Revista de Sistemas de Informação da FSMA
n. 25 (2020) pp. 2-18

SISTEMAS DE INFORMAÇÃO
http://www.fsma.edu.br/si/sistemas.html

TRILHA PRINCIPAL

ERI·GO

Implantação de múltiplos gateways IoT definido por software e virtualizado para campus inteligente

Divino Alves Ferreira Júnior, *Instituto de Informática (INF) - Universidade Federal de Goiás (UFG) & Instituto Federal de Goiás (IFG) - Campus Senador Canedo*
João Paulo Oliveira Cabral, *Instituto de Informática (INF) - Universidade Federal de Goiás (UFG)*
Ciro José Almeida Macedo, *Instituto de Informática (INF) - Universidade Federal de Goiás (UFG) & Instituto Federal de Goiás (IFG) - Campus Cidade de Goiás*
Tércio Alberto dos Santos Filho, *Instituto de Biotecnologia (IBiotec) - Universidade Federal de Catalão (UFCAT)*
Sand Luz Corrêa, *Instituto de Informática (INF) - Universidade Federal de Goiás (UFG)*
Waldir Moreira, *Fraunhofer Portugal AICOS*
Kleber Vieira Cardoso, *Instituto de Informática (INF) - Universidade Federal de Goiás (UFG)*
Antonio Carlos de Oliveira Júnior, *Instituto de Informática (INF) - Universidade Federal de Goiás (UFG) & Fraunhofer Portugal AICOS*

***Resumo*—A Internet das Coisas - IoT é baseada em objetos ("coisas"), capazes de se comunicar e transferir dados. O fato gera um aumento no fluxo de dados trafegados pela rede. Para suportar esse aumento, a heterogeneidade de dispositivos e a distribuição geográfica tem-se a necessidade de uma infraestrutura que comporte esta demanda. O Instituto de Informática da Universidade Federal de Goiás propôs um *gateway* IoT definido por software com suporte a várias tecnologias de comunicação sem fio e integração de ambiente neblina/nuvem. Um protótipo foi implantado para a fila do Restaurante Universitário com uso de *gateway* IoT único, que foi posteriormente avaliado com bom desempenho. Nesse artigo, são apresentados os resultados da avaliação do protótipo, bem como, a expansão do sistema em um projeto piloto no Campus Samambaia (UFG), com implementação de múltiplos *gateways* IoT, visando coletar dados sobre a quantidade de resíduos orgânicos, monitoramento de níveis de lixo em lixeiras seletivas, temperatura, umidade, nível de gás e monitoramento de vídeo. Os experimentos realizados tinham como intuito analisar o funcionamento e a capacidade do sistema para gerenciar o fluxo de dados, desde os sensores até a aplicação do usuário.**

***Palavras-chave*—Campus Inteligente, Internet das Coisas (IoT), Gateway, SDN, Computação na névoa, Orquestração, Virtualização.**

Deployment of multiple software-defined and virtualized IoT gateways for smart campus

***Abstract*—The Internet of Things (IoT) is based on objects (" things "), which are able to communicate and transfer data. The fact generates an increase in the flow of data trafficked over the network. In order to support this increase, the heterogeneity of devices and the geographic distribution have the need for an infrastructure which can handle this demand. The Institute of Informatics of the Federal University of Goias has proposed a software-defined IoT gateway thats supports for various wireless communication technologies and fog / cloud environment integration. A prototype was deployment for the queue at the University Restaurant using a single gateway IoT, which was afterwards evaluaed with good performance. In this article, the results of the evaluation of the prototype are presented, as well as the expansion of the system in a pilot project at Campus Samambaia (UFG), with the implementation of multiple gateways IoT, aiming to collect data on the amount of organic waste , monitoring of garbage levels in selective bins, temperature, humidity, gas level and video monitoring. The experiments carried out were intended to analyze the working and the capacity of the system to manage the data flow, from the sensors to the user's application.**

***Index Terms*—Smart Campus, Internet of Things (IoT), Gateway, SDN, Fog Computing, Orchestration, Virtualization.**

Autor Correspondente: Antonio Oliveira-Jr, *antoniojr@ufg.br*

I. Introdução

INTERNET das Coisas (*Internet of Things* - IoT) é um conceito que colabora diretamente com o que chamamos de revolução tecnológica. IoT refere-se a objetos do cotidiano, tais como lâmpadas domésticas, relógios, aparelhos hospitalares, carros (denominados "coisas" no contexto de IoT), que se conectam a uma rede e desempenham uma ou mais funcionalidades que agregam valor ao dia a dia, tanto no aspecto profissional como pessoal. L. Atzori *et al.* [1] definem IoT como sendo a presença difusa de uma variedade de objetos inteligentes que, por meio de comunicação usando tecnologias de redes, são capazes interagir uns com os outros de maneira cooperativa para alcançar objetivos comuns.

