Arthur Brito Cosmi

Análise de Desempenho de Protocolos de Comunicação para Internet das Coisas em Dispositivos com Baixa Capacidade de Processamento e Energia

Vitória - ES

Dezembro de 2018

Arthur Brito Cosmi

Análise de Desempenho de Protocolos de Comunicação para Internet das Coisas em Dispositivos com Baixa Capacidade de Processamento e Energia

Projeto de Graduação do curso de Engenharia de Computação apresentado ao Departamento de Informática da Universidade Federal do Espírito Santo

Universidade Federal do Espírito Santo - Ufes

Orientador: Vinícius Fernandes Soares Mota

Vitória - ES

Dezembro de 2018

Arthur Brito Cosmi

Análise de Desempenho de Protocolos de Comunicação para Internet das Coisas em Dispositivos com Baixa Capacidade de Processamento e Energia

Projeto de Graduação do curso de Engenharia de Computação apresentado ao Departamento de Informática da Universidade Federal do Espírito Santo

Monografia apresentada ao Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Engenheiro de Computação.

Dsc. Vinícius Fernandes Soares Mota
Departamento de Informática - Ufes -
Vitória

Dsc. José Gonçalves Pereira Filho
Departamento de Informática - Ufes -
Vitória

PhD. Rodolfo da Silva Villaça
Departamento de Tecnologia Industrial -
Ufes - Vitória

Vitória - ES
Dezembro de 2018

Agradecimentos

Agradeço, primeiramente, à Deus por todas as graças recebidas e por encontrar o conforto necessário nos momentos mais difíceis nesse período tão importante da minha vida.

Agradeço a minha família, por me incentivar na escolha desse curso e por todo apoio que recebi durante minha graduação. Agradeço muito por está realizando um sonho e isso devo graças a vocês.

Agradeço a Mirella, que esteve sempre ao meu lado nos melhores e nos piores momentos e que me deu forças para continuar. Muito obrigado por ser tão importante na minha vida.

Agradeço aos meus orientadores, Dsc. Vinícius Fernandes Soares Mota e Dsc. José Gonçalves Pereira Filho, que foram compreensivos e objetivos e por me passar a experiência necessária para discutir sobre esse assunto. Levarei para sempre o apoio e o ensinamento de vocês.

Por fim, agradeço aos amigos que conheci através da universidade e que me ajudaram a vencer os obstáculos da graduação e, também, aos que estavam longe, mas me apoiaram de outras formas possíveis.

Resumo

Este trabalho fez uma análise entre os protocolos com mais destaques em relação a Internet das Coisas a fim de determinar onde cada um se aplica da melhor maneira possível. Foi feito um levantamento da atuação das principais empresas do mercado com a finalidade de delimitar cenários de aplicação para ajudar na escolha dos protocolos. Além disso, comparou-se dois dos principais protocolos para o mundo da IoT em uma abordagem diferente. Foi utilizado como principal referência um dispositivo com baixo poder de processamento e energia, que tem ganhado espaço pelo baixo custo e por recursos como conexão *WiFi* já integrada ao *chip*, o que torna fácil a interação com sistemas residenciais e empresariais. Os protocolos escolhidos foram o *Message Queue Telemetry Transport* (MQTT) e o *Constrained Application Protocol* (CoAP) e o dispositivo escolhido foi o ESP8266. Os testes foram feitos com e sem tráfego de fundo utilizando um roteador comum. Foi feita uma análise da forma como os protocolos se comportam nesses dispositivos em relação ao tamanho das mensagens e ao intervalo entre envio de mensagens e como esses parâmetros podem afetar uma estrututa de rede comum. Com isso, observou-se que o protocolo MQTT, mesmo utilizando o TCP como base, conseguiu enviar mensagens sem muitas retransmissões em relação ao CoAP. Além disso, com intervalos pequenos, os dois protocolos são influenciados pelo tráfego de fundo do roteador.

Palavras-chave: Internet das Coisas, MQTT, CoAP, ESP8266, Análise de desempenho.

Abstract

This work made an analysis between the protocols with more highlights in relation to the Internet of Things in order to determine where each one applies in the best possible way. It was made a survey of the performance of the main companies in the market with the purpose of delimiting application scenarios to assist in the choice of protocols. In addition, we compared two of the main protocols to the Internet of Things in a different approach. The main reference was a device with low processing power and energy, which has gained space at low cost and features such as *WiFi* connection already integrated to the *chip*, which makes it easy to interact with residential systems and business. The protocols chosen were *Message Queue Telemetry Transport* (MQTT) and *Constrained Application Protocol* (CoAP) and the device chosen was ESP8266. The tests were done with and without background traffic using a common router. An analysis was made of how protocols behave in these devices in relation to the size of the messages and the interval between sending messages and how these parameters can affect a common network structure. With this, it was observed that the MQTT protocol, even using TCP as base, was able to send messages without many retransmissions in relation to CoAP. In addition, at small intervals, the two protocols are influenced by the background traffic of the router.

Keywords: Internet of Things, MQTT, ESP8266, CoAP, Performance Analysis.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AMQP	*Advanced Message Queuing Protocol*
CoAP	*Constrained Application Protocol*
DARPA	*Defense Advanced Research Projects Agency*
DNS	*Domain Name System*
DTLS	*Datagram Transport Layer Security*
FTP	*File Transfer Protocol*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IoT	*Internet of Things*
IP	*Internet Protocol*
MOM	*Message-Oriented Middleware*
MQTT	*Message Queue Telemetry Transport*
MQTT-SN	*Message Queue Telemetry Transport for Sensor Networks*
M2M	*Machine to Machine*
QoS	*Quality of Service*
REST	*REpresentational State Transfer*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
UFES	Universidade Federal do Espírito Santo
UTF-8	*8-bit Unicode Transformation Format*
WEB	*World Wide Web*

Sumário

1 Apresentação do tema

1.1 Introdução

A Internet das Coisas vem ganhando destaque nos últimos anos e se caracteriza por um objeto capaz de recolher informações do ambiente e conseguir enviá-las a outros dispositivos. Esse objeto pode ser qualquer coisa, desde um sensor em um forno de uma siderúrgica até mesmo um sensor conectado a uma pastilha de freio em um carro. A proposta desse conceito é recolher e processar os dados, atuar de maneira imediata em situações para minimizar uma falha ou atraso contido em uma resposta humana. Essas informações recolhidas podem ser utilizadas para prevenção de acidentes, automatização de tarefas, maximização do uso de recursos diminuindo o desperdício em uma empresa e, por consequência, o preço dos produtos, monitoramento de pacientes com marcapasso ou outros equipamentos que precisam de acompanhamento em tempo real, entre outras formas de uso que facilitam e deixam a vida mais segura.

Pelo crescente número de aplicações possíveis, a estrutura física dos dispositivos ligados aos sensores deve ser maleável incluindo *hardwares* com tamanho restrito. Isto implica em pouco poder de processamento de dados, em pouco espaço de armazenamento de informações e na construção do dado a ser processado. Além disso, a questão energética torna-se importante, pois muitas vezes o local de instalação desses dispositivos não possibilita a ligação direta em uma rede elétrica ou a troca da bateria deve ser feita em períodos espaçados de tempo.

Por conta disso, alguns parâmetros importantes devem ser levados em consideração na hora de escolher qual o sistema atende as necessidades de um usuário de maneira mais eficiente aproveitando o *hardware* restrito disponível. O protocolo da camada de aplicação, de transporte, a forma de roteamento e de acesso ao meio de cada dispositivo e a conexão da camada física são parâmetros que direcionam para uma escolha mais eficiente. Entretanto, propor novos protocolos em algumas das camadas citadas pode causar certos transtornos como a falta de compatibilidade com os equipamentos que já estão em uso. Outro problema é fazer com que esses protocolos atinjam a mesma proporção daqueles que são, atualmente, padrões na Internet em geral. Isso pode levar anos ou décadas para acontecer.

Por isso, analisar os protocolos da camada de aplicação acaba sendo uma possibilidade plausível, já que esses protocolos estão mais próximos as aplicações dos usuários e não afetam diretamente serviços de roteamento e de criação de sessão. Por conta disso, o número de protocolos voltados para IoT tem crescido para suprir as necessidades de

diferentes empresas que estão se interessando no assunto.

O grande problema é que não existe uma definição muito precisa de quando se deve utilizar determinado protocolo nem quais os impactos provocados por essa escolha. A partir da análise das aplicações disponíveis no mercado é possível mapear cenários em que esses protocolos podem ser utilizados de maneira mais eficiente e quais métricas são mais importantes para cada um desses cenários.

1.2 Motivação e Justificativa

O ramo de pesquisa sobre Internet das Coisas tem se mostrado bastante promissor, visto que, o conforto proporcionado pela automatização de tarefas cotidianas aliado ao baixo custo na produção dos microcontroladores e dos sensores vem chamando atenção de grande parte do mercado consumidor. Outro ponto importante é a diversidade de cenários que podem ser explorados a partir da redução no tamanho dos dispositivos e qual o impacto gerado pela troca de informação pela mesma rede utilizada para fins gerais. Como as tecnologias de roteamento e transporte de dados já seguem um padrão adotado no mundo inteiro, analisar o protocolo que está ligado diretamente a aplicação é uma forma de usufruir ao máximo da capacidade dos dispositivos sem que isso dificulte a vida do usuário final.

Assim, com o crescente interesse nessa área, vários protocolos da camada de aplicação foram e estão sendo desenvolvidos para suprir as necessidades de grandes empresas. Muitas aplicações estão sendo desenvolvidas, porém não é fácil encontrar estudos comparativos sobre quando e quais impactos o uso desses protocolos tem em uma infraestrutura de rede já definida. Ainda mais se o foco for dispositivos com baixo poder de processamento e energia.

Diante do exposto, este trabalho justifica-se pela carência de estudos relacionados a classificação de alguns dos protocolos da camada de aplicação mais utilizados atualmente nos cenários residencial, comercial e industrial. Além disso, alguns dos protocolos foram testados em um cenário residencial com um dispositivo restrito a fim de examinar o desempenho em uma rede com e sem tráfego de fundo.

1.3 Objetivos

1.3.1 Objetivo Geral

Analisar o melhor cenário para alguns dos principais protocolos da camada de aplicação para Internet das Coisas, além de buscar o impacto da utilização dos protocolos MQTT e CoAP em uma rede local com e sem tráfego de fundo utilizando um dispositivo de baixo

poder de processamento e energia.

1.3.2 Objetivos Específicos

Fundamentar e analisar a base teórica do tema proposto a fim de extrair um sólido conhecimento dos protocolos MQTT, MQTT-SN, CoAP, AMQP e DDS. Determinar, a partir da teoria e entre os protocolos citados anteriormente, qual o melhor cenário - residencial, comercial ou industrial - para sua utilização. Utilizar métodos para analise do comportamento dos protocolos MQTT e CoAP em dispositivos com recursos restritos e em relação ao tráfego de fundo em uma infraestrutura de rede no cenário residencial. Apresentar e discutir os resultados dos impactos que a rede pode produzir em cima de cada protocolo, além do seu desempenho em relação ao tamanho da mensagem e a frequência de envio de mensagens.

1.4 Principais Contribuições

Após a análise das especificações dos protocolos, da teoria sobre eles e do estudo da aplicação dos mesmos em IoT, além dos testes realizados para a elaboração desse trabalho, segue as principais contribuições obtidas.

- Classificar e categorizar os protocolos segundo cenários de aplicação específicos;
- Corrigir e propor mudanças, além de distribuir as correções em algumas das bibliotecas utilizadas;
- Implementar, testar e validar o uso dos protocolos com e sem tráfego de fundo.

2 Fundamentação Teórica

2.1 Arquitetura de Rede

A comunicação entre dois dispositivos é bastante complexa e envolve a integração entre *software* e *hardware* para a troca de informações. Algumas dessas dificuldades foram descritas por (STALLINGS, 2007) como:

- Ao enviar um dado, um dispositivo deve informar à rede a intenção de iniciar uma comunicação para evitar colisões;
- Deve ter a certeza que o destino está ativo e pronto para receber informação;
- A aplicação no destino deve ser capaz de receber o dado e filtrar qual usuário pode ter acesso;
- Deve traduzir a informação caso o formato adequado que o dispositivo de destino possa entender.

Por causa desse grande número de atividades que devem ser executados para resolver os problemas citados, a implementação da arquitetura de rede foi desenvolvida pensando na modularidade criando as conhecidas camadas de rede onde cada camada presta um serviço para a camada superior e estes serviços são independentes podendo ser atualizados sem implicar diretamente na modificação de outras camadas (KUROSE; ROSS, 2012). Portanto, essas camadas devem ser implementadas nos dois dispositivos envolvidos.

Dessa forma, as camadas formam uma estrutura vertical de comunicação. No entanto, para que haja uma interação entre os serviços de camadas semelhantes, em dispositivos diferentes, foram criados os protocolos de rede. Esses protocolos são regras que mostram como os serviços são acessados e como os dados são tratados. Assim, cada camada possui inúmeros protocolos, sendo alguns deles bastante conhecidos como o FTP, HTTP e o IP.

A associação de protocolos de diferentes camadas é conhecida como pilha de protocolos. A pilha TCP/IP, desenvolvida pela DARPA e exemplificada na Figura 1, é a base da comunicação entre dispositivos até os dias de hoje e, por mais que receba esse nome, é composta por inúmeros outros protocolos que estão distribuídos em cinco camadas: Física, Enlace, Rede, Transporte e Aplicação (STALLINGS, 2007).

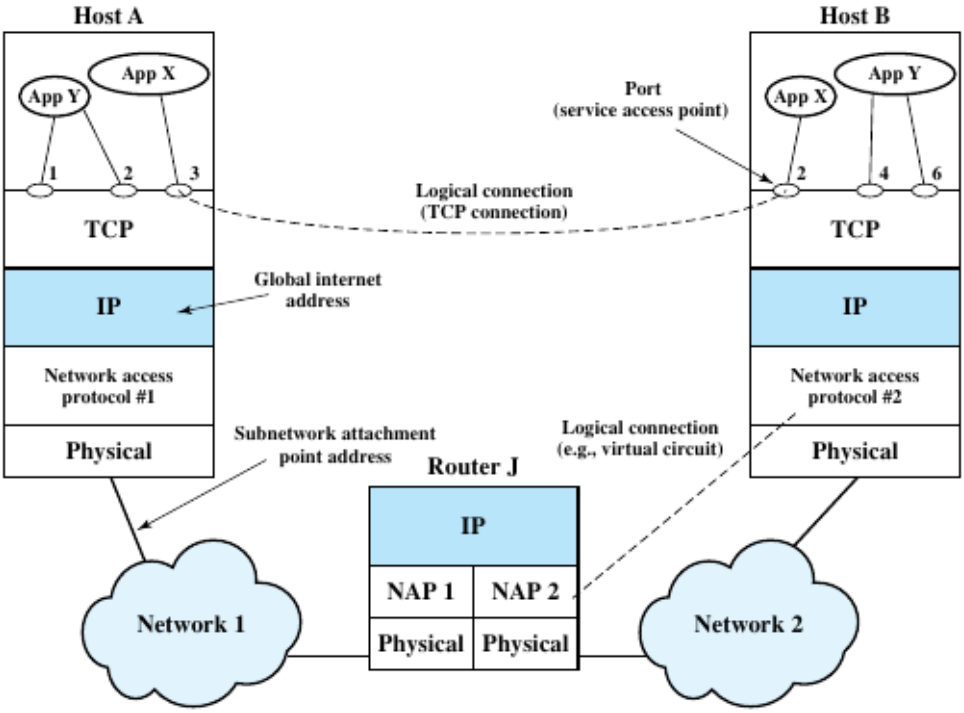


Figura 1 – Pilha de Protocolos TCP/IP

Fonte – Stallings,2007, p. 36

2.1.1 Camada Física

A camada Física é implementada em *hardware* e tem a função de enviar os *bits* de um dispositivo até o próximo nó da rede. Nesse ponto, não há informação do destino e os protocolos são ligados diretamente ao meio que é usado para comunicação, como cabo de par trançado, cabo coaxial, redes sem fio, entre outros (KUROSE; ROSS, 2012). Além disso, características como tipo do sinal (analógico ou digital) e taxa de transferência são gerenciados nessa camada (STALLINGS, 2007).

2.1.2 Camada de Enlace

A camada de Enlace é responsável por agrupar os dados em pacotes conhecidos como *frames* e enviá-los para o próximo dispositivo da rede com um endereço local do destino, ou seja, antes de ser direcionado para a internet por um roteador (KUROSE; ROSS, 2012). A camada de Enlace ainda é ligada ao meio em que a comunicação será estabelecida e faz o controle de acesso à esse meio para evitar as colisões dos *frames*, ou seja, para diminuir a perda de pacotes e o consumo de recursos dos equipamentos em tentativas de reenvio de informações.

2.1.3 Camada de Rede

A camada de Rede tem a função principal de rotear um pacote, conhecido nessa camada como datagrama, ou seja, transmitir esse pacote para qualquer lugar do mundo. Um dos mais famosos protocolos está nessa camada: o protocolo IP. Ele define o endereço de destino e algumas ações que os roteadores ou que outros dispositivos comuns devem executar ao receber o datagrama. Nessa camada ainda existem outros procotolos que são específicos para atualização de tabelas de roteamento entre os roteadores (KUROSE; ROSS, 2012).

2.1.4 Camada de Transporte

A camada de Transporte é a primeira camada que realiza a comunicação entre aplicações. Os pacotes nessa camada são chamados de segmentos. Ela é responsável por organizar o fluxo de dados para as aplicações com o uso das portas, além de fragmentar dados muitos extensos e ordená-los para que seja possível remontá-los no dispositivo de destino com alta confiabilidade. Um protocolo que faz este serviço é o protocolo TCP (STALLINGS, 2007). Existe também o protocolo UDP, contudo, não realiza uma entrega segura. (KUROSE; ROSS, 2012).

2.1.5 Camada de Aplicação

A camada de Aplicação é onde os aplicativos dos usuários estão sendo executados. Diversos protocolos são executados nessa camada como o HTTP, FTP e DNS. As informações geradas nessa camada são mais complexas e maiores e são conhecidas como mensagens (KUROSE; ROSS, 2012). Devido a essa proximidade com o usuário, os protocolos são mais robustos para suprir necessidades que aparecem ao longo do tempo tanto em questão de segurança quanto no uso das aplicações que tendem cada vez mais à computação na nuvem aumentando consideravelmente o tráfego de informações na rede.

2.2 Protocolos da Camada de Aplicação

Os protocolos da camada de aplicação são alvos para a implementação da Internet das Coisas, pois é nessa camada que a aplicação de um sensor, por exemplo, transforma a informação recolhida em algo que pode ser utilizado por outro dispositivo e este sabe exatamente o que é e o que fazer com aqueles dados. Quanto mais sofisticada for a aplicação, ou seja, quanto mais recursos uma aplicação dispõe ao usuário, maior o uso do processador do dispositivo para montar ou desmontar a informação recebida e, consequentemente, maior é o consumo energético (NAIK, 2017).

Outro ponto que fortalece a necessidade de modificar os protocolos de aplicação é que a Internet das Coisas está inserida em um mundo já implementado com seus protocolos de camadas físicas e enlace, que são protocolos implementados em *hardware*, já estabelecidos e uma mudança brusca pode não ser eficiente para as pessoas que buscam adicionar uma funcionalidade sem a necessidade de mudar toda uma infraestrutura para tal.

Uma característica que pode ajudar na escolha do melhor protocolo é a arquitetura de comunicação. Ela é classificada como Telemetria, Consulta, Comando e Notificação.

> "Em Telemetria, a informação flui dos dispositivos para a nuvem informando mudanças de estados nos dispositivos [...] Para o padrão Consulta, as requisições vêm dos dispositivos para a nuvem para coletar informações requeridas [...] No padrão Comandos, comandos são enviados dos sistemas para os dispositivos para desempenhar atividades específicas. [...] Por fim, em Notificação, a informação flui dos sistemas para os dispositivos, lidando com mudanças de estados no mundo físico [...]"(MAZZER; FRIGIERI; PARREIRA, 2015, p. 6).

Assim, inúmeros protocolos estão sendo criados para encaixar o cenário de eficiência no uso de recursos descrito anteriormente com esses padrões de comunicação. Alguns se destacaram nos últimos anos pela simplicidade e pela facilidade de implementação, além de resultados interessantes para o mundo dos dispositivos restritos. Alguns deles serão brevemente comentados a seguir.

