Adaptabilidade e tolerância a falhas no uso de recursos de IoT em uma plataforma de cidades inteligentes

Rafael R. Silva, Fábio M. Costa (Orientador), Bruno Silvestre (Coorientador)

[1]Instituto de Informática
Universidade Federal de Goiás (UFG)

silva_rafael@discente.ufg.br, fmc@inf.ufg.br, brunoos@inf.ufg.br

Resumo

As plataformas de Cidades Inteligentes surgiram como base para o desenvolvimento e integração de aplicações distribuídas para as cidades, fornecendo serviços comuns e agilizando a construção de soluções. Contudo, a utilização de recursos por parte das aplicações exigem a combinação de dados de diversos sensores para que um resultado seja obtido. Conforme presente na literatura, existem diversos modelos de mapeamento entre recursos e sensores, de forma que tornam mais efetivo o uso dos dados. Nesse contexto, este trabalho introduz o conceito de virtualização de recursos para o e envio de mensagens à plataforma InterSCity através da implementação de um componente para a virtualização, permitindo assim a flexibilização dos recursos da plataforma. Portanto, o trabalho porpõe uma infraestrutura que possibilita a adaptabilidade dos recursos da plataforma à diferentes aplicações e domínios de sensores através da virtualização.

1. Apresentação

Conforme a literatura da última década previa [Domingue et al. 2011], durante os últimos anos obtivemos diversos avanços tecnológicos na área da internet das coisas, como sensores e microcontroladores. Aliado ao desenvolvimento tecnológico, diversas aplicações foram criadas para solucionar problemas de uma cidade moderna. A internet das coisas (IoT) é um dos elementos principais da infraestrutura de cidades inteligentes, [Zanella et al. 2014].

Nesse contexto, devido à grande demanda de aplicações em uma cidade moderna, é promissor o fato de que o cenário de desenvolvimento para aplicações em cidades inteligentes integre o desenvolvimento de aplicações de IoT através do uso de uma plataforma centralizada. A centralização é uma alternativa que visa proporcionar o suporte necessário para que diversas aplicações possam acessar diversos recursos da infraestrutura de uma cidade de maneira uniforme, independente de tecnologias específicas [Santana et al. 2017].

A plataforma InterSCity é um exemplo desse tipo de plataforma, [Del Esposte et al. 2019]. Desenvolvida através da colaboração que envolve pesquisadores de diversas universidades brasileiras, no contexto da INCT da Internet do Futuro para Cidades Inteligentes.

Nos dias de hoje o processo de desenvolvimento de aplicações para cidades inteligentes envolve a implantação dos sensores, a coleta dos dados pelas plataformas e o processamento dos dados, no qual no cenário atual é realizado pela aplicação ou pela

própria plataforma de cidades inteligentes, o que gera mais custos para a plataforma [Ogawa et al. 2019b].

Estudos exploratórios na área [Nitti et al. 2015] e [Bhajantri et al. 2020], citam soluções para o uso eficiente de sensores para cidades inteligentes, onde é utilizada a abordagem de combinação de sensores, ou até mesmo a sua agregação, um cenário de N recursos reais para 1 recurso lógico.

O presente trabalho atua no escopo referente à coleta e processamento dos dados de sensores para plataformas de cidades inteligentes. Propomos a inclusão de um componente responsável pela virtualização de recursos para o envio de mensagens no contexto da plataforma [Del Esposte et al. 2019].

1.1. Plataforma InterSCity

A plataforma InterSCity é uma plataforma de código aberto para aplicações em cidades inteligentes baseada em microsserviços que possui uma arquitetura focada em escalabilidade. A plataforma foi desenvolvida pelo INCT Internet do Futuro para Cidades Inteligentes, sediado no Instituto de Matemática e Estatística da Universidade de São Paulo, e do qual participam outras universidades brasileiras, dentre as quais a UFG. O objetivo principal da plataforma é fornecer serviços e APIs para auxiliar no desenvolvimento de novas aplicações para cidades inteligentes.

