IoT Redirector: um redirecionador para gerenciamento da heterogeneidade de dados em aplicações IoT

Renato Oliveira[1,2], Carlos A. Kamienski[1]

[1]Universidade Federal do ABC (UFABC)
Brasil

[2]Instituto Federal de São Paulo (IFSP)
Brasil

renato.bueno@ifsp.edu.br, cak@ufabc.edu.br

Abstract. *The amount of data grows annually and most of it comes from IoT devices. However, this data can be generated and transmitted in different formats. Data can be generated, by a sensor, as a temperature measurement and transformed, for example, into JSON, XML, HTML, UltraLight (UL), text, base64 format. This is a problem for applications as the heterogeneity of data makes interoperability in IoT systems difficult. The purpose of this work is to deal with the proposed formats, develop a solution (module or generic enabler) capable of identifying the data format and redirecting them to an IoT agent responsible for its management. The proposed architecture used FIWARE and in the tests performed the solution identified and redirected messages with a high degree of accuracy.*

Resumo. *A quantidade de dados cresce anualmente e a maioria deles advêm de dispositivos IoT. No entanto, esses dados podem ser gerados e transmitidos em formatos diferentes. Um dado pode ser gerado, por um sensor, como uma medida de temperatura e transformado, por exemplo, em formato JSON, XML, HTML, UltraLight (UL), texto, base64. Isso é um problema para as aplicações, pois a heterogeneidade dos dados dificulta a interoperabilidade em sistemas IoT. A proposta deste trabalho é tratar os formatos mencionados, desenvolver uma solução (módulo ou generic enabler) capaz de identificar esses formatos dos dados e redirecioná-los para um agente IoT responsável pelo seu gerenciamento. A arquitetura proposta utilizou o FIWARE e nos testes realizados a solução identificou e redirecionou as mensagens com elevado grau de acurácia.*

1. Introdução

A quantidade de dados cresce anualmente e eles são oriundos de diversas fontes como dispositivos eletrônicos, sensores (temperatura e humidade, movimento, luminosidade). Um comparativo no estudo (Arun Solanki, Adarsh Kumar, 2021) indica que a quantidade de dispositivos inteligentes conectados cresce de uma maneira exponencial muito mais acentuada que a quantidade de pessoas no mundo.

Atualmente, em 2022, há 16,4 bilhões de dispositivos IoT conectados e há expectativa de crescimento para 30,9 bilhões de dispositivos em 2025. Desde 2020, há mais conexões IoT (carros conectados, dispositivos de casas inteligentes, equipamentos industriais conectados) do que conexões não IoT (smartphones, laptops e computadores). (LASSE LUETH, 2020)

Uma das características dos dados é que seu conteúdo pode ser de vários tipos como vídeos, arquivos compartilhados, documentos de texto e PDF, assim como podem ser representados em formatos distintos. O protocolo HTTP possui no seu cabeçalho o tipo de conteúdo (*content-type*) usado para indicar o tipo de mídia do recurso. Um dos desafios da IoT é a utilização de padrões para manipulação, processamento e armazenamento de dados dos sensores (BANAFA, 2017). Dados gerados por sensores podem estar, por exemplo, em formato XML, JSON, HTML, UltraLight (UL), texto e base64. Para que eles possam ser compreendidos pela aplicação que recebe esse dado existe a necessidade de padronização. A interoperabilidade entre os dispositivos IoT fica limitada quando não são adotados padrões (Khatoon e Ahmed, 2021). Existem padrões utilizados para comunicação entre dispositivos IoT, por exemplo, o protocolo MQTT. Esse protocolo utiliza dois conceitos que seriam a arquitetura *publish/subscribe* e o conceito de tópicos e subscrições (Soni, Dipa & Makwana, 2017). O MQTT trabalha com uma arquitetura que pode ser dividida em cliente e *broker*. O cliente publica as mensagens para os usuários interessados e o *broker* recebe todas as mensagens que são publicadas.

A API REST é um conjunto de regras de arquitetura (RED HAT, 2020) que permite utilizar métodos como POST para envio de dados, PUT para atualização, GET para requisição e DELETE para remoção de dados (Al-Fuqaha et al., 2015). Nesse artigo a API REST é utilizada em conjunto com o protocolo MQTT para redirecionamento dos dados recebidos do *broker*. Os dados chegam ao *broker* através do protocolo MQTT e são redirecionados através da API REST. Isso permite maior flexibilidade para integração de dados entre os sistemas.

Para que os dados possam ser processados existem elementos responsáveis por essa tarefa. Na plataforma FIWARE[1] os agentes IoT são responsáveis por receber os dados oriundos dos sensores e eles atuam como se fossem *gateways* ou tradutores entre os protocolos que os dispositivos usam para enviar ou receber informações e uma linguagem comum é utilizada em toda a plataforma FIWARE: o NGSI (FOX, 2021). Existem agentes IoT que trabalham com diferentes formatos de dados como, por exemplo, JSON, UL e LoRaWAN.

Um agente IoT central que recebe primeiramente esses dados, pode interpretá-los e dizer para onde deve ser redirecionado baseado no seu *payload* ou conteúdo de sua mensagem. Para isso, ele precisa subscrever a um tópico do *broker* que recebe todos os dados de diversos tipos de sensores. A solução proposta faz o reconhecimento do seu *payload*, analisa o conteúdo da mensagem e decide para qual agente responsável irá redirecionar o dado. A Figura 1 ilustra a API de redirecionamento pelos formatos de dados.

