Um Framework para Gerenciamento da Comunicação de Múltiplos Recursos e Observadores em Internet das Coisas

Marlon Santos[1], **Antônio Abelém**[1], **and André Riker**[1]

[1]Universidade Federal do Pará (UFPA) – Belém, PA – Brasil

marlon.santos.santos@icen.ufpa.br
abelem@ufpa.br, ariker@ufpa.br

Abstract. *IoT platforms play a central role in IoT architecture as they facilitate the development of services and applications. IoT platforms must satisfy multiple application requirements and clients preferences, but they must also avoid consuming the scarce resources of IoT devices. This article proposes a framework called OBSERVICE that aims to support communication management in IoT environments involving multiple stakeholders (observers) in consuming IoT data from multiple producers. The proposed framework allows multiple entities, i.e. clients, to simultaneously specify their communication preferences. The implementation of the framework had web compatibility as a priority, in addition to code reuse, open source tools and integration with real code embedded in IoT devices. The results of validation tests show that OBSERVICE is capable of managing communication involving multiple IoT entities and devices, with low resource consumption and improvement in network management aspects.*

Resumo. *As plataformas IoT desempenham um papel central na arquitetura IoT, pois facilitam o desenvolvimento de serviços e aplicações. As plataformas IoT devem satisfazer os múltiplos requisitos das aplicações e preferências dos clientes, mas também devem evitar o consumo dos escassos recursos dos dispositivos IoT. Este artigo propõe um framework chamado OBSERVICE que visa suportar o gerenciamento da comunicação em ambientes IoT envolvendo múltiplas partes interessadas (observadores) em consumir dados IoT de vários produtores. O framework proposto permite que várias entidades, isto é usuários, especifiquem simultaneamente suas preferências de comunicação. A implementação do framework teve a compatibilidade com a web como prioridade, além do reuso de código, ferramentas de código aberto e a integração com código real embarcado em dispositivos IoT. Os resultados dos testes de validação mostram que o OBSERVICE é capaz de gerenciar a comunicação envolvendo múltiplas entidades e dispositivos IoT, com baixo consumo de recursos e melhora em aspectos de gerência de redes.*

1. Introdução

Internet das Coisas, do inglês *Internet-of-Things* (IoT), é um conceito da computação que se refere a interconectar objetos ou coisas aos sistemas digitais de acesso global [Botta et al. 2016]. IoT é uma tendência global que permite o surgimento de aplicações de monitoramento e tomada de decisão inteligente em diversos ambientes, tais como transporte, indústria, meio ambiente, cidades e agricultura.

Uma ideia extremamente difundida é que as diversas aplicações IoT devem compartilhar uma camada comum de software, pois isto reduz o custo de implementação e padroniza as aplicações [Ismail et al. 2018]. Esta camada de software é uma plataforma que visa a reutilização de código, suporta protocolos de comunicação, faz a gestão de

dados e dispositivos e suporta mecanismos de segurança. Do ponto de vista do mercado, não se mostra vantajoso ter que investir no desenvolvimento novo e completo cada vez que uma nova aplicação surge. Portanto, a plataforma IoT possui grande importância para a adoção de IoT no mercado.

Em um futuro próximo, plataformas de IoT devem suportar mecanismos de comunicação adaptados às necessidades das aplicações e cientes da capacidade dos dispositivos. Suportar aquilo que as aplicação esperam e estar adaptado aos recursos dos dispositivos significa ter sistemas IoT mais eficientes e funcionais. Dessa forma, espera-se que as plataformas tenham suporte às aplicações e que também estejam customizadas para melhor desempenho dos dispositivos IoT.

As redes de comunicação IoT utilizam dispositivos com restrição de recursos computacionais, tais como processamento, memória e largura de banda. Por isso, as redes IoT economizam ao máximo recursos computacionais no nível dos dispositivos. Neste contexto, uma característica das aplicações IoT de monitoramento inteligente é a necessidade em receber dados periodicamente sobre um determinado alvo. Por exemplo, em aplicações industriais, pode ser desejável receber dados dos maquinários a cada 10 segundos. Este tipo de comunicação não terá bom rendimento se for utilizado o modelo do tipo requisição-resposta. A fim de economizar recursos dos dispositivos, é necessário que a comunicação seja assíncrona com reduzido número de requisições. Ou seja, que a aplicação envie uma única requisição e os dispositivos sejam capazes de enviar dados periodicamente sem a necessidade de novas requisições.

É um desafio propor e implementar soluções que sejam capazes de gerenciar a comunicação entre múltiplos usuários e produtores de dados de forma assíncrona. Um dos aspectos desse desafio é que existem poucas plataformas IoT de código aberto. Isto deve-se ao fato de que grande parte do investimento feito em plataformas IoT resulta em plataformas proprietárias.