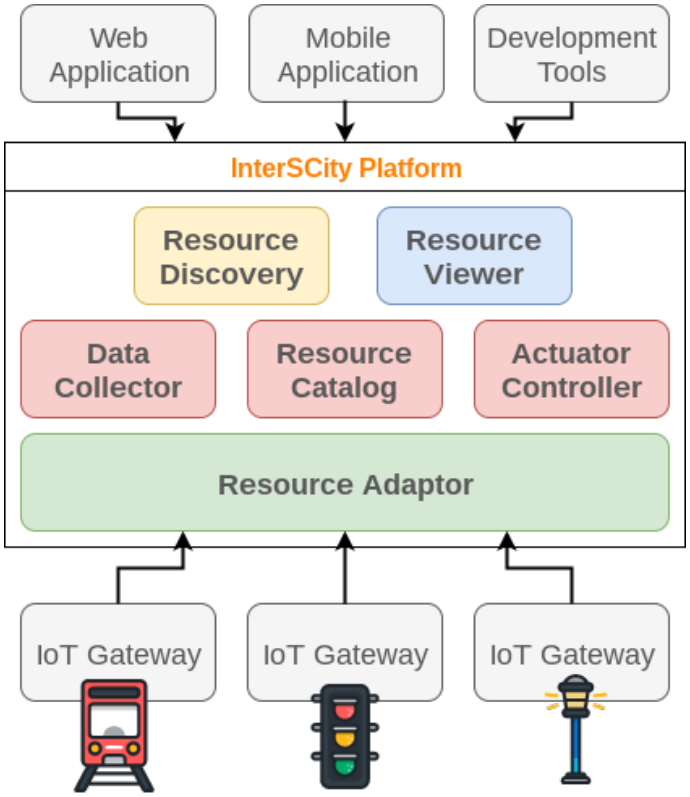

Figura 1. Arquitetura da Plataforma InterSCity [Del Esposte et al. 2019]

Conforme a arquitetura exposta na Figura 1, a plataforma recebe dados enviados pelos recursos da infraestrutura da cidade (sensores e atuadores, por exemplo) por meio de *IoT Gateways*, os quais por sua vez os enviam ao microsserviço *Resource Adaptor*.

Esse microsserviço é responsável por realizar a comunicação entre os *IoT Gateways* e os demais microsserviços, bem como cadastrar recursos (sensores e atuadores) e interagir com os recursos por meio de consultas e do envio de comandos.

O *Resource Catalog* é o responsável pelos dados estáticos da plataforma, age como um catalogo dos recursos presentes na plataforma, atribui uma nova UUID [1] para cada novo recurso na plataforma. O *Data Collector* é o responsável por armazenar a prover o acesso aos dados dos recursos aos outros microsserviços quando requisitado. O *Actuator Controller* possibilita que a plataforma envie comandos para interagir com o estado de recursos da cidade. O *Resource Discovery* provê uma API para as aplicações permitindo assim a busca de recursos na plataforma. Por ultimo, o *Resource Viewer* é um microsserviço *front-end* que possibilita uma representação visual dos recursos da plataforma.

O processo de cadastro e envio de dados dos recursos à InterSCity é realizado através de requisições REST[2], contendo informações sobre o sensor (como capacidades, descrição e tipos de dados que serão enviados). Após realizar o cadastro, a plataforma gera um UUID[3] responsável pela identificação do recurso. A partir daí, o acesso e envio de dados é realizado por meio de requisições REST, endereçando os dados ao recurso usando seu UUID.

O *IoTGateway* é a estrutura responsável por enviar dados coletados à plataforma InterSCity. A forma com que esses dados são enviados ao *IoTGateway* é abstraída nos níveis superiores.

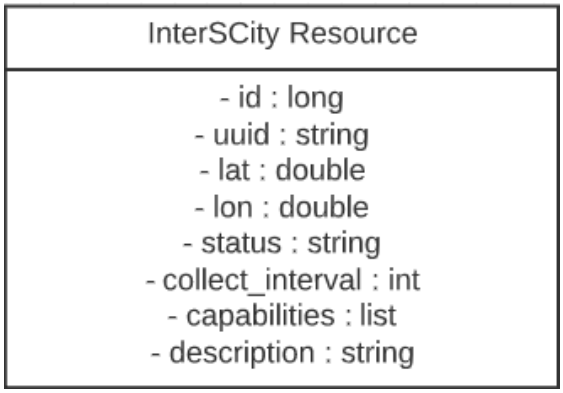

Figura 2. Recurso para a Plataforma InterSCity

A InterSCity utiliza o conceito de recursos para a identificação dos dados, conforme a Figura 2. Recursos são, em sua maioria, mapeados individualmente por sensores, onde cada recurso contem os dados referentes à um sensor real. Com isso, com o objetivo de promover uma infraestrutura mais flexível, na próxima subseção apresentamos o conceito de virtualização de recursos, que possibilitam a agregação e um pré-processamento dos dados dos sensores.

[1] *Universally Unique IDentifier*
[2] *Representational State Transfer*
[3] *Universally Unique IDentifier*

1.2. Virtualização de Recursos

A virtualização de recursos tem como principal objetivo, permitir a inclusão de recursos na plataforma InterSCity, os quais possuem dados provenientes de outros recursos presentes na plataforma através da agregação de sensores.

O conceito de virtualização de recursos abordada nesse trabalho é semelhante à virtualização de dispositivos abordada pelo trabalho Fed4IoT [Ogawa et al. 2019a], no qual é apresentada uma proposta que realiza o pré-processamento dos dados de sensores, através do compartilhamento dos dados entre os nós intermediários na infraestrutura, conforme ilustrado na Figura 3.

