínimo corresponde a 50% dos valores da amostra. O Quartil 3 apresenta 7.45s também tendo início no valor mínimo que corresponde a 75% das amostras e, por fim, o valor máximo de 7,97s que corresponde ao valor de 100% das amostras tendo como início o valor mínimo. Esta amostragem não apresentou valores discrepantes. Com o tratamento e análise destes dados,

podemos constar que uma média de amostragem similar será confiável se ficar entre 6,77s e 7,20s. Em outras palavras, de cada 100 amostras 95 estarão neste intervalo de confiança.

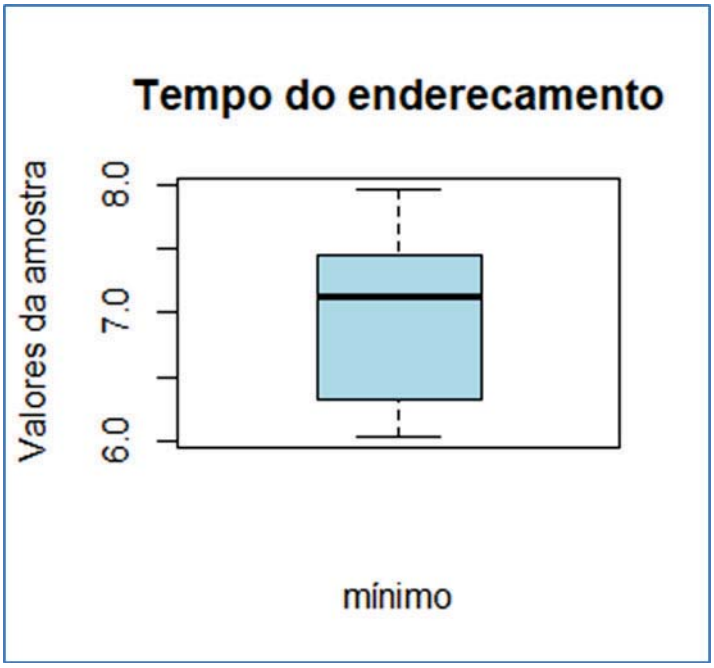

Figura 16.Box Plot Enderçamento

Com a média de 6,98s dividida por 30 postes, afirma-se com 95% de certeza que o endereçamento de cada poste leva em média 0,23s. Conforme [29], de fato o 6LoWPAN utiliza parte do MAC para compor o endereçamento reduzindo tempo de processamento em IoT, deixando a métrica de endereçamento semelhante ao IPv6 sem adaptação conforme podemos constar em [30] que a velocidade do DHCPv6 é em milissegundos e mesmo com poder de processamento menor ocorreu em 6LoWPAN, mantendo o desempenho de endereçamento.

4.1.2. Roteamento RPL

A principal característica do protocolo RPL, bem como o que o difere dos demais protocolos de roteamento é a capacidade de gerar um grafo acíclico dirigido também conhecido por DAG. A rede é organizada com um nó raiz que concentra o recebimento de dados de todos os demais nós e esta estrutura recebe o nome de DODAG. Na sessão 2.3.2 está detalhado o funcionamento da DAG e da DODAG pode-se observar o mesmo na Figura 7. A disposição da DODAG simulada é representada na Figura 17, bem como o nó raiz pode ser melhor observado

na Figura 10 e 17, representado pelo ID 1 em cor verde ao centro. Mas para melhor compreensão do resultado da composição da DODAG, precisa-se observar a Figura 17:

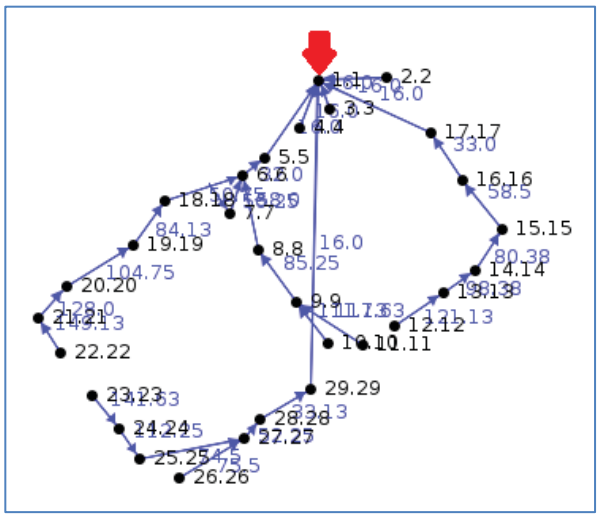

Figura 17. DODAG Simulada

4.1.2.1. Discussão dos resultados – Tempo de formação DODAG

O nó raiz que é representado pelo ID 1 e apontado com a seta vermelha na Figura 17, recebe as informações de todos os outros nós e sua orientação leva à criação do grafo acíclico dirigido, gerando uma DODAG que terá sua coleta e tratamento de dados de tempo em segundos explanados neste capítulo.