Para abordar os desafios apresentados, este artigo propõe um framework chamado OBSERVICE. A solução proposta visa gerenciar a comunicação entre múltiplos clientes e servidores IoT. O OBSERVICE é uma solução em código-aberto que integra os dispositivos IoT aos mecanismos propostos na plataforma IoT, atendendo as necessidades das aplicações e executa ações de coleta de dados ciente das restrições dos dispositivos. O OBSERVICE é baseado na ferramenta Californium [Kovatsch et al. 2014], a qual é vastamente utilizada em aplicações reais IoT. As entidades que recebem os dados são denominadas de observadores, os quais podem determinar seus interesses e preferência de comunicação. Os dispositivos fontes de dados são observados e produzem notificações de acordo com as preferências determinadas pelos observadores.

O restante desse artigo está dividido como segue. A Seção 2 apresenta os trabalhos relacionados. O framework proposto é apresentado na Seção 3. A validação desse framework, cenário de testes e resultados, são mostrados na Seção 4. Por fim, as considerações finais na Seção 5.

2. Trabalhos Relacionados

Os autores [Ishaq et al. 2016] apresentam uma implementação real que envolve comunicação periódica de redes IoT. Neste trabalho, há um roteador de borda que agrega todas as mensagens CoAP recebidas de nós, enviando uma única mensagem CoAP para o destino final. Portanto, o objetivo é reduzir o número de mensagens de observação de um recurso.

A pesquisa apresentada por [Iglesias-Urkia et al. 2018] propõe duas novas opções para o protocolo CoAP com o objetivo de melhorar a comunicação periódica. A primeira melhoria permite que um cliente observe um recurso sem obter o atual estado do recurso, usando mensagens do tipo POST. Outra melhoria proposta permite que um recurso que

seja do tipo "configuração"possa ser criado ou alterado por mensagem CoAP do tipo PUT ou POST.

Em [Suwannapong and Khunboa 2021] e [Suwannapong and Khunboa 2019] há soluções de controle de congestionamento para tráfego CoAP periódico. A ideia é gerenciar o buffer de dados e também os algoritmos de backoff responsáveis pelo retransmissão de acordo com o nível de congestionamento da rede. As soluções são capazes de melhorar a taxa de perda de mensagens e o tempo para se iniciar a comunicação periódica.

O artigo de [Lai et al. 2020] aborda um cenário onde os clientes registram seus interesses junto aos dispositivos IoT e esses permanecem enviando notificações com dados. Neste cenário, alguns clientes possuem diferentes demandas em termos de intervalo máximo e mínimo para receber as notificações. A fim de evitar o desperdício de notificações, o artigo propõe o uso de proxies para coordenar o envio de notificações dos dispositivos para os múltiplos clientes interessados nos mesmos dados. Esta solução é capaz de fazer armazenamento e fusão de mensagens, atingindo melhor desempenho em termos de energia consumida.

O artigo de [Mišic et al. 2018] apresenta uma arquitetura para suportar a comunicação periódica em redes IoT. Na arquitetura proposta há também o uso de proxies, o qual é responsável por manter os dados atualizados de acordo com as preferências dos clientes, além de fazer o controle de congestionamento.

De um modo geral, existem muitos trabalhos que abordam o cenário IoT onde clientes registram quais são seus interesses em termos de dados e suas preferências de comunicação periódica. As soluções propostas resolvem problemas relacionados à rede IoT. Por exemplo, o alto nível de congestionamento da rede ao ter muitos clientes interessados. Porém, existe uma falta de trabalhos que suporte e facilite esse tipo de comunicação no nível da plataforma IoT e promova uma solução integrada com o código embarcado nos dispositivos.

3. *Framework* OBSERVICE

O framework OBSERVICE é proposto para gerenciar a comunicação dados assíncrona entre múltiplos clientes e servidores IoT, permitindo aos cliente definir e alterar suas preferências em termos de configuração de comunicação. A seguir estão apresentados aspectos relevantes da especificação, arquitetura e implementação do OBSERVICE. Na Seção 3.1, as aplicações e cenários alvos são apresentados com o objetivo de estabelecer os cenários práticos para os quais a proposta é pensada e também fornecer uma visão geral de como o framework proposto se insere em ambientes IoT. A Seção 3.2 apresenta os requisitos que o OBSERVICE deve cumprir. A Seção 3.3 descreve a arquitetura geral do framework proposto e a Seção 3.4 apresentação os detalhes da implementação do OBSERVICE.