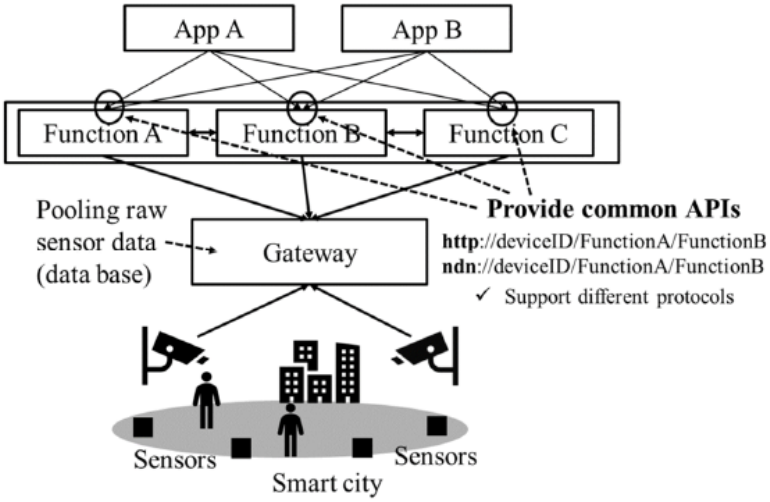

Figura 3. Conceito de virtualização de dispositivos, [Ogawa et al. 2019a]

Na proposta da Feed4IoT existem dois nós responsáveis por realizar a comunicação dos dados às aplicações. Os *gateways*, responsáveis pela camada de comunicação com os sensores e os nós *function*, responsáveis pelo pré-processamento dos dados oriundos dos sensores. A arquitetura é composta por microsserviços, os quais possuem funções pré-definidas, que realizam a comunicação com os demais componentes através de um protocolo pub-sub.

Uma outra abordagem para a virtualização de dispositivos é encontrara em [Cheng et al. 2017], que apresenta um *framework* para computação em núvem chamado FogFlow. Que trata a virtualização de recursos como uma forma de promover a conectividade de dispositivos que possuem diferentes formas de comunicação à plataforma de cidades inteligentes.

Conforme é ilustrado na Figura 4, a virtualização re recursos é ligada ao conceito de intermediação de comunicação entre sensores reais e a aplicação, semelhante à figura do IoT Gateway presente na infraestrutura da InterSCity, explicitada na seção 1.1.

Semelhante à FogFlow, outro trabalho que introduz o conceito de virtualização de dispositivos é o Virtual Fog [Li et al. 2017]. Nesse caso o autor realiza a implementação do conceito de *Virtual Object*(VO), que faz referência ao sensor físico. Conforme ilustrado na Figura 5, o VO é uma camada que representa o sensor real internamente.

Por fim, tento em vista os trabalhos analisados, aliado à analise do fluxo de envio

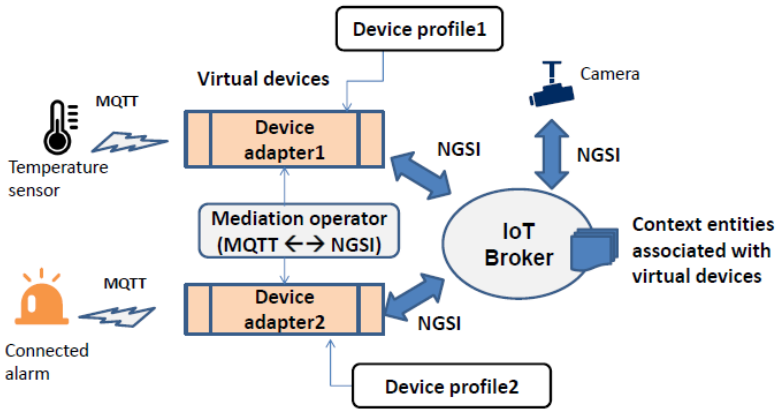

Figura 4. Ilustração do conceito de virtualização de recursos para o FogFlow, [Cheng et al. 2017]

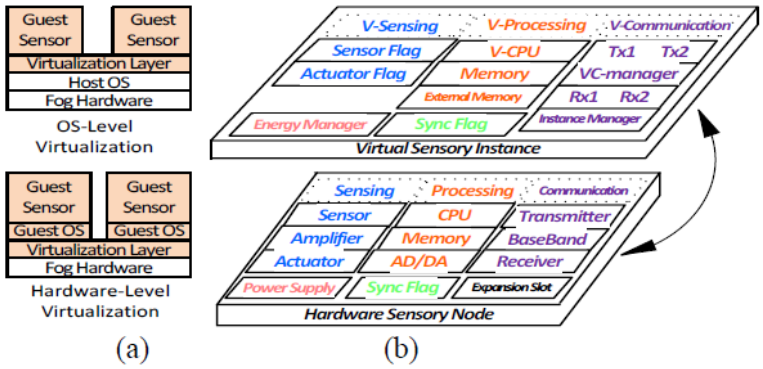

Figura 5. Virtualização de Objetos na Virtual Fog, [Li et al. 2017]

de mensagens à plataforma interSCity ilustrado na Figura 1. Introduzimos o conceito de Virtualização de Recursos com o objetivo de agregar recursos e permitir com que o processamento dos dados fossem descentralizados. Solução abordada na Seção 3.