Tabela 7. Tratamento de dados tempo DODAG

Recurso	**Configuração**
População	30 simulações
Desvio Padrão	3,2
Nível de Confiança	95%
Intervalo de Confiança	2,39
Limite inferior	209,14
Média	210,34
Limite Superior	211,53
Valor mínimo	205,95
Quartil 1	207,33
Mediana (Quartil 2)	210,56

Quartil 3	213,47
Máximo	215,17

Os valores dados na Tabela 7 estão em rol de ordem crescente e, assim como na Tabela 6, representam os quartis. O primeiro quartil, Q1 até 207,33s representa 25% das observações. A mediana representa 50% dos valores da amostra até 210,56s. Enquanto que o terceiro quartil, Q3, deixa 75% das observações igual ou abaixo de 213,47s., e o número 215,17s representa o número máximo de 100% das amostras partindo do menor valor. Estas informações são representas na Figura 18:

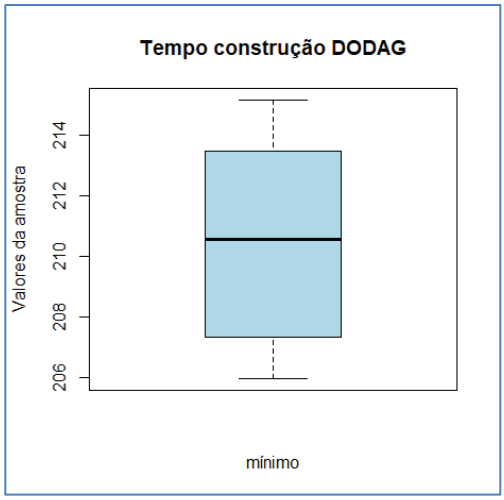

Figura 18. Tempo de construção da DODAG

O intervalo de confiança fica em 2,39s, o que dá a confiança que de cada 100 amostras 95 terão a média de composição da DODAG entre os valores de 209.14s e 211.53s para um grafo de 30 objetos (postes). Conclusão, por demorar para compor o grafo acíclico de roteamento dirigido, recomenda-se utilizar o RPL apenas para nós fixos, apesar de [9] propor uma avaliação em ambientes de mobilidade, mas como é citado, o rádio precisaria ficar ligado mais tempo para poder enviar e receber mensagens e assim atualizar a posição um DODAG ou até mesmo ingressar em um novo DODAG.

4.1.2.2. Taxa de Entrega RPL

Conforme [9], em redes LLNs, a perda de pacotes já é esperada, e é exatamente por este motivo que o RPL foi desenvolvido. A taxa de entrega de pacotes é contabilizada pela razão dos pacotes recebidos por pacotes enviados, ou seja, se na transmissão de 100 pacotes forem recebidos 90, tem-se a taxa de entrega em 90%. Logo, quanto maior for a taxa de entrega mais eficiente é o protocolo [21].

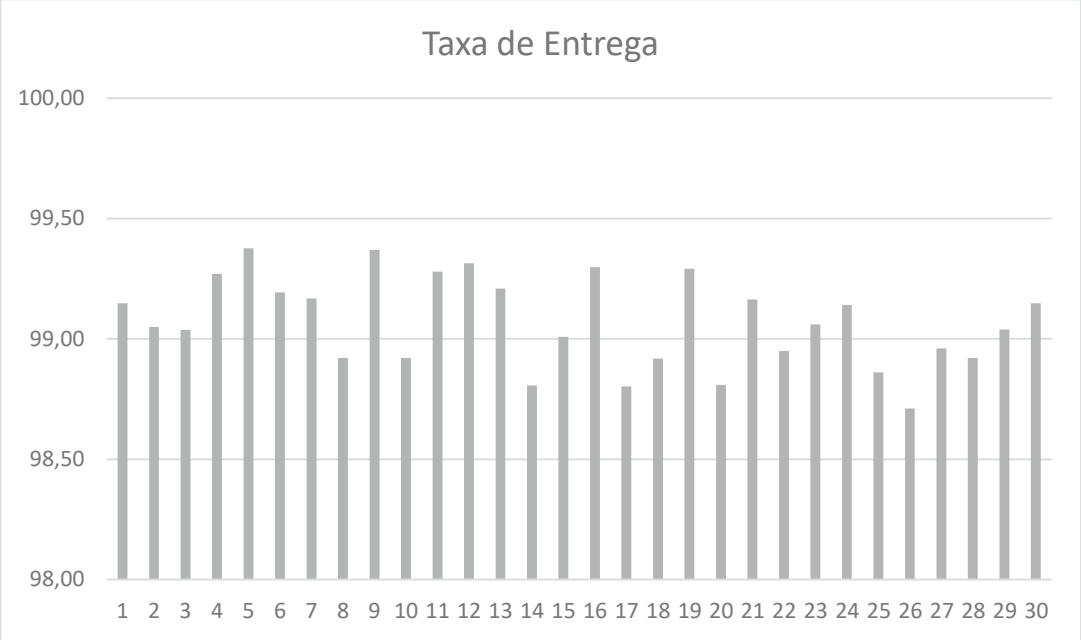

Figura 19. 30 Simulações taxa de entrega com RPL

A média de taxa de entrega para RPL, transportando mensagens de 3 sensores em nós fixos com o protocolo UDP é de 99,07%, sendo superior à de protocolos de roteamento de mobilidade em um comparativo com os protocolos testados em [13]. Ao analisar o desempenho na criação da DODAG, percebe-se que o RPL tem tempo elevado para composição do grafo de roteamento, não sendo indicado para nós móveis em função disto, mas em contrapartida, ao analisarmos a taxa de entrega, observa-se que é uma ótima escolha para nós fixos, pois consegue de forma muito eficiente contornar o problema de perdas em redes LLNs, conforme é exposto na Figura 19, onde se percebe que na simulação 26 em que ocorreu menor taxa de entrega foram entregues 98.71% dos pacotes.

4.1.2.3. Contagem de Neighbor

Segundo [5], a descoberta de vizinhança foi implementada para automatizar e facilitar a comunicação entre os hosts, e entre as suas funções está a de localizar dispositivos da rede. Sendo assim, o 6LoWPAN utiliza deste processo para garantir que não haja duplicação de endereços, pois terá a determinação dos endereços físicos de rede. Segundo [2], algumas características deste processo podem ser exploradas por usuários mal intencionados para realização de ataques. É de fundamental importância conhecer a estrutura de vizinhança e ter acesso sobre as informações compostas por ela.