3.1. Aplicações Alvo

Em diversas aplicações IoT existe a necessidade de monitorar periodicamente dados aferidos por dispositivos IoT que estão em rede. Neste tipo de aplicação, o usuário está interessado em acompanhar, de forma constante, um determinado tipo de dado produzido e comunicado pela rede.

Em uma aplicação de Indústria 4.0, o usuário pode estar interessado em receber dados referentes ao nível de CO_2 em uma parte da fábrica. Para isso basta que uma mensagem seja enviada aos nós correspondentes informando que existe um observador para o tipo de dado CO_2 com preferência de recebimento de dados a cada 60 segundos. O mesmo usuário pode estar interessado em observar outra área da fábrica, especificamente o nível de pressão de tanques de armazenamento a cada 10 segundos.

Nesta aplicação ilustrativa de Indústria 4.0, o usuário pode ser um funcionário responsável por fiscalizar alguns indicadores do ambiente e dos equipamentos. Fica claro neste exemplo que o usuário deve ser capaz de se tornar observador de vários tipos de dados, envolvendo diferentes grupos de dispositivos IoT. Além disso, a comunicação dos dados deve ser feita de forma assíncrona e que seja evitado que uma mensagem de requisição seja enviada aos mesmos dispositivos de forma repetida a cada novo intervalo de tempo. O envio repetido de mensagens de requisição gera um consumo excessivo de recursos da rede e dos dispositivos. Dessa forma, estes envios podem ser suprimidos.

3.2. Definições e Análises de Requisitos

Dado as aplicações alvo descritas na seção 3.1, as seguintes definições são importantes para o restante do trabalho:

Definição 1. Um recurso de dados é um tipo de dado produzido por um dispositivo IoT e que está disponível para monitoramento.
Definição 2. Um observador é uma entidade que tem interesse em um recurso de dados e tem preferências de como essa comunicação vai ser executada.
Definição 3. Uma notificação é uma mensagem assíncrona enviada de um dispositivos IoT para um observador.

Baseado nas aplicações alvo, o framework OBSERVICE deve satisfazer os seguintes requisitos:

A. Gerenciar os observadores simultâneos de múltiplos recursos de dados envolvendo vários grupos de dispositivos;
B. Satisfazer as preferências dos observadores e suportar as observações de forma assíncrona.

O requisito A trata de permitir que haja vários observadores para vários recursos de dados produzidos por múltiplos dispositivos IoT. Isto significa que o framework deve criar, armazenar, editar e finalizar os interesses dos observadores e suas preferências, evitando os conflitos que possam surgir entre as preferências dos vários observadores.

O requisito B estabelece que o framework deve ser capaz de se comunicar com os dispositivos IoT de forma assíncrona, satisfazendo as preferências dos observadores. Para isso, o framework deve manter um canal de comunicação aberto para recebimento dos dados. Por outro lado, os dispositivos IoT devem ser capazes de executar os interesses e preferências estabelecidas pelo observador.

3.3. Arquitetura OBSERVICE

A arquitetura do framework OBSERVICE foi especificada com o objetivo de satisfazer os requisitos analisados na seção 3.2. A Figura 1 apresenta os componentes dessa arquitetura, os quais são: Interface, Gerenciador de observações, Californium e Cliente CoAP.

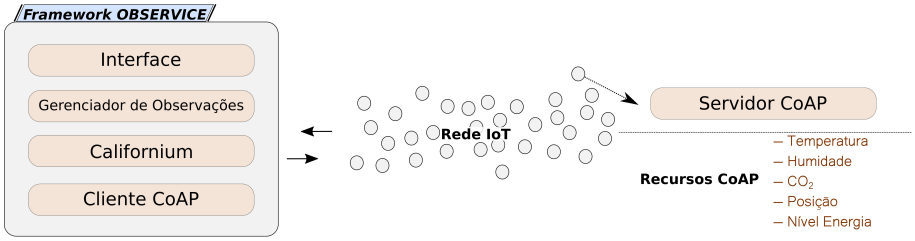

Figura 1. Arquitetura OBSERVICE

A Interface é responsável por interagir com o usuário, isto inclui capturar dados sobre os interesses e preferências do observador e apresentar os dados comunicados pelos dispositivos IoT.

O gerenciador de Observações é o componente responsável em criar, editar e armazenar os interesses e preferências dos observadores. Isto inclui a emissão de mensagens para os dispositivos IoT relacionados ao interesse do observador. Este componente também é responsável por gerenciar os dados obtidos pelas observações recebidas, entregando os dados para o observador que iniciou o monitoramento.