2. Metodologia

Inicialmente, realizamos buscas na literatura atual com o objetivo de encontrar exemplos e diferentes formas de gerenciamento,mapeamento e relacionamento de recursos.

Tendo em vista que o trabalho atual é a continuação de uma pesquisa anterior, na qual um modelo de IoTGateway [4] foi implementado, para integrar diferentes tipos de sensores à plataforma InterSCity. Para uma avaliação inicial, idealizamos um modelo de virtualizador para o tratamento dos recursos dentro da arquitetura do IoTGateway, com o objetivo de avaliar a virtualização de recursos no ambiente de envios de dados à plataforma InterSCity. Posteriormente, realizamos a implementação de um componente nomeado IoT Virtualizer [5], responsável pela virtualização de recursos. Desta forma, o IoT Virtualizer possibilita a virtualização de recursos sobre um domínio maior de sensores reais, uma vez que a arquitetura atual permite que o componente seja posicionado em

[4]Implementação disponível em: https://github.com/r4faelrs/IoTGateway
[5]Implementação disponível em: https://github.dev/r4faelrs/IoTVirtualizer

diferentes camadas da *fog* na arquitetura de mensagens à plataforma.

As tecnologias utilizadas para o desenvolvimento da pesquisa foram Raspberry Pi Model B (ARM11, 512 MB) para instâncias do IoTGateway, um laptop (i5-8250U, 8 GB) para instâncias do IoTVirtualizer e uma VM na Google Cloud (e2-medium, 2 vCPU, 4 GB) para uma instância da InterSCity.

3. Resultados e discussão

Nesta seção, são descritos: a arquitetura do IoTVirtualizer proposto para a plataforma InterSCity; o processo de envio de mensagens virtualizadas; e os testes realizados com o objetivo de validar o componente proposto.

3.1. Virtualização de Recursos para a plataforma InterSCity

Baseado no conceito de associação de objetos virtuais com sensores reais previsto em um *Survey* por [Nitti et al. 2015], que cita a possibilidade de associação de muitos para muitos. Introduzimos a proposta de virtualização de recursos através de um componente nomeado de IoTVirtualizer.

O componente proposto realiza a virtualização de recursos através dos conceitos utilizados pela plataforma InterSCity. Como abordado na seção 1.1, a plataforma possui os conceitos de *Resource* e de *Capabilities*, os quais, definem as bases de um recurso. O virtualizador realiza a adequação desses conceitos para que seja possível a agregação de diversos sensores à um recurso virtual.

Um recurso virtual, possui duas partes principais: sua definição como recurso, que será cadastrada na plataforma InterSCity; e os sensores pertencentes à ele, tais sensores podem ser definidos a partir das informações contidas na plataforma sobre eles, como UUID [6], latitude ou longitude.

Uma *capability* de um recurso virutal, além dos dados referentes ao seu cadastro na plataforma, possui uma definição para o processamento de dados. Essa definição contém a operação a ser realizada, como também o tipo de dados utilizados para esse cálculo, por exemplo, o cálculo médio de dados de temperatura. A definição das *capabilities* segue um padrão semântico previamente definido.

Conforme a Figura 6, a virtualização de recursos acontece numa camada intermediária, possibilitando assim com que o domínio de sensores possíveis para a virtualização seja maior.

3.2. Arquitetura do IoTVirtualizer

A arquitetura do IoTVirtualizer é composta por seis componentes, *Receiver*, *Catalog*, *Data Processor*, *Register*, *Sender* e um *Manager*. Conforme a Figura 7.

O *Receiver* é responsável pela API REST do Virtualizer, é responsável por receber os dados e por receber todas as conexões provenientes dos sensores. O *Catalog* é responsável por armazenar os dados de sensores e recursos virtuais no banco de dados, como também, pelo fornecimento dessas informações para os outros componentes do IoTVirtualizer.