Esse artigo propõe um módulo chamado de IoT Redirector que realiza o redirecionamento de mensagens dependendo do formato dos dados enviados pelos sensores. As mensagens foram geradas em diversos formatos através de dois geradores de mensagens. Um gerador envia mensagens no formato base64 com a tecnologia LoRa e o outro nos formatos XML, HTML, JSON, UL, TEROS 12[2] e Indefinido. Com a geração de mensagens foi possível realizar testes de acurácia no redirecionamento dos dados e de desempenho como consumo de CPU, memória e vazão.

[1] https://www.fiware.org/developers/

[2] https://www.metergroup.com/en/meter-environment/products/teros-12-soil-moisture-sensor

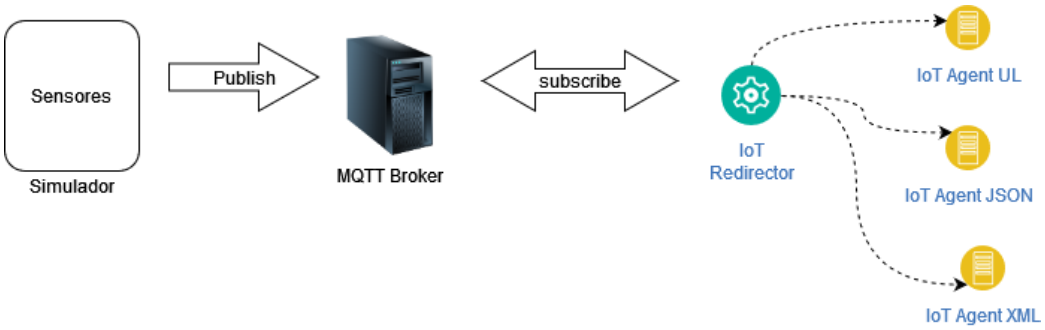

Figura 1 – IoT Redirector

No restante desse artigo, a Seção 2 trata da motivação e trabalhos relacionados, enquanto que a Seção 3 mostra a arquitetura do IoT Redirector. A Seção 4 apresenta a metodologia e os cenários de teste e na Seção 5 são apresentados os resultados. A Seção 6 faz uma discussão dos principais resultados enquanto que conclusões e trabalhos futuros são apresentados na Seção 7.

2. Motivação e trabalhos relacionados

As aplicações que trabalham com IoT precisam identificar o formato dos dados para poderem processá-los. Uma aplicação que consiga estabelecer um padrão de comunicação em que identifique diferentes formatos de dados advindos de sensores é necessária porque pode melhorar a interoperabilidade entre os dispositivos.

A interoperabilidade caracteriza até que ponto duas implementações de sistemas ou componentes de diferentes fabricantes podem coexistir e trabalhar juntos, simplesmente confiando nos serviços uns dos outros conforme especificado por um padrão comum (Steen e Tanenbaum, 2018). Ela pode ser categorizada em 4 níveis (WANG, 2020) que são a interoperabilidade técnica, sintática, semântica e a organizacional. A técnica está associada com protocolos de comunicação, por exemplo, como o protocolo MQTT. Ela também está associada com a infraestrutura necessária para que os protocolos operem como *middlewares*, *gateways* e adaptadores. A interoperabilidade sintática está associada com formato de dados, sintaxe e *encodings*, independentemente do significado do dado transferido, por exemplo, HTML, XML e JSON. Algumas soluções para esse tipo de interoperabilidade são os conversores, *middlewares* e adaptadores. A interoperabilidade semântica é responsável por um entendimento comum do significado da informação sendo trocada. A interoperabilidade organizacional possui a habilidade de efetivamente comunicar e transferir informação mesmo entre diferentes sistemas sobre infraestruturas heterogêneas, possivelmente até sobre diferentes regiões geográficas e culturas.

Apesar da categorização encontrada no estudo, não é apresentada uma solução prática para gerenciamento da heterogeneidade dos formatos dos dados. O IoT Redirector proposto neste artigo se caracteriza por trabalhar a interoperabilidade sintática, pois trata o formato dos dados. Um dos componentes que fazem parte da solução proposta para trabalhar a interoperabilidade sintática é baseada em agentes IoT. Segundo (Savaglio et al., 2020) os agentes ajudam a superar desafios da internet das coisas como a interoperabilidade e a heterogeneidade. Na prática, o agente é responsável por permitir que as simulações com sensores sejam realizadas pois ele irá traduzir os dados recebidos

do redirecionador para o padrão NGSI[3] do FIWARE. Esse padrão traz como benefícios uma API que define um modelo de dados para informações de contexto, interface para troca de informações por meio de operações de consulta, assinatura (*subscribe*) e atualização (*update*).

Existem trabalhos que têm como objetivo tornar a integração e a interoperabilidade dos dados dos sensores em IoT mais plausível (Aggarwal, Ashish e Sheth, 2013) através do World Wide Web Consortium (W3C)'s Semantic Sensor Networks Incubator Group (SSN-XL) conforme citado por (Karkouch et al., 2016). Isso seria possível através de ontologias, metadados, anotações, serviços web, interfaces, *frameworks* como *Resource Description Framework* (RDF), *Web Ontology Language* (OWL). Esses *frameworks* proveem mecanismos para descrever os dados para facilitar sua busca, recuperação e processamento. Uma plataforma brasileira chamada Dojot[4] foi desenvolvida para a utilização de tecnologias para as cidades inteligentes com foco em segurança pública, mobilidade urbana e saúde.

Este artigo propõe uma arquitetura baseada no FIWARE[5] que é uma plataforma que utiliza o gerenciamento de contexto. Com ele é possível utilizar Docker[6] para integrar serviços como IoT Agents, *broker* MQTT, banco de dados MongoDB[7], Orion context *broker*[8]. A plataforma FIWARE permite integração com serviços de gerenciamento de dados de contexto, processamento, análise e visualização das informações.