O OBSERVICE faz uso de funcionalidades da biblioteca Californium, a qual implementa protocolos de comunicação IoT. O principal protocolo disponibilizado pelo Californium é o CoAP (Constrained Application Protocol). O CoAP é um protocolo de camada de aplicação similar ao Protocolo de Transferência de Hipertexto (HTTP). O protocolo CoAP é RESTful e adaptado para ser executado em dispositivos com restrição de energia, memória e processamento. O cliente CoAP é executado junto com as funcionalidades do framework OBSERVICE, enquanto o servidor CoAP está nos dispositivos IoT. Uma requisição enviada por um cliente solicita uma ação em um recurso, identificado por uma URI, em um servidor CoAP.

3.4. Implementação

A implementação do framework OBSERVICE foi feita usando a linguagem Java, assim como o Californium, e utiliza threads para permitir comunicações simultâneas. No OBSERVICE, existem dois níveis de threads: um para o recurso observado e outro para as observações recebidas. Quando um recurso passa a ser observado, uma thread, denominada thread-grupo, é criada. Na thread-grupo, há a criação de um segundo nível de threads, denominado de thread-nó. Existe a criação de uma thread-nó para cada nó envolvido por aquele recurso. Isto deve-se a razão de que a observação de um recurso envolve a coleta de dados de um grupo de dispositivos.

Cada thread-nó executa um código para iniciar a observação, enviando uma requisição CoAP informando a URI a ser observada e as preferências do observador para a comunicação dos dados. Após a mensagem de início de observação, a thread-nó fica à espera da chegada assíncrona de mensagens de notificação. Para cessar de observar um recurso, basta finalizar a thread responsável por observar aquele recurso, isso gerará uma mensagem ao nó informando o fim da observação.

A Figura 2 apresenta a interação entre os componentes OBSERVICE na criação de um novo observador. Nota-se que esse processo envolve a criação de threads e a gestão das informações recebidas. Um único observador pode ser responsável pela emissão de uma grande quantidade de mensagens CoAP, visto que um observador pode estar interessado em receber notificações de muitos dispositivos IoT. É importante ressaltar também que, uma vez que as mensagens CoAP são enviadas para os dispositivos IoT, não é necessário mais enviar mensagens de requisição de dados. Os dispositivos passam a enviar as notificações de acordo com as preferências informadas.

A Figura 3 apresenta um fluxograma especificado para descrever o funcionamento da interface. Este fluxograma começa com o usuário inserindo a URL do roteador de borda. Este roteador disponibiliza informações sobre as rotas ativas na rede IoT e quais dispositivos estão conectados. Exibir uma lista de dispositivos conectados é útil para o usuário criar os interesses do observador. Ao clicar em um endereço IP lista, a interface exibe os recursos que podem ser observados naquele dispositivo. Caso o usuário tenha interesse em observar aquele recurso, basta clicar em adicionar. Por fim, o usuário clica em ObsGrupo para criar um observador e iniciar a observação. A interface exibe os dados coletados e atualiza assim que novas mensagens são recebidas.