[6] *universally unique identifier*

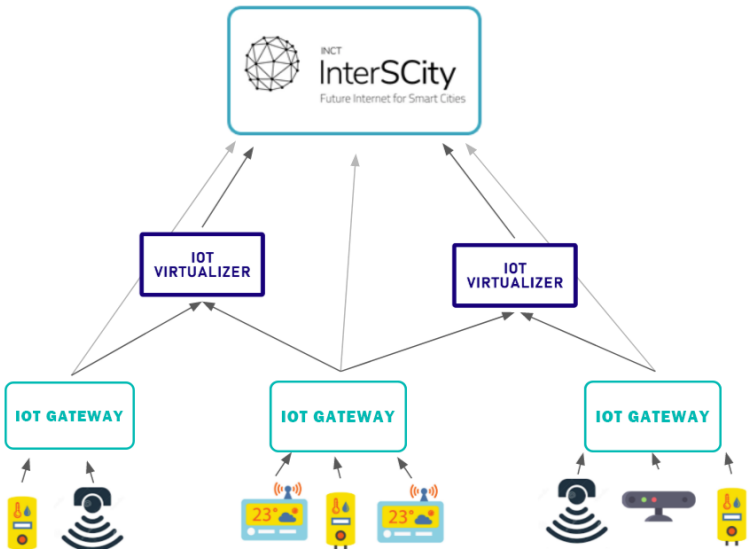

Figura 6. Modelo de envio de mensagens com virtualização de recursos.

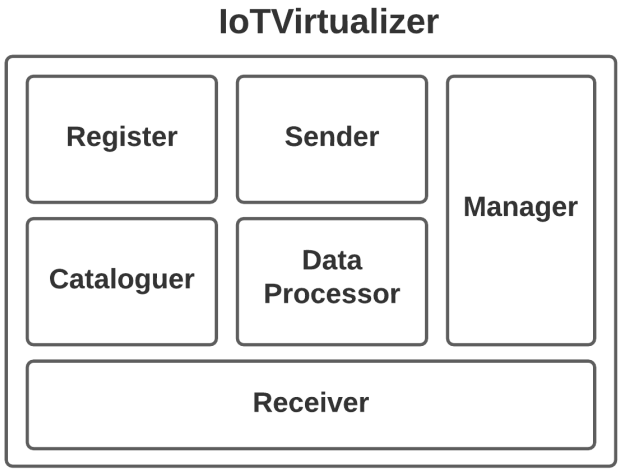

Figura 7. Arquitetura do IoTVirtualizer

O *Data Processor* é o componente responsável pelo processamento dos dados oriundos dos sensores, de forma que, o processamento é realizado sob uma definição previamente estabelecida no registro das *capabilities* do recurso virtual. É um componente pode ter inúmeras implementações diferentes. No momento atual, existe uma implementação que realiza o processamento de dados simples, como cálculo média, máximo e mínimo. Contudo, em trabalhos futuros podemos ter abordagens mais complexas para esse componente.

O *Register* é responsável pelo cadastro de recursos virtuais e de *capabilities* na plataforma InterSCity, como também do registro de interesse nos IoTGateways, através de um pub-sub, dessa forma, os dados dos sensores são destinados ao IoTVirtualizer.

O *Sender* é responsável pelo envio de dados processados pelo *Data Processor* à plataforma. Temos também um componente de gerenciamento, o *Manager*, responsável

pela orquestração dos componentes internos.

3.2.1. Fluxo Interno

O IoTVirtualizer possui uma API REST [7] que possibilita a comunicação com os sensores e possíveis administradores da rede de sensores. Conforme o diagrama presente na Figura 8, o processo de envio de mensagens possui três processos principais: O cadastro de um recurso virtual (1 à 7); Cadastro das *capabilities* (8 à 13); e o envio (14 à 16) e processamento (17 à 20) dos dados dos sensores, conforme.