Tabela 8. Tratamento de dados para Neighbor

Recurso	**Configuração**
População	30 simulações
Desvio Padrão	1.104179
Nível de Confiança	95%
Intervalo de Confiança	0.84
Limite inferior	4.40
Média	4.82
Limite Superior	5.24
Valor mínimo	3
Quartil 1	4
Mediana (Quartil 2)	5
Quartil 3	5
Máximo	7

Conforme pode ser analisado na Tabela 9, em média cada nó teve 5 vizinhos, sendo que o menor número de vizinhos foi de 3 e maior número de vizinho foi de 7. Com a garantia do nível de confiança, uma cidade que utilizar as quadras de suas ruas com disposição de postes em 30 metros, cada objeto inteligente terá em média 5 vizinhos. Sete vizinhos é um discrepante da amostra, ou seja, é um valor isolado que não pode ser contabilizado com nível de segurança, conforme exibido na Figura 20.

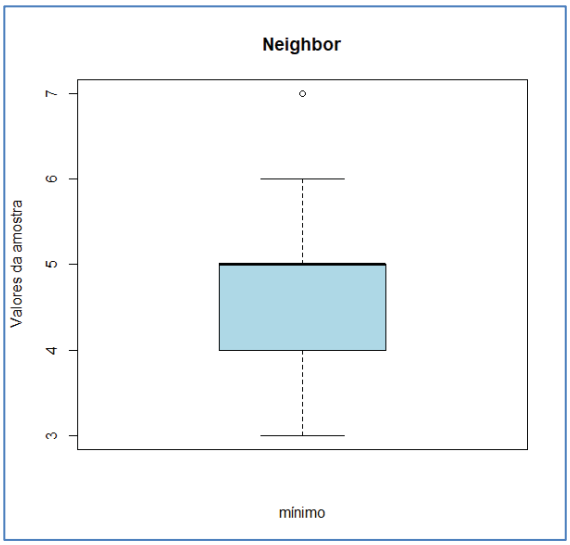

Figura 20. Boxplot Neighbor

Esta informação é importante, pois como a estrutura DODAG permite a conexão com mais de um nó, vizinhos indesejados podem se agregar à rede. Nestas simulações foram testadas apenas a capacidade de encontrar vizinhos e registrá-los sobre um IPv6 formado com o MAC de cada dispositivo, como o endereço físico é composto pelo IEEE em conjunto com fabricante, pode-se chegar à conclusão que a composição da rede foi realizada por equipamentos Tmotes *Sky*, mas isto não garante a segurança, pois o MAC pode ser alterado facilmente.

4.2. DADOS SIMULADOS NOS SENSORES

De encontro ao objetivo da dissertação, a simulação proposta pretende não apenas pesquisar a comunicação e endereçamento da rede, mas sim gerar dados de forma concisa como ocorre em IoT. Por este motivo foram simulados os Tmotes *Sky* com sensores de umidade, luz e temperatura, mas diferentemente do que aconteceu com as informações de rede, onde os dados analisados foram de 30 simulações. Neste caso, utilizaremos do conceito de estatística de amostragem aleatória, da qual foi separada a amostragem 6, pois os dados coletados dos sensores são fictícios, não havendo a necessidade do tratamento de dados. O importante na coleta é mostrar como seria possível trabalhar com essas informações. Suas coletas serão demonstradas a seguir, onde 0 significa sem registro.