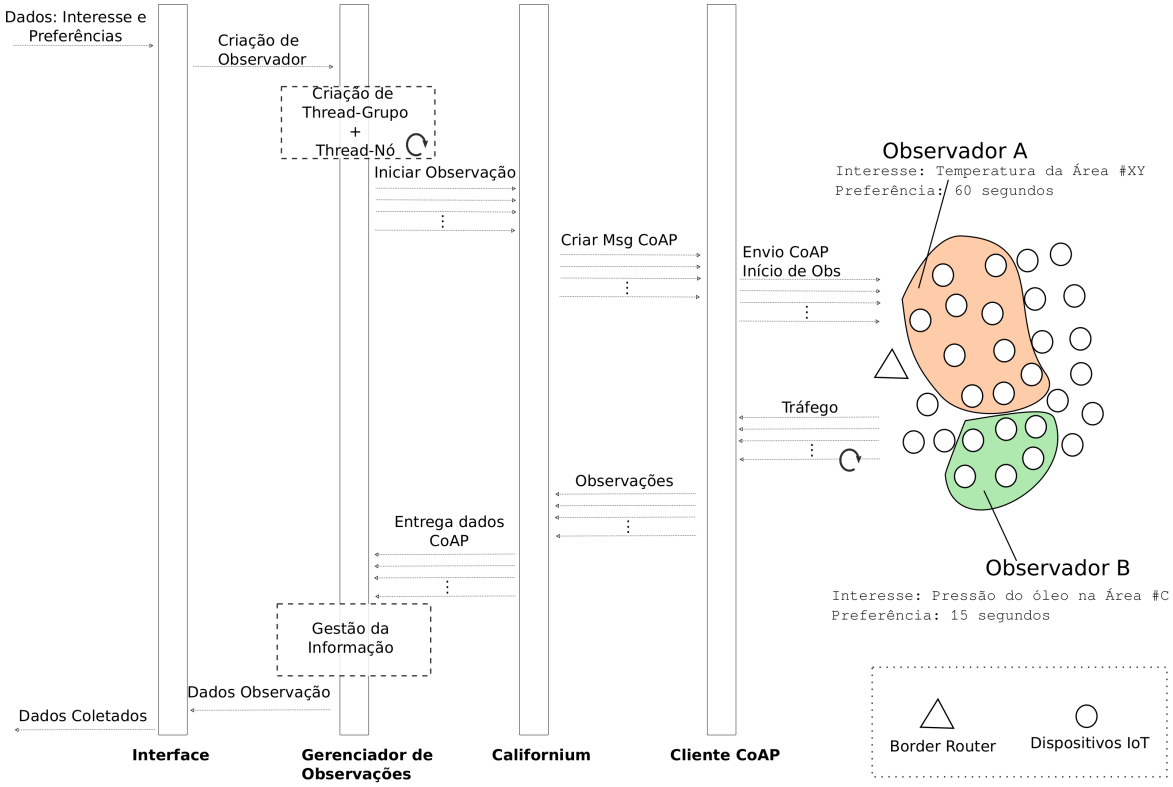

Figura 2. Interação entre os componentes OBSERVICE desde a criação de um observador.

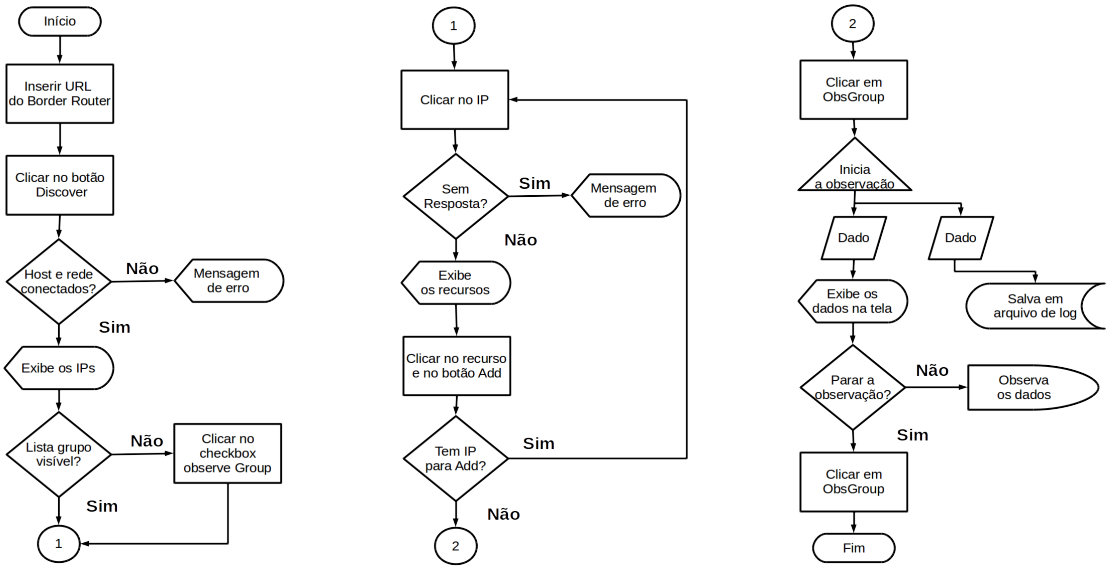

Figura 3. Fluxograma da função *OBSERVE*.

Como pode ser vista na Figura 4, a ferramenta desenvolvida possibilita visualizar quais recursos estão disponíveis e observar múltiplos recursos. Inicialmente, o usuário deve inserir e usar a funcionalidade Discover colocando o IP do roteador de borda da rede, como indicado no número 1 e 2 da Figura 4. Caso a funcionalidade *Discover* seja executada com sucesso, surgirão informações de Neighbors, Resource Mote e Routes, nos campos 5, 6 e 7 da Figura 4. Essas informações permitirão ao usuário selecionar

vários recursos e agrupá-los usando o botão “Add” (9). Ao fazer isso, todos os recursos selecionados serão apresentados no campo 8 da Figura 4. O conteúdo das notificações recebidas são exibidas no campo 10. Os botões 3 e 4 são usados para fazer o GET e observar um único recurso, respectivamente.

