COSGP-IoT-SG: Uma Proposta de Integração para Smart Grid em Dispositivos com Capacidade Limitada

Gabriel de Mello, Luís A. Silva, Valderi R.Q. Leithardt

[1]Universidade do Vale do Itajaí (UNIVALI), Itajaí, Brasil, 88302-901
Laboratório de Sistemas Embarcados e Distribuídos

{gabrieldemello, luis.silva}@edu.univali.br, valderi@univali.br

Resumo. *Neste artigo é proposto e implementado um modelo de gateway para a Internet das Coisas (Internet of Things - IoT), considerando os modelos de transporte e comunicacão CoAP (Constrained Application Protocol) e OSGP (Open Smart Grid Protocol). Por se tratarem de protocolos que buscam cobrir as principais deficiências em sistemas para Cidades Inteligentes (Smart Cities - SC), inferiu-se necessária uma integracão rápida e eficiente entre ambos.*

1. Introdução

A IoT está cada vez mais presente no cotidiano das pessoas, e sua popularização possibilitou o surgimento de redes inteligentes. Um dos principais pilares das redes inteligentes é a utilização de sensores e atuadores inteligentes para o gerenciamento de contexto e a descoberta de serviços [Perera et al. 2014]. Com isso, mais dispositivos são conectados à rede, ocasionando a necessidade de estudos na área de intercompatibilidade. As SCs são exemplos de sistmas inteligentes em ascensão e que dependem da utilização de diversos sensores.

As SCs são constituídas por sistemas inteligentes que trabalham em conjunto, com o propósito de fornecer soluções eficientes para a distribuição de recursos de uma cidade. Um exemplo desses sistemas inteligentes são as SGs, que possibilitam a coleta de uma grande quantidade de dados referentes ao uso de energia elétrica, fornecendo maior controle do consumo de energia aos consumidores e prestadores de serviço [ETSI 2012]. Em ambientes diversos, os dados podem ser obtidos através de sensores dispostos dentro do contexto de inúmeras aplicações, gerando um grande fluxo de informações de origens distintas. Sendo assim, a transmissão dos dados pode ser realizada atráves de modelos como o CoAP e o OSGP.

O objetivo do OSGP é permitir a coleta de dados em ambientes diversos, possibilitando o uso eficiente de energia elétrica dos consumidores e prestadores de serviço, provendo um serviço automatizado, sem a necessidade de mão de obra humana [Alliance 2015]. Assim, então surge o desafio da interoperabilidade entre equipamentos, fornecendo suporte a dispositivos (e.g. sensores) com recursos limitados, como memória, processamento, armazenamento e energia. Para tanto, o modelo CoAP tem este propósito. O CoAP é um protocolo leve, de transporte e comunicação M2M (*machine-to-machine*), que opera nos moldes de requisição/resposta do HTTP [Iglesias-Urkia et al. 2017]. Com base no contexto apresentado, surge a necessidade da integração dos protocolos OSGP e CoAP.

Uma rede que retrate a IoT é formada por dispositivos heterogêneos, demandando uma aplicação que consiga relacionar todos eles, visando suas limitações es-

pecíficas e modelos de comunicação como os que vigoram no atual estado-da-arte. Dessa forma, o modelo proposto consiste em mapear os pacotes enviados por ambos os lados da comunicação entre os protocolos CoAP e OSGP. As estruturas definidas pelas especificações dos protocolos foram mantidas, tornando esta implementação escalável ao desenvolvimento e futuras implementações em diversos ambientes. Sendo assim, o trabalho realizado tem como seu escopo a integração por atribuição de requisições/respostas análogas entre os modelos escolhidos, desenvolvendo as lógicas de tradução dos dados contidos nos cabeçalhos, opções e payloads dos pacotes.