3. Arquitetura do IoT Redirector

A arquitetura foi numerada indicando como o fluxo de dados passa pelos diferentes dispositivos, conforme Figura 3. Na Figura 2, é ilustrado com mais detalhes o simulador utilizado na arquitetura da Figura 3. O passo 1 do simulador representa o início em que o dado é gerado através da coleta de valores de temperatura dos sensores. Essa coleta foi realizada de maneira simulada, ou seja, é realizada uma requisição do tipo GET para a URL da API do OpenWeather[9] e o dado que a API retorna, entra no simulador em diversos formatos. O JSON é o formato padrão, porém é possível especificar o formato XML e o HTML. São várias medidas que chegam ao simulador como temperatura mínima, temperatura máxima, visibilidade, velocidade do vento. No entanto, a medida que foi utilizada na simulação foi a temperatura principal (*main temp*) que corresponde à temperatura média.

As coordenadas para coleta de temperatura dos sensores da API podem ser mapeadas para coletar dados de locais onde os sensores poderiam ser instalados. Para essa arquitetura foram selecionados 4 pontos: USP, UFABC Santo André, UFABC São Bernardo do Campo e centro de São Bernardo do Campo. As latitudes e longitudes podem ser consultadas através do site latlong[10] e informadas no site da API para que os sensores retornem os valores de temperatura desses locais.

Os valores recebidos da API precisam passar por uma análise e posterior seleção dos dados que farão parte do envio para o tópico do *broker* MQTT. Dentro do simulador

[3] http://telefonicaid.github.io/fiware-orion/api/v2/stable/
[4] https://dojot.com.br/sobre-a-dojot-iot/
[5] https://fiware-tutorials.readthedocs.io/en/latest/
[6] https://www.docker.com/
[7] https://www.mongodb.com/
[8] https://fiware-orion.readthedocs.io/en/master/
[9] https://openweathermap.org/api
[10] https://www.latlong.net/

existem funções que permitem manipular o *payload* da mensagem recebida. Uma dessas funções seleciona um atributo desejado e envia para o *broker*. No caso em questão o atributo é a temperatura principal. Na Figura 2 é esquematizada a aquisição, conversão, seleção, tratamento e envio do dado através do simulador.

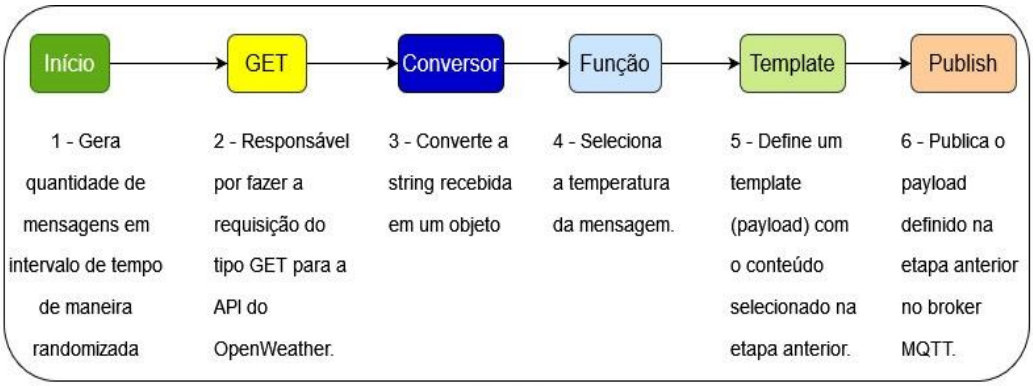

Figura 2 – Simulador elaborada pelo autor

Na Figura 3 é realiza a publicação do *payload* da mensagem no *broker* MQTT. Uma mensagem em formato JSON, por exemplo, será enviada para o tópico da seguinte forma: "3{"t":"19.5", "ts": "1638322260686"}". O valor 3 foi uma padronização adotada para o tipo identificado, que nesse caso é o JSON. O parâmetro ts corresponde ao timestamp e foi utilizado para que seja possível identificar mudança no contexto pelo Quantum Leap. O *payload* da mensagem publicada no *broker* pode ser visualizado na Tabela 1.