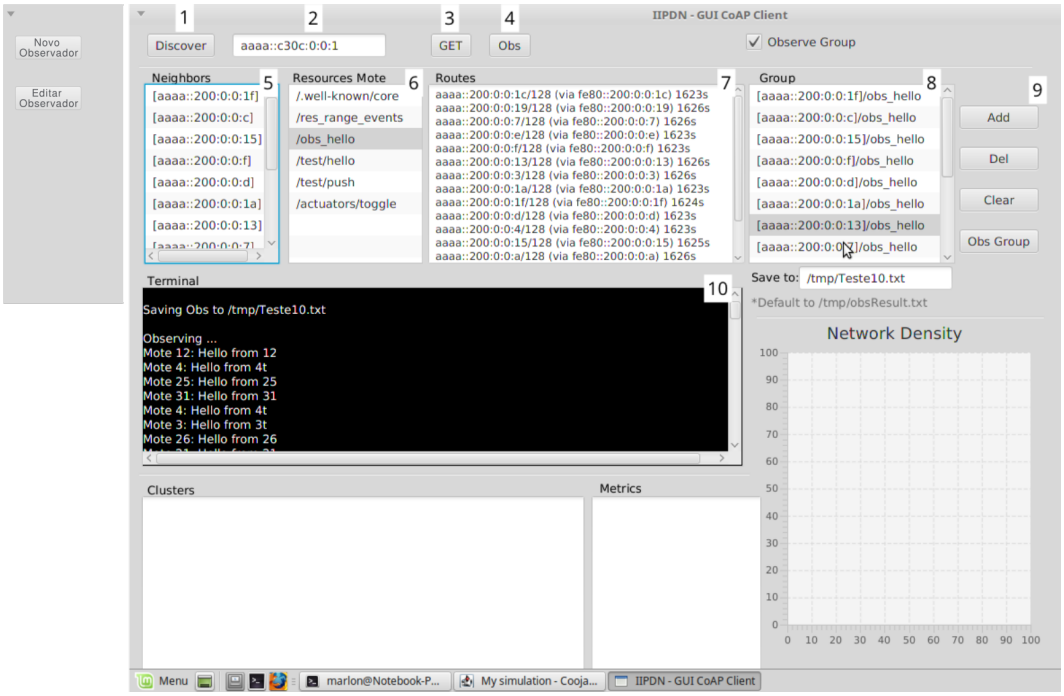

Figura 4. GUI da Observação de Múltiplos IPs.

O código [1] do framework OBSERVICE está disponibilizado para fins acadêmicos como um software de código aberto.

4. Validação

Esta seção apresenta a validação do framework OBSERVICE. A Seção 4.1 apresenta o cenário e os objetivos dos testes de validação. A Seção 4.2 descreve os resultados obtidos e a Seção 4.3 discute os resultados.

4.1. Cenário de Teste e Métricas de Avaliação

Para esta avaliação, criou-se uma rede de 30 nodos que são servidores CoAP. Além disso, nesta rede há um roteador de borda que atua na ligação entre o framework OBSERVICE e a rede de dispositivos IoT. Todos estes nodos foram emulados e a comunicação entre eles foi simulada usando a ferramenta Cooja. Para isso, todos os nodos foram embarcados com o sistema operacional embarcado Contiki.

Cada servidor CoAP possui um recurso observável. Assim que a simulação da rede era iniciada, o OBSERVICE foi programado para carregar todos os recursos observáveis que estavam disponíveis na rede. Logo após início da simulação, o OBSERVICE executava o código responsável por observar múltiplos recursos e a criação de vários observadores. Nesta fase, o OBSERVICE envia uma requisição CoAP para todos os dispositivos a serem observados. Depois dessa fase, a aferição dos testes iniciava. Cada teste de validação teve duração de dez minutos, com 5 observadores, um intervalo de notificação de 20 segundos. Os protocolos IEEE 802.15.4 e RPL foram utilizados em todos os servidores CoAP.

[1] https://github.com/MarlonWSantos/DENX

As seguintes medidas foram aferidas durante o período de observações com o intuito de validar as funcionalidades propostas. A Tabela 1 apresenta as métricas e objetivos das métricas usadas na validação.

Tabela 1. Métricas e Objetivos da Validação.

Métrica	Objetivo
Número de mensagens entregues e perdidas ao longo do tempo	Analisar a capacidade de manter as observações ativas longo do teste
Número de requisições CoAP depois do início da observação	Verificar o número de reqisições necessárias para manter a observação ativa
Número total de mensagens entregue por nodo	Validar a composição do tráfego recebido no destino

Como pode ser observado na Tabela 1, o intuito do teste de validação não é apenas verificar se o OBSERVICE é capaz de iniciar a observação dos recursos CoAP. Entre os objetivos do teste é saber se o OBSERVICE é capaz de observar múltiplos recursos, entregando as notificações a múltiplos observadores ao longo do tempo.