2. Gateway

Com base no trabalho realizado em [Shin et al. 2017], o gateway COSGP-IoT-SG foi desenvolvido utilizando os métodos e recursos fornecidos pela biblioteca libcoap2, disponível na linguagem C. Neste trabalho, o servidor é o responsável por atribuir as devidas lógicas de funcionamento dos métodos GET, PUT, POST e DELETE definidos pela própria especificação do protocolo CoAP, que segue a arquitetura CoRE (Constrained Restful Environment). São levadas em consideração as definições Write/Read e Full/Partial do modelo OSGP, além do suporte aos Pending Events Descriptors (PEDs), que agem como os ativadores das Pending Tables, essenciais para manutenções e escalabilidade de forma geral.

Assim, o gateway funciona de modo que, na chamada individual de cada requisição e resposta, um método correspondente de tradução, similar ao apresentado na Figura 1, é invocado, mapeando os dados ao extrair certos atributos, como: identificador da mensagem, tipo de requisição/resposta, tamanho do pacote e token. O repositório de dados do servidor CoAP são recursos que podem ser adicionados no contexto da aplicação, por onde os métodos são chamados.

Os testes realizados visaram comparar o tempo de processamento de cada etapa de tradução dos métodos. Dois microcontroladores de capacidades computacionais distintas foram escolhidos propositalmente para simular os efeitos que limitações no hardware de um dispositivo podem ter em uma aplicação, e a maneira como isso afeta o desempenho geral do processo, assim possibilitando a comparação e comprovando a eficácia da implementação. Este trabalho não contou com a utilização de um cliente OSGP, que levaria à necessidade de simular a comunicação realizada através de uma rede elétrica, por não ser o objetivo da atual fase do projeto. Por conta disso, o cliente OSGP foi substituído por estruturas cujos componentes foram definidos de forma a também seguir as especificações do protocolo. Para as requisições CoAP, realizamos a implementação de um cliente em Python, devido à facilidade no uso das bibliotecas disponíveis para a plataforma.

O ambiente de testes consistiu na implementação, em hardware, do código desenvolvido. As plataformas de desenvolvimento escolhidas, com foco nas implementações de integração elaboradas no trabalho, se dividiram entre clientes e servidor. Para os clientes CoAP optou-se por utilizar o hardware Raspberry PI 3 Model B v1.2, possuindo 1GB de RAM e uma CPU ARM Cortex A53 com quatro núcleos operando a 1,2GHz, e o BeagleBone Black, que possui 512MB de RAM, uma CPU ARM Cortex A8 AM3358 com um núcleo operando a 1GHz e rodando o sistema Debian 9.4. O servidor, contendo a implementação COSGP-IoT-SG, contou com o sistema operacional Ubuntu 16.04 LTS 64 bits, processador AMD-FX 6300 com seis núcleos de 3,5 GHz e 8 GB de memória.

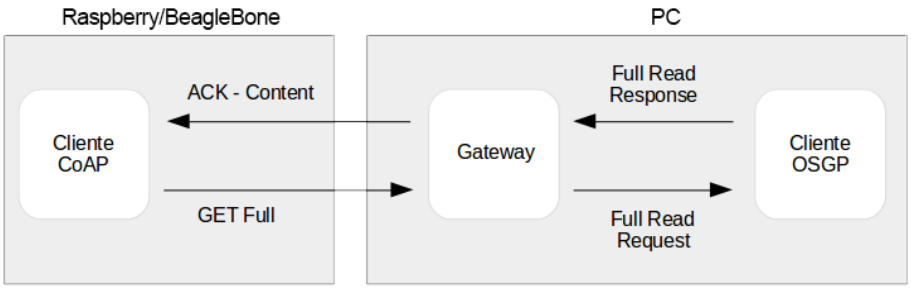

Figura 1. Diagrama demonstrando o funcionamento do método GET Full em caso de comunicação bem sucedida.