2.2.1 Message Queue Telemetry Transport - MQTT

O MQTT é um protocolo de comunicação criado pela IBM e padronizado pelo consórcio OASIS com versão mais difundida sendo a 3.1.1 e é muito utilizado na cenário de *IoT* devido a seu cabeçalho reduzido, ou seja, pela pequena quantidade de informações que os dispositivos utilizam para identificar e extrair corretamente os dados partir de um pacote (alguns pacotes possuem apenas dois bytes), e sua simplicidade de implementação (MAZZER; FRIGIERI; PARREIRA, 2015). O MQTT funciona por um método conhecido como *Publisher/Subscriber* onde não há comunicação direta entre os clientes, ou seja, os clientes sempre se conectam ao *Broker*. Assim, para que um cliente receba informações (*Subscribers*, por exemplo, um celular que deseja saber a informação de temperatura próxima a um forno) ele precisa se inscrever em um tópico, que é uma estrutura hierárquica que explicita um assunto de interesse e que é acessado de maneira bem próxima a uma navegação entre pastas em um sistema de arquivos de um sistema operacional. Por outro lado, existem os clientes que enviam informações (*Publishers*, por exemplo, sensores de temperatura, movimento, distância, entre outros) que recolhem a informação do ambiente e enviam para um tópico específico. Assim, o MQTT possui uma característica de comunicação onde a informação publicada por apenas um cliente em um tópico pode chegar a vários outros clientes que estão inscritos naquele tópico (OASIS, 2014). Por essas características, é possível entender as dimensões desses dispositivos na rede. O *Broker*

é uma estrutura que possui maior capacidade de processamento e armazenamento por ser responsável por manter a comunicação entre os dispositivos. Ele deve ser responsável por manter informações de clientes conectados a ele, de mensagens que podem ser armazenadas para posterior envio, deve conhecer todos os tópicos e verificar informações de segurança para saber se um cliente pode acessar aquele tópico. Os clientes possuem tamanho reduzido, pois são geralmente sensores ou atuadores que são posicionados nos mais diversos tipos de locais, sendo muitas vezes, locais de difícil acesso. Isso também expõe uma característica importante desses tipos de dispositivos que é o uso de rede sem fio.

O MQTT por ser um protocolo da camada de aplicação, ou seja, um protocolo de nível mais alto, define seu funcionamento em cima da pilha de protocolos TCP/IP e, por isso, funciona com um padrão de acesso ao meio e de camada física bastante difundido pelo mundo que é o IEEE 802.11. Por conta do protocolo de transporte utilizado, o MQTT se aproveita de algumas características importantes como a fragmentação dos pacotes, que podem ter até duzentos e cinquenta e seis *MegaBytes*, controle de fluxo e de retransmissão de pacotes, além de delegar a manutenção da sessão aberta entre o broker e o cliente, não implementando mecanismos para resolver os problemas citados (OASIS, 2014). Porém, para redes muito sensíveis tanto em relação à quantidade de dados que circulam nela ou para sistemas que precisem de uma forma garantia que aquela informação foi realmente recebida pelo destinatário, o MQTT possui três níveis de QoS (LEE et al., 2013), que são exemplificados na Figura 2:

- QoS 0: também conhecido como “ no máximo um ” é um nível que não requisita confirmação de recebimento por parte do destinatário. Ele deixa a entrega do pacote sobre somente as capacidades da rede.
- QoS 1: também conhecido como “ no mínimo um ” é um nível que garante que pelo menos uma mensagem seja entregue ao destinatário. É uma camada a mais de confiabilidade em cima da capacidade da rede. Ele é caracterizado pelo envio de uma mensagem de confirmação de recebimento por meio do destinatário.
- QoS 2: também conhecido como “ exatamente um ” é um nível que garante dois passos de confirmação de recebimento entre os dispositivos que estão se comunicando.

Divulgada em dezembro de 2017, a nova versão do protocolo, chamada de MQTTv5, trouxe algumas modificações importantes que aumentam a faixa de aplicabilidade do protocolo. São medidas que tornam o envio de mensagens mais genérico de forma que a aplicação possa tratar os dados enviados mais facilmente. Além disso, foi criado um sistema de controle de fluxo acima da camada TCP facilitando e garantindo a entrega de

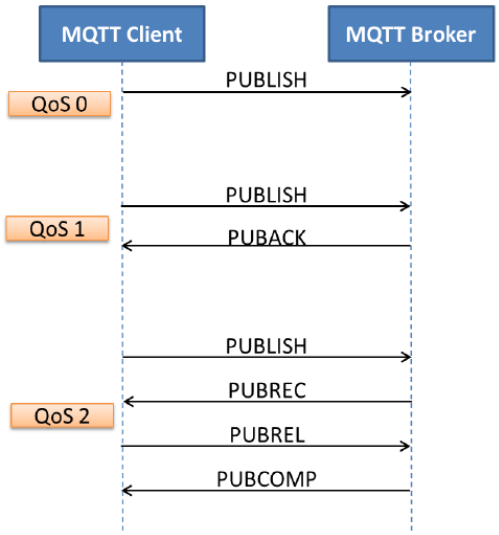


Figura 2 – Mensagens Trocadas a Cada Nível de QoS no MQTT

Fonte – Kodali,2017, p. 2

mensagens em grande quantidade de tamanho e/ou frequência de envio. Foi desenvolvido também, uma forma alternativa para indicação do tópico a ser publicado com a possibilidade de criar um identificador numérico único no *Broker* para um tópico e os clientes enviam suas mensagens indicando esse identificador ao invés de mandar um conjunto de caracteres em UTF-8, como na versão anterior. Outro ponto de destaque é na segurança. Foi implementado um novo tipo de pacote específico para autenticação sendo possível utilizar sistemas de autenticação baseados no SASL (Simple Authentication and Security Layer) e usando criptografia ponto a ponto com o TLS (Transport Layer Security). Porém, isso tudo tem um preço. O aumento dos recursos necessários para aproveitar essas melhorias pode restringir o número de dispositivos com pouco poder de processamento que estão disponíveis hoje no mercado (OASIS, 2017).

2.2.2 Message Queue Telemetry Transport for Sensor Networks - MQTT-SN

Por mais que o MQTT seja um protocolo leve, como mostrado anteriormente, ele possui algumas características que podem diminuir a sua aplicação em redes de sensores. O principal problema, é a utilização do protocolo TCP para o transporte dos dados. Por mais que seja um protocolo seguro, o seu cabeçalho relativamente grande com mais de vinte *bytes* junto com a troca de informações antes da criação da conexão, como visto em (STALLINGS, 2007), geram um tráfego muito grande de informações que não são úteis quando considera-se uma rede de sensores com mil ou dez mil equipamentos. Outro problema que pode acontecer pelo tamanho do pacote é a retransmissão desnecessária em redes com um nível alto de ruídos. Nessa situação, quanto maior o datagrama, maiores são as chances de que um ruído atrapalhe a entrega do dado fazendo com que o transmissor tenha que enviar outro dado igual. Isso gera mais tráfego na rede aumentando a chance

de colisão em uma rede deficitária como mostrado em (MUN; DINH; KWON, 2016). Essa situação só tende a piorar criando um problema que deve ser evitado com a utilização de outro protocolo que está na camada de transporte: o UDP. O MQTT-SN, na verdade, foi desenhado para qualquer protocolo de camada de transporte que tenha uma comunicação bidirecional podendo ser utilizada também em sistema como *Zigbee* (TRUONG; LINH, 2013).

O MQTT-SN foi feito para ter uma certa compatibilidade com o protocolo MQTT. Com isso, o mesmo concentrador do MQTT pode ser utilizado no MQTT-SN, mas houve a criação de um novo dispositivo denominado *gateway*. Esses *gateways* podem ser também só uma camada de *software* que faz a tradução do protocolo diretamente no concentrador. Os *gateways* podem ser de dois tipo: agregados e transparentes. Este último é uma implementação mais simples onde para cada cliente conectado ao *gateway*, haverá uma sessão direta para conexão no concentrador. Já o *gateway* do tipo agregador é mais complexo de ser programado, porém cria uma sessão única com o concentrador e, todos os dados de todos os clientes conectados devem passar por esse canal (TRUONG; LINH, 2013).

Para lidar com esses problemas, o MQTT-SN implementa algumas características interessantes como a criação de um mecanismo de identificadores de tópicos, onde cada cliente pode pedir ao concentrador que gere um número para o tópico e, ao invés de utilizar o nome do tópico em UTF-8 para a troca de mensagens, é utilizado aquele identificador diminuindo o tamanho de até vinte e três caracteres para apenas dois bytes. Outra implementação é a criação de tópicos pré-definidos conhecidos tanto pelo concentrador tanto pelos clientes. A grande vantagem desse tipo de identificação é que não há a necessidade de registro, portanto dispositivos que entram na rede podem se comunicar sem gerar tráfego de rede adicional. Esse protocolo ainda implementa uma forma de diminuir os pacotes de estabelecimento de conexão permitindo o envio de *"Will messages"* logo após o pacote inicial permitindo assim que publicadores que não desejam usar esse recurso diminuir o tamanho do pacote (TRUONG; LINH, 2013).

O MQTT-SN implementa uma função de descoberta de novos dispositivos onde estes conseguem requisitar a rede informações do concentrador para começar uma comunicação. Somente nessa situação há a comunicação direta entre clientes sem passar pelo concentrador. Os concentradores também podem enviar suas informações pela rede em certos intervalos de tempo que devem ser cerca de quinze minutos para que não haja congestionamento na rede (TRUONG; LINH, 2013).

2.2.3 Constrained Application Protocol - CoAP

O CoAP é um protocolo que foi desenhado para ser compatível com o modelo REST e consequentemente com servidores Web, com uma fácil tradução de entre o protocolo

HTTP, e dispositivos que utilizam *WebServices*. Uma característica herdada do REST é o modelo de comunicação que é baseado no estilo requisição/resposta, onde os dispositivos assumem o papel de cliente e servidor a cada transição sendo o cliente aquele que está requisitando a informação e servidor aquele que está enviando a informação requisitada (MAZZER; FRIGIERI; PARREIRA, 2015). Em um modelo REST padrão, o conceito de servidor/cliente é geralmente mais rígido, pois os servidores são mais robustos e capazes de gerenciar muitas requisições, como os servidores Web, e os clientes são mais simples e estão interessados apenas na informação, como um navegador Web. No CoAP, essa distinção de capacidade não existe, portanto os dispositivos assumem o papel de cliente ou servidor a cada transmissão (BAHIA; CAMPISTA, 2016). Essa é outra característica importante de dispositivos IoT que é a capacidade de comunicação M2M. Porém, diferentemente do HTTP, o protocolo CoAP foi desenhado para ser pequeno, com um cabeçalho fixo de quatro *bytes*, e utiliza como protocolo de transporte o UDP e, portanto, foi desenvolvido primeiramente sem a opção de ser fragmentado o que fez com que os desenvolvedores criassem uma documentação anexa modelando um sistema de blocos inteligentes para dividir e ordenar o envio de grandes quantidades de informações como especificado em (BORMANN; SHELBY, 2016).

Mesmo assim, o CoAP teve que implementar sua própria forma de confiabilidade para entrega de mensagens. Por isso, foram criados quatro tipos de envio de mensagens sendo eles o CON, que obriga o dispositivo que recebe a mensagem a enviar uma resposta de reconhecimento, o ACK, que é enviado como resposta à um CON, o NON, que é uma mensagem que não necessita de resposta por parte quem recebe a informação, e o RESET que é utilizado para fechar conexões quando um erro é encontrado e não necessita de resposta. Para diminuir ainda mais o tráfego de rede, o protocolo permite o envio de mensagens em *piggyback*, ou seja, é possível o envio da informação requisitada diretamente na mensagem ACK, se esse dado for rápido de ser construído, utilizando de um mecanismo de identificação de mensagens simples chamado de *MessageID* para controle de respostas imediatas. Caso contrário, o CoAP utiliza um outro nível de identificação chamado de *Token*. Se o dado em um CON for mais difícil de ser criado, uma ACK é enviado sem informação útil e quando o dado estiver pronto, o servidor enviará uma mensagem CON com o token enviado anteriormente para que o cliente saiba que aquela mensagem contém uma informação que já havia sido requisitada (SHELBY; HARTKE; BORMANN, 2014).

Ainda sobre algumas características de ser RESTful, o CoAP realiza acesso aos recursos através de URI's. Outro ponto em comum é a adoção de quatro maneiras de requisição de informação que são também utilizados no RESTful, sendo eles o GET que é o comando que pede o envio de informações para o requisitante, o POST que envia um comando a ser executado em outro dispositivo, o PUT que envia uma informação diretamente ao destinatário atualizando um determinado dado e o DELETE que apaga o recurso do dispositivo (SHELBY; HARTKE; BORMANN, 2014).

O CoAP também permite a implementação de dispositivos como *Proxies* com funções de cache de respostas, já que o REST possui ferramentas para identificar a validade de uma mensagem, e também para a tradução dos protocolos tanto entre CoAP e HTTP quanto no caminho contrário (SHELBY; HARTKE; BORMANN, 2014).

O CoAP aproveita ainda do *multicast* disponível pelo UDP possibilitando envio de mensagem para mais de um dispositivo e, assim, dois recursos importantes para o IoT que são o modo observador e um método de *discovery*. O método observador possibilita ao CoAP o funcionamento no modelo de comunicação *publish/subscribe* que se assemelha ao MQTT. Um dispositivo que deseja saber quando um recurso muda de estado pode se inscrever em um servidor CoAP e, a cada momento que o recurso trocar de estado, ele enviará uma mensagem para todos os dispositivos que estavam interessados naquele recurso. Se um *proxy* de *cache* estiver disponível, esse recurso pode ser utilizado para deixar a resposta ainda mais rápida e pode permitir que o sensor principal entre em modo de economia de energia e, então, o *proxy* responderá por ele (HARTKE, 2015). O outro recurso comentado, chamado de *discovery*, é muito importante para comunicação M2M, pois aumenta a flexibilidade da rede de sensores ao permitir a adição de dispositivos sem uma prévia configuração de endereço de um concentrador (SHELBY; HARTKE; BORMANN, 2014).

Ao final do ano de 2017, foi publicada uma versão do CoAP utilizando como base o TCP. A motivação principal para criação desse sitema foi que, pensando nos caminhos e nas diversas redes que um pacote pode passar, alguns deles podem ter algum bloqueio de tráfego UDP o que tornaria esse protocolo mais restrito. Porém, na própria especificação é informado que de 2 a 4% das redes possuem esse tipo de bloqueio. Outra motivação foi que os dispositivos que fazem o *Network Address Translation* (NAT) acabam mantendo sessões de comunicação a partir do protocolo de transporte fazendo com que os dispositivos que utilizam o UDP como base, precisem enviar mais mensagens para manter uma sessão. Isso aumentaria o fluxo de mensagens na rede de maneira desnecessária. Novamente, a própria especificação informa que este número de mensagem existe em ambientes muito específicos, informando que o melhor, até esse momento, seria a utilização do protocolo padrão, sobre o UDP (BORMANN et al., 2018).

2.2.4 Advanced Message Queuing Protocol - AMQP

O AMQP começou a ser desenvolvido em 2013 por John O'Hara em uma empresa de investimentos bancários e serviços financeiros, a JPMorgan Chase. A versão mais atual é a 1.0 é padronizada pelo consórcio OASIS. O protocolo se fundamenta na necessidade de padronizar a comunicação entre MOM's de forma a garantir a interoperabilidade entre eles (APPEL; SACHS; BUCHMANN, 2010). MOM's são sistemas, de *hardware* ou *software*, que tem a finalidade de interligar sistemas distribuídos tanto de mesmos fabricantes

quanto de fabricantes diferentes. Assim, ele foi desenvolvido para suportar quase todos os tipos de sistemas distribuídos ou sistemas empresariais (LUZURIAGA et al., 2015) como sistemas bancários e financeiros e a comunicação com ou entre *data-centers*.

Para manter a interoperabilidade, o AMQP divide as interações em filas que consegue se adequar aos diversos modelos de comunicação como *Publisher/Subscriber* utilizado no MQTT, *Request/Response* utilizado no CoAP e P2P (APPEL; SACHS; BUCHMANN, 2010). Possui também características de segurança para os dados enviados, pois eles são opacos e imutáveis durante a transmissão. Outra importante característica é a possibilidade de envio de mensagens grandes com identificadores para os fragmentos da mensagem implementados na camada de aplicação (LUZURIAGA et al., 2015). O protocolo estabelece campos de propriedades da mensagem a ser enviada que são definidas pelo nó de origem e que podem ser modificados ou acrescidos pelos componentes intermediários.

O protocolo estabelece estruturas e como estas podem se comunicar. Abaixo elas são listadas (OASIS, 2012) e ilustradas nas Figuras 3 e 4.

- Nós: Estruturas que possuem mensagens. Podem ser as filas, clientes produtores ou consumidores de conteúdo.
- Containers: Estruturas com identificadores que agregam um ou mais nós. Aqui há a criação de *Brokers* contendo filas e alguns clientes embutidos.
- Filtro: Estruturas que contém pré-definições de quais tipos de mensagens podem ser aceitas por aquele nó.
- Links: Caminho de ligação entre o filtro de uma fonte de dados e o filtro de um destino para os dados.

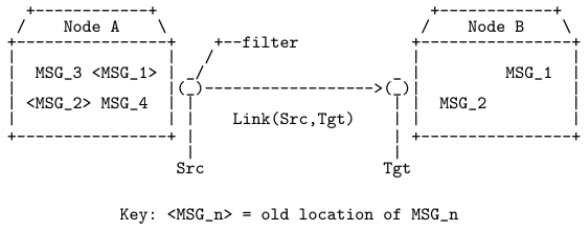


Figura 3 – Estrutura dos Nós no AMQP

Fonte – OASIS,2012, p. 31

Outro ponto de destaque do protocolo é a um sistema de acordo antes da conexão ser estabelecida mais complexo onde pode ser definidos vários aspectos que na teoria já são tratados protocolo de transporte base do AMQP que é o TCP. Nele é possível estabelecer o tamanho dos pacotes que serão enviados, além de realizar um controle de fluxo baseado em créditos.

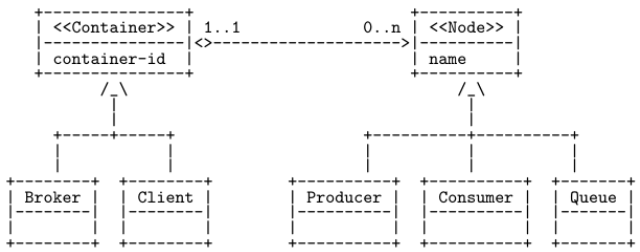


Figura 4 – Estrutura dos *Containers* no AMQP

Fonte – OASIS,2012, p. 31

O AMQP tem a vantagem de manter as mensagens no servidor e enviá-las para o consumidor assim que este estiver disponível. Assim, vários dispositivos podem utilizar funções de *sleep* e economizar energia o que é bastante aceitável em aplicações IoT. Outra vantagem é a segurança em cima das mensagens enviadas. A característica de mensagens imutáveis, citada anteriormente, garante o envio das mensagens de ponto a ponto. O ponto negativo para IoT é o tamanho do cabeçalho que pode ser grande dependendo do comportamento desejado para a aplicação. O próprio *handshake* inicial define várias funções que irão acontecer na comunicação como controle de fluxo e anotações o que pode ser um problema para dispositivos muito restritos em processamento e memória já que o próprio AMQP roda em cima do TCP necessitando de *buffers* maiores nos dispositivos.

2.2.5 Data Distribution Service - DDS

O DDS é um *middleware* homologado pela OMG (Object Management Group) que possui como propósito a entrega eficiente e robusta a uma grande quantidade de dispositivos em tempo real. Por isso, ele pode ser implementado em cima dos dois principais protocolos de transporte (TCP e UDP). Ele utiliza um modelo conhecido como DCPS (Data Centric Publish-Subscribe) estabelecendo uma "nuvem de dados"onde os dispositivos não se preocupam com questões de infraestrutura, pois a rede seria capaz de descobrir se o destino está acessível ou qual um caminho alternativo para aquele dado, além de recolher informações sobre o formato que o recebedor aceita aquele dado, podendo inclusive, ser diferente do formato enviado (OMG, 2015). Essa "nuvem"é conhecida como *Global Data Space* (GDS). Essa estrutura está exemplificada na Figura 5 (OMG, 2014). Os dispositivos possuem funções específicas como forma de utilizar dispositivos com menor capacidade de processamento e, também, para tornar a rede mais distribuída e menos susceptível a falhas totais se alguns dos nós perderem conexão. Para garantir ainda mais interoperabilidade, o DDS possui um protocolo de descoberta de dispositivos conhecido como *Simple Discovery Protocol* que executa as funções descritas acima e, além disso, é possível utilizar outros protocolos o que pode ser útil se alguma empresa já possuir um projeto desse tipo em andamento (OMG, 2015).