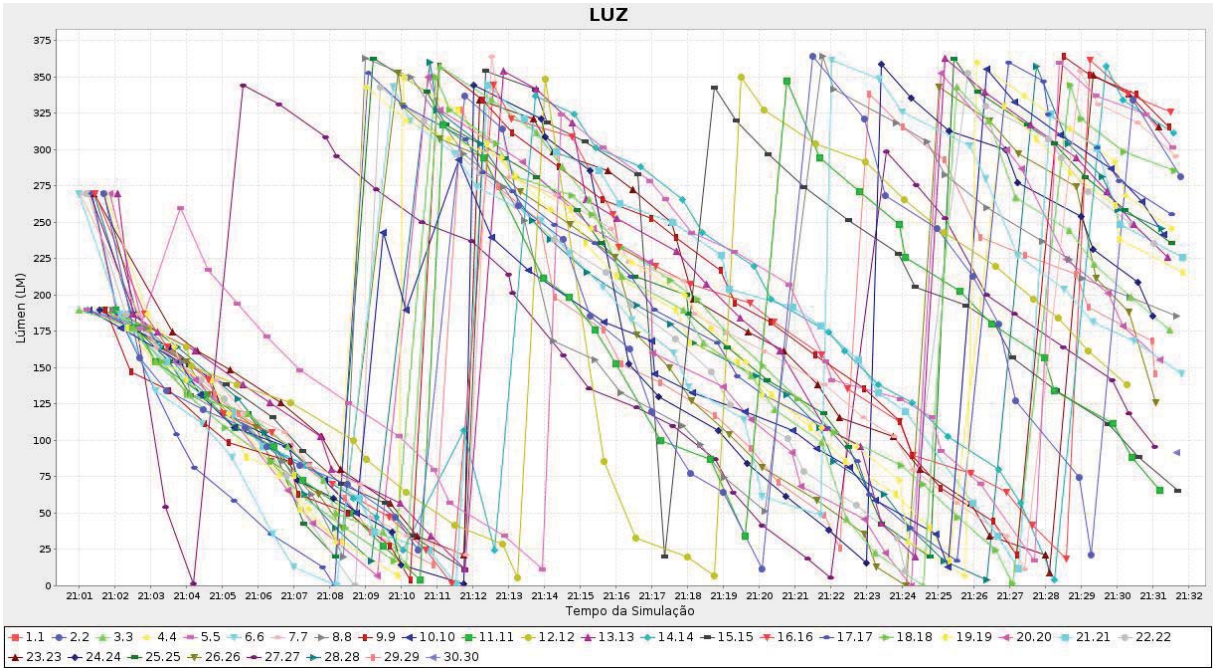

Figura 21. Dados do sensor de luminosidade

Na Figura 21, existem os dados do sensor de luminosidade, sendo que na simulação realizada foram coletadas 31 amostras durante o período de 30 minutos. Pode se notar que os postes tiveram elevação de luminosidade com mais de 350 lúmens e voltam a declinar. Esta simulação pode ser entendida como uma pessoa caminhando na rua, os postes acendendo conforme a aproximação e apagando conforme o afastamento do pedestre. Estes dados assim como todos tratados neste capítulo forma transmitidos do poste até o RPL *Collect*.

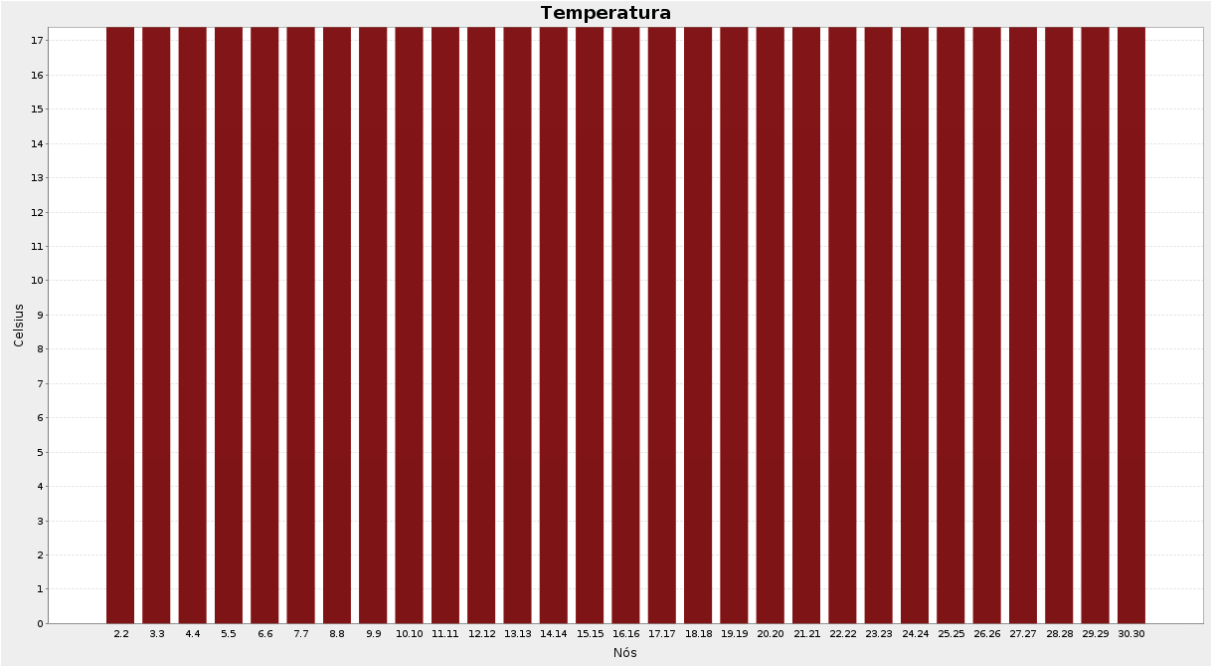

Figura 22. Dados do sensor de temperatura

A temperatura coletada refere-se ao período de 30 minutos, assim como pode se observar na Figura 22, os dados são constantes em 17.3ºC, pois assumiu-se a situação que a temperatura seria constante. Os dados foram capturados de forma perfeita, pois a simulação coletada era em 2 quadras que obviamente tem a mesma temperatura ou pouca diferença não sendo registrada na escala do gráfico.