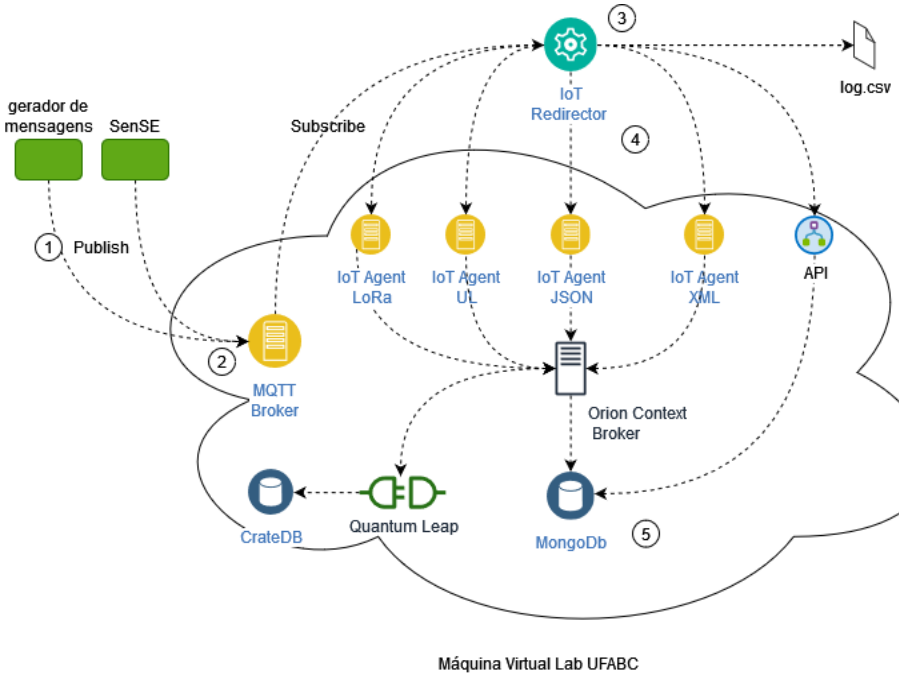

Figura 3 – arquitetura para gerenciamento da heterogeneidade de dados

No passo 3 da arquitetura uma solução de subscrição ao *broker* foi desenvolvida para que ocorra a subscrição ao tópico e faça a gravação do *payload* da mensagem no arquivo de log. Essa gravação no arquivo de log é realizada para que seja possível visualizar o dado que chega ao IoT Redirector e que será redirecionado.

Tabela 1. Exemplos de dados publicados no *broker* MQTT

Content-type	***payload***
xml	<measure device=dht22 key=4jggokgpepnvsb2uv4s40d59ov><t value=20.07 /><ti value = XML /><ts value = 1638322262635 /> </measure>
html	<!DOCTYPE html> 20.07 HTML 1638322260513
json	{t:20.06, ti:JSON, ts:1638322260686}
ul	t|20.12|ti|UL|ts|1638322261109
indefinido	<div title=Current Temperature>17
LoRa	{applicationID:5, applicationName:mestrado_simulacao, deviceName:DHT11, devEUI:221597e4529df57d, rxInfo: [{gatewayID:000000ffff001000, name: ExpGW, rssi:-57, loRaSNR:7, location: {latitude:0, longitude:0, altitude:0}}], txInfo: {frequency: 915000000, dr:5}, adr: false, fCnt: 0, fPort: 1, data:dHN8MTYzODMyMjI2MTU4Mnx0aXxsb3Jh}
TEROS12	t|20.12|ti|TEROS12|ts|1638322262013

O passo 4 da Figura 3 é responsável pelo redirecionamento da mensagem que chega ao tópico dependendo do tipo identificado. Porém, a lógica para detecção do tipo de dado identificado é baseada na assinatura da mensagem. Para um *payload* que possui o caractere "|" (pipe) é realizado o redirecionamento para o IoT Agent UL(UltraLight). Foi realizada a seguinte categorização que é detalhada na Figura 4:

- contém o caractere "<" e o caractere ">" foram classificados como XML
- possuem o caractere "{" como JSON
- com "|" como UL
- com "<!" como HTML,
- mensagem contém gatewayId ou o campo data como LoRa
- mensagem contém três espaços entre os dados e o caractere *carriage return* como Teros
- dado desconhecido caso contrário.

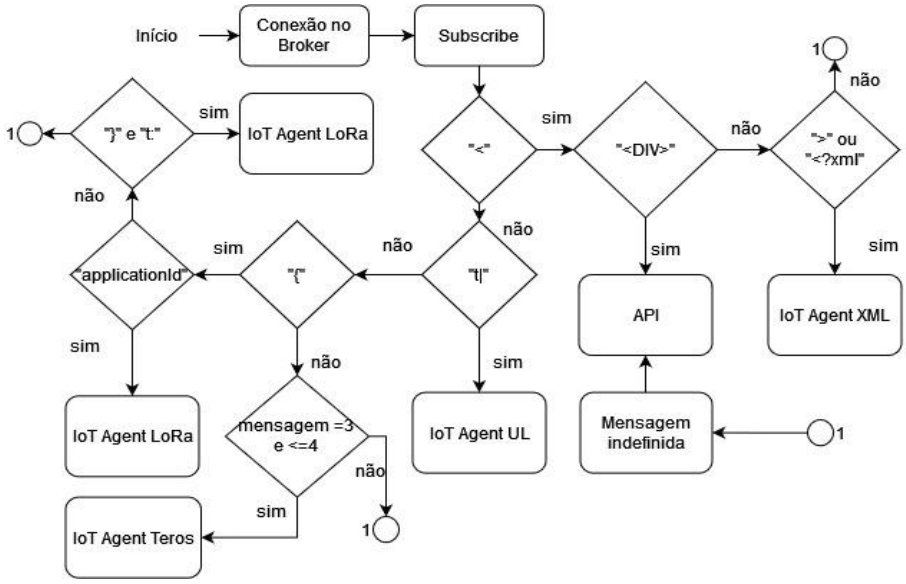

Figura 4 - Fluxograma IoT Redirector

A persistência dos dados é realizada por um componente chamado Quantum Leap, que faz a integração entre os dados e diversos tipos de bancos de dados. Nessa arquitetura é utilizado o CrateDB.

No passo 5 ocorre o monitoramento do redirecionamento, armazenamento do dado, consumo de recursos de máquinas virtuais como memória, CPU e tráfego de rede. Para esse tipo de monitoramento foi utilizado um código-fonte[11] em Python para realizar o monitoramento.

4. Metodologia e cenário de testes

Os testes foram elaborados para analisar o tempo de *parsing* de cada conteúdo das mensagens e a eficácia do redirecionamento das mensagens através do cálculo da acurácia. Foram criados três casos de testes (CT) em que foram aumentadas as quantidades de mensagens enviadas e reduzido o tempo de envio entre cada mensagem. No CT1 foram mil mensagens em 10 min e intervalo de tempo de envio de 5s, no CT2 três mil mensagens em 10 min e intervalo de tempo de envio de 4s, por último no CT3 foram cinco mil mensagens em 10 min e intervalo de tempo de envio de 3s.