Outro ponto de destaque desse *middleware*, é o sistema de identificação de prioridade, nível de tolerância a falhas e controle de fluxo das mensagens que conta com vinte e três políticas diferentes tornando o sistema mais robusto (OMG, 2015).

O DDS se apresenta ao mundo IoT como uma solução que desacopla os dispositivos sem a necessidade de instalação de um concentrador e, mesmo assim, consegue ser uma solução que proporciona baixa latência e confiabilidade no envio de dados.

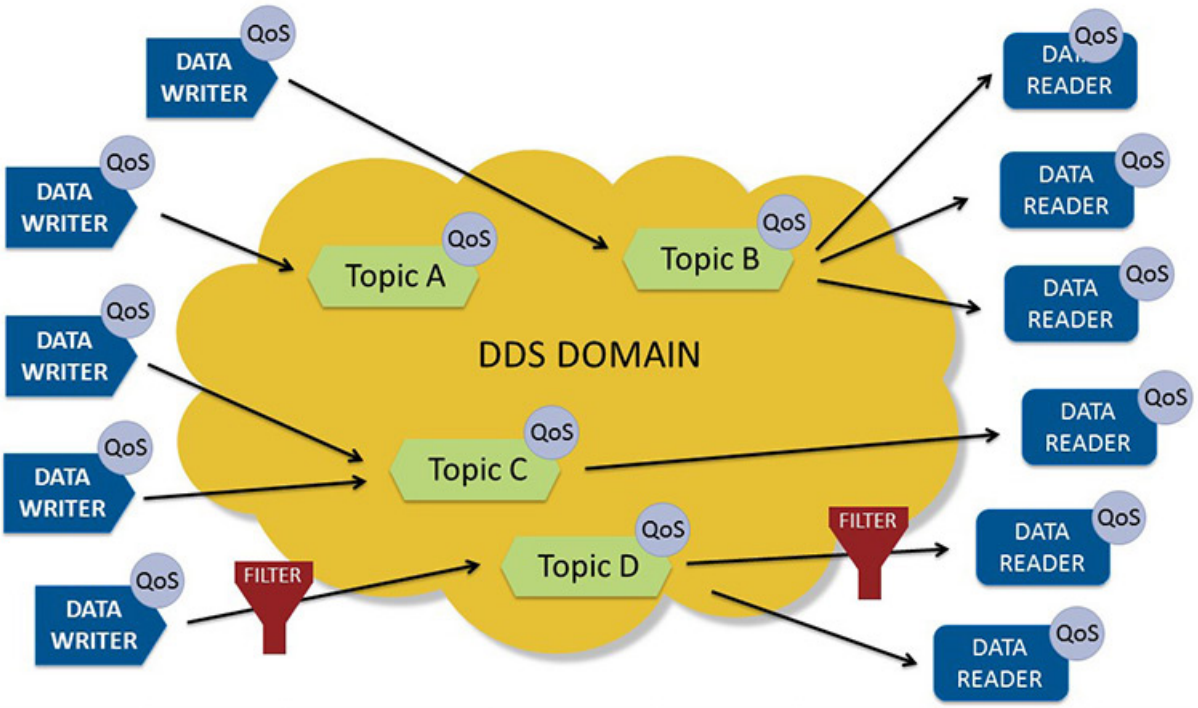


Figura 5 – Estrutura de funcionamento do DDS

Fonte – https://www.omgwiki.org/dds/what-is-dds-3/

2.3 Dispositivos ESP8266

Produzido pela empresa chinesa Espressif, o dispositivo escolhido para o trabalho vem ganhando destaque em implementações empresariais e residenciais pelo seu baixo custo, um bom número de entradas e saídas suportadas, um bom poder de processamento aliado ao baixo consumo energético com possibilidade de agendar os momentos de envio de informação para que nesse tempo ocioso o dispositivo possa dormir diminuindo ainda mais o consumo (a Tabela 1 mostra o consumo típico desses dispositivos), e pela sua integração com algumas IDE's, como a do próprio *Arduíno* (CURVELLO, 2015).

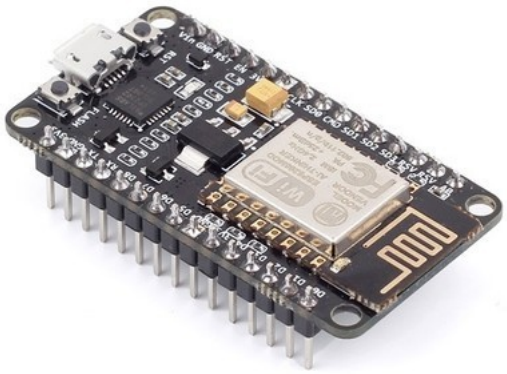


Figura 6 – ESP8266 12E NodeMCU

Fonte – https://www.vidadesilicio.com.br/nodemcu-esp8266-wifi-esp-12e

Uma grande vantagem do ESP8266 é ter WiFi embutido o que possibilita uma fácil integração em sistemas com uma infraestrutura já implementada. Assim, o dispositivo pode ser integrado com os sensores pelas entradas GPIO e se comunicar com outros dispositivos de maneira sem fio utilizando protocolos conhecidos como o TCP/UDP (CURVELLO, 2015).

2.3.1 Restrições de Hardware

O ESP8266 não possui todas as suas informações de *hardware* publicadas. Pela documentação oficial observa-se um processador *Tensilica L106 32-bit RISC* com 80 MHz de clock como padrão, podendo chegar a 160 MHz. Já para memória RAM, a documentação não é clara e apenas informa que está disponível cerca de 50 *Kbytes* que podem ser utilizados com *heap* pela aplicação (TEAM, 2018a). Alguns intusiastas desse meio conseguiram, através de engenharia reversa, mapear a memória disponível nos ESP8266, como explica (GEORGE, 2016), e como é mostrado na Figura 7. Com isso, observa-se existem dois

Tabela 1 – Consumo Energético Típico do ESP8266

Modo	**Típico**
Transmit 802.11b, CCK 1Mbps, POUT=+19.5dBm	215 mA
Transmit 802.11b, CCK 11Mbps, POUT=+18.5dBm	197 mA
Transmit 802.11g, OFDM 54Mbps, POUT =+16dBm	145 mA
Transmit 802.11n, MCS7, POUT=+14dBm	135 mA
Receive 802.11b, packet length=1024 byte, -80dBm	60 mA
Receive 802.11g, packet length=1024 byte, -70dBm	60 mA
Receive 802.11n, packet length=1024 byte, -65dBm	62 mA
Standby	0.9 mA
Deep sleep	10 uA
Power save mode DTIM 1	1.2 mA
Power save mode DTIM 3	0.86 mA
Total Shutdown	0.5 uA

Fonte – https://www.embarcados.com.br/modulo-esp8266

blocos de memória sendo uma específica para dados e outra específica para instruções o que pode se categorizar como uma arquitetura Harvard. Dos 80 *Kbytes* para o usuário mostrados na Figura 7, apenas cerca de 50Kbytes estão disponíveis sendo que os outros 30 *Kbytes* são utilizados por rotinas de funcionamento do próprio sistema, incluindo do Wifi. Nesse espaço encontra-se ainda a pilha de execução que se limita a 4 *Kbytes*, o que torna o uso de *buffers* na aplicação bem restrito. O armazenamento é feito por uma memória do tipo SPI NOR flash com suporte a até 16 *Mbytes*.

2.3.2 Restrições de Software

A Espressif disponibiliza duas formas de utilização do ESP8266 onde a primeira promete que o sistema possa ser utilizado como tempo real e outra que promete o funcionamento como um Sistema Operacional comum. Elas são denominadas como *Software Development Kits* (SDK's) RTOS e NON-OS, respectivamente e são escritas em C/C++ (TEAM, 2018b). A SDK NON-OS foi escolhida para ser integrada à IDE do *Arduíno* o que acabou disseminando o uso da placa pelo mundo, já que algumas outras funções já implementadas no *Arduíno* puderam ser utilizadas para facilitar o uso de algumas funções do ESP8266.

Pela restrição de memória RAM, mesmo com a implementação em C, ainda era necessário reduzir o consumo de memória pela pilha TCP/IP o que fez com que a Espressif optasse por utilizar uma variação conhecida como *lwIP*. Essa versão foi escrita originalmente por Adam Dunkels do Swedish Institute of Computer Science (SISC) e possibilita o acesso a internet em dispositivos com algumas dezenas de memória RAM, além de pouco ROM para armazenar seu código. Atualmente, essa pilha implementa grande parte dos principais protocolos utilizados na internet como o TCP UDP, DNS, ICMP, DHCP, entre

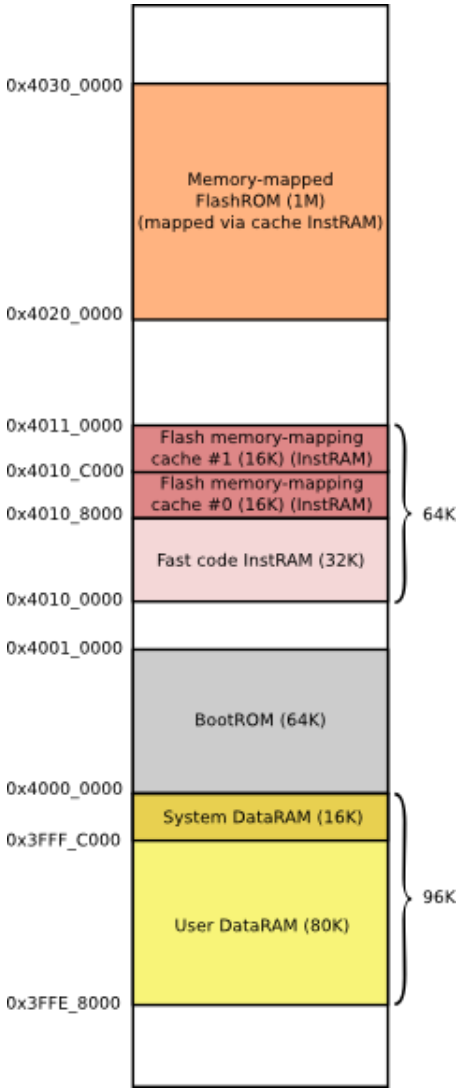


Figura 7 – Mapeamento da Memória do ESP8266

Fonte – GEORGE, 2016

outros.

Uma das implicações em utilizar o lwIP é ter como padrão o algoritmo de Nagle implementado. Esse algoritmo aumenta a eficiência na entrega de dados, pois evita que mensagens muito pequenas, ou seja, mensagem em que o cabeçalho dos protocolos é maior que o conteúdo da mensagem, sejam enviadas aumentando o tráfego de informações úteis na rede. Assim, as mensagens esperam em um *buffer* até que essa quantidade seja atingida para, então, serem enviadas (DUNKELS, 2001).

2.4 Cenários

Ao longo dos anos, a Internet das Coisas vem se tornando cada vez mais importante e está atingindo as mais diversas áreas de atuação como residências, comércios, indústrias,

hospitais e até mesmo cidades inteiras criando uma gama de possibilidades. Delimitar e categorizar onde e como os dispositivos e protocolos envolvidos estão atuando ajudam na análise mais precisa dos impactos causados por esse tipo de tecnologia.

A partir de exemplos de aplicações que estão em uso pelas grandes empresas do ramo da tecnologia tanto para venda de serviços quanto para melhoria de produtividade, é possível traçar o perfil de cada tipo de usuário e levantar quais as métricas mais indicadas para cada área de atuação. Com isso, a partir da definição dessas métricas, fica mais fácil determinar qual o melhor dispositivo a ser utilizado, além do protocolo de rede ou *middleware* que é necessário.

Por fim, com o estudo dessas aplicações implementadas por grandes empresas da área, das especificações dos protocolos e dos trabalhos acadêmicos realizados sobre cada um dos protocolos é possível criar cenários de IoT e, então, determinar qual protocolo pode ter um melhor desempenho em cada âmbito de atuação. Requisitos como número de dispositivos, quantidade de dados transferidos e tamanho máximo de cada mensagem e função ajudam na classificação desses produtos das grandes empresas e são levados em consideração na hora da análise. As seções abaixo, dividem as aplicações estudadas em tipos diferentes levando em consideração os cenários residencial, comercial e industrial.

2.4.1 Cenário Residencial

As aplicações residenciais se subdividem em duas categorias diferentes ao que diz respeito a integração entre equipamentos, segundo (BRUSH et al., 2011). São elas: aplicações feitas ou implementadas por entusiastas e aplicações desenvolvidas e/ou instaladas por empresas especializadas.

Com a facilidade de acesso a informação e aos dispositivos como ESP8266, *Arduínos*, *Raspberries*, muitos intusiastas no assunto começaram a criar seus próprios sistemas de automação realizando funções de ligar uma lâmpada pelo celular até acessar relatórios do sistema de câmeras de uma residência. O acesso a sites de tecnologia do exterior com equipamentos independentes que, às vezes, executam apenas uma função também ajudou na disseminação da automação residencial. Essas duas situações podem ser englobadas em uma mesma categoria, pois não há um controle rigoroso quanto a segurança em caso de falha de um equipamento e as soluções encontradas não são integradas aumentando o tráfego de rede (BRUSH et al., 2011).

Outra categoria é representada por empresas que oferem serviços de automação muitas vezes utilizando integração com sistemas de grandes empresas, mas podem utilizar soluções próprias. Geralmente esses sistemas são mais robustos e fazem uma automação de várias partes das residências (BRUSH et al., 2011).

Sistemas de aquecimento ou resfriamento de ambientes, monitoramento de segurança,

sistema de iluminação inteligente para economia de energia, sistemas de controle de luminosidade para ambientes, sistemas para automatização de tarefas em horários específicos do dia, entre outros são opções de aplicações em residências.

2.4.2 Cenário Comercial

As aplicações comerciais estão se destacando como verdadeiras centrais de recebimento de dados criando assim, uma camada de abstração onde as empresas, residências e indústrias são pontos que se ligam em um barramento principal administrado pelas principais companhias tecnológicas, como *Microsoft*, *Google*, *IBM* e *Intel* (DERHAMY et al., 2015). Essa estrutura é mostrada na Figura 8.

Produtos como *Azure IoT* da Microsoft, *Amazon AWS* da Amazon, *Google Cloud IoT* da Google trabalham com mais de um protocolo para gerenciar os dispositivos nas pontas da estrutura e depois para levar um grande volume de dados para os *Datacenters* para tratamento e disponibilização dos mesmos no sistema.

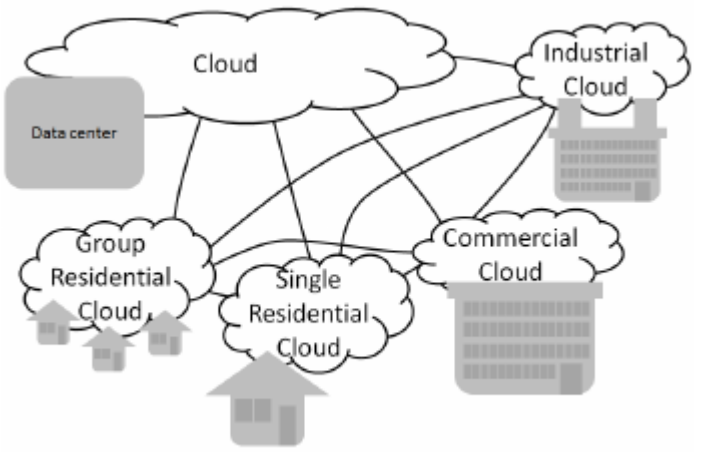


Figura 8 – Estrutura de Ação de Empresas na Área de IoT

Fonte – DERHAMY et al., 2015

2.4.3 Cenário Industrial

No âmbito industrial, observa-se uma grande quantidade de sensores que precisam comunicar entre si e com um concentrador, portanto a utilização da rede de uso comum pode ser facilmente prejudicada devido a grande quantidade de informação que é trocada por eles. Existe a necessidade de tratar e armazenar esses dados de forma a programar uma melhora nos processos, acompanhar o funcionamento de equipamentos, manutenção, transporte de produtos, entre outros.

Sistemas médicos onde o paciente está sempre conectado enviando informações, por exemplo, de frequência cardíaca, sistemas anti propagação de incêndio, controle e rastrea-

mento de cargas em tempo real, além de monitoramento dos processos desde a plantação até o consumo de alimentos são exemplos que mostram a complexidade de aplicações industriais que podem utilizar IoT (XU; HE; LI, 2014).

2.4.4 Características Importantes para cada Cenário

Com a quantidade de protocolos disponíveis e com as possibilidades alavancadas pelo aumento da tecnologia dos dispositivos com baixo poder de processamento é importante determinar qual o melhor protocolo para os cenários de automação residencial, comercial e industrial. Assim, a partir do estudo dos protocolos citados acima, eles foram analisados em um conjunto de características. A Tabela 2 resume os resultados obtidos a partir de dispositivos com baixo poder de processamento. Pôde-se estabalecer um paralelo entre o resultado e o dispositivo utilizado nesse trabalho. O DDS possui mais campos sem respostas devido a falta de fontes de informação com testes que pudessem ser comparados com os demais protocolos. Abaixo, cada característica é explicada, além de mostrar a sua importância na escolha do protocolo.

- **Restrição severa de processamento**: Essa característica refere-se a quantidade de memória e ao poder de processamento do dispositivo. A referência mínima utilizada nesse caso é o ESP8266 com os recursos descritos no Item 2.3.1. As respostas possíveis são *Sim* e *Não* que representam aplicações já realizadas nesses dispositivos como demonstrado em (MUN; DINH; KWON, 2016) e (NAIK, 2017). Com a crescente demanda para aplicação de sensores em locais específicos, o tamanho físico dos dispositivos está cada vez mais importante e isso reflete diretamente no poder de processamento. Em aplicações industriais, onde os dispositivos estão localizados em locais estratégicos dentro de equipamentos tanto para monitoramento quando para acionamento ou até mesmo em casas automatizadas que precisam de descrição em dispositivos de segurança, esse quesito torna-se relevante. Essa característica é importante nos cenários **Residencial** e **Industrial** devido a necessidade de automação de tarefas o que implica em processamento de informações. No caso industrial, o recolhimento de informações, dependendo do sensor, pode necessitar de um processamento maior. Os protocolos MQTT, CoAP e MQTT-SN, possuem cabeçalhos pequenos o que ajuda a diminuir a necessidade de **buffers** e de dados processados. O AMQP e o DDS possuem ferramentas adicionais de transmissão e gerenciamento de mensagens o que necessita maior processamento.

- **Restrição severa de energia**: Essa característica refere-se ao consumo energético dos dispositivos em modo de operação. A refêrencia, novamente, é no mínimo o ESP8266 com os valores descritos na Tabela 1. As respostas possíveis são *Sim* e *Não* que são baseados em estudos já realizados tanto diretamente em dispositivos

quanto utilizando medidores de corrente a fim de medir apenas o efeito do protocolo para geração e envio dos pacotes. Os testes realizados podem ser encontrados em (BANDYOPADHYAY; BHATTACHARYYA, 2013), (MUN; DINH; KWON, 2016) e (NAIK, 2017). As aplicações seguem o mesmo modelo exemplificado acima. Sensores em locais de difícil acesso precisam ter energia por longos períodos de tempo sem que seja necessária a troca de fontes energéticas. Sensores em locais de difícil acesso dentro de equipamentos ou em locais altos, não podem ter manutenção diária para troca de bateria, pois além de caro, outros setores de produção podem ser atralhados, por exemplo. Essa característica é importante nos cenários **Residencial** e **Industrial**, pois nesses cenários há uma infinidade de locais onde os dispositivos podem ser instalados e a manutenção de energia acaba se tornando um problema. A análise dos protocolos é diretamente ligada ao item de restrição de processamento.

- **Qualidade do sinal da rede**: Essa característica não refere-se diretamente à um dispositivo, mas está relacionado a rede em que ele utiliza para comunicação. Isso acaba influenciando os dispositivos com baixo poder de processamento, pois quando não é possível enviar um dado, este permanacerá em um *buffer* na memória do dispositivo o que pode gerar efeitos no envio de mensagens posteriores. Para esta análise, foi feito um estudo a partir de um sistema que simulava a perda de pacotes em uma rede, como mostra (BANDYOPADHYAY; BHATTACHARYYA, 2013), ou através de estudos comparando rede cabeada, rede sem fio padrão *WiFi* e redes sem fio móveis, como descrito em (CHAUDHARY; PEDDOJU; KADARLA, 2017) e (CARO et al., 2013). A escala de respostas pode ser de *Ruim* (com mais de 20% de perda), *Média* (entre 0 e 20% de perda) e *Boa* onde não há perda. Essa característica é relevante nos cenários **Residencial**, **Comercial** e **Industrial** já que os mecanismos de retransmissão e controle de erro dos protocolos podem trazer problemas para os demais dispositivos que estejam conectados à mesma rede dos sensores. Protocolos sobre o UDP, como CoAP e MQTT-SN, sofrem com essa característica, pois precisam implementar seus próprios sistemas de controle de erro o que pode diminuir a eficiência do protocolo.