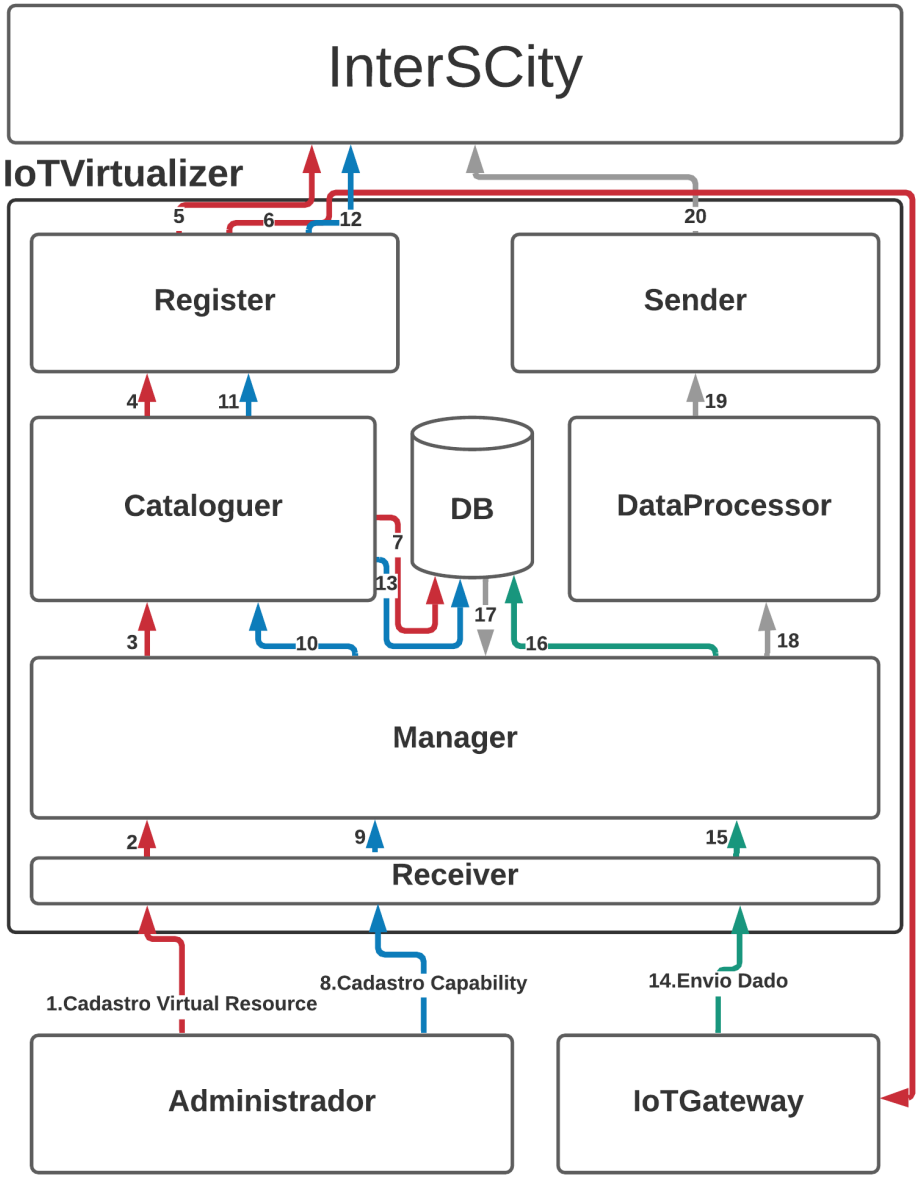

Figura 8. Diagrama de do envio de mensagens ao IoTVirtualizer

[7] *Representational State Transfer*

O cadastro do recurso virtual é requisitado através de um administrador responsável pela modelagem da rede de virtualização. Ao receber a requisição através do *Receiver* (2), a mensagem é identificada e encaminhada para o fluxo de cadastro (3). O *cataloguer* é ativado para que verifique o cadastro, caso não exista um cadastro do recurso em questão, o *register* é acionado (4), o qual, realiza o cadastro do recurso virtual como um *resource*, na plataforma e busca dos sensores reais vinculados ao recurso virtual na plataforma (5), e quando encontrados, realiza o cadastro de interesse nos IoTGateways responsaveis pelo dado (6), para que os dados dos sensores sejam enviado ao virtualizador. Posteriormente ao cadastro, os dados do recurso virtual são guardados numa base de dados local (7).

Semelhante ao cadastro de um recurso virtual o cadastro de uma capability é recebida e encaminhada ao *cataloguer* onde é verificado a existência de outras capabilities com o mesmo nome (8 à 10), caso contrário, o *Register* realiza seu cadastro na plataforma (11 e 12). Ao fim do processo de cadastro os dados da capability são guardados na base de dados (13).

Por último, o envio de dados através de sensores reais é recebido da mesma forma que as outras requisições (15), porém, é apenas armazenado na base de dados (16), pois o processamento não é realizado no momento em que um dado é recebido. Na implementação atual, o processamento de mensagens é realizado a partir de um tempo pré-definido, como à cada um minuto. Durante cada período de execução o *Manager* realiza o processamento das *capabilities* de todos os recursos virtuais cadastrados na instancia atual do virtualizador. O *Manager* verifica os dados de sensores recebidos armazenados na base de dados e classifica-os de acordo com os recursos virtuais aos quais cada dado pertence (17) e encaminha os dados para o processamento no *Data Processor* (18). Por fim, o *Data Processor* realiza o processamento dos dados conforme a definição da *Capability* determina. Após o processamento, o dado é enviado à plataforma pelo *Sender* (19 e 20).

3.3. Avaliação de desempenho do IoTVirtualizer

Com o objetivo de validar o componente desenvolvido, realizamos testes referentes ao *overhead* de envio de mensagens à plataforma, como também um teste de estresse.

O teste com o objetivo de aferir o *overhead* foi realizado a partir dos componentes de *hardware* citados na seção 2. Para a realização das analises tivemos como foco principal o envio de mensagens dos sensores que englobam o, envio de dado do IoTGateway ao IoTVirtualizer, o processamento do dado e o envio do dado à plataforma.

Para a coleta de dados, os tempos foram consultados nos pontos 14, 17 e 20, conforme a Figura 8. A medição do tempo do processo no ponto 14 faz referência ao tempo de envio de um dado ao virtualizador, no ponto 17 o tempo de consulta e processamento dos dados, e o ponto 20 para identificação do tempo de envio à plataforma.