Foram executados vários experimentos para validar a atuação do IoT Redirector. A simulação de envio de mensagens com diferentes conteúdos para a plataforma FIWARE foi realizada com um código em Java através do comando java -jar na linha de comando dessa forma: java -jar simulador.java 1000 600000. O parâmetro 1000 representa a quantidade de mensagens e 600000 representa o tempo em milissegundos ou 10 min. No CT1 o parâmetro da quantidade de mensagens foi mil, no CT2 foi três mil e no CT3 foi cinco mil.

[11] https://github.com/renatobdo/mestrado/blob/master/desempenho.py

Tabela 2 – Descrição dos testes

Descrição dos testes	Geração de mensagens de sensores com diferentes formatos de dados para o serviço de pub/sub do *broker* MQTT Mosquitto.
Propósito do teste	Analisar o tempo de parsing do IoT Redirector para cada formato de dados e a acurácia do redirecionamento das mensagens
Métrica	Tempo de parsing, utilização de CPU, utilização de memória, vazão e acurácia do redirecionamento
Caso de teste 1	Simular o envio de 1000 mensagens de sensores aleatórios com os tipos de dados: XML, JSON, UL, HTML, indefinido, LoRa, TEROS12. Com intervalo de tempo entre cada envio de 5 segundos.
Caso de teste 2	Simular o envio de 3000 mensagens de sensores aleatórios com os tipos de dados: XML, JSON, UL, HTML, indefinido, LoRa, TEROS12. Com intervalo de tempo entre cada envio de 4 segundos.
Caso de teste 3	Simular o envio de 5000 mensagens de sensores aleatórios com os tipos de dados: XML, JSON, UL, HTML, indefinido, LoRa, TEROS12. Com intervalo de tempo entre cada envio de 3 segundos.
Passos para execução	1) Executar o IoT Redirector e o algoritmo para medir o desempenho na VM3. 2) Iniciar os IoT Agents na VM 2 3) Iniciar o gerador de mensagens na VM 4 e o SenSE na VM1 4) Analisar os resultados
Resultado esperado	Contabilização de aproximadamente 15% do total de mensagens geradas de cada tipo. Por exemplo, em uma simulação de 1000 mensagens, 15% ou 150 mensagens devem ser do tipo JSON, 150 do tipo XML e assim por diante. Persistir no banco de dados as mesmas quantidades de mensagens geradas pelos simuladores sem perdas.

5. Resultados

Ao se realizar a contagem da quantidade de cada tipo de mensagem, obtivemos os seguintes resultados:

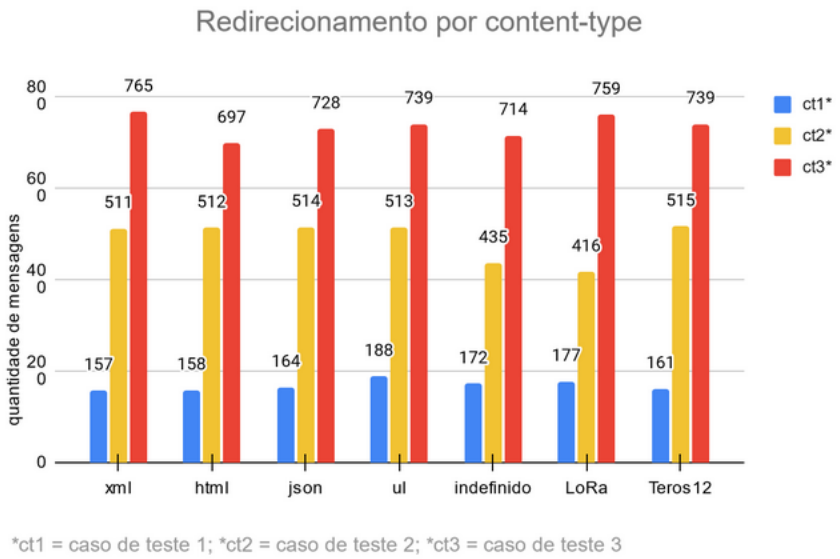

Figura 5 – Resultado do redirecionamento

Os diferentes tipos de dados gerados mostraram que o resultado de cada caso de teste (CT) encontrou aproximadamente 15% de cada tipo. No caso de teste 1, por exemplo, com 1000 mensagens sendo geradas, 157 foram identificadas como sendo do tipo XML, isso representa 15% do total de mensagens. As mensagens que foram redirecionadas para os IoT Agents XML, JSON, UL e LoRa passaram pelo Orion. As mensagens que não possuem um IoT Agent para gerenciá-las foram encaminhadas para uma API que as persistiu no banco de dados CrateDB.

Com uma letra correspondendo a um byte (PAULO FEOFILOFF, 2018), uma mensagem XML possui 127 bytes e uma mensagem LoRa 303 bytes aproximadamente. Uma tabela com a quantidade de bytes pode ser visualizada abaixo:

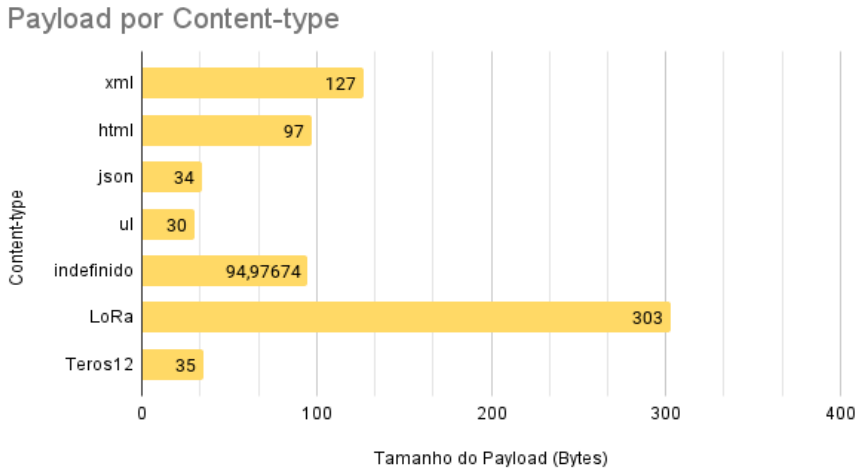

Figura 6 - Tamanho do *Payload* por *Content-type*

A quantidade de bits em 10 minutos de testes foi analisada. Essa vazão foi calculada baseada no tamanho do *payload* das mensagens e obteve-se o seguinte resultado:

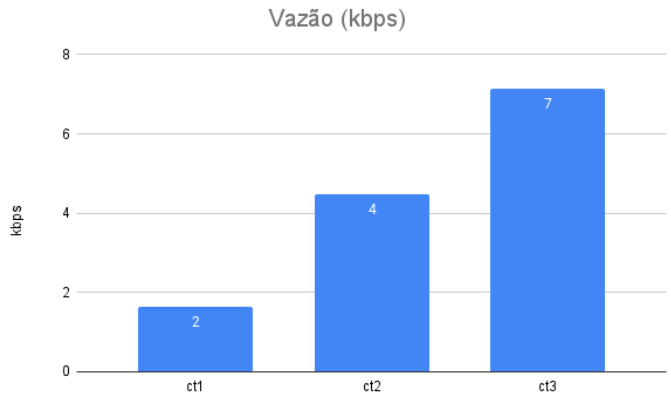

Figura 7 - vazão

O tempo de parsing de cada mensagem foi analisado em milissegundos e foi utilizada uma função que ao receber a mensagem do *broker* contabiliza o tempo inicial com a função do java *currentTimeMillis()* e quando o IoT Redirector identifica a mensagem contabiliza o tempo final subtraindo do tempo inicial. Foi utilizada uma média aritmética para se obter o tempo de parsing, pois todos os valores coletados se apresentaram de maneira uniforme, ou seja, não houve *outliers*. Observou-se que mensagens do tipo XML e HTML demoram mais para serem analisadas. Na média os outros formatos apresentaram resultados parecidos próximos de 1 ms. Apesar da quantidade de mensagens geradas no caso de testes 1 ser menor, o tempo de *parsing* foi maior quando comparado com os casos de testes 2 e 3, conforme pode ser visualizado na Figura 8.

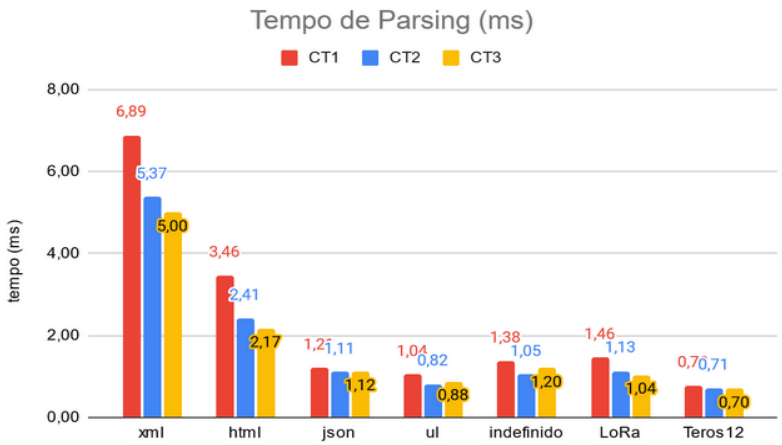

Figura 8 – Tempo de *parsing*

As máquinas virtuais utilizadas no experimento possuem processador Intel Xeon E5620 com 2.4 GHz, memória RAM de 4 Gb e sistema operacional Linux Ubuntu 16.04. A média de consumo de CPU se manteve praticamente a mesma para cada caso de testes, porém houve variação do consumo de memória. O caso de teste 3 exigiu 10% a mais de utilização de memória quando comparado com o caso de teste 1. Isso indica que possíveis exceções podem ser lançadas devido ao aumento do volume de dados e ao consumo de memória.

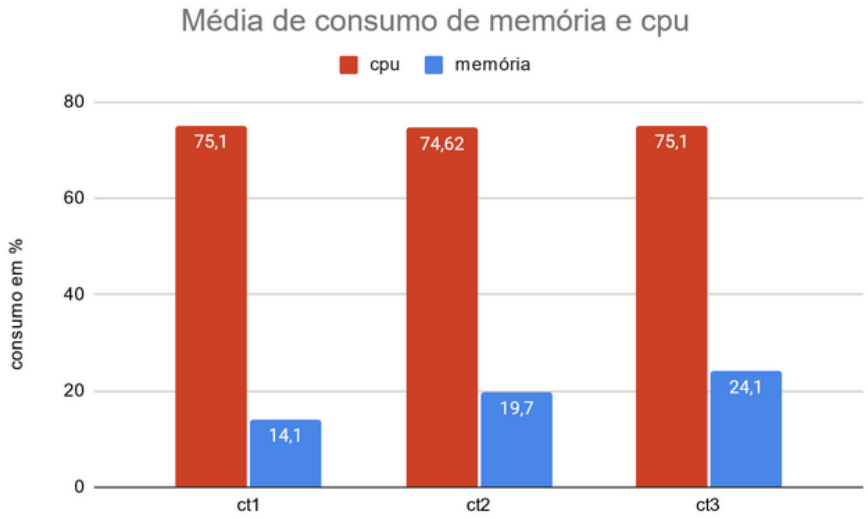

Figura 9 - Utilização de CPU e memória em porcentagem

Na Figura 10 pode-se observar a comparação entre o consumo de memória, nos três casos de testes em porcentagem, durante os 10 minutos de testes.