- **Possibilidade de Concentrador**: Essa característica define se a estrutura de rede do local onde os dispositivos estão instalados permite ou necessita de um concentrador de dados. As repostas possíveis são *Sim* e *Não* e estão ligadas diretamente a estrutura de comunicação apresentada pelo protocolo. Aqueles que utilizam o modelo *Publisher/Subscriber* naturalmente necessitam de um concentrador. O funcionamento normal do CoAP não exige que o tenha, permitindo o acesso direto ao dado no dispositivo. As informações foram retiradas das especificações como podem ser analisadas nos Itens 2.2.1, 2.2.2, 2.2.3, 2.2.4 deste trabalho. Essa característica é importante nos cenários **Residencial** e **Industrial** já que a localização dos senso-

res pode não permitir a instalação de um componente maior com maior capacidade de processamento. Em uma rede de sensores em que as ações são tomadas pelos dispositivos através dos dados recebidos por sensores, o uso do concentrador pode se tornar dispensável.

- **Capacidade de Recursos do Concentrador**: Essa característica define se o concentrador citado acima necessita de muitos recursos para gerenciar os dados recebidos. Dentro desse tópico foi levado em consideração se o concentrador gerencia filas de pacotes, informações sobre a sessão com o dispositivo e a quantidade de informações, além do modelo de gerenciamento dos dados que está ligado mais ao MQTT-SN por possuir dois estilos de concentrador, como descrito no Item 2.2.2. As outras informações foram levantadas pela leitura das especificações, descritas nos Itens 2.2.1 e 2.2.4 e em (TORRES; ROCHA; SOUZA, 2016). Essa análise é importante, pois mostra se é necessária a aquisição de um servidor, um simples *Raspberry Pi* ou até mesmo um outro dispositivo com baixo poder de processamento é suficiente para implementar esse serviço. Geralmente, quando trata-se de envio de dados para análise, em nuvem ou localmente, faz-se necessária a aquisição de uma máquina com *hardware* mais avançado. Porém, em comunicação entre dispositivos, para leitura e acionamento automático de atuadores em uma indústria, menor o custo para o concentrador maior será a possibilidade de utilizar esse serviço. A escala de respostas possíveis trata como *Baixa* quando há a possibilidade de utilizar outro dispositivo com baixo poder de processamento, *Média* quando esse concentrador por ser um *Raspiberry* ou equipamentos similares e *Alta* quando há a necessidade de mais processamento. Essa característica é importante nos cenários **Residencial** e **Industrial**, mas principalmente no cenário Industrial. Dependendo da quantidade de sensores, o concentrador precisa tem uma grande capacidade para lidar e processar as informações que passam por ele. Alguns protocolos, como o MQTT-SN, possibilita a instalação de *Gateways* em outros dispositivos restritos, contudo é importante saber se esse dispositivo é capaz de processar todas as informações que são passadas a ele.

- **Capacidade de reconhecimento de outros dispositivos**: Essa característica define se, em uma comunicação M2M, o protocolo possui a capacidade de descobrir outros dispositivos, inclusive dispositivos concentradores. Esse tipo de resposta é importante em locais onde a troca de dispositivos é constante ou quando os dispositivos permanecem ligados por uma quantidade fixa de tempo, possuindo assim, rotas alternativas para o envio dos dados. Nessa situação, não há espaço para parar o serviço e recompilar todos os dispositivos se, por exemplo, um concentrador tem um problema. A análise foi feita em cima das especificações nos Itens 2.2.1, 2.2.2 e 2.2.4 deste trabalho e também em (VILLAVERDE et al., 2014). As respostas

possíveis são *Sim* e *Não* e representam se os protocolos possuem a característica de descoberta de outros dispositivos ativa. Essa característica é importante nos cenários **Comercial** e **Industrial**, pois nem sempre a identificação dos concentradores ou dos dispositivos pode ser fixa. Redes dinâmicas em que há uma grande quantidade de dispositivos que podem dormir ou se desconectar a qualquer momento precisam de dinamismo para encontrar os concentradores. Geralmente os protocolos que utilizam como base o UDP possuem esse serviço. Assim, os protocolos CoAP e MQTT-SN encaixam-se nessa categoria. O DDS, que também pode utilizar o TCP como base, possui esse sistema em sua camada de aplicação.

- **Envio em tempo real de dados**: Essa característica define se um protocolo é recomendado para uso em situações em que o tempo real é importante. As respostas possíveis são *Sim* e *Não* e tem como base resultados obtidos por (MUN; DINH; KWON, 2016), (NAIK, 2017), (CARO et al., 2013) e (AMARAN et al., 2015). Esse tipo de análise é importante, pois algumas aplicações **Industriais** e **Comerciais** necessitam desse tipo de resposta para resolver questões de segurança em controle contra incêndios, terremotos ou para registrar informações de tempo em que um evento efetivamente ocorreu. As respostas estão ligadas aos protocolos de transporte utilizados como base, além de resultados experimentais que mostram que as implementações dos protocolos podem torná-los burocráticos a ponto de mesmo utilizando uma base de transporte rápida, o protocolo não responde bem a baixa latência exigida. O caso clássico é o MQTT-SN que mesmo rodando pelo UDP, teve resultados ruins devido as políticas de retransmissão e/ou pela implementação em si das bibliotecas. O DDS possui níveis de QoS capazes de tornar a comunicação em tempo real com grande eficiência.

- **Comunicação com dados muito grandes**: Essa característica mostra se o protocolo está preparado para o envio de mensagens com grandes quantidades de informação. Ela pode ser implementada tanto no nível do protocolo de aplicação que desenvolve seu próprio sistema de fragmentação de pacotes ou pode ser implementada a nível da camada de transporte, onde esse controle pode ser feito ou não. As respostas possíveis podem ser *Sim* e *Não* e foram obtidas a partir da leitura das especificações dos protocolos e em (NAIK, 2017). Essa característica é importante nos cenários **Residencial**, **Comercial** e **Industrial**, pois algumas aplicações recolhem informações grandes e o envio destas pode ser afetada diretamente pela capacidade dos dispositivos restritos.

- **Confiabilidade na entrega dos dados**: Essa característica define se o protocolo consegue uma entrega confiável de dados para uma aplicação que exija tal requisito. As respostas possíveis são *Sim* e *Não* e representam estudos das especificações e de (NAIK, 2017). Essa análise é importante nos cenários **Residencial**, **Comercial** e

Industrial, pois aplicações que interagem a partir de sensores geralmente precisam de dados corretos e constantes. O não recebimento de um dado por um ou dois intervalos de tempo podem representar uma falha de segurança em uma empresa ou ter reflexos mais negativos ainda em uma indústria. Como visto, nem sempre o protocolo de transporte interfere diretamente no resultado sendo a implementação do protocolo a mesma importância. Os protocolos CoAP, MQTT-SN, AMQP e DDS possuem seus próprios sistemas de confiabilidade sobre as camadas de transporte TCP e UDP. O MQTT possui nos níveis de QoS 1 e 2, porém no nível mais comum de utilização, o QoS 0, o protocolo deixa esses controle para a camada de transporte.

- **Interoperabilidade entre fabricantes**: Essa característica define a possibilidade de interação com plataformas já disponíveis por grandes empresas. Alguns protocolos como o AMQP e o DDS tem isso como fundamento e possuem implementações compatíveis com os grandes fornecedores desse tipo de recurso, como listados no Item 2.4. Porém, outros protocolos são utilizados como parte de um sistema em um dos cenários, porém ficam restritos a ações dos protocolos de transporte que são utilizados na base limitando, assim, o controle de fluxo e a conexão entre dois dispositivos de maneira personalisada, como acontece no AMQP. As respostas possíveis são *Sim* e *Não* e foram obtidas a partir das especificações bem como por (NAIK, 2017). Essa característica é importante para os cenários **Comercial** e **Industrial**.

- **Envio constante de informações**: Essa característica define se um protocolo está apto ao envio constante de dados. As respostas possíveis são *Sim* e *Não* e foram obtidas em testes onde os dispositivos não influenciaram no desempenho dos protocolos, como pode ser visto em (CHAUDHARY; PEDDOJU; KADARLA, 2017) e (CARO et al., 2013). O envio constante está ligado ao envio de mensagens em um curto período de tempo. Os resultados foram feitos a partir do número de dados enviados em um intervalo de um segundo. Esse tipo de medida é interessante em cenários onde o registro de informações é importante, como em produtos de segurança em empresas, serviços de vigilância em residências e comércios, recolher informações de sensores em equipamentos industriais, entre outros. Por isso, essa analise é importante para os cenários **Residencial**, **Comercial** e **Industrial**.

Tabela 2 – Comparação Entre os Protocolos Estudados

Características	**MQTT**	**MQTT-SN**	**CoAP**	**AMQP**	**DDS**
Restrição severa de processamento	Sim	Sim	Sim	Não	–
Restrição severa de energia	Não	Sim	Sim	Não	–
Qualidade do sinal da rede	Boa	Ruim	Ruim	Boa	Ruim
Possibilidade de concentrador	Sim	Sim	Não	Sim	Não
Capacidade de recursos do concentrador	Média	Média/Baixa	–	Alta/Média	–
Capacidade de reconhecimento de outros dispositivos	Não	Sim	Sim	Não	Sim
Envio em tempo real de dados	Não	Não	Sim	Não	Sim
Comunicação com dados muito grandes	Sim	Não	Não	Sim	Sim
Confiabilidade na entrega dos dados	Sim	Sim	Não	Sim	Sim
Interoperabilidade entre fabricantes	Não	Não	Não	Sim	Sim
Envio constante de informações	Sim	Sim	Sim	Sim	Sim

Fonte – Produzida pelo autor

3 Trabalhos Relacionados

Dois artigos foram escolhidos como base para análise dos resultados obtidos nesse trabalho. Eles tem em comum a comparação entre os protocolos que foram analisados nesse trabalho o que torna possível a comprovação dos resultados em cenários parecidos.

O primeiro artigo tem como motivação ajudar os desenvolvedores na escolha do protocolo de mensagens em ambientes restritos com a comparação de cinco protocolos (MQTT, CoAP, MQTT-SN, WEBSOCKET, TCP). Medindo desempenho, eficiência energética e uso de recursos, o objetivo é entender a execução dos protocolos sobre diferentes condições de redes e de tamanho de pacotes, ajudar os programadores a escolher o melhor protocolo para IoT e inferir oportunidades de melhorar algumas questões nos protocolos (MUN; DINH; KWON, 2016). Foram utilizadas três métricas:

- Desempenho: tempo total de transmissão que leva para enviar mensagens e receber o ACK.
- Eficiência energética: consumo total de energia para o dado tempo de execução.
- Uso de recursos: o uso médio do heap Java medidos por 500 ms e o tempo total de CPU (MUN; DINH; KWON, 2016).

A estrutura utilizada para os testes consistia em um *Raspberry Pi model B*, que foi utilizado como cliente e três servidores: um local, um em outro lugar no mesmo país e outro em Tóquio, Japão. Os testes realizados foram feitos a partir do envio de mensagens com diferentes tamanhos e o tempo foi medido entre o envio da mensagem e o recebimento de uma mensagem de confirmação por parte do destino. O teste foi repetido 20 vezes para cara tamanho de pacote (MUN; DINH; KWON, 2016).

Como resultados apresentados, quanto ao desempenho, os melhores em todas as situações foram o TCP e o WebSocket. Coap e MQTT foram melhorando o tempo com o aumento do tamanho da mensagem com vantagem para o MQTT (possivelmente por trabalhar com TCP), porém para tamanhos pequenos de mensagem o MQTT foi o pior em todos os cenários. O MQTT-SN foi o pior quando o tamanho da mensagem era grande (possivelmente UDP). Para pacotes menores que 1024 *bytes*, o CoAP sempre foi melhor que o MQTT (possivelmente, este é o maior valor antes da fragmentação). Em relação à eficiência energética, os resultados das medidas energéticas de cada protocolo foram os mesmos do desempenho. Já em relação ao uso de recursos, o MQTT-SN utiliza o maior uso de processamento principalmente com o aumento do pacote. O CoAP teve o pior desempenho em memória utilizada quando o envio de dados foi para Tóquio. Teve

uma piora nos outros servidores sempre com pacotes maiores que 1024 *bytes* indicando que no CoAP a retransmissão de pacotes influencia diretamente no consumo de memória. Nos outros cenários, ele se comparou com o MQTT-SN (ambos utilizando UDP) (MUN; DINH; KWON, 2016).

Como conclusão, para que o CoAP seja utilizado em dispositivos restritos, o tamanho do pacote deve ser menor que 1 KB. O MQTT, para o mesmo fim, deve ter até 1,5 KB. O MQTT-SN não está preparado para a aplicação em sistemas externos (MUN; DINH; KWON, 2016).

O segundo trabalho relacionado teve como motivação realizar um estudo direcionado para entender qual o melhor protocolo (MQTT,CoAP e AMQP) para o envio constante de dados em redes cabeadas, wifi e 2/3/4G. Foram definidas algumas propriedades de dispositivos IoT, tais como: 1. o protocolo deve caber em pouca memória; 2. o protocolo não deve ser difícil de montar, ou seja, deve utilizar pouco processamento para criação e envio das mensagens, 3. o protocolo deve ter interoperabilidade (CHAUDHARY; PEDDOJU; KADARLA, 2017). As métricas utilizadas foram diferentes do trabalho anterior sendo elas:

- *Overhead* do pacote: número de pacotes gerados pelos protocolos para entrega de um pacote.
- Vazão das mensagens: número de mensagens enviadas por unidade de tempo.
- Uso da largura de banda.

A estrutura utilizada consistia em um servidor de alto desempenho sendo utilizado como broker do MQTT e do AMQP e como servidor do CoAP. Um *Raspberry* Pi-3 é utilizado como cliente MQTT,AMQP e CoAP. Um notebook HP utilizado como cliente dos protocolos e está ligado via cabo/wireless com o servidor Z820 e *wireless* com o *Raspberry*. O *Raspberry* também está conectado a um *broker* externo via conexão celular. Os dados são recolhidos do *Raspberry* e do notebook HP para comparação dos valores utilizando o *Wireshark*. A Figura 9 mostra como ficou a estrutura descrita anteriormente. Os testes realizados faziam o envio sucessivo, com intervalos pequenos entre envios, de mensagens com tamanhos diferentes com base 10 (exemplo, 10,100,1000,100000), simulando um uso mais robusto dos protocolos (CHAUDHARY; PEDDOJU; KADARLA, 2017).

Como resultados apresentados, observou-se que para o *overhead* de pacotes quanto mais instável a rede (a rede celular) e quanto maior o nível de QoS aplicado, maior é o overhead do protocolo MQTT devido às retransmissões. Para o envio de mensagens pequenas, o *overhead* dos protocolos é insignificante. Para a vazão de mensagens quanto maior a mensagem, menor a vazão. O MQTT possui grande vazão em redes instáveis quando utilizado no QoS nível 0. E para a largura de banda o CoAP utiliza pouca

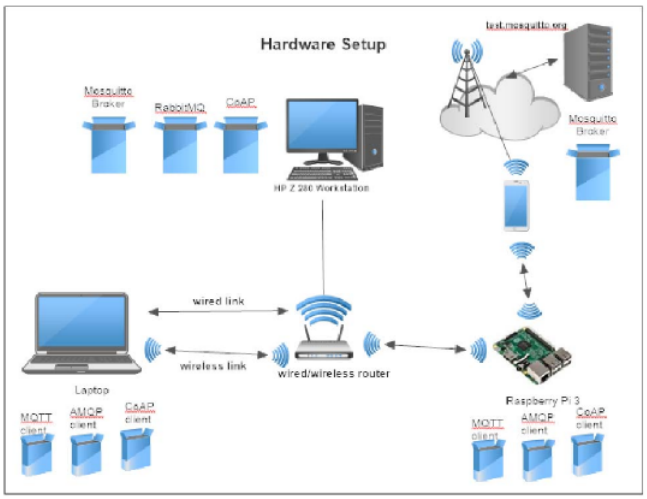


Figura 9 – Estrutura Apresentada no Segundo Artigo Estudado

Fonte – CHAUDHARY; PEDDOJU; KADARLA, 2017

largura de banda mesmo enviando mensagens maiores. Ele se manteve constante nos testes. O CoAP e o AMQP utilizaram mais largura da banda quando em redes instáveis (CHAUDHARY; PEDDOJU; KADARLA, 2017).

Concluiu-se então que em aplicações que desejam alta taxa de entrega de mensagens, o MQTT QoS1,2 faz um uso excessivo da largura de banda por causa da retransmissão de pacotes, mas pelo estilo pub/sub e pela grande adoção dos últimos anos, faz seu uma ótima opção. O CoAP consome muito pouco recurso dos dispositivos e trabalha de maneira estável em redes com alta taxa de perdas, porém o estilo client/server pode atrapalhar na utilização do mesmo. Já o AMQP, não está preparado para aplicações com componentes restritos pela dificuldade em encontrar bibliotecas para tais equipamentos (ESP8266 e *Arduino*) e pela grande memória utilizada pelo dispositivo (CHAUDHARY; PEDDOJU; KADARLA, 2017).

4 Materiais e Métodos

4.1 Materiais

Para realizar os experimentos, foi necessário utilizar um computador com alta capacidade de processamento para receber e analisar as mensagens enviadas pelos dispostivos com poucos recursos. Para representar esses dispositivos com baixo poder de processamento foi escolhido o ESP8266 pela grande aplicabilidade adquirida por já implementar uma pilha TCP/IP, por ter algumas das bibliotecas estudadas implementadas, pela facilidade de programação e pelo baixo custo. Essas características também ajudaram no aumento da utilização desses dispositivos na comunidade. Um roteador comum foi utilizado para normalizar os testes com e sem tráfego. Além disso, outro computador com alta capacidade foi utilizado para validação de alguns testes. Segue, abaixo, a lista dos materiais utilizados nos experimentos:

- Um dispositivo ESP8266 12E NodeMCU;
- Um roteador D-Link 615 Dir;
- Um computador HP Pavilion p7-1060Br com processador Intel Core i5 2300 2.8 GHz, 6 *GBytes* de memória RAM e conexão *Wireless* e a cabo;
- Um notebook Samsung NP370E4K-KD3BR com processador Intel Core i3 2.0 GHz, 4 *GBytes* de memória RAM e conexão *Wireless* e a cabo.

4.2 Metodologia

Com a finalidade de analisar os protocolos em dispositivos com baixo poder de processamento, os experimentos foram realizados com um desses dipositivos gerando e enviando mensagens para um computador que capturava as informações para posterior análise. O objetivo é analisar dois dos protocolos estudados nesse trabalho, o CoAP e o MQTT, em um cenário residencial utilizando como parâmetros algumas das características que representam esse cenário. Os parâmetros escolhidos foram o tamanho da mensagem representando a característica de envio de grandes quantidades de dados e o monitoramento em tempo quase real que representa a característica de envio constante de informações. As características estão detalhadas na Tabela 2.

Uma forma de comparar os resultados foi utilizando uma rede com e sem tráfego de fundo com a finalidade de extrair o comportamento dos protocolos com as métricas

de número de mensagens recebidas em um intervalo pequeno de tempo e o número de retransmissões dos pacotes. Esses testes representam outra característica citada na Tabela 2 que é a qualidade da rede, pois o tráfego de fundo impacta em como o roteador irá gerenciar os pacotes o que pode ocasionar perdas. Nos testes com tráfego de fundo, as aplicações em *backgroud* que estavam em execução eram aplicativos como *YouTube*, *Netflix*, *Dropbox*, navegação Web, ou seja, a rede não estava sendo estressada propositalmente. Nas aplicações sem tráfego de fundo, houve um pequeno tráfego, não constante, de aplicações que estavam instaladas no computador tentando requisitar informações dos servidores externos. O cliente NTP e o *Dropbox* são exemplos dessas aplicações. Abaixo, os passos da metodologia aplicada neste trabalho são explicados detalhadamente.

4.2.1 Topologia Aplicada

O ESP8266 estava a aproximadamente dois metros do roteador sem paredes ou partes de madeira entre eles. Esse roteador, por sua vez, estava ligado em um modem disponibilizado pela operadora de internet. Para testes com tráfego de fundo, o modem era colocado em modo *Bridge*, ou seja, todo o tráfego era gerenciado pelo roteador D-Link e os demais aparelhos da residência se conectavam a ele. Para testes sem tráfego de fundo, o roteador era desconectado do modem e apenas o computador e o ESP8266 permaneciam ligados à ele. A topologia pode ser vista na Figura 10.