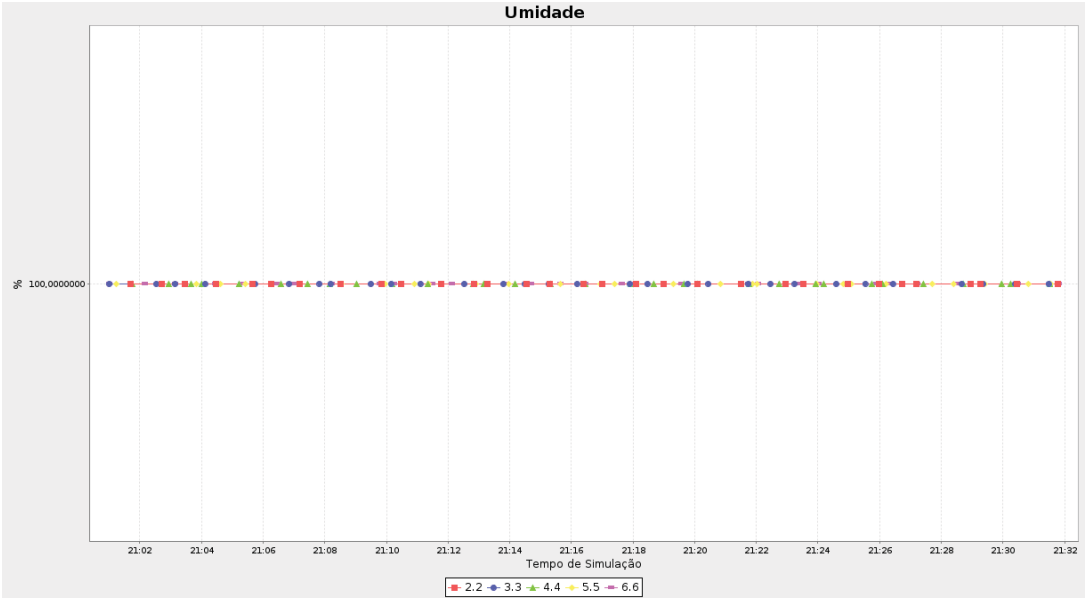

Figura 23. Dados do sensor de umidade

Do mesmo modo que assumiu-se valor constante para a temperatura, o mesmo ocorreu com a umidade relativa do ar, como exposto da Figura 23. A umidade relativa do ar estava em 100%, simulando uma situação em que o ar resfriou, possibilitando a chuva. Os dados foram coletados conforme o esperado, porém na Figura 23 por questões de visualização foi representado apenas os dados dos sensores dos Tmotes *Sky* 2,3,4,5 e 6.

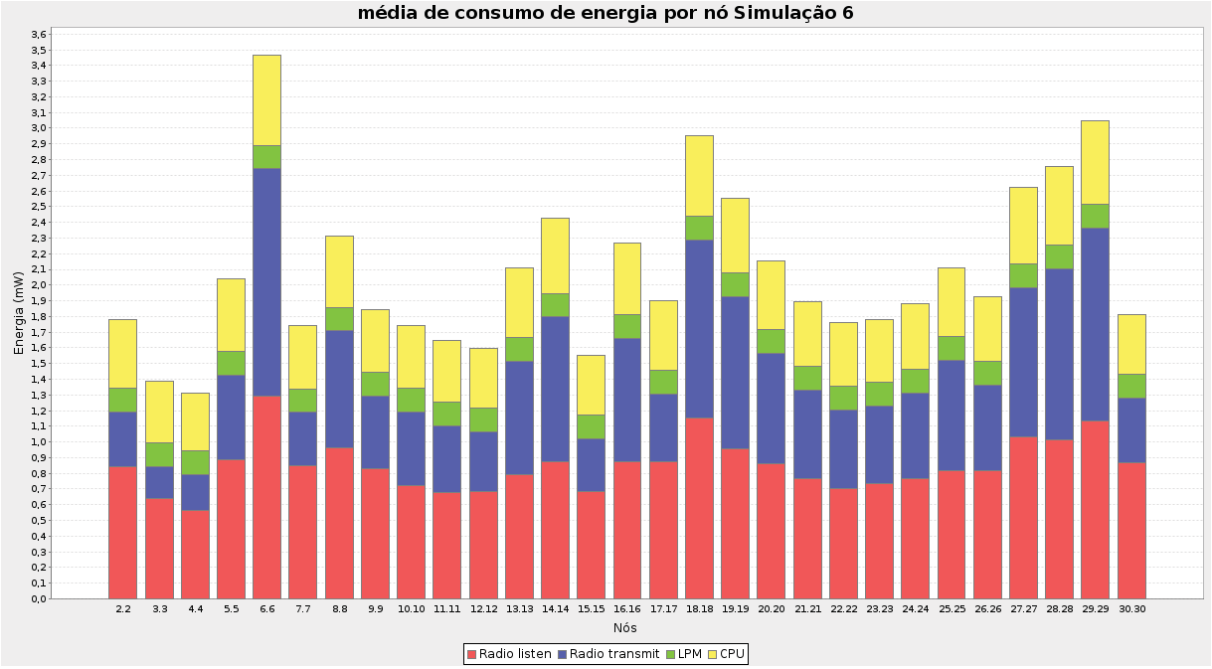

Figura 24. Média de consumo por nó

Na Figura 24, pode-se observar a média de consumo de energia por nó, no que diz respeito aos dados de escuta e transmissão do rádio, processamento da CPU e no modo *Low Power Mode*, que é o modo de baixo consumo automático. Nesse caso, a máquina tem a capacidade de se alterar automaticamente para o modo de baixo consumo se ficar inativa durante o período em que não está se comunicando. Os rádios que mais escutaram foram o 6 e o 18, pois sua posição, conforme a Figura 15, fica próxima ao nó 30 que simula a comunicação fora da quadra.

4.3. RESULTADOS COAP E MQTT

A aplicação CoAP foi simulada através de um exemplo que vem junto com o próprio COOJA, chamada de *Erbium Rest*, simulando uma *Representational State Transfer* (REST). O

CoAP simulado no contiki, permite várias implementações de métodos CRUD como POST, GET, PUT e DELETE.