4.2. Resultados Obtidos

A Figura 5 apresenta o número de mensagens entregues em cada minuto de teste. É possível observar também o número de mensagens CoAP perdidas e o número extra de requisições CoAP. A rede gera por minuto, em média, 90 notificações CoAP. Na prática, de acordo com a Figura 5, o número de mensagens entregues é em média 81,6. É possível notar também que houve em média 5 envios de requisições extra CoAP, para reiniciar as observações depois de instabilidade nas rotas.

A perda de mensagens CoAP ocorre por duas razões principais. No primeiro caso, a mensagem enviada pelo nó observado pode ser perdida devido à baixa confiabilidade da comunicação característica dos protocolos utilizados. No segundo caso, a conectividade da rede pode estar instável e fazer com que as notificações do servidor CoAP sejam perdidas. A instabilidade das rotas pode gerar perda de comunicação. Neste caso, o OBSERVICE faz um novo envio de requisição CoAP para reiniciar as observações. A porcentagem média de perda de notificações CoAP foi de 9,3%.

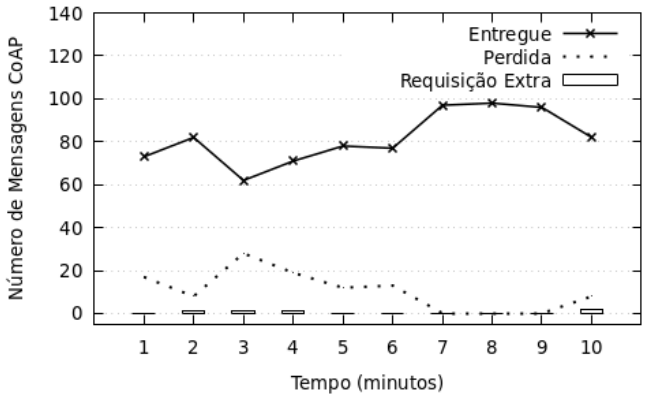

Figura 5. Gráfico de mensagens enviadas e de falhas.

Outro ponto relevante na validação do framework proposto é qual a composição do tráfego entregue, isto pode ser visto na Figura 6. Esta figura ilustra o número de

mensagens que cada nó entregou aos observadores. É possível notar que todos os nós participaram com proporções semelhantes. Grande parte dos nodos teve participação muito próxima de 30 notificações. O menor valor corresponde aos nodos 13 e 16. O maior valor corresponde aos nodos 1, 2, 19, 25, 26 e 27.

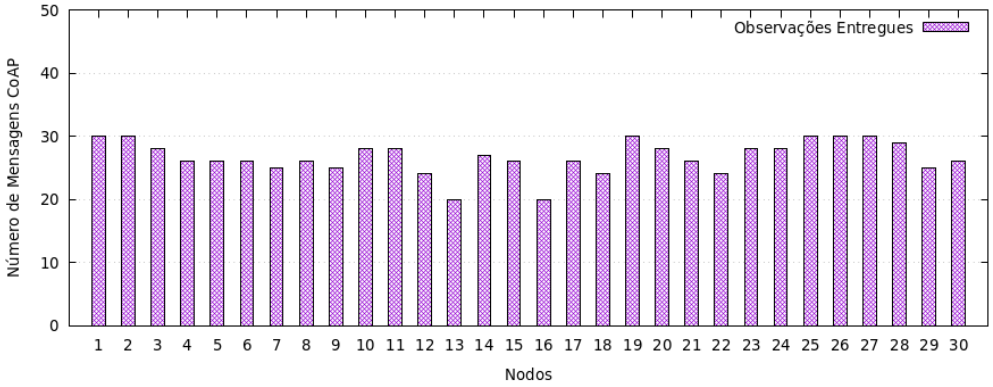

Figura 6. Gráfico de mensagens enviadas por cada nodo.

Por fim, a Figura 7 apresenta a porcentagem acumulativa ao longo do tempo do tráfego recebido. De um modo geral, é possível notar que o recebimento das observações é recebido pelos observadores de maneira constante, sem grandes perdas no fluxo cumulativo de informações recebidas. Em uma amostra de dez minutos a rede entregou 4,08 MBytes de informação, sendo perdido 9,3% do total gerado.