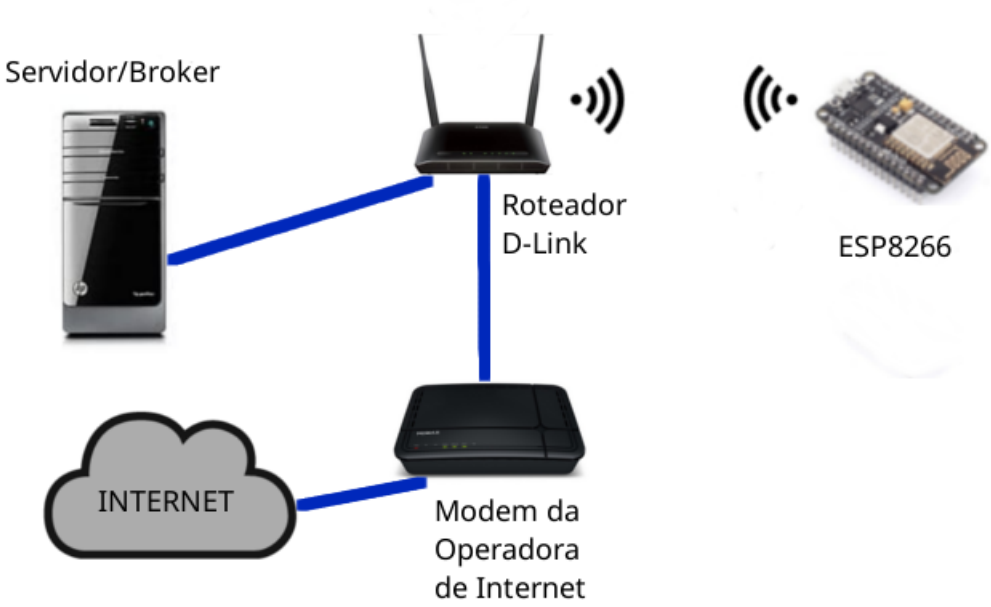


Figura 10 – Topologia Utilizada nos Testes

Fonte – Produzida pelo autor

4.2.2 Configurações de Software Utilizadas

Segue abaixo, as versões dos *softwares* que foram utilizados para os testes.

- Sistema Operacional KDE Neon 5.13 x64 baseado no Ubuntu 16.04.1 com kernel 4.15.0-38-generic;
- Arduíno IDE versão 1.8.6. Foi utilizada para programar o ESP8266 e enviar os dados para o dispositivo;
- Wireshark 2.6.2 packaged as 2.6.3-1 ubuntu16.04.1. Foi utilizado para captura dos pacotes enviados entre o computador e o ESP8266. Foi feita a filtragem por protocolo a fim de retirar as informações essenciais para análise;
- Python 3.5.2. Foram desenvolvidos *scripts* para análise dos dados gerados pelo *Wireshark*. Os gráficos foram feitos utilizando bibliotecas em Python;
- Zsh 5.5.1. Foi a *shell* utilizada para executar os *scripts* para o início de captura dos testes;
- Mosquitto 1.4.8. Disponibiliza um *broker* MQTT para Linux;
- Libcoap 1.0. Disponibiliza um cliente e um servidor para o protocolo CoAP.

4.2.3 Bibliotecas Utilizadas

Não foi possível testar todos os protocolos comentados na fundamentação teórica. Os únicos que tinham uma versão para a IDE do *Arduíno* que eram compatíveis com o ESP8266 foram o MQTT e o CoAP. Mesmo assim, houve certa dificuldade na hora de executar os testes devido a limitações impostas nos códigos das bibliotecas para manter certa estabilidade no dispositivo, devido a quantidade restrita de memória.

A biblioteca escolhida para o MQTT foi a ArduinoMqtt (KOVALENKO, 2018). Ela é escrita em C/C++ e durante o período de testes, ela possuiu uma maior estabilidade para envio em curtos períodos de tempo em relação a outra biblioteca que também é bastante conhecida para os ESP8266, que é a PubSubClient (O'LEARY, 2018). A biblioteca ArduinoMQTT foi configurada para permitir que o fosse enviado 1000 *bytes*, onde o padrão era 128 *bytes*. No servidor, foi utilizado o *software* Mosquitto como o *Broker*. A aplicação funcionou com as configurações padrões.

A biblioteca CoAP escolhida foi a ESP-CoAP (NAGESH, 2018). Ela é escrita em C e foi modificada para enviar mais de 256 *bytes*. Após a alteração, ela pode enviar mais de 1000 *bytes* de conteúdo na mensagem. Ela possui quase todas as funções REST que o CoAP dispõe tanto para servidor quanto para o cliente. No servidor foi utilizado o libcoap como cliente para enviar as mensagens ao ESP8266. O código do programa foi alterado

para que gerasse novos identificadores nas mensagens para facilitar o análise dos pacotes capturados. Após essa alteração, o programa foi recompilado e nenhum parâmetro foi alterado na execução do cliente. Foi utilizado o protocolo NTP (NTP..., 2018) que foi importante para os testes com intervalos de tempo menores do que 1 segundo.

4.2.4 Métricas

As métricas utilizadas nesse trabalho foram baseadas nos trabalhos relacionados descritos no Item 3. Foram utilizadas quatro métricas:

Número de mensagens recebidas: Quantidade de mensagens recebidas do ESP8266 por segundo. Foi utilizada nos protocolos MQTT e CoAP. Ela permite analisar se as mensagens geradas pelo ESP estão chegando ao destino e em qual frequência isto está acontecendo. Quanto mais próximo de uma ou duas linhas, melhor é o resultado.

Tempo médio entre recebimento de pacotes: Mostra a média do tempo entre captura dos pacotes no *Broker* e foi utilizada para o MQTT. Ela permite analisar um tempo geração mais o tempo de latência. Abaixo, a equação utilizada:

$$T_{dm} = \frac{\sum_{i=0}^{N-1} t_{i+1} + t_i}{N_{PUB}}$$

Onde, t_i é o tempo de recebimento da mensagem i, t_{i+1} é o tempo da próxima mensagem e N_{PUB} é o número de mensagens publicadas no experimento.

Tempo médio de transmissão: Trata do tempo médio entre o envio e o recebimento da mensagem no protocolo CoAP, ou seja, o *Round Time Trip* (RTT). Ela permite analisar o tempo de processamento do mensagem pelo ESP mais o tempo de transporte do pacote. Quanto mais estável e menor esse valor, melhor será o resultado. A fórmula utilizada foi a seguinte:

$$T_m = \frac{\sum_{i=0}^{N-1} \frac{t_{ACK} - t_{GET}}{2}}{N_{ACK}}$$

Onde, t_{ACK} é o tempo em que a mensagem de confirmação de recebimento foi capturada, t_{GET} é o tempo em que a mensagem de requisição foi enviada (elas possuem a resposta enviadas como *piggyback*) e N_{ACK} é o número de mensagens de recebimento de confirmação durante o experimento.

Número de retransmissão dos pacotes: número de pacotes que foram retransmitidos até que a mensagem fosse recebida pelo computador. Ela mostra como o roteador conseguiu gerenciar os pacotes enviados e o impacto do dos mecanismos de

retransmissão dos protocolos estudados. Esses dados foram retirados da captura dos experimentos.

4.2.5 Cenários de Experimentos

Para o protocolo MQTT, foi utilizado o QoS nível 0 com o dispositivo sendo o cliente responsável por gerar e publicar as mensagens para o *Broker*, que era o computador principal onde foi feita a captura dos pacotes.

Não foi utilizado o nível 1 do QoS para o MQTT, pois a estrutura do protocolo para esse nível, não possibilita verificar o poder de processamento dos dispositivos para geração das mensagens assim como é feito pelo CoAP, tornando a comparação injusta. As bibliotecas disponíveis para o ESP8266 também são limitadas quanto a implementação desse nível devido aos recursos limitados dos dispositivos.

Para o protocolo CoAP, o dispositivo ESP8266 foi utilizado como servidor, pois teria o mesmo efeito de geração da mensagem original e, com isso, as informações do tempo obtidas na captura incluiriam os tempos de transmissão e também o tempo de processamento da mensagem. *Wireshark*.

Foram realizados dois testes que possibilitaram observar as métricas estudadas nesse trabalho. Cada um deles foi feito com e sem tráfego de fundo na rede. Os testes foram os seguintes:

Monitoramento em tempo (quase) real: Para cada protocolo foram enviados pacotes de 30 *bytes* para intervalo por meio de envio de mensagens entre 100 e 900 milissegundos. Os testes duraram cerca de 10 minutos. O conteúdo das mensagens era composto por um identificador do dispositivo, um identificador da mensagem e por um *timestamp* que foi acrescentado no momento de geração da mensagem. O restante foi preenchido com letras 'a'. Esses dados foram capturados para posterior análise. Este teste simula o uso de sensores que enviam dados de maneira constante. Sistemas de monitoramento em tempo real podem utilizar esse tipo de teste.

Variação no tamanho da mensagem: para intervalos entre envio de mensagens de 1, 5, 10, 30 e 60 segundos com pacotes de tamanho 10, 100, 500 e 1000 *bytes* durante 1 hora. O conteúdo das mensagens era composto por um identificador do dispositivo, um identificador para a mensagem e o restante era preenchido com letras 'a'. Esse teste busca simular um uso mais comum dos dispositivos em aplicações residenciais que reportam dados de maneira esporádica ou que não há necessidade de informação em tempo real.

Cada teste citado acima foi realizado duas vezes de maneira sequencial. Como os resultados foram parecidos apenas um deles foram utilizados para análise. Vale ressaltar

que os testes foram executados durante o dia e a noite o que pode influenciar em alguns dos resultados, já que o fluxo de dados não era constante. Essa variável é importante, pois os sistemas reais possuem característica semelhante. Assim, os gráficos gerados na próxima seção são *snapshots* de um período de testes, ou seja, não é uma média dos valores encontrados.

Um teste extra para provar uma possível limitação na rede foi realizado entre o computador principal e o notebook, listados no Item 4.1, este foi colocado no lugar do ESP8266, utilizando também a conexão sem fio, à mesma distância que os dispositivos com baixo poder de processamento.

Os testes foram executados a partir de *scripts* em *bash script* onde a captura utilizando o *Wireshark* era iniciada. Os comandos para cada protocolo eram executados e após o término do tempo estabelecido, dois arquivos eram gerados: um com o filtro do protocolo especificado no formato *JSON* e outro original com todas as informações capturadas.

Com o arquivo *JSON* gerado na etapa anterior, foi criado um *script* em *Python* para filtrar as informações relevantes com intuito de mapear as métricas listadas abaixo.

5 Resultados e Análises

A análise dos testes é descrita abaixo por protocolo estudado e, no final, há uma comparação entre os resultados obtidos pelos dois. Os protocolos foram avaliados pelos testes de monitoramento em tempo (quase) real e de variação do tamanho das mensagens descritos no Item 4.2.5.

5.1 CoAP

5.1.1 Teste de monitoramento em tempo (quase) real

Ao analisar a sequência numérica inserida no pacote na hora da geração do mesmo, observa-se que ela se mantém em ordem crescente, ou seja, o ESP8266 consegue enviar as mensagens no intervalo especificado. Ao comparar os gráficos da Figura 11, observa-se que sem tráfego na rede as mensagens são recebidas quase que em sua totalidade no tempo esperado mostrando, novamente, que o ESP8266 consegue enviar dados naqueles pequenos intervalos de tempo. A Tabela 3 mostra a relação entre o número de mensagens enviadas pelo protocolo e o número de mensagens capturadas durante o teste. Ela mostra que houve uma variação durante o período de testes nos testes com tráfego de fundo.

O gráfico obtido é um reflexo da forma como o roteador trata os pacotes. Como o CoAP trabalha sobre o UDP e, portanto, não exige confiabilidade na entrega, os pacotes esperam no *buffer* e/ou podem ser descartados forçando o cliente a enviar uma nova requisição já que o *timeout* (tempo limite máximo que um cliente espera para receber uma resposta do servidor) definido pelo protocolo foi atingido e o requisitante não recebeu *feedback* do servidor. Os pontos no zero em sequência, mostrados na Figura 11, são causados pelo mecanismo de retransmissão utilizado no protocolo. A implementação do Libcoap, que é utilizado como cliente nos testes, tem a característica de esperar dois a três segundos antes de enviar um novo pacote mesmo com a especificação informando que o primeiro intervalo de tempo seria menor que um segundo e depois cresceria exponencialmente.

Uma comparação entre dois dispositivos com alto poder de processamento e entre um ESP8266 e um computador pode ser vista na Figura 12. Observa-se que, mesmo no primeiro cenário, há um problema na entrega das mensagens quando o roteador precisa gerenciar o tráfego de fundo, principalmente quando o intervalo entre envio de mensagens é de cem milissegundos. Outra diferença, é que o ESP8266 não cria novas *threads* para gerenciar novos requisições, logo, se um dos pedidos tiver um atraso na mesma ordem de grandeza do intervalo de envio de mensagens, o ESP pode estar processando uma requisição e aquele não será atendido gerando um maior número de pacotes retransmitidos

como mostrado na Figura 12 (a).

A partir de setecentos milissegundos, o roteador consegue gerenciar melhor o fluxo de dados e, durante o experimento, não foi observado retrasmissões em grandes quantidades.

Assim, para os testes de monitoramento em tempo real, o CoAP mostrou que quanto menor o intervalo de geração de mensagens maior será a chance do pacote não chegar ao destino no tempo esperado quando há tráfego de fundo, como mostra a Figura 11.

Tabela 3 – Porcentagem de mensagens do CoAP em relação ao total capturado com e sem tráfego de fundo no teste de monitoramento em tempo (quase) real

Intervalo entre envios (ms)	**Com tráfego**	**Sem tráfego**
100	13 %	49,3 %
200	7,2 %	34,3 %
300	5 %	26,3 %
400	22,4 %	21,3 %
500	43,4 %	17,9 %
600	10,9 %	18 %
700	1 %	21,4 %
800	4,3 %	23,8 %
900	3,7 %	20,1 %

Fonte – Produzida pelo autor

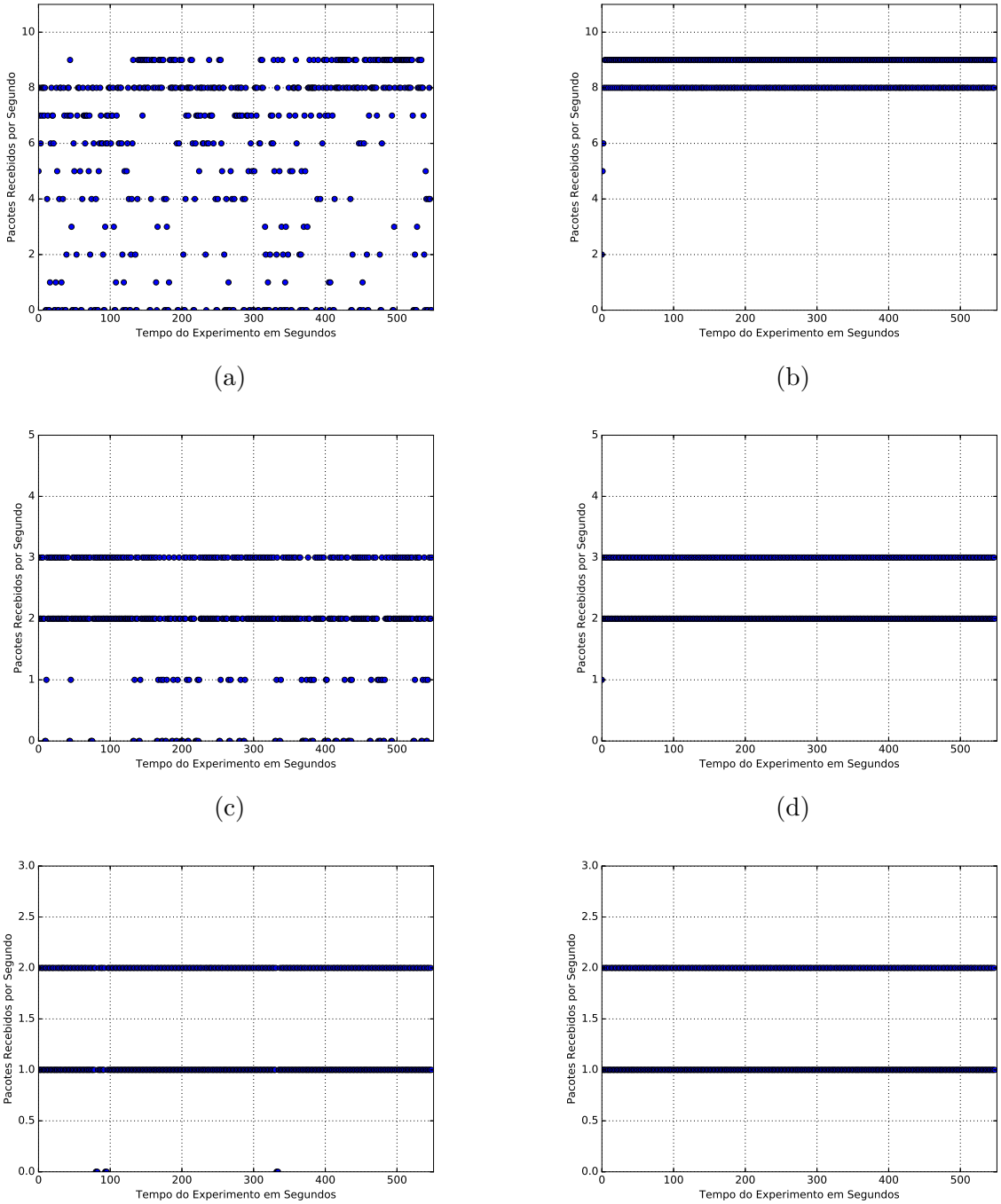
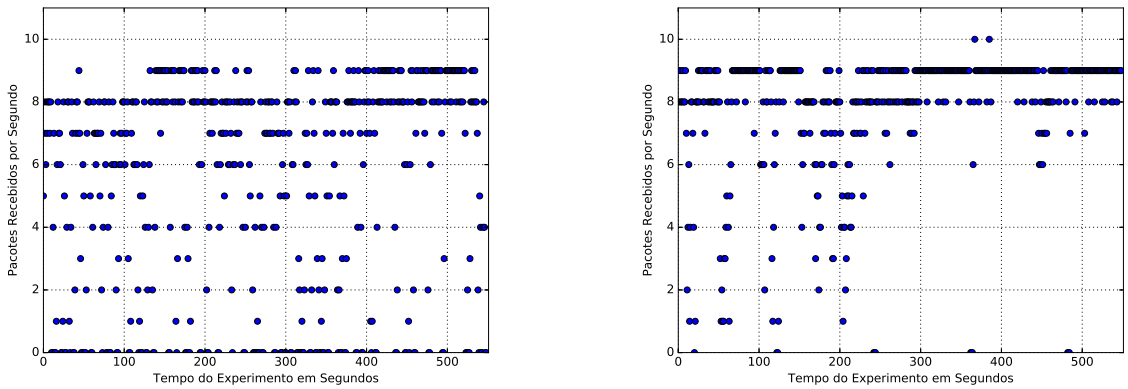
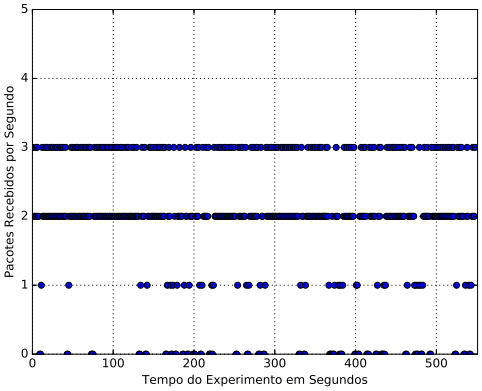


Figura 11 – Comparação do CoAP com e sem Tráfego com intervalos menores que um segundo: (a) com tráfego de fundo e intervalo de cem milissegundos (b) sem tráfego de fundo e intervalo de cem milissegundos (c) com tráfego de fundo e intervalo de quatrocentos milissegundos (d) sem tráfego de fundo e intervalo de quatrocentos milissegundos (e) com tráfego de fundo e intervalo de setecentos milissegundos (f) sem tráfego de fundo e intervalo de setecentos milissegundos