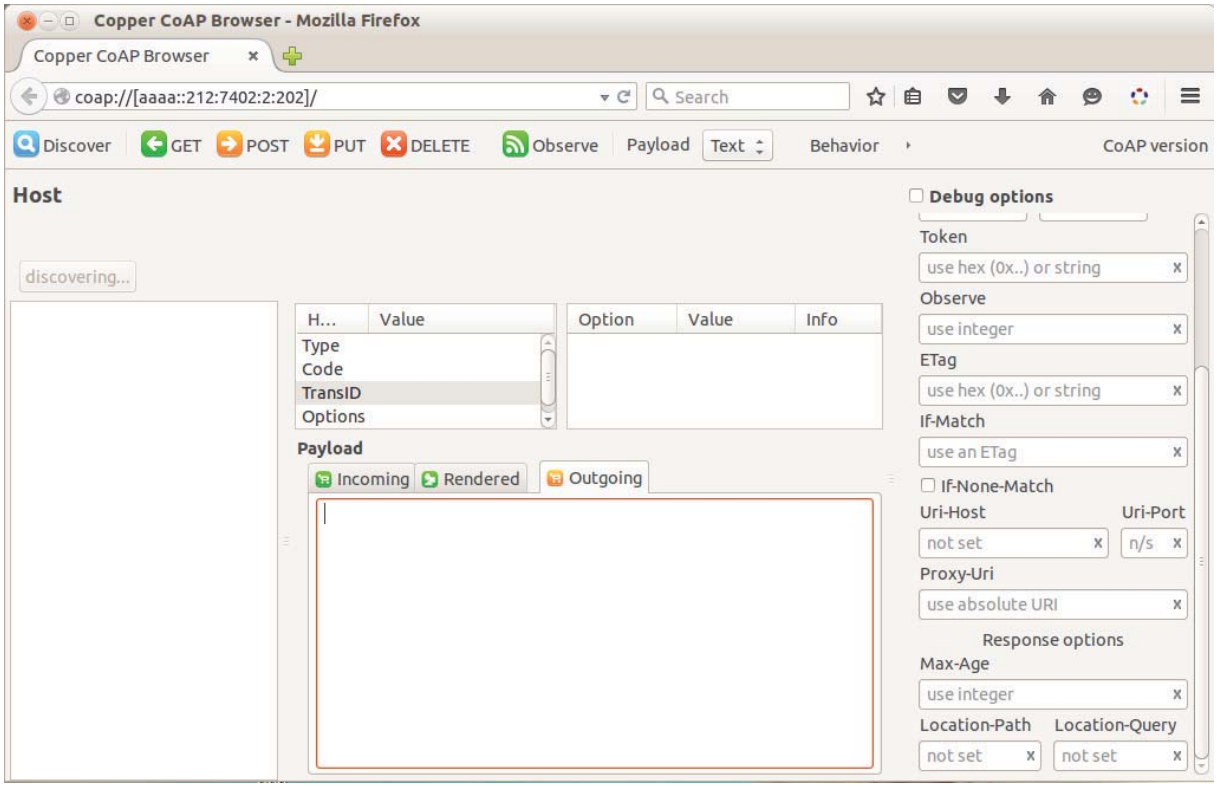

Figura 25. Tela de manipulação do CoAP

Na Figura 25, pode-se observar o navegador Copper, que interage com o protocolo CoAP. Ele foi testado com o recurso “hello”, que assim que selecionado alterou a URI. Foram testados o envio de POST acrescentando o número 1 em “Outgoing”, que serviu com envio de comando simulando o botão interruptor para acender um dos leds e o mesmo funcionou perfeitamente. O segundo teste foi receber a temperatura do sensor com GET, que também funcionou recebendo a mensagem: “*Current Temperature is:17.3*”, exatamente a temperatura configurada na simulação.

O CoAP criou um subconjunto da arquitetura *Representational State Transfer* [REST] da Web, que se assemelhou ao HTTP com os métodos de solicitação GET (1), POST (2), PUT (3) e DELETE (4). Esses métodos puderam ser usados para criar, atualizar, consultar e excluir os recursos no nó servidor que representa eventos de um aplicativo IoT.

Para que se pudesse ter os mesmos resultados, foi simulado o Procolo MQTT-SN, que foi desenvolvido para ser o mais parecido possível com o MQTT original, porém ele usa comunicação UDP. Ele é específico para redes de sensores e não depende do TCP/IP para

operar, também foi otimizado para aplicações de baixo custo, baixo consumo, junto com a simulação usou-se o *Really Small Message Broker* (RSMB), simulando um o *broker* do MQTT-SN que rodou na porta 1884 UDP6, porém esta implementação nos possibilitou apenas obter informações sobre os clientes MQTT-SN em modo texto ou em imagens similares a Figura 26, onde temos a coleta de dados do sensor de luminosidade.