3. Resultados Preliminares

As medições, que se encontram dispostas em milissegundos na Tabela 1 e Tabela 2, foram realizadas com base no tempo de resposta da requisição dos clientes e com os recursos provenientes das bibliotecas padrões da linguagem Python. A média da latência existente entre a comunicação dos dispositivos foi de 10,761 ms no Raspberry e 0,356 ms no BeagleBone Black para cada pacote enviado ou recebido. Apenas os métodos PUT e GET Fulls foram testados, considerando a complexidade da descrição dos resultados. Os payloads dos pacotes CoAP contaram com arquivos JSON de tamanhos variando entre 1024, 2048, 4096 e 8192 bytes, que foram divididos em pacotes de 1024 bytes por conta da especificação do protocolo CoAP. Cada tamanho de pacote foi testado quatro vezes em cada método, totalizando uma amostra de 120 pacotes enviados ou recebidos.

A Tabela 1 representa o tempo de resposta de um PUT Full para cada tamanho de pacote. A tendência do crescimento linear é exposta pelo aumento no número de requisições feitas, considerando que a especificação CoAP conta com, no máximo, 1024 bytes para cada envio ou resposta. É importante ressaltar as diferenças de latência entre os dispositivos, onde cada tempo de requisição feita pelo Raspberry conta com cerca de 60% de interferência pela latência de fluxo da rede, enquanto o BeagleBone Black contou apenas com 1% de interferência.

Tabela 1. PUT Full

Hardware	1024	2048	4096	8192
Raspberry	17,749	33,850	70,206	133,329
BeagleBone	38,336	55,705	97,612	177,95

A relação para o método GET Full é ilustrada na Tabela 2. A diferença no tempo entre os dois métodos testados é causada pelas funções de tradução, onde o processamento é maior no caso da necessidade de se adicionar o conteúdo do payload em um dos recursos do servidor para o monitoramento da chegada dos pacotes.

A eficácia dos testes julgou-se comprovada em razão à quantidade e tamanho dos dados utilizados na amostragem, que denotam excelente comportamento no caso de um grande tráfego de dados.

Tabela 2. GET Full

Hardware	1024	2048	4096	8192
Raspberry	32,224	55,581	113,245	233,298
BeagleBone	54,639	113,388	227,196	449,753

4. Conclusão

Este trabalho visou suprir a demanda por interoperabilidade entre dispositivos de capacidade limitada no âmbito da IoT. Os resultados se mostraram satisfatórios dentro do contexto aplicado e da amostragem utilizada. Apesar da aplicação ter sido bem explorada, a falta de simulações realizadas através de uma rede elétrica real pode limitar o escopo deste trabalho.

Sendo assim, em trabalhos futuros destacam-se as seguintes necessidades: simular comunicações e traduções feitas por meio de uma rede elétrica, com a finalidade de avaliar a solução proposta; testar em dispositivos com capacidades computacionais ainda mais baixas; comparar o desempenho com outros protocolos, implementados em diferentes linguagens e ambientes; e aferir resultados obtidos.

Agradecimento

Os resultados deste trabalho foram possíveis devido ao apoio recebido através do projeto de pesquisa e extensão da Universidade do Vale do Itajaí através do Artigo 171 - Projetos de Pesquisa. Também agradecemos o uso da infraestrutura fornecida pelo Laboratório de Sistemas Integrados e Distribuídos (LEDS) da Univali.

Referências

Alliance, O. (2015). The open smart grid protocol. *Available in http://www. osgp. org/*. Acessado em Dezembro de 2018.

ETSI (2012). GS OSG v1.1.1 (2012-01) - Open Smart Grid Protocol (OSGP). Acessado em Dezembro de 2018.

Iglesias-Urkia, M., Orive, A., and Urbieta, A. (2017). Analysis of CoAP implementations for industrial internet of things: A survey. *Procedia Computer Science*, 109:188 – 195. doi: 10.1016/j.procs.2017.05.323.

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials*, 16(1):414–454.

Shin, I.-J., Song, B.-K., and Eom, D.-S. (2017). International electronical committee (iec) 61850 mapping with constrained application protocol (coap) in smart grids based european telecommunications standard institute machine-to-machine (m2m) environment. *Energies*, 10(3). doi:10.3390/en10030393.