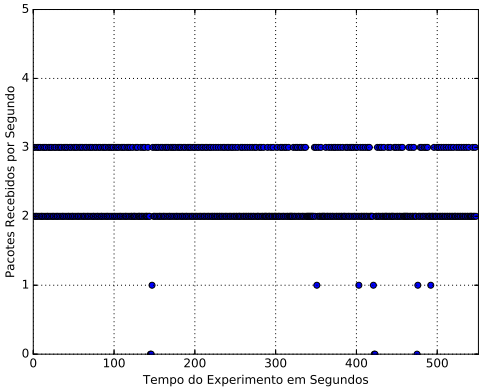


(a)

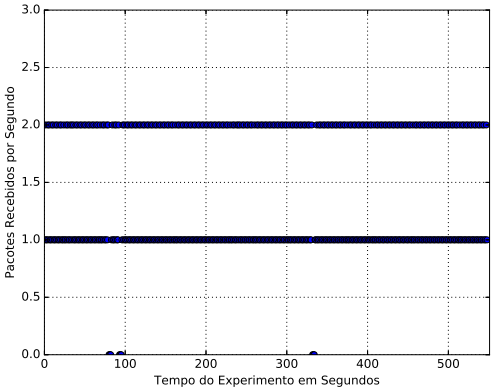
(b)



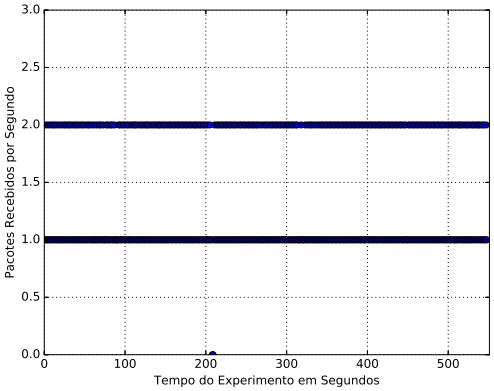
(c)



(d)



(e)



(f)

Figura 12 – Comparação do CoAP com Tráfego entre ESP8266 e um computador e entre dois computadores: (a) entre o ESP8266 e um computador com intervalo de cem milissegundos (b) entre o notebook e um computador com intervalo de cem milissegundos (c) entre o ESP8266 e um computador com intervalo de quatrocentos milissegundos (d) entre o notebook e um computador com intervalo de quatrocentos milissegundos (e) entre o ESP8266 e um computador com intervalo de setecentos milissegundos (f) entre o notebook e um computador com intervalo de setecentos milissegundos

5.1.2 Teste de variação no tamanho da mensagem

Ao alterar o tamanho das mensagens, o tempo de transmissão aumentou de acordo com a variação do *payload*. Essa é uma característica normal para um protocolo de rede ainda mais quando a camada de transporte não tem suporte a fragmentação para dividir melhor os dados. Com isso, o tamanho máximo dos pacotes é o MTU da rede. A Figura 14 mostra esse resultado. Observa-se que sem tráfego, o tempo de transmissão é o mesmo para todos os testes. Porém, quando há tráfego de fundo observou-se distorções como mensagens maiores com tempo de transmissão menor que mensagens com menos conteúdo. A Tabela 4 mostra a relação entre o total de mensagens enviadas pelo CoAP e o total de mensagens capturadas durante o experimento. Nota-se que o teste com tráfego para o *payload* de 10 *bytes* havia um tráfego de fundo maior em relação ao restante dos testes o que pode ter influenciado no resultado.

A partir desses testes, foi possível observar como o protocolo reage à uma rede comum com suas adversidades, já que em alguns horários existe um fluxo grande facilitando a perda do pacote e os mecanismos de retransmissão devem entrar em ação, como ilustrado na Figura 15. O CoAP se mostrou vulnerável ao tráfego de fundo sendo que situações de retransmissão de pacotes, como mostradas na Figura 13, aconteceram mais de uma vez o que aumentou consideravelmente o tempo da média. Já em outras situações, isso não aconteceu distorcendo a linearidade obtida no teste com menos tempo de captura. Outro fator importante é que retransmissões consecutivas são piores que as esporádicas devido ao tempo de restransmissão exponencial. Portanto, se uma mensagem foi restransmitida 3 vezes para chegar ao destino o resultado será pior que se 3 mensagens forem perdidas em intervalos de tempo diferentes.

```
8537 1885.561216476   192.168.0.25   192.168.0.23   CoAP    50 CON, MID:9871, GET, /500
8547 1888.126912081   192.168.0.25   192.168.0.23   CoAP    50 CON, MID:9871, GET, /500
8548 1888.149514666   192.168.0.23   192.168.0.25   CoAP   549 ACK, MID:9871, 2.05 Content (text/plain)
8549 1889.152450031   192.168.0.25   192.168.0.23   CoAP    50 CON, MID:3586, GET, /500
8554 1891.154552249   192.168.0.25   192.168.0.23   CoAP    50 CON, MID:3586, GET, /500
8562 1895.158698240   192.168.0.25   192.168.0.23   CoAP    50 CON, MID:3586, GET, /500
8563 1895.208258666   192.168.0.23   192.168.0.25   CoAP   549 ACK, MID:3586, 2.05 Content (text/plain)
```

Figura 13 – Retransmissão de mensagens CoAP

Fonte – Produzida pelo autor.

Tabela 4 – Porcentagem de mensagens do CoAP em relação ao total capturado com e sem tráfego de fundo no teste de variação do tamanho da mensagem para um segundo

Tamanho do payload (bytes)	**Com tráfego**	**Sem tráfego**
10	29,5 %	42,2 %
100	39,9 %	50,4 %
500	41,6 %	50,3 %
1000	46 %	49,6 %

Fonte – Produzida pelo autor

5.2 MQTT

5.2.1 Teste de monitoramento em tempo (quase) real

Para o protocolo MQTT quanto menor o intervalo entre envios de mensagens, maior a chance do pacote enviado pelo dispositivo não chegar ao *Broker*, como mostra a Figura 17. A Tabela 5 mostra a relação entre o número de mensagens enviadas pelo protocolo e o número de mensagens capturadas durante o experimento. Como pode ser visto, o experimento com tráfego de fundo tem uma variação nessa realção, entretanto os valores sem tráfego continuam maiores. Mesmo utilizando o TCP na camada de transporte, o que garante confiabilidade na entrega do pacote, observa-se que não há recebimento de mensagens em alguns instantes durante a captura de pacotes. A explicação refere-se ao algoritmo de Nagle que é implementado de maneira padrão no *lwIP*. Esse algoritmo pode parar, por um período de tempo, o envio de mensagens quando uma destas não recebe um ACK. Assim, todas as outras mensagens que poderiam ser entregues permanecem no *buffer* até que ele receba um ACK ou até que o tamanho combinado no *handshack* do TCP seja atingido. Nos experimentos, esse algoritmo pode ser percebido no *Wireshark*, por que toda vez que acontecia essa situação era exibida uma mensagem de erro *"TCP Spurious Retransmission"* seguida de um pacote com mais de uma mensagem MQTT, como mostra a Figura 16. Uma comparação entre a resposta com e sem o algoritmo de Nagle estar habilitado pode ser vista na Figura 18. Nota-se que sem o Nagle, o comportamento se assemelha ao MQTT sem tráfego de fundo. Isso acontece por que não há a retenção das mensagens e o fluxo de dados consegue seguir livremente mesmo com uma outra perda.

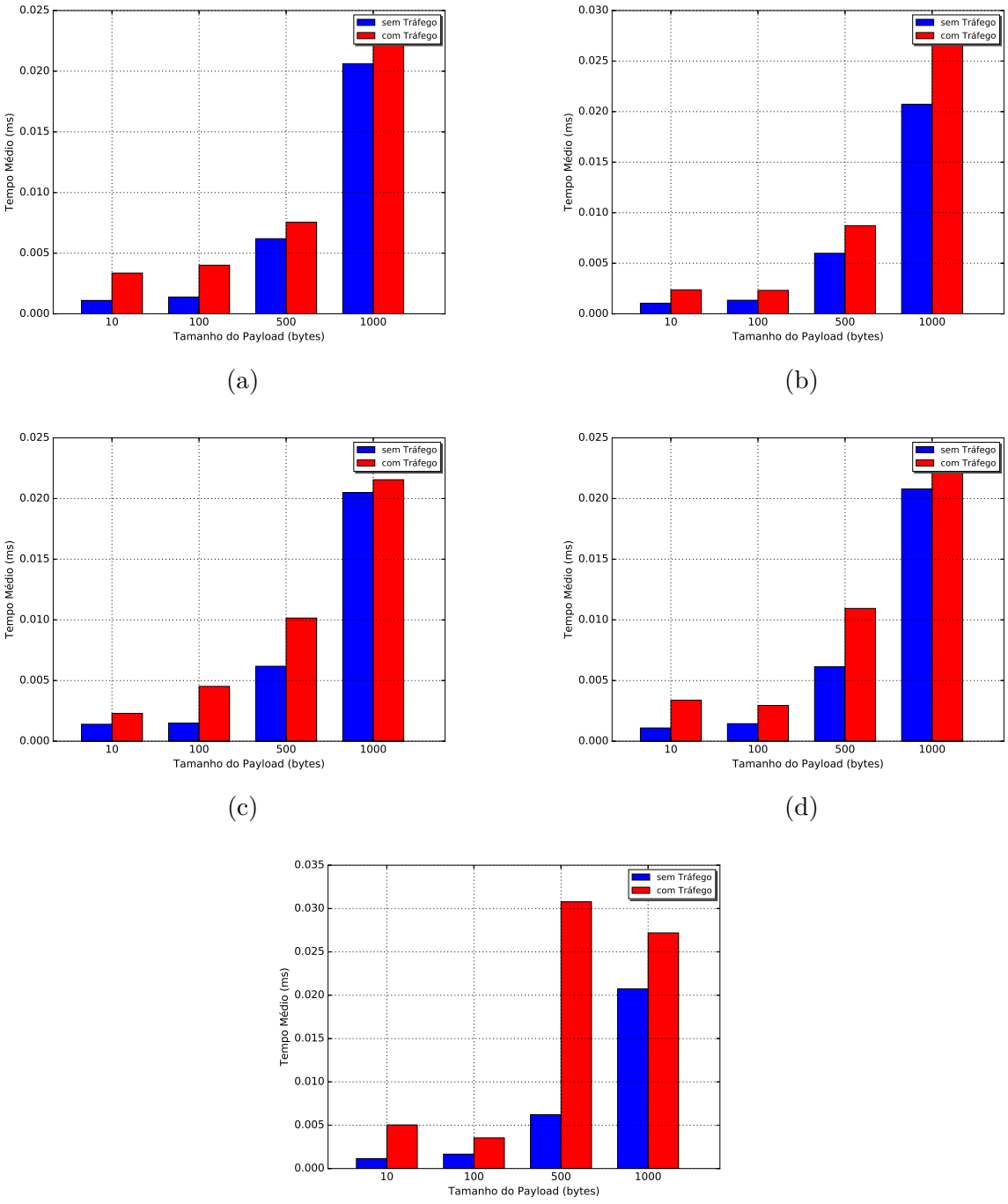


Figura 14 – Comparação do CoAP com e sem Tráfego com intervalos maiores que um segundo com pouco tempo de captura: (a) com intervalo de um segundo (b) com intervalo de cinco segundos (c) com intervalo de dez segundos (d) com intervalo de trinta segundos (e) com intervalo de sessenta segundos

Outra métrica analisada foi o tempo médio entre recebimento de mensagens. As Figuras 19 e 20 mostram que quando não há tráfego de fundo, as respostas são estáveis ficando próximas ao intervalo especificado. Já quando há tráfego de fundo, as respostas são variadas informando que a entrega das mensagens estão sofrendo efeito por esse fluxo

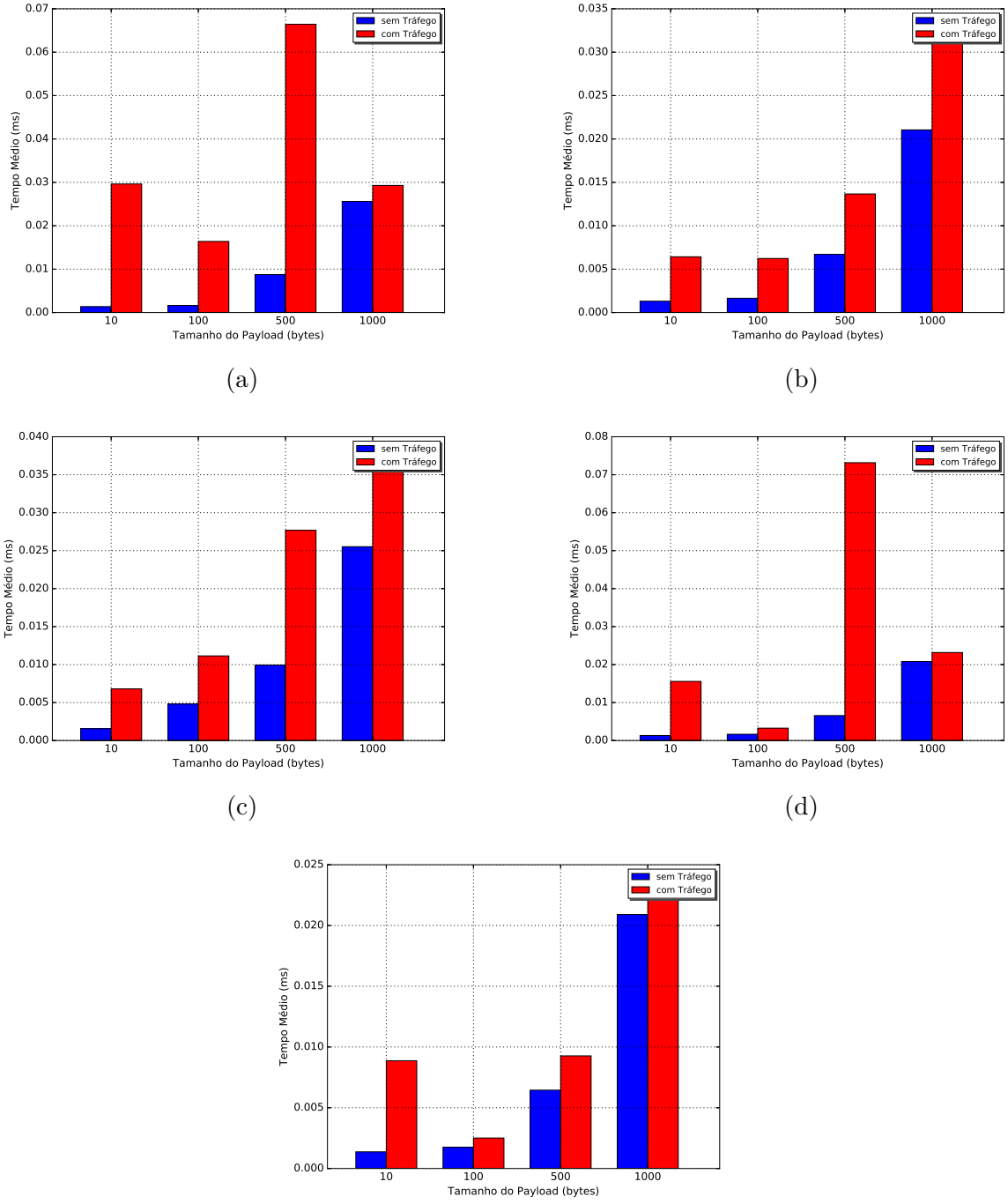


Figura 15 – Comparação do CoAP com e sem Tráfego com intervalos maiores que um segundo: (a) com intervalo de um segundo (b) com intervalo de cinco segundos (c) com intervalo de dez segundos (d) com intervalo de trinta segundos (e) com intervalo de sessenta segundos

de dados. Os testes foram realizados com o algoritmo de Nagle habilitado o que explica a não uniformidade do resultado.

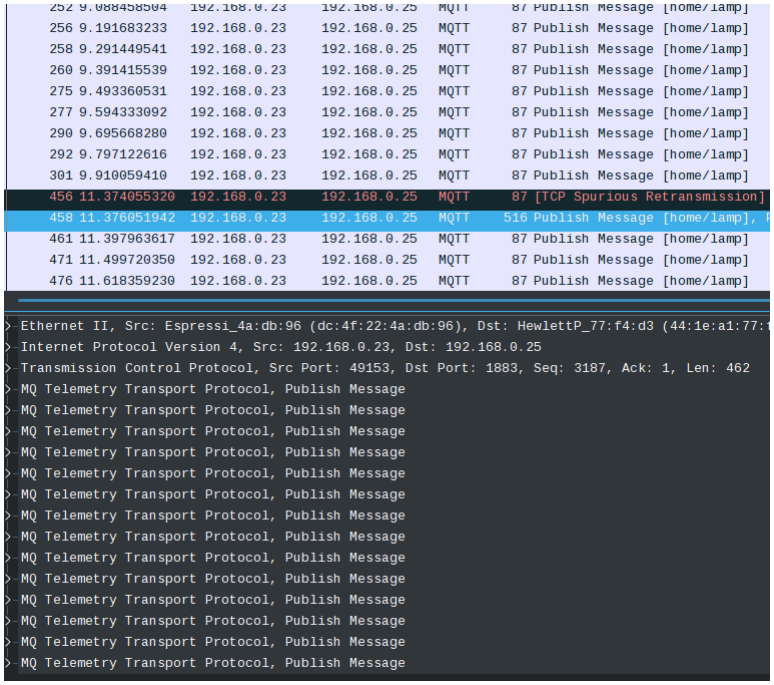


Figura 16 – MQTT: Exemplo de erro ao não receber um ACK.

Fonte – Produzida pelo autor.

Tabela 5 – Porcentagem de mensagens do MQTT em relação ao total capturado com e sem tráfego de fundo no teste de monitoramento em tempo (quase) real

Intervalo entre envios (ms)	**Com tráfego**	**Sem tráfego**
100	6,9 %	26,5 %
200	4,6 %	18,9 %
300	3,1 %	20,3 %
400	2 %	21,3 %
500	2 %	19,9 %
600	2,5 %	18,5 %
700	21,7 %	15,6 %
800	12,8 %	13,8 %
900	2 %	12,4 %

Fonte – Produzida pelo autor

5.2.2 Teste de variação no tamanho da mensagem

Para o MQTT é possível observar que, de modo geral, o tamanho do *payload* não interfere na geração e envio de mensagens, como mostra a Figura 21. Entretanto, é possível notar que a diferença média de tempos sem tráfego de rede é maior que a média quando há tráfego de rede, excluindo 21a. Isso mostra, na verdade, que com informações na rede, ela se torna mais instável para entrega dos pacotes e isso gera números mais baixos de média. Se a rede é estável, a diferença sempre será igual como pode ser visto em todos os gráficos sem tráfego que mantém praticamente a mesma média. A Tabela 6 mostra a relação entre as mensagens enviadas pelo protocolo e todas as mensagens