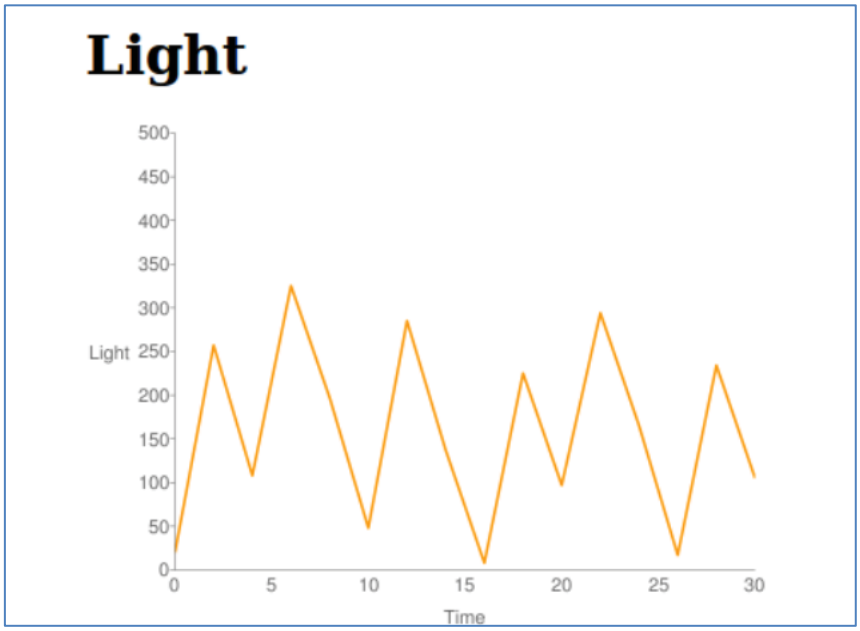

Figura 26. Informação do sensor de luminosidade

Através dos testes constatou-se que ambos protocolos cumprem as características descritas na bibliografia pesquisada, porém pelo fato do MQTT original usar TCP, não foi foco compará-lo com o protocolo CoAP. Logo, usou-se o MQTT-SN, que possui menos recursos que os protocolos citados neste capítulo, e com objetivo de testar a usabilidade e confiança na troca de mensagens, notamos que os protocolos atendem o proposto em suas RFCs.

Como trabalhos futuros, sugere-se que com a análise dos protocolos sendo executados em um ambiente controlado não é possível determinar qual deles seria a melhor aplicação para M2M. Neste caminho, indicam-se testes com *testebeds* físicos em ambientes não controlados, bem como a comparação dos protocolos CoAP e MQTT-SN quanto à segurança de dados. Sugere-se ainda testar a segurança na distribuição de endereços 6LoWPAN, uma vez que o mesmo compõe a rede através da camada de enlace e *Mac Address*, visto que após as pesquisas, acredita-se que seja possível a infiltração na rede por um dispositivo renegado.

5. CONCLUSÃO

O objetivo principal deste estudo foi simular uma rede RSSF, com dados de 3 sensores e com a aplicabilidade em uma cidade inteligente, observando o endereçamento 6LoWPAN, o roteamento RPL com a composição dos *neighbor*, e em cima desta estrutura operar 2 protocolos de aplicação o CoAP e o MQTT.

Para a realização dos experimentos foi criada uma simulação de 500 objetos, sendo que os dados coletados foram de 30 objetos. A RSSF foi simulada 30 vezes por 30 minutos para que os dados pudessem ser coletados tratados e analisados o mais fidedigno possível com a realidade das RSSF.

Após a revisão da literatura chegou-se à estrutura da RSSF e dos sensores que fariam sua composição, porém com a dificuldade de equilibrar as métricas dos protocolos MQTT e CoAP, que operavam com as mesmas informações, e com finalidades diferentes. Buscou-se um meio de equilibrar informações usando o MQTT-SN, no entanto, por ser uma aplicação adaptada ainda não apresenta todos recursos do modelo original. Mesmo assim, pode-se observar que o protocolo CoAP nestas simulações demonstrou-se mais flexível, dinâmico e de fácil operação, interagindo enviando e recebendo dados. Já com o MQTT-SN foi possível apenas obter a informação dos sensores sem enviar dados para os Tmotes *sky*, assim como ocorreu com o CoAP. Para melhor compreensão destes protocolos sugere-se, como trabalho futuro, implementá-los com *testbeds* físicos em ambiente não controlado. Desta forma, mais dados serão coletados e mais interações poderão ser testadas.

Quanto ao endereçamento da RSSF, conseguiu-se comprovar as características explanadas na literatura pesquisada. A rede levou em média 0,23s para o endereçamento de cada nó, confirmando a eficiência do 6LoWPAN em trabalhar com pouco processamento e memória, usando informações de outras camadas de rede para compor o endereçamento. Conclui-se que o 6 LoWPAN, realiza o endereçamento de forma eficiente e rápida, porém como utiliza parte do MAC para composição dos endereços, seria necessário em trabalhos futuros testar a segurança deste endereçamento.

Com a descoberta *neighbor*, a DODAG, levou mais de 3 minutos para composição da tabela de roteamento e do gráfico acíclico, sendo desta forma descartado o protocolo RPL para o roteamento móvel como sugerem algumas bibliografias, pois a cada modificação levaria 3 minutos para recompor a rede, sendo inviável para *manets*.

REFERÊNCIAS

[1] Abellá-García, A.; Ortiz-De-Urbina-Criado, M.; De-Pablos-Heredero, C.; Juan, R. 2015. The ecosystem of services around Smart cities: An exploratory analysis. *Procedia - Procedia Computer Science*. 64, (2015), 1075–1080.

[2] Baccelli, E.; Raggett, D. 2015. *The Internet of Things and The Web of Things*.

[3] Bahia, J.G.; Elias, M.; Campista, M. 2016. *Um Mecanismo de Comuta ao de Servidores CoAP para Aumento de Disponibilidade dos Serviços de IoT*.

[4] Barata, D.; Louzada, G.; Carreiro, A.; Damasceno, A. 2013. System of acquisition, transmission, storage and visualization of Pulse Oximeter and ECG data using Android and MQTT Selection and/or peer-review under responsibility of SCIKA – Association for Promotion and Dissemination of Scientific Knowledge. *Procedia Technology*. 9, 9 (2013), 1265–1272.

[5] Chakrabarti, S.; Shelby, Z.; Nordmark, E. *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*.

[6] Chalappuram, A.; Sreesh, P.R.; ## J.M.; ## G. 2016. ScienceDirect Global Colloquium in Recent Advancement and Effectual Researches in Development of 6LoWPAN in Embedded Wireless System. *Procedia Technology Science and Technology*. 25, (2016), 513–519.

[7] Chooruang, K.; Mangkalakeeree, P. 2016. ScienceDirect Wireless Heart Rate Monitoring System using MQTT. *Procedia Computer Science*. 86, (2016), 160–163.