Conforme os resultados obtidos no teste de *overhead* ilustrados na Figura 9. Obtivemos um resultado no qual não nos permite concluir nenhuma perda de desempenho referente ao tamanho do dado que passa pelo virtualizador. Porém, os testes efetuados apenas contabilizam o tempo gasto em um caso base, onde o dado bruto apenas passa pelo virtualizador, ou seja, não é tratado. Em um ambiente onde vários dados são processados é esperado que o tempo de processamento aumente de acordo com o processamento

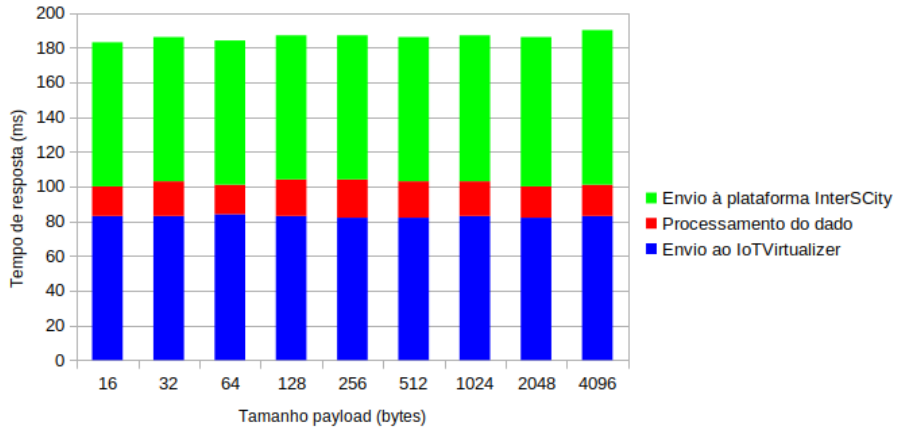

Figura 9. Resultados do teste *overhead*

realizado.

Para uma melhor avaliação do cenário atual, e de um possível gargalo no componente, pontuamos a necessidade da realização de mais testes. Testes que possibilitam descontar o tempo gasto nas requisições *HTTP*[8] observados nos pontos que englobam envios de mensagens presentes na Figura 9.

Realizamos também, um teste de estresse, o qual permitiu a realização de uma avaliação de uso do IoTGateway ao enviar dados para um IoTVirtualizer.

A coleta de dados desse experimento foi realizada a partir dos pontos 14 e 15, conforme a Figura 8. O ponto 14 faz referência ao tempo de envio de um dado e o 15 para a verificação do recebimento desse dado.

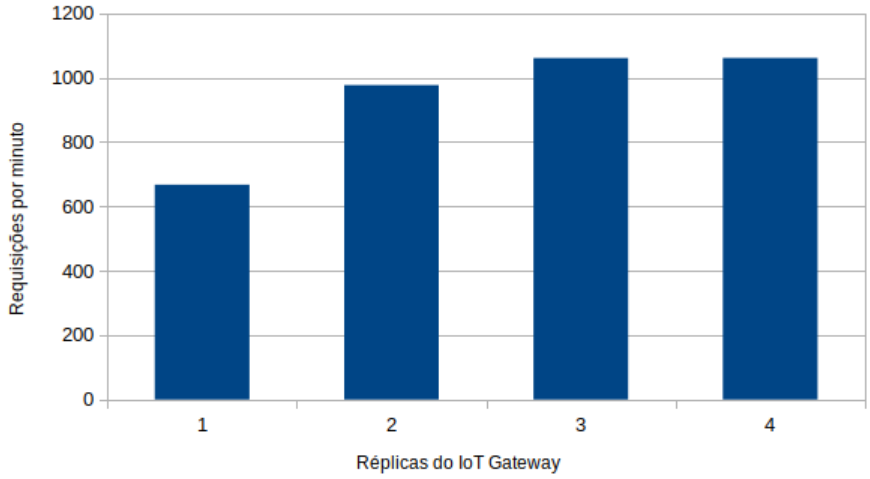

Figura 10. Resultados do teste de carga.

A condução desse teste se deu através da replicação das *threads* de processamento de um IoTGateway, no qual foi realizados o envio sequencial dos dados. Conforme obser-

[8] *Hypertext Transfer Protocol*

vado na Figura 10 a quantidade de dados recebidos pelo IoTVirtualizer não dobra ao passo que os processos dobram, porém, há um aumento no número de requisições processadas.