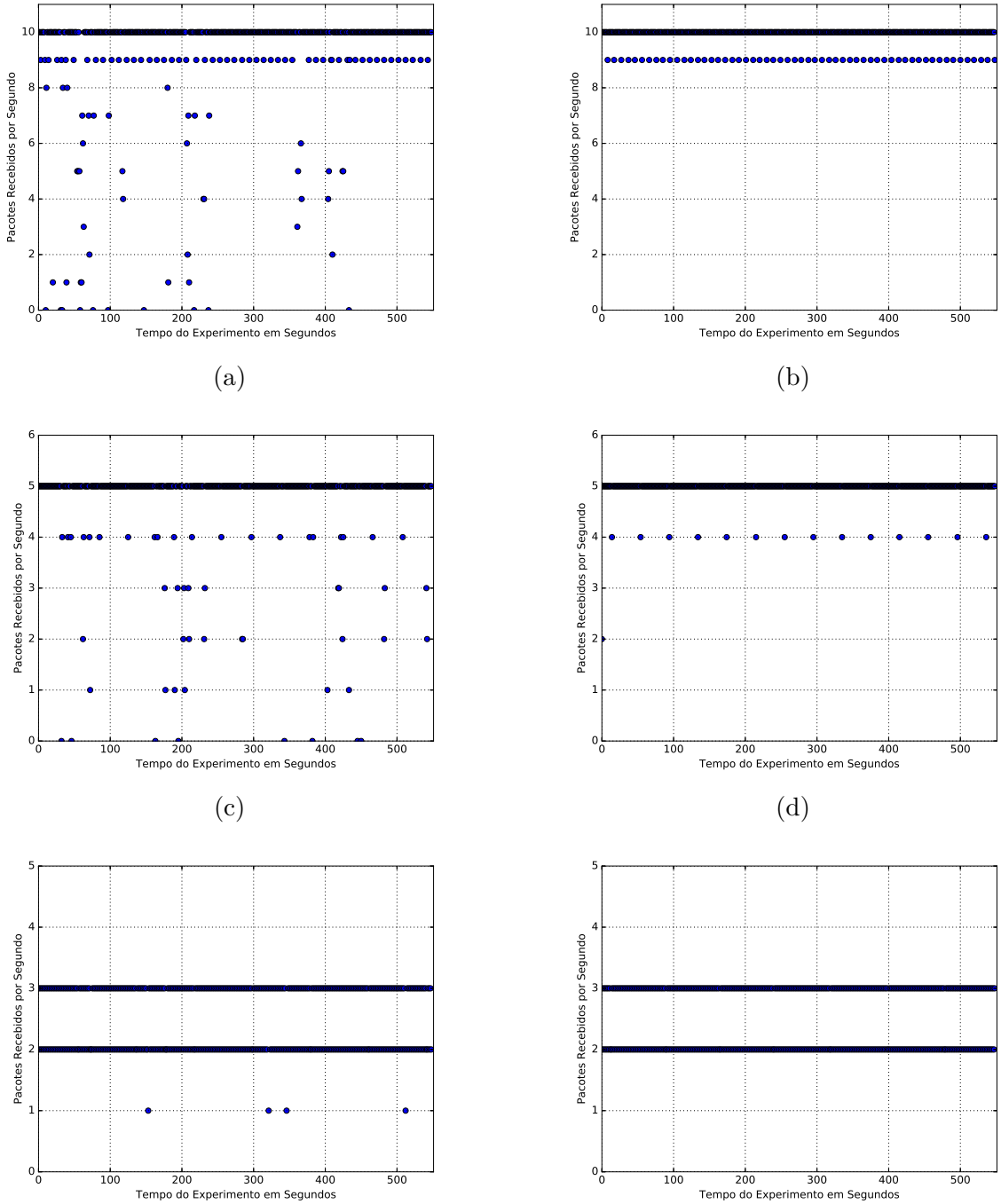
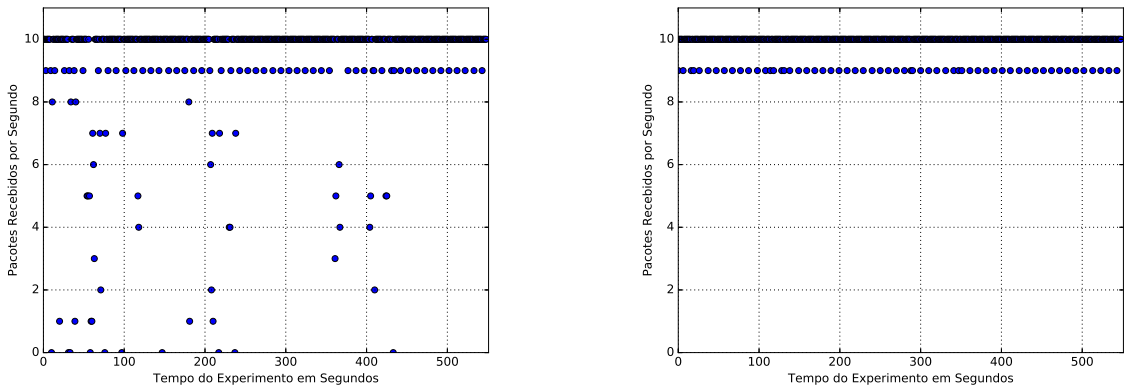


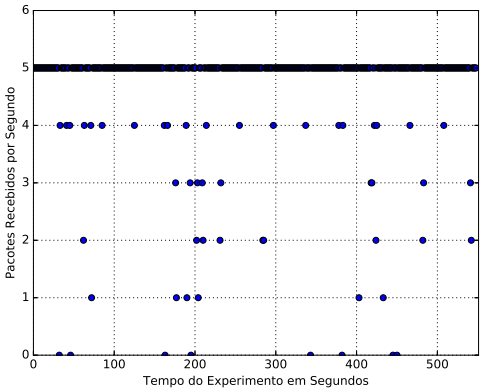
Figura 17 – Comparação do MQTT com e sem Tráfego com intervalos menores que um segundo com Nagle habilitado: (a) com tráfego de fundo e intervalo de cem milissegundos (b) sem tráfego de fundo e intervalo de cem milissegundos (c) com tráfego de fundo e intervalo de duzentos milissegundos (d) sem tráfego de fundo e intervalo de duzentos milissegundos (e) com tráfego de fundo e intervalo de quatrocentos milissegundos (f) sem tráfego de fundo e intervalo de quatrocentos milissegundos

capturadas durante o experimento. Nota-se que o experimento com tráfego de fundo as

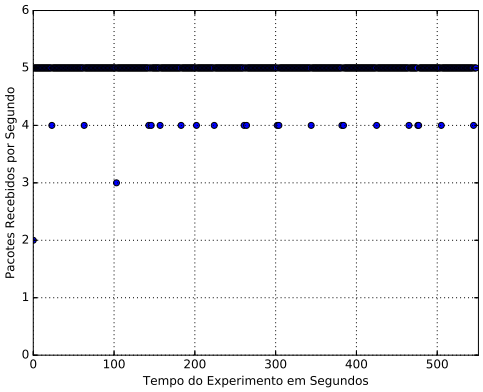


(a)

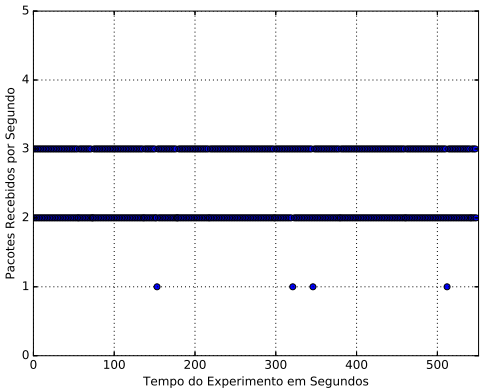
(b)



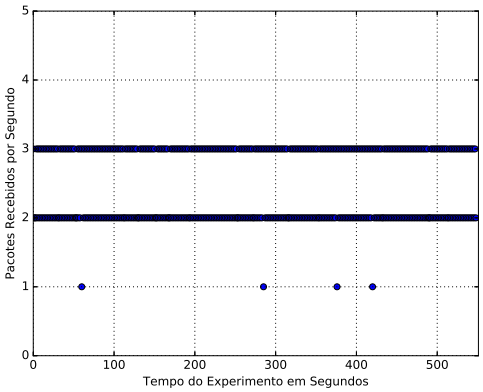
(c)



(d)



(e)



(f)

Figura 18 – Comparação do MQTT com Tráfego com intervalos menores que um segundo com e sem Nagle habilitado: (a) com Nagle habilitado e intervalo de cem milissegundos (b) sem Nagle habilitado e intervalo de cem milissegundos (c) com Nagle habilitado e intervalo de duzentos milissegundos (d) sem Nagle habilitado e intervalo de duzentos milissegundos (e) com Nagle habilitado e intervalo de quatrocentos milissegundos (f) sem Nagle habilitado e intervalo de quatrocentos milissegundos

mensagens do protocolo foram bem menores em relação ao total, indicando que o roteador

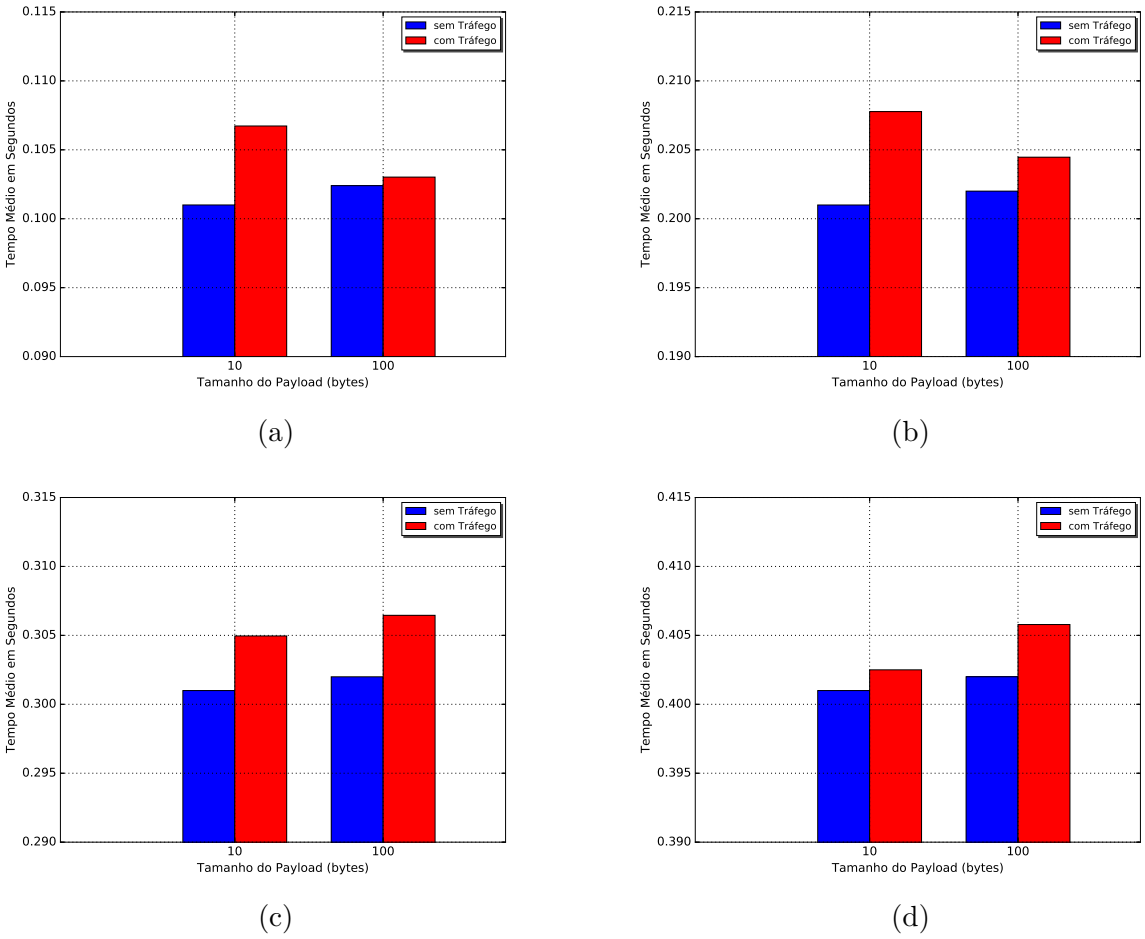
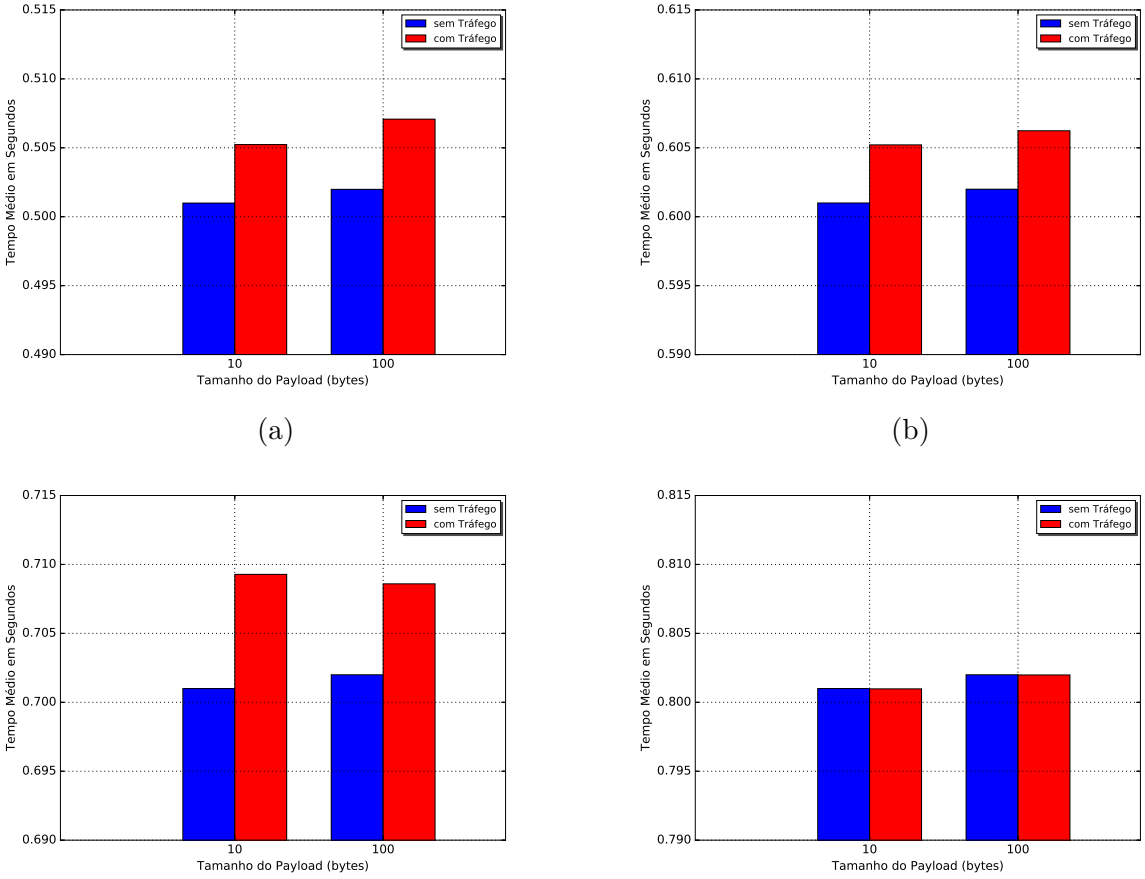


Figura 19 – Comparação de tempo médio de chegada entre MQTT com e sem Tráfego com intervalos entre 100 e 400 ms: (a) MQTT com intervalo de cem milissegundos (b) MQTT com intervalo de duzentos milissegundos (c) MQTT com intervalo de trezentos milissegundos (d) MQTT intervalo de quatrocentos milissegundos

estava em intenso trabalho.

Alguns dos experimentos possuem características distintas dos demais. A Figura 21a mostra que no momento da captura aconteceram algumas perdas de pacotes na camada TCP e foram feitas algumas retransmissões. Ocorreu também o efeito do algoritmo de Nagle em alguns momentos e o ESP parou de enviar pacotes por uns três a oito segundos segundos influenciando na média dos resultados. As Figuras 21b, 21c e 21d mostram que nesses intervalos quase não houve instabilidades na rede por causa dos valores bem parecidos, enquanto em (e), a rede ficou mais instável. Isso pode ser explicado pelo tempo de manutenção de sessão, o *keep alive time*, que era nessa ordem de grandeza e algumas vezes era enviado quase que ao mesmo tempo que a mensagem, gerando tráfego em uma rede que já possuía bastante fluxo.



(a) (b)

(c) (d)

Figura 20 – Comparação de tempo médio de chegada entre MQTT com e sem Tráfego com intervalos entre 500 e 800 ms: (a) MQTT com intervalo de cem milissegundos (b) MQTT com intervalo de duzentos milissegundos (c) MQTT com intervalo de trezentos milissegundos (d) MQTT intervalo de quatrocentos milissegundos

Tabela 6 – Porcentagem de mensagens do MQTT em relação ao total capturado com e sem tráfego de fundo no teste de variação do tamanho da mensagem para um segundo

Tamanho do payload (bytes)	**Com tráfego**	**Sem tráfego**
10	1,2 %	28,9 %
100	1,6 %	29 %
500	2,1 %	28,8 %
1000	14,6 %	20,2 %

Fonte – Produzida pelo autor

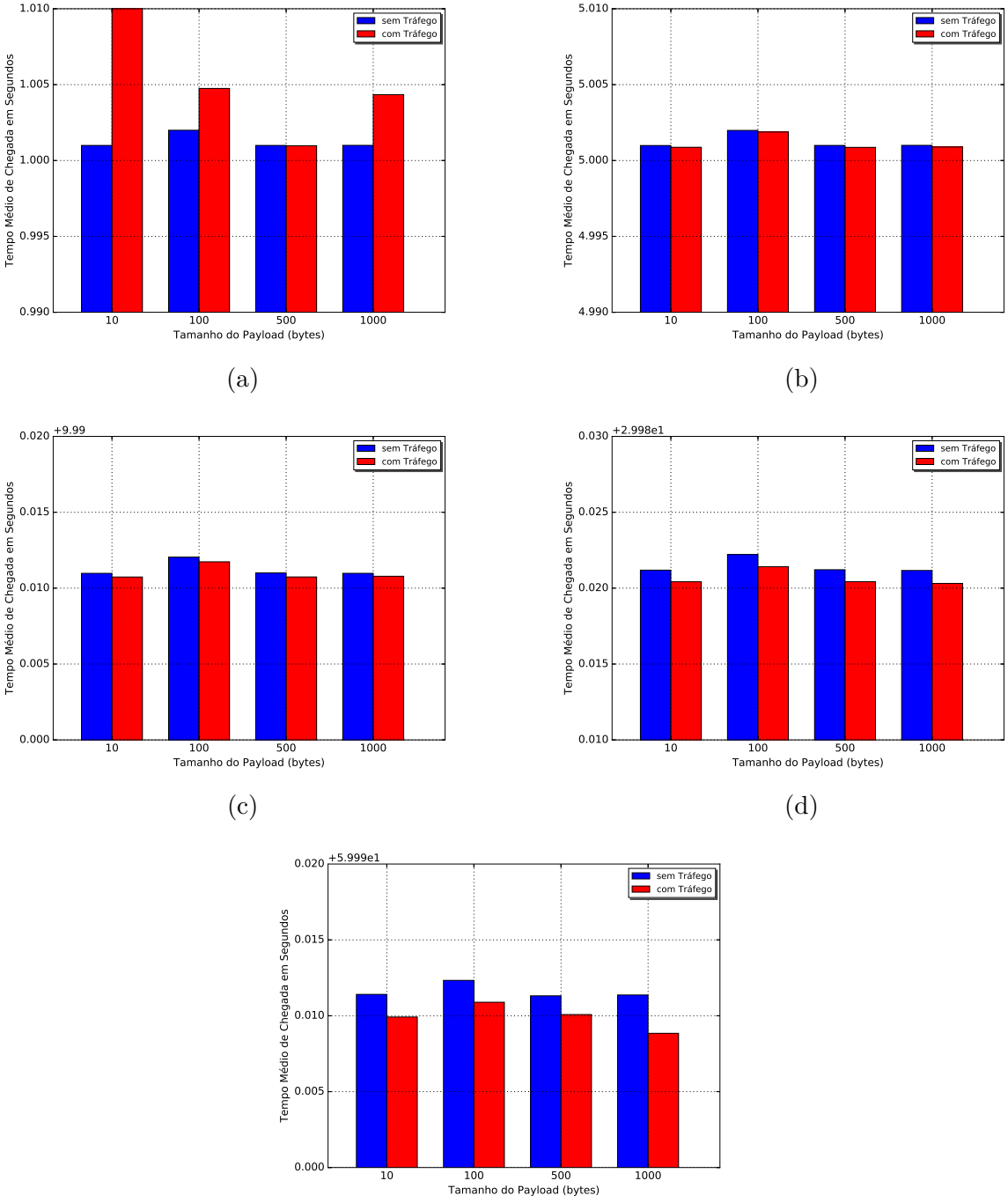


Figura 21 – Comparação do MQTT com e sem Tráfego com intervalos maiores que um segundo: (a) com intervalo de um segundo (b) com intervalo de cinco segundos (c) com intervalo de dez segundos (d) com intervalo de trinta segundos (e) com intervalo de sessenta segundos

5.3 Comparação entre CoAP e MQTT

5.3.1 Teste de monitoramento em tempo (quase) real

Colocando lado a lado os dois protocolos, observa-se que mesmo o MQTT com a camada de transporte sendo o TCP com todos os seus mecanismos de confiabilidade, al-

goritmo de Nagle, controle de fluxo e erro, ele consegue superar o CoAP para intervalos menores que quinhetos milissegundos. Depois desse valor, até o limite de novecentos milissegundos, os resultados ficam muito próximos. Com o algoritmo de Nagle desabilitado, o MQTT torna-se ainda mais interessante.

Em relação ao número de mensagens retransmitidas, para o MQTT a verificação é feita pela detecção de *"TCP Spurious Retransmission"* que acabaram gerando reflexos no resultado final por causa do algoritmo de Nagle. As retransmissões no CoAP acontecem pelo lado do requisitante que deve implementar seu sistema de monitoramento para mensagens que necessitem de confirmação. Dito isso, observa-se que o protocolo MQTT possui menos erros, o que era de se esperar pela confiabilidade do TCP em relação ao UDP. As Tabelas 7 e 8 mostram esses dados recolhidos no experimento.