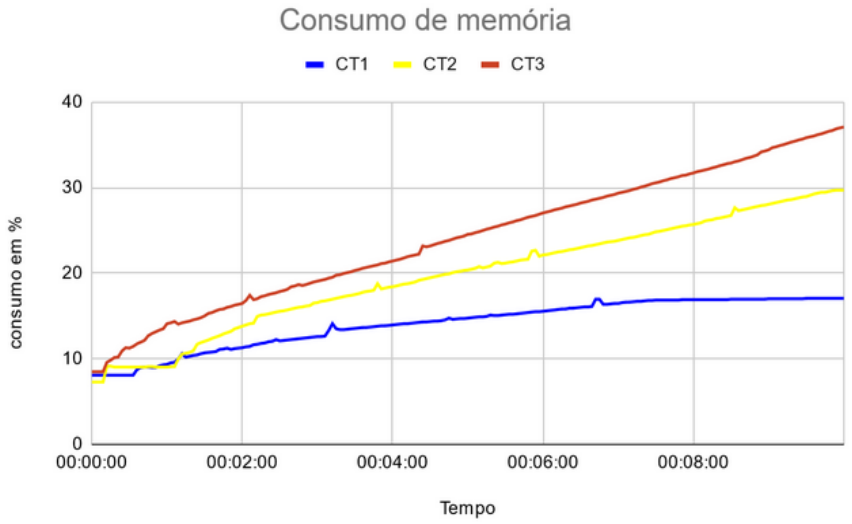

Figura 10 - Comparativo de consumo de memória em porcentagem

Com relação a acurácia no redirecionamento das mensagens foi utilizado um cálculo adaptado de (MAVROGIORGOU et al., 2019) em que a acurácia é calculada com a seguinte fórmula: Acurácia do dado = (Total de registros – Total de redirecionamentos errados) /Total de registros. Portanto, para cada caso de teste obteve-se 100% de acurácia porque não houve redirecionamentos errados.

6. Discussão

O propósito dos experimentos foi verificar o comportamento do IoT Redirector quanto a utilização de recursos computacionais e acurácia no redirecionamento, quando a quantidade de mensagens aumenta e quando o tempo de envio entre cada mensagem diminui.

Notou-se que em um cenário de geração de mais mensagens e um cenário com o tempo de envio menor existe a necessidade de mais recursos computacionais como memória. No entanto, o impacto percebido pela utilização de CPU se manteve constante com praticamente o mesmo valor para os três casos de testes.

O IoT Agent é um elemento importante em um ambiente de gerenciamento de dados, pois ele é o responsável por tratar a mensagem, assim como é responsável por encaminhá-la para um gerenciador de contexto. O Orion permite que seja realizado o *subscribe* no seu contexto e outros componentes que fazem parte desse ecossistema irão conseguir extrair informações.

A geração de mensagens pelos simuladores permitiu simular um ambiente heterogêneo de dados, no entanto, algumas mensagens não possuíam um IoT Agent capaz de tratá-las. Foram os casos das mensagens com conteúdos do tipo indefinidos, HTML e TEROS12. Dessa forma, a criação de uma API responsável por persistir os dados no banco de dados foi acrescentada.

Essa API fez com que todas as mensagens que chegassem ao IoT Redirector pudessem ser persistidas no MongoDB. Com isso, o IoT Redirector exerce a função de identificar as mensagens pelo seu conteúdo e encaminhar aquelas que ele não conseguiu identificar para essa API. Portanto, essas mensagens não são identificadas pelo Orion.

A acurácia da identificação das mensagens que chegaram ao *broker* foi de 100%, conforme experimentos realizados, pois o a solução proposta consegue identificar os 7 tipos de mensagens que os simuladores geraram. Porém, o IoT Redirector poderá encaminhar outros tipos de mensagens, mesmo sem conhecê-las. Ele poderá classificá-la como indefinida, mas ao conhecer um padrão na mensagem pode ser acrescentado esse tratamento na solução. Isso não é realizado de maneira automática, ou seja, um programador deve implementar na solução um encaminhamento da mensagem baseada no padrão.

A quantidade máxima de mensagens que o IoT Redirector conseguiu encaminhar foi de 7 mensagens por segundo. Isso representa uma vazão de 7 kbps conforme foi observado na Figura 7. A vazão foi calculada baseada na quantidade de bytes de cada mensagem enviadas durante os testes de 10 minutos. Apesar dos geradores utilizarem o protocolo MQTT para publicação no *broker*, o redirecionamento das mensagens é realizado através do padrão REST. Esse padrão possibilita a utilização de verbos como o POST para as URLs onde se encontram os *IoT Agents*. Isso permite maior flexibilidade no gerenciamento das mensagens.

7. Conclusão

O gerenciamento de dados de contexto tem se mostrado um importante avanço para lidar com dados provenientes de dispositivos IoT. É possível saber o que está acontecendo, quando e porquê através dos dados de contexto. Esse trabalho identificou a existência de um problema no gerenciamento dos dados advindo da sua heterogeneidade de formatos na plataforma FIWARE. Diferentes fabricantes podem enviar dados de sensores com distintos protocolos de comunicação, por exemplo, mensagens do tipo UL utilizam pipe. Mensagens do fabricante de sensor TEROS12 utilizam espaços entre os dados e no final o *carriage return*.