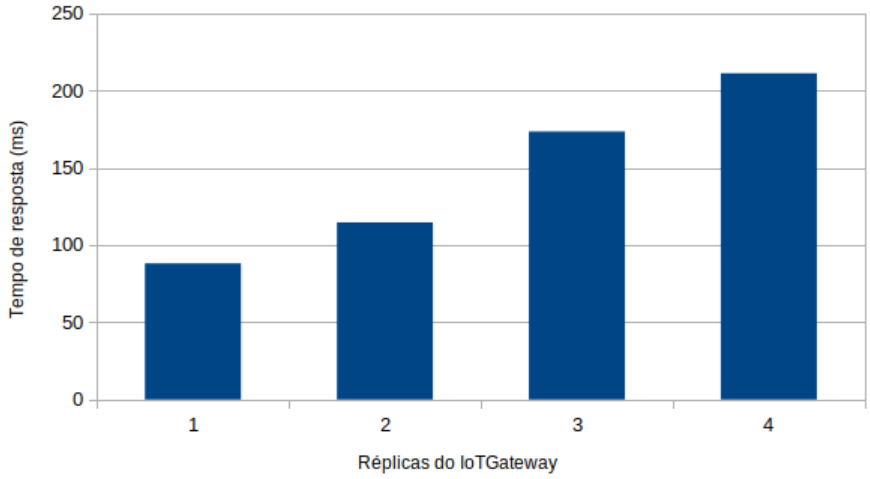

Figura 11. Tempo médio de resposta obtidos no teste de carga.

Contudo, conforme os resultados obtidos no teste de carga ilustrados na Figura 10 combinados com os dados presentes na Figura 11, nota-se que houve um acréscimo no tempo médio de resposta ao passo que aumentamos as réplicas dos IoTGateways.

Portanto, para um melhor aproveitamento no número de requisições enviadas de um IoTGateway à um IoTVirtualizer é possivel verificar que o tempo de resposta das requisições é afetado.

4. Considerações finais

Neste trabalho, apresentamos a proposta da inclusão do conceito de virtualização de recursos para o contexto de envio de mensagens à plataforma InterSCity através da implementação de um componente nomeado de IoTVirtualizer. O virtualizador de recursos proposto permite a flexibilização dos recursos presentes na platafoma, possibilitando o mapeamento de vários sensores reais para um recurso virtual.

A implementação atual do virtualizador possui alguns pontos para trabalhos futuros, como a elaboração de testes com o objetivo de identificar gargalos da arquitetura atual. Também como trabalho futuro temos a elaboração de um modelo estruturado para a definição das cabilities, bem como uma validação de sua sintaxe dentro do virtualizador. Por último, a implementação de um componente para o processamento de eventos complexos para uma futura versão do *Data Processor*.

Referências

Bhajantri, L. B. et al. (2020). A comprehensive survey on resource management in internet of things. *Journal of Telecommunications and Information Technology*.

Cheng, B., Solmaz, G., Cirillo, F., Kovacs, E., Terasawa, K., and Kitazawa, A. (2017). Fogflow: Easy programming of iot services over cloud and edges for smart cities. *IEEE Internet of Things Journal*, 5(2):696–707.

Del Esposte, A. d. M., Santana, E. F., Kanashiro, L., Costa, F. M., Braghetto, K. R., Lago, N., and Kon, F. (2019). Design and evaluation of a scalable smart city software platform with large-scale simulations. *Future Generation Computer Systems*, 93:427–441.

Domingue, J., Galis, A., Gavras, A., Zahariadis, T., Lambert, D., Cleary, F., Daras, P., Krco, S., Müller, H., Li, M.-S., et al. (2011). *The Future Internet: Future Internet Assembly 2011: Achievements and Technological Promises*. Springer Nature.

Li, J., Jin, J., Yuan, D., and Zhang, H. (2017). Virtual fog: A virtualization enabled fog computing framework for internet of things. *IEEE Internet of Things Journal*, 5(1):121–131.

Nitti, M., Pilloni, V., Colistra, G., and Atzori, L. (2015). The virtual object as a major element of the internet of things: a survey. *IEEE Communications Surveys & Tutorials*, 18(2):1228–1240.

Ogawa, K., Kanai, K., Nakamura, K., Kanemitsu, H., Katto, J., and Nakazato, H. (2019a). Iot device virtualization for efficient resource utilization in smart city iot platform. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 419–422. IEEE.

Ogawa, K., Sekine, H., Kanai, K., Nakamura, K., Kanemitsu, H., Katto, J., and Nakazato, H. (2019b). Performance evaluations of iot device virtualization for efficient resource utilization. In *2019 Global IoT Summit (GIoTS)*, pages 1–6. IEEE.

Santana, E. F. Z., Chaves, A. P., Gerosa, M. A., Kon, F., and Milojicic, D. S. (2017). Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Computing Surveys (Csur)*, 50(6):1–37.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32.

A. Informações complementares

Devido à choques de horários e ao baixo número de palestras oferecidas pelo programa Diálogos em Pesquisa e Inovação, não consegui participar das atividades realizadas durante o período pandêmico através do canal do YouTube da instituição.