Em relação ao tamanho máximo da mensagem enviada, a biblioteca utilizada para o MQTT conseguiu ser mais estável e enviar mensagens de até 1500 *bytes* sem que utilizasse toda a memória RAM do ESP. Já para o CoAP, após a alteração da biblioteca, foi possível enviar até 1024 *bytes* antes que o mesmo efeito acontecesse.

Tabela 7 – Número retransmissões na camada de aplicação do MQTT nos testes com intervalo menor que um segundo

Intervalo entre envios (ms)	**Retransmissões**	**Total**
100	27	5611
200	24	2876
300	15	1962
400	5	1490
500	5	1194
600	9	995
700	7	859

Fonte – Produzida pelo autor

Tabela 8 – Número de mensagens retransmitidas em relação ao total de mensagens do protocolo CoAP nos testes com intervalo menor que um segundo

Intervalo entre envios (ms)	**Retransmissão**	**Total**
100	68	6520
200	64	3622
300	55	2713
400	42	2342
500	50	1766
600	6	1898
700	3	1651

Fonte – Produzida pelo autor

5.3.2 Teste de variação no tamanho da mensagem

A comparação direta entre esses protocolos não pode ser feita pelos gráficos anteriores, pois as métricas utilizadas são diferentes. Porém, nota-se que os dois tiveram problemas ao utilizar uma rede com tráfego de fundo. Novamente, o resultado obtido pelo CoAP mostra que esse protocolo sofre mais em redes com grande quantidade de fluxo de dados.

Observa-se que o número de mensagens retransmitidas pelo CoAP foi maior com intervalos menores de tempo. Após 5 segundos de intervalo, as retransmissões ocorrem, porém em número bem menor. As Tabelas 9 e 11 mostram os resultados mais relevantes do CoAP em relação ao número de retransmissões. Os testes que não foram citados não tiveram erros.

Para o MQTT, apenas no intervalo de um segundo entre envios com tráfego de fundo obteve uma acentuada diferença no número de erros. Isso é resultado da forma com o TCP faz o controle de confirmação dos dados já que com intervalos maiores, o *timeout* para receber um ACK era menor que a diferença de tempos. A Tabela 10 mostra o resultado mais relevante do MQTT para as restransmissões.

Em relação ao tamanho dos pacotes, o CoAP teve um aparente aumento do tempo de resposta com o aumento do *payload*. O MQTT teve um desempenho mais uniforme comparado com os outros tamanhos. Vale ressaltar que o TCP estava fragmentando os pacotes com tamanho máximo de 536 *bytes* que é implementado como padrão no *lwIP* disponível no ESP8266. Já o CoAP, estava mandando o pacote inteiro como característica do UDP, tendo como máximo valor 1024 *bytes* devido à limitação do dispositivo.

Tabela 9 – Número de mensagens retransmitidas em relação ao total de mensagens do protocolo CoAP nos testes com intervalo de um segundo com tráfego de fundo

Tamanho da mensagem	**Retransmissão**	**Total**
10	96	3960
100	225	6983
500	253	6459
1000	220	6800

Fonte – Produzida pelo autor

Tabela 10 – Número de mensagens retransmitidas em relação ao total de mensagens do protocolo MQTT nos testes com intervalo de um segundo com tráfego de fundo

Tamanho da mensagem	**Retransmissão**	**Total**
10	40	3600
100	18	3601
500	12	3609
1000	0	3584

Fonte – Produzida pelo autor

Tabela 11 – Número de mensagens retransmitidas em relação ao total de mensagens do protocolo CoAP nos testes com intervalo de cinco segundos com tráfego de fundo

Tamanho da mensagem	**Retransmissão**	**Total**
10	12	1436
100	18	1438
500	7	1433
1000	10	1424

Fonte – Produzida pelo autor

6 Considerações Finais e Trabalhos Futuros

Este trabalho buscou analisar o papel de alguns protocolos de rede que estão ganhando espaço no mercado de Internet das Coisas. Com o crescente desenvolvimento tecnológico, a criação de novas plataformas vem aumentando, assim como as possibilidades de utilização de equipamentos de IoT. Com isso, criar cenários é uma maneira de otimizar as escolhas dentro de uma grande quantidade de opções. Após a observação das aplicações disponibilizadas por grandes empresas do ramo da tecnologia que estão investindo na área de Internet das Coisas, foi possível a classificar alguns protocolos em cenários de aplicação. Feito isso, os protocolos MQTT, MQTT-SN, CoAP, AMQP e DDS foram analisados e, a partir das observações de suas principais características, foi proposta a divisão em três cenários: residencial, comercial e industrial.

Com isso, a partir de informações como qualidade da rede onde os dispositivos estão instalados, utilização de concentradores de dados, interoperabilidade entre redes e aplicações distintas, aplicações em tempo real, entre outras, verificou-se que o MQTT encaixa-se nos cenários residenciais e comerciais por utilizar uma base sólida e confiável, o TCP. Aplicações que necessitem troca de mensagens com grande quantidade de dados (limitados aos dispositivos envolvidos) em redes locais também podem utilizar o MQTT como base. Devido a grande adoção desse protocolo pela facilidade de manipulação, existem versões para dispositivos com baixo poder de processamento o que aumenta ainda mais as possibilidades de utilização. As aplicações de IoT da *Amazon* e da *Microsoft* dão suporte a esse protocolo para troca de mensagens entre dispositivos locais ou até servidores próximos.

O MQTT-SN encaixa-se nos cenários residencial e industrial com ressalvas. Por utilizar UDP como base, aplicações de envio constante de dados podem tirar proveito desse protocolo que utiliza o formato *Publish/Subscribe* para envio mais fácil em *multicast*, já que a mensagem é primeiro enviada para um dispositivo com mais processamento para depois ser enviada aos demais dispositivos na rede. A característica de descoberta de novos *gateways* deixa a aplicação ainda mais independente. A ressalva desse protocolo é a falta de bibliotecas abertas para dispositivos com baixo poder de processamento. Isso acaba restringindo os testes com o protocolo.

O CoAP é um dos mais versáteis tendo vantangens em todos os cenários. A arquitetura cliente/servidor utilizada no protocolo permite que os dispositivos economizem energia em rotinas programadas. O uso do UDP torna o fluxo de dado menor o que é ideal para aplicações industriais. A função de descoberta e de observação são trunfos que podem ser utilizados na indústria em equipamentos que tem alta rotatividade. A proximidade com

o padrão HTTP torna o uso mais amigável para pessoas que já desenvolvem aplicações Web e pelo fato de não precisar de um concentrador economiza espaço e torna o sistema independente. Por isso, o CoAP é melhor utilizado em aplicações industriais e residenciais.

O AMQP encaixa-se melhor em aplicações comerciais ou industriais. A grande utilização desse protocolo está ligado ao fato da segurança no transporte de uma grande quantidade de informação e nos mecanismos de controle de fluxo e erro que ajudam na interoperabilidade dos sistemas. Empresas como *Microsoft*, *IBM* e *Intel* já utilizam esse protocolo para envio de dados entre *datacenters*.

O DDS, assim como o AMQP, encaixa-se melhor em aplicações comerciais e industriais. O fato de focar em interoperabilidade entre sistemas com vários níveis de *QoS* que permitem envio em tempo real de informações ou grande quantidade de dados e que funcionam tanto em UDP como em TCP, são grandes atrativos para empresas e indústrias. Aplicações médicas com envio de informações do cliente diretamente para o hospital ou para o médico responsável, já está em uso em alguns sistemas. Aplicações de comunicação em tempo real em indústrias também são contempladas pelo modelo de implementado pelo DDS. O próprio sistema de descoberta de novos dispositivos ajuda empresas que precisam de troca constantes de equipamentos.

A partir dos experimentos apresentados no capítulo anterior, foi possível constatar alguns resultados do uso dos protocolos MQTT e do CoAP em dispositivos com baixo poder de processamento e energia aplicados em um cenário residencial. O primeiro ponto de destaque é que o tráfego de fundo em uma rede local tem um impacto importante na entrega de mensagens, pois por causa dele o número de retransmissão de pacotes pode aumentar tornando o sistema mais susceptível a falhas. Esse ponto também aumenta o número de mensagens que circulam na rede causando efeitos colaterais em outros dispositivos ou sistemas, considerando, por exemplo, uma rede compartilhada.

Sobre o MQTT, observou-se que o protocolo, por não possuir nenhum controle de fluxo ou erros, utiliza todas as vantagens do TCP e isso acabou deixando o uso mais estável como um todo. O ponto negativo, é ficar atrelado a como o protocolo de transporte gerenciará o envio das mensagens. O efeito do algoritmo de Nagle que vem habilitado como padrão para melhorar a eficiência da rede aumentando a quantidade de dados úteis tem como *tradeoff* retirar a frequência desejada causando espaços de tempos sem envio de mensagem. Isso acaba impactando diretamente nas características de envio constante de dados e na qualidade do sinal da rede apresentada neste trabalho. Porém, é possível retirar esse parâmetro e o sistema responde da maneira planejada.

Já o CoAP utiliza como base o UDP, um protocolo sem nenhum tipo de controle para evitar perda de pacotes. A partir dessa limitação, o CoAP contruiu seu próprio sistema de retransmissão criando uma camada de confiabilidade. Contudo, esse mecanismo acabou trazendo problemas na frequência de mensagens enviadas, pois o modelo de retransmissão

com tempos exponenciais acaba deixando alguns segundos sem mensagens. O uso do UDP também aumenta a chance do pacote ser ignorado pelo roteador ou sofrer com atrasos de *buffer* já que esse protocolo não tem característica de ser confiável. Esse resultado impacta diretamente nas características de qualidade do sinal e do envio constante de dados que são importantes no cenário residencial.

Sobre o *hardware* do ESP8266, ele se manteve constante com tempo de geração de mensagens de no máximo dois milissegundos. A ressalva a ser feita é em relação a implementação das bibliotecas para o dispositivo, pois algumas delas não possuem mecanismos para gerenciar a memória RAM, que é bastante restrita, e isso acaba limitando a quantidade de dados que pode ser enviada por mensagem. Isso acaba impactando na caraterística de grande quantidade de dados por mensagem, pois o máximo conseguido era de 1024 *bytes*. Mesmo assim, observou-se que o protocolo CoAP foi mais sensível a tamanhos maiores de *payload* enquanto o MQTT, que utiliza a fragmentação de pacotes do TCP, conseguiu se manter mais estável durante os testes com tráfego de rede, ainda mais se o algoritmo de Nagle for desabilitado.

Comparando os resultados com os trabalhos relacionados, observa-se que as limitações impostas pelo ESP8266 não permitiram uma comparação melhor sobre o tamanho das mensagens obtidas no artigo do (MUN; DINH; KWON, 2016). Este artigo ainda relata que o CoAP teve um desempenho melhor que o MQTT para *payloads* menores que 1024 *bytes*. Os resultados obtidos neste trabalho foram diferentes, devido principalmente, a implementação das bibliotecas. Outro ponto importante é que os testes do artigo foram realizados em um computador com alto poder de processamento que não possui alguns parâmetros de eficiencia habilitados por padrão, como o algoritmo de Nagle do lwIP. Já sobre o artigo (CHAUDHARY; PEDDOJU; KADARLA, 2017), os resultados foram semelhantes. As topologias entres os trabalhos foram semelhantes e resultados como o MQTT ser mais estável permitindo o envio de mensagens em tempo quase real estão de acordo. Outro resultado que está de acordo é que o CoAP envia mais mensagens quando em redes instáveis devido ao mecanismo de retransmissão de pacotes.

Além disso, as bibliotecas utilizadas nos dispositivos com recursos restritos são dependentes dos seus criadores. A especificação serve como uma base, porém alguns elementos são alterados para manter a estabilidade dos dispositivos o que torna a escolha desse quesito crucial para o desenvolvimento de sistemas. Os ajustes realizados nas bibliotecas para os experimentos e as diferenças em comparação com as especificações mostram que elas ainda estão em fase de aprimoramento.

Isto posto, a implementação de sistemas com vários dispositivos deve ser analisada, pois dependendo da aplicação o tráfego gerado pode atrapalhar o seu funcionamento. Sistemas que compartilhem a mesma conexão entre usuários e sensores serão ainda mais afetadas se o número de dispositivos for grande. O uso do TCP como camada de transporte

gera um tráfego a mais de informação, porém os resultados são mais estáveis do que aqueles que utilizam o UDP utilizando esse mesmo ambiente do teste. Assim, o MQTT mostra-se um protocolo mais estável e que consegue ser utilizado em uma grande faixa de aplicações residenciais.

Após a realização desse trabalho, espera-se que este seja utilizado como ponto de partida para que pesquisadores e entusiastas de Internet das Coisas possam escolher qual o melhor protocolo da camada de aplicação segundos os cenários estudados. Uma possibilidade de trabalho que pode ser desenvolvido é verificar a implementação de bibliotecas dos outros protocolos que foram estudados para o ESP8266 e, assim, pode ser feita uma analise mais precisa em utilizando as métricas desse projeto. Isto tornaria possível também, testar os protocolos nos outros cenários para confirmar os resultados propostos.

Outros pontos a serem testados futuramente são as características propostas na Tabela 2 a partir da criação de métricas que se relacionam com essas características. Após isso, analisar o efeito nos três cenários propostos.

Por fim, sobre o dispositivo utilizado nos testes, seria interessante um estudo da condição energética durante o processo de geração e transmissão de mensagens para determinar o protocolo mais eficiente em locais que necessitem de equipamantos com baterias.

Referências

AMARAN, M. H. et al. A comparison of lightweight communication protocols in robotic applications. *Procedia Computer Science*, Elsevier, v. 76, p. 400–405, 2015. 34

APPEL, S.; SACHS, K.; BUCHMANN, A. Towards benchmarking of AMQP. *Proceedings of the 4th ACM International Conference on Distributed Event-Based Systems, DEBS 2010*, n. October 2016, p. 99–100, 2010. Disponível em: <http://dx.doi.org/10.1145/1827418.1827438>. 22, 23

BAHIA, J. G.; CAMPISTA, E. M. Um Mecanismo de Controle de Demanda no Provimento de Serviços de IoT Usando CoAP. 2016. 21

BANDYOPADHYAY, S.; BHATTACHARYYA, A. Lightweight internet protocols for web enablement of sensors using constrained gateway devices. In: IEEE. *Computing, Networking and Communications (ICNC), 2013 International Conference on*. [S.l.], 2013. p. 334–340. 32

BORMANN, C. et al. *CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets*. [S.l.], 2018. 22

BORMANN, C.; SHELBY, Z. *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*. 2016. RFC7959. Disponível em: <http://tools.ietf.org/rfc/rfc7959.txt>. Acesso em: 03.05.2018. 21

BRUSH, A. et al. Home automation in the wild: challenges and opportunities. In: ACM. *proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. [S.l.], 2011. p. 2115–2124. 29

CARO, N. D. et al. Comparison of two lightweight protocols for smartphone-based sensing. In: IEEE. *Communications and Vehicular Technology in the Benelux (SCVT), 2013 IEEE 20th Symposium on*. [S.l.], 2013. p. 1–6. 32, 34, 35

CHAUDHARY, A.; PEDDOJU, S. K.; KADARLA, K. Study of internet-of-things messaging protocols used for exchanging data with external sources. p. 666–671, 2017. 32, 35, 38, 39, 65

CURVELLO, A. *Apresentando o módulo ESP8266*. 2015. Disponível em: <https://www.embarcados.com.br/modulo-esp8266/>. Acesso em: 23.06.2018. 26

DERHAMY, H. et al. A survey of commercial frameworks for the Internet of Things. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, v. 2015-Octob, 2015. ISSN 19460759. 30

DUNKELS, A. Design and implementation of the lwip tcp/ip stack. *Swedish Institute of Computer Science*, v. 2, p. 77, 2001. 28

GEORGE, D. *MicroPython on the ESP8266: beautifully easy IoT*. 2016. Disponível em: <https://www.kickstarter.com/projects/214379695/micropython-on-the-esp8266-beautifully-easy-iot/posts/1501224>. 26

HARTKE, K. *Observing Resources in the Constrained Application Protocol (CoAP).* 2015. RFC7641. Disponível em: <http://tools.ietf.org/rfc/rfc7641.txt>. Acesso em: 08.04.2018. 22

KODALI, R. K.; SORATKAL, S. R. MQTT based home automation system using ESP8266. *IEEE Region 10 Humanitarian Technology Conference 2016, R10-HTC 2016 - Proceedings*, n. October, 2017.

KOVALENKO, O. *ArduinoMqtt: MQTT Client library for Arduino based on the Eclipse Paho project*. 2018. Disponível em: <https://github.com/monstrenyatko/ArduinoMqtt>. 42

KUROSE, J. F.; ROSS, K. W. *Computer Networking: A Top-Down Approach (6th Edition)*. 6th. ed. [S.l.]: Pearson, 2012. ISBN 0132856204, 9780132856201. 14, 15, 16

LEE, S. et al. Correlation analysis of MQTT loss and delay according to QoS level. *International Conference on Information Networking*, p. 714–717, 2013. ISSN 19767684. 18

LUZURIAGA, J. E. et al. A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. *2015 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015*, p. 931–936, 2015. ISSN 2331-9860. 23

MAZZER, D.; FRIGIERI, E.; PARREIRA, L. Protocolos M2M para Ambientes Limitados no Contexto do IoT: Uma Comparação de Abordagens. *Inatel.Br*, 2015. Disponível em: <http://www.inatel.br/smartcampus/imgs/protocolos-para-iot-pt.pdf>. 17, 21

MUN, D.-H.; DINH, M. L.; KWON, Y.-W. An assessment of internet of things protocols for resource-constrained applications. v. 1, p. 555–560, 2016. 20, 31, 32, 34, 37, 38, 65

NAGESH, L. P. P. *ESP-CoAP: This is a Arduino Library for the ESP8266 12E*. 2018. Disponível em: <https://github.com/automote/ESP-CoAP>. 42

NAIK, N. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*, 2017. 16, 31, 32, 34, 35

NTP: Connect to a NTP server. 2018. Disponível em: <https://github.com/arduino-libraries/NTPClient>. 43

OASIS. OASIS Advanced Message Queuing Protocol OASIS Standard. *OASIS Standard*, n. October, p. 124, 2012. Disponível em: <http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html>. 23

OASIS. MQTT Version 3.1.1. *OASIS Standard*, n. October, p. 81, 2014. Disponível em: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>. 17, 18

OASIS. *MQTT Versão 5.0*. 2017. Disponível em: <http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf>. 19

O'LEARY, N. *PubSubClient: a library provides a client for doing simple publish/subscribe messaging with a server that supports MQTT*. 2018. Disponível em: <https://github.com/knolleary/pubsubclient>. 42

OMG. *What is DDS?* 2014. Disponível em: <https://www.omgwiki.org/dds/what-is-dds-3/>. Acesso em: 15.11.2018. 24

OMG. *Data Distribution Service*. 2015. Disponível em: <https://www.omg.org/spec/DDS/About-DDS/>. 24, 25

SHELBY, Z.; HARTKE, K.; BORMANN, C. *The Constrained Application Protocol (CoAP)*. 2014. RFC7252. Disponível em: <http://tools.ietf.org/rfc/rfc7252.txt>. Acesso em: 06.04.2018. 21, 22

STALLINGS, W. *Data and Computer Communications*. Pearson/Prentice Hall, 2007. (Alternative eText Formats Series). ISBN 9780132433105. Disponível em: <https://books.google.com.br/books?id=c_AWmhkovR0C>. 14, 15, 16, 19

TEAM, E. I. *ESP8266 Datasheet*. 2018. Disponível em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex$_d atasheet_e n$>. 26

TEAM, E. I. *ESP8266 Datasheet*. 2018. Disponível em: <https://www.espressif.com/en/products/hardware/esp8266ex/resources>. 27

TORRES, A. B.; ROCHA, A. R.; SOUZA, J. N. de. Análise de desempenho de brokers mqtt em sistema de baixo custo. In: *Anais do XXXVI congresso da sociedade brasileira de computaç ao. Sociedade Brasileira de Computaçao*. [S.l.: s.n.], 2016. 33

TRUONG, A. S.-C.; LINH, H. MQTT For Sensor Networks (MQTT-SN) Protocol Specification. *International Business Machines Corporation (IBM*, v. 1, p. 28, 2013. Disponível em: <http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf>. 20

VILLAVERDE, B. C. et al. Service discovery protocols for constrained machine-to-machine communications. *IEEE Communications Surveys and Tutorials*, v. 16, n. 1, p. 41–60, 2014. 33

XU, L. D.; HE, W.; LI, S. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, IEEE, v. 10, n. 4, p. 2233–2243, 2014. 31