[8] Contiki: The Open Source Operating System for the Internet of Things: *http://www.contiki-os.org/*. Accessed: 2018-06-17.

[9] Cotrim, Jeferson Rodrigues; Kleinschmidt, J.H. 2016. Avaliação de Desempenho do Protocolo RPL em Ambientes com Mobilidade. *XXXIV Simpósio Brasileiro de Telecomunicações*. September (2016), 45–49.

[10] Council, S.C. 2015. *Smart Cities Readness Guide.* 2013.

[11] Domingues, M.A. de O. 2004. Uma Abordagem Para Validação De Protocolos De Comunicação Em Ambientes De Simulaçao.

[12] Eclipse Mosquitto: *http://www.mosquitto.org/*. Accessed: 2018-06-17.

[13] Ferronato, J.J. 2015. *Análise De Protocolos De Roteamento Em Vanets De Diferentes Densidades Em Ambiente Urbano.*

[14] Firyaguna, A.C. de O.C.F. 2012. *Implementação E Simulação De Redes Ad Hoc De Comunicações Sem Fio Com Múltiplas Antenas*.

[15] Gaddour, O.; Koubâa, A. 2012. RPL in a nutshell: A survey. *Computer Networks*. 56, 14 (2012), 3163–3178.

[16] Internet of Things (IoT) for Real World Projects | Devices and Cloud Software: *https://www.advanticsys.com/shop/?lang=en*. Accessed: 2018-06-26.

[17] Ismael Rodrigues Martins3 and José Luís Zem 2014. *Estudo Dos Protocolos De Comunicação Mqtt E Coap Para Aplicações Machine-To-Machine E Internet Das Coisa.*

[18] Jaloudi, S. 2015. Open source software of smart city protocols current status and challenges. *2015 International Conference on Open Source Software Computing (OSSCOM)* (Sep. 2015), 1–6.

[19] Kazala, R.; Taneva, A.; Petrov, M.; Penkov, S. 2015. Wireless Network for Mobile Robot Applications. *IFAC-PapersOnLine*. 48, 24 (2015), 231–236.

[20] Kushalnagar, N.; Montenegro, G.; Culler, D.E.; Hui, J.W. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks.*

[21] Lorente, G.G.; Lemmens, B.; Carlier, M.; Braeken, A.; Steenhaut, K. 2016. ARTICLE IN PRESS BMRF: *Bidirectional Multicast RPL Forwarding*. (2016).

[22] Loseto, G.; Ieva, S.; Gramegna, F.; Ruta, M.; Scioscia, F.; Sciascio, E. Di 2016. ScienceDirect Linking the Web of Things: LDP-CoAP mapping. *Procedia Computer Science*. 83, (2016), 1182–1187.

[23] M2M and the Internet of Things: A guide | *ZDNet*: 2013. *http://www.zdnet.com/article/m2m-and-the-internet-of-things-a-guide/*. Accessed: 2016-07-16.

[24] Montenegro, G.; Kushalnagar, N.; Hui, J.; Culler, D. 2007. *RFC 4944*. (2007).

[25] MQTT V3.1 Protocol Specification: *http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html*. Accessed: 2016-10-13.

[26] Pegden, C.D. 1986. Introduction to SIMAN. *Proceedings of the 18th conference on Winter simulation - WSC '86* (New York, New York, USA, 1986), 95–103.

[27] Rede, S.E.E.D.E. 2014. Internet Das Coisas E 6LoWPAN.

[28] *RFC-7252* The Constrained Application Protocol (CoAP): 2014. *http://www.ietf.org/rfc/rfc7252.txt*. Accessed: 2016-08-21.

[29] Santos, B.P.; Silva, L.A.M.; Celes, C.S.F.S.; Borges, J.B. 2016. Internet das Coisas: da Teoria à Prática. *Minicursos do XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos — SBRC 2016*. (2016), 52.

[30] Schumacher, C.P.P., Kushalnagar, N.; Montenegro, G. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals.

[31] Solanki, V.K.; Katiyar, S.; BhashkarSemwal, V.; Dewan, P.,;Venkatasen, M.; Dey, N. 2016. Advanced Automated Module for Smart and Secure City. *Procedia Computer Science*. 78, (2016), 367–374.

[32] T. Winter, P.; Thubert; A. Rock, K. 2012. *RFC 6550 - RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*.

[33] Thomson, C. 2016. Cooja Simulator Manual.

[34] Using Streetlights to Strengthen Cities | Data-Smart City Solutions: 2016. *http://datasmart.ash.harvard.edu/news/article/using-streetlights-to-strengthen-cities-895*. Accessed: 2016-09-16.

[35] Vasseur, J.; Fellow, C.; Systems, C., Agarwal; N., Leader, T.; Hui, J. 2011. RPL: The IP routing protocol designed for low power and lossy networks Internet Protocol for Smart Objects (IPSO) Alliance. (2011).

[36] Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. 2014. Internet of Things for Smart Cities. *IEEE Internet of Things Journal*. 1, 1 (Feb. 2014), 22–32.

[37] Zhou, J.; Hu, L.; Wang, F.; Lu, H.; Zhao, K. 2013. An efficient multidimensional fusion algorithm for IoT data based on partitioning. *Tsinghua Science and Technology*. 18, 4 (Aug. 2013), 369–378.

[38] ZigBee usa agora 6loWPAN! Sua próxima lâmpada terá IPv6? *http://ipv6.br/post/zigbee-usa-agora-6lowpan-sua-proxima-lampada-tera-ipv6/*. Accessed: 2018-05-11.