A principal contribuição desse trabalho foi desenvolver uma solução de software chamada IoT Redirector capaz de gerenciar a heterogeneidade dos dados ao identificar o tipo de conteúdo das mensagens enviadas pelos dispositivos IoT. Para se realizar a análise do conteúdo das mensagens é necessário identificar padrões que possibilitem categorizá-las. Essas categorizações foram realizadas com base em 7 tipos de conteúdos: XML, HTML, JSON, UL, LoRA, Indefinido e TEROS12. Contudo, outros protocolos podem ser identificados com base em padrões. A medição do tempo de parsing de cada conteúdo pelo IoT Redirector permite avaliar qual mensagem leva menos tempo para identificação. Novos agentes podem ser criados, adaptando-se a novas tecnologias e protocolos, traduzindo as mensagens para o padrão NGSI.

Durante os testes realizados a quantidade de mensagens geradas pelos simuladores permitiu que o IoT Redirector identificasse as várias mensagens aleatórias que chegavam ao *broker*. Todas as mensagens foram identificadas com taxa de acurácia de 100%, ou seja, nenhuma mensagem identificada foi redirecionada incorretamente. Quando as mensagens chegavam ao Orion ele notificava o Quantum Leap que as persistia no CrateDB. As mensagens com conteúdos indefinidos, HTML e TEROS12, foram persistidas diretamente no banco de dados MongoDB por uma API desenvolvida com o objetivo de receber as mensagens que não possuem um IoT Agent responsável pelo seu gerenciamento.

Nesse trabalho o IoT Redirector desenvolvido conseguiu identificar as mensagens simuladas de diferentes fabricantes, mostrando que é possível gerenciar a heterogeneidade de dados em aplicações inteligentes baseada na IoT. Entretanto, existem *trade-offs* quando mais mensagens são geradas por sensores como maior consumo de memória e tempo de processamento na identificação das mensagens. Em um contexto real de IoT serão necessários ajustes no código-fonte do IoT Redirector que realiza o redirecionamento, porém os dados podem ser identificados com elevado grau de acurácia.

Para pesquisas futuras novos filtros poderão ser adicionados na identificação dos *content-types* permitindo um redirecionamento maior de dados. Criação de novos *IoT Agents* para permitir o gerenciamento das mensagens com conteúdos similares. Utilização de gramática formal e processamento de linguagem natural para classificação automatizada do formato das mensagens. No IoT Redirector as mensagens recebidas poderiam ser identificadas e classificadas de acordo com a proximidade com outro padrão. Por exemplo, os dados de um sensor poderiam ser classificados como UL porque se apresentam de forma parecida a esse padrão. Isso poderia trazer como benefício um maior gerenciamento de dados provenientes de dispositivos desconhecidos. Avaliação da performance dos *IoT Agents* com uma maior quantidade de mensagens recebidas, pois os dados gerados por minuto são muito grandes e a tendência é crescer esse número, principalmente com o 5G. Replicação e balanceamento de carga com aumento da vazão. Classificador de mensagens com utilização de inteligência artificial. Modularizar o IoT Redirector de forma a aceitar classificadores de forma plugável.

Referências

AGGARWAL, C. C.; ASHISH, N.; SHETH, A. The Internet of Things: A Survey from the Data-Centric Perspective. In: Managing and Mining Sensor Data. Boston, MA: Springer US, 2013. p. 383–428.

AL-FUQAHA, A. et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials, v. 17, n. 4, p. 2347–2376, 2015.

ARUN SOLANKI, ADARSH KUMAR, A. N. Digital Cities Roadmap: IoT-Based Architecture and Sustainable Buildings. 1. ed. [s.l.] John Wiley & Sons, 2021, 2021.

BANAFA, A. Three Major Challenges Facing IoT - IEEE Internet of Things. Disponível em: <https://iot.ieee.org/newsletter/march-2017/three-major-challenges-facing-iot.html/>. Acesso em: 19 jan. 2022.

FOX, J. What is an IoT Agent? Disponível em: <https://fiware-tutorials.readthedocs.io/en/stable/iot-agent/>. Acesso em: 18 out. 2021.

KARKOUCH, A. et al. Data quality in internet of things: A state-of-the-art survey. Journal of Network and Computer Applications, v. 73, p. 57–81, set. 2016.

KHATOON, P. S.; AHMED, M. Semantic Interoperability for IoT Agriculture Framework with Heterogeneous Devices. In: [s.l: s.n.]. p. 385–395.

LASSE LUETH, K. State of the IoT 2020: 12 billion IoT connections. Disponível em: <https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/>. Acesso em: 22 dez. 2021.

MAVROGIORGOU, A. et al. IoT in Healthcare: Achieving Interoperability of High-Quality Data Acquired by IoT Medical Devices. Sensors 2019, Vol. 19, Page 1978, v. 19, n. 9, p. 1978, 27 abr. 2019.

PAULO FEOFILOFF. Bytes, números e caracteres. Disponível em: <https://www.ime.usp.br/~pf/algoritmos/aulas/bytes.html>. Acesso em: 8 dez. 2021.

RED HAT. What is a REST API? Disponível em: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>. Acesso em: 19 jan. 2022.

SAVAGLIO, C. et al. Agent-based Internet of Things: State-of-the-art and research challenges. Future Generation Computer Systems, v. 102, p. 1038–1053, jan. 2020.

SONI, DIPA & MAKWANA, ASHWIN. A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT). Agharam Road, Selaiyur, Chennai: 2017.

STEEN, M. VAN; TANENBAUM, A. S. Distributed Systems. [s.l: s.n.].

WANG, Y. Enhancing interoperability for IoT based smart manufacturing. [s.l: s.n.]. Acesso em: 23 jun. 2020.