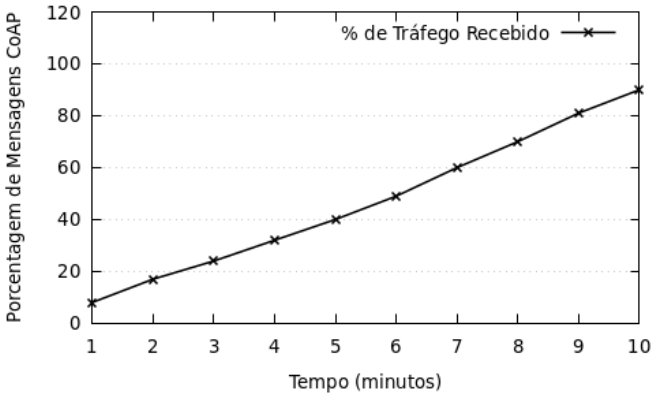

Figura 7. Porcentagem acumulativa de dados recebidos.

4.3. Discussão

Os resultados apresentados demonstram a capacidade do framework proposto em manter as observações ativas ao longo do tempo, mesmo que a rede enfrente problemas de perda de conexão e mensagens. Além disso, os resultados mostram que o OBSERVICE foi capaz de gerar uma participação equilibrada entre os 30 nodos. Isto é importante porque ter um nodo que participasse pouco do tráfego gerado seria um indicativo de problemas de comunicação ou falha na gestão dos observadores.

A taxa de perda de mensagens, em média 9,3%, é aceitável para o cenário utilizado, onde o meio sem fio está sujeito a colisões na transmissão. Com essa taxa de perda, uma projeção para 24 horas, a rede teria entregue cerca de 573 MBytes de informação.

Um dos benefícios do OBSERVICE é a redução do número de requisições CoAP. Em uma projeção de 24 horas, o número de requisições extra CoAP seria algo em torno de 720. Em comparação à uma comunicação sem o uso do OBSERVICE, seriam necessários enviar 129600 requisições CoAP.

5. Considerações Finais

As plataformas IoT são objetos centrais para o sucesso de IoT. Plataformas robustas permitem o desenvolvimento de serviços e aplicações de forma rápida e eficiente. Neste contexto, a comunicação de alto rendimento torna-se uma peça chave para o bom desempenho das plataformas IoT. Para isso, as plataformas devem estar adaptadas ao tipo de comunicação executada e aos recursos disponibilizados pelos dispositivos IoT.

Este artigo propõe um framework chamado OBSERVICE que visa gerenciar a comunicação assíncrona em ambientes IoT que possuam múltiplos interessandos em receber dados de várias fontes. O framework proposto permite que várias entidades recebam dados de várias fontes de dados, segundo configurações de comunicações estabelecidas e preservando o consumo de recursos escassos dos dispositivos. A implementação do framework teve como prioridade o reuso de código, ferramentas de código aberto e a integração com código real embarcado em dispositivos IoT. Os resultados dos testes de validação mostram que o OBSERVICE é capaz de criar e manter comunicação periódica envolvendo múltiplas entidades, com baixo consumo de recursos, mesmo em situações adversas de rede onde há perda de conexão.

Referências

Botta, A., De Donato, W., Persico, V., and Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey. *Future generation computer systems*, 56:684–700.

Iglesias-Urkia, M., Casado-Mansilla, D., Mayer, S., and Urbieta, A. (2018). Enhanced publish/subscribe in coap: describing advanced subscription mechanisms for the observe extension. In *Proceedings of the 8th International Conference on the Internet of Things*, pages 1–8.

Ishaq, I., Hoebeke, J., Moerman, I., and Demeester, P. (2016). Observing coap groups efficiently. *Ad Hoc Networks*, 37:368–388.

Ismail, A. A., Hamza, H. S., and Kotb, A. M. (2018). Performance evaluation of open source iot platforms. In *2018 IEEE global conference on internet of things (GCIoT)*, pages 1–5. IEEE.

Kovatsch, M., Lanter, M., and Shelby, Z. (2014). Californium: Scalable cloud services for the internet of things with coap. In *2014 International Conference on the Internet of Things (IOT)*, pages 1–6. IEEE.

Lai, W.-K., Wang, Y.-C., and Lin, S.-Y. (2020). Efficient scheduling, caching, and merging of notifications to save message costs in iot networks using coap. *IEEE Internet of Things Journal*, 8(2):1016–1029.

Mišic, J., Ali, M. Z., and Mišic, V. B. (2018). Architecture for iot domain with coap observe feature. *IEEE Internet of Things Journal*, 5(2):1196–1205.

Suwannapong, C. and Khunboa, C. (2019). Congestion control in coap observe group communication. *Sensors*, 19(15):3433.

Suwannapong, C. and Khunboa, C. (2021). Encoco-red: Enhanced congestion control mechanism for coap observe group communication. *Ad Hoc Networks*, 112:102377.