Um *gateway IoT* tem como função estabelecer a comunicação entre vários dispositivos para formar uma rede, compartilhando recursos e informações, e permitindo a intercomunicação entre os equipamentos com diferentes protocolos de rede [2]. O *gateway* é a ponte entre redes de sensores sem fio e redes de comunicação tradicionais ou Internet [2]. A base para a evolução da IoT está na comunicação, definindo desta forma a importância de um *gateway* IoT.

Considerando este contexto, diversos tipos de *gateways* IoT vêm surgindo para atender as demandas das tecnologias emergentes em escopos maiores, como cidades e campi inteligentes, sendo então importante sua performance em relação ao número de dispositivos gerenciados.

Dentre as tecnologias mais utilizadas para lidar com a variedade de dispositivos IoT, podemos destacar os *containers*, que são um recurso utilizado para que processos rodem de forma isolada em um host com o mesmo sistema operacional. Esta tecnologia se sobressai em virtude da performance sobre técnicas de virtualização tradicionais, como máquinas virtuais baseadas em Hypervisor [3]. Provedores de serviços IoT e entusiastas que fazem uso de *containers* tem feito avaliações de desempenho majoritariamente com ênfase em consumo de *hardware* e eletricidade [4].

Gateways IoT são comuns [5] e [6], assim como *gateways* de rede baseados em *hardware*. No entanto, há uma tendência de *gateways* IoT empregando a tecnologia de rede definda por *software* - SDN. Dada a expansão do uso de redes sem fio no cenário de IoT é de suma importância trabalhos que também visam o desempenho de rede em suas avaliações, como nos estudos [7] e [8].

Normalmente, dispositivos IoT utilizam uma comunicação baseada em tecnologia de redes sem fio para ter acesso à Internet. Alguns exemplos de tecnologias para comunicação em redes sem fio são: Wi-Fi Halow, BLE (*Bluetooth*), ZigBee, Z-Wave, NFC (*Near Field Communication*), LoRa/LoRaWAN, LTE-M (*Long Term Evolution - Machine*) e NB-IoT (*Narrowband Internet of Things*). Cada uma dessas tecnologias tem uma característica própria, considerando inclusive o alcance de transmissão.

Uma solução de campus inteligente pode demandar o uso de uma ou mais dessas tecnologias e, neste contexto, identifica-se a necessidade de *gateways* que suportem múltiplas tecnologias para que os dispositivos IoT tenham acesso à internet. Algumas tecnologias de comunicação como, por exemplo, Wi-Fi Halow, LTE-M1 estão em padronização, desenvolvimento ou até mesmo em projeto. Assim, deve-se ter cautela na definição dos dispositivos IoT, uma vez que, essas tecnologias podem sofrer atualizações ou serem descontinuadas.

Um projeto desenvolvido pela Universidade Federal de Goiás (UFG) e financiado pela Rede Nacional de Ensino e Pesquisa (RNP), desenvolveu uma solução de comunicação (*gateway*) virtualizada, definida por software, com suporte à tecnologia *fog computing* integrada com a computação em nuvem, denominado SOFTWAY4IoT [9], que suporta a comunicação de múltiplas tecnologias sem fio. Para um

TABELA I
Características de tecnologias de comunicação usada em dispositivos IoT

Tecnologia	**Freq. Transm..**	**Taxa Transm.**	**Alcance**
BLE	2.4 - 2.5 GHz	1Mbps	77 m
ZigBee	2.4 - 2.5 GHz	250kbps	290 m
Z-WAVE	915MHz (US-ISM band), 868MHz (EU-RFID band)	100kbps	100 - 200 m
LoRa	868MHZ (Europa) e 915MHZ (USA / Brasil)	290bps a 50 Kbps	15 km Zona Rural 2 - 5 km Zona Urbana
WiFi HaLow	900MHz	150kbps a 40mbps	1000 m
LTE-M1	1.4 MHz	100kbps	Curto
NB-IoT	180 KHz	50kbps	Longo
NFC	13.56MHz	424kpbs	0.1 m

solução com estas características torna-se importante o uso de um sistema IoT definido por software, pois permite criar redes virtuais isoladas e implementar políticas de segurança de rede de maneira rápida e flexível. A criação de redes isoladas viabiliza o uso de várias tecnologias de comunicação em um *gateway* definido por *software*. Algumas vantagens são identificadas como:

- Sistema mais leve permitindo maior escalabilidade. Com o uso de virtualização e *container* é possível tornar o sistema mais leve, permite ainda usar múltiplas tecnologias de comunicação com o recurso de fatiamento de redes o que melhora a escalabilidade.
- A migração de serviços para a borda da rede propicia uma redução de custos operacionais. Entende-se por borda a parte do sistema mais próxima dos sensores/dispositivos IoT ligados em rede. O custo operacional ao processar os serviços na nuvem é maior se comparar com o custo de processamento de serviços na borda, desta forma parte ou todos os recursos podem ser manipulados na borda e enviados resultados para armazenamento na nuvem.
- Uso de SDN permite adicionar funcionalidades de controle de tráfego, segurança, entre outras, a um controlador centralizado, facilitando o gerenciamento da rede. Com o emprego de SDN as atualizações e melhorias implementadas no gerenciamento de funcionalidade de redes fica centralizada, e uma vez modificado as alterações são atualizadas para a rede a que se aplica, facilitando o lado operacional da gerencia de rede.

A solução é direcionada especialmente aos campi inteligentes, que são vistos com várias características semelhantes a uma cidade inteligente, por serem um ambiente com alta densidade de pessoas tendo problemas relacionadas à mobilidade, estacionamento, segurança, alimentação, convívio social, etc. Por outro lado, em comparação com uma cidade, o campus possui tamanho reduzido, gestão mais simples e possibilidade de acesso a toda a infraestrutura física de campus o que viabiliza a implantação de soluções de IoT e experimentos de maneira mais rápida.

Em um primeiro momento, o foco do projeto foi o desenvolvimento de uma solução/protótipo que estabeleça a comunicação dos dispositivos IoT com um *gateway* integrando isto a uma rede, considerando os conceitos de SDN, virtualização, integração da *fog/cloud*, suporte à múltiplas tecnologias como apresentado por D. A. Ferreira Jr *et al.* [10]. Isto posto, o sistema foi implantado com uma aplicação para monitoramento da fila de um restaurante universitário, pois esta serviria como bom modelo para a avaliação da solução e por ser de interesse de múltiplas pessoas, permitindo testes práticos com múltiplos usuários. Os dados coletados pelo sensor são enviados pela rede e alimentam a base de dados, que poderá ser acessada via aplicação, por alunos ou demais interessados em verificar o status da fila. A Figura 1 apresenta a interface da aplicação da Fila RU. O objetivo do protótipo foi o de avaliar a comunicação estabelecida e o desempenho do sistema.

Fig. 1. Interface da aplicação de monitoramento da Fila RU - UFG

Um cenário de campus inteligente vai além de uma fila de restaurante universitário. Existem demandas desde controle de temperatura, os níveis de gás e até um sistema de monitoramento, cenário este que motivou a continuidade da pesquisa. Em uma nova etapa, aprimorou-se a solução para suportar múltiplos *gateways* estendendo as aplicações envolvidas e implanta-se um piloto do SOFTWAY4IoT a partir do protótipo apresentado em [10].

Este artigo tem como objetivos:

- Apresentar uma solução que estabeleça a comunicação dos dispositivos IoT, usando múltiplos *gateways*, sendo um *master* e outros *slave*, de modo virtualizado, definido por *software*, empregando ferramentas de orquestração de recursos, *containers*, fatiamento de redes e integrando os ambientes de *fog/cloud*;
- Implantar o piloto do sistema, tendo o Campus Samambaia como ambiente de implantação, considerando aplicações de controle de resíduos sólidos, monitoramento de vídeo, controle de níveis de lixo em lixeiras seletivas, coletar dados de temperatura, umidade e gás;
- Realizar os testes de implantação, checagem de funcionamento dos módulos, criação e configuração de *slices*, conectividade dentro do *slice* e o teste da aplicação piloto.

Para tanto este artigo é organizado da seguinte forma: após a Introdução, a seção 2 dedica-se aos trabalhos relacionados. A Seção 3 apresenta a implantação, avaliação e resultados dos experimentos relativos ao protótipo. A Seção 4 mostra o piloto do sistema aplicado ao ambiente de campus inteligente, considerando a arquitetura, implantação e testes realizados. As considerações finais e trabalhos futuros são discutidas na Seção 5.

II. Trabalhos relacionados

O Ambiente de *smart city, smart campus* ou cenários que se aplicam o uso de dispositivos IoT estão em ascensão e despertam tanto o interesse comercial quanto o científico, a fim de aprimorar as pesquisas e evoluir o contexto de internet das coisas.

A Canonical, desenvolvedora do Ubuntu, apresentou um tutorial que permite transformar seu Raspberry 4 em um *gateway* de borda. Segundo K. Galem [11], o *EdgeX Foundry* não é apenas um *software* de código aberto, mas também um dispositivo que enfatiza a interoperabilidade. Esses fatores se combinam para catalisar um ecossistema de componentes que federam o espaço da IoT.

A Intel também possui no mercado dispositivos como o Intel IoT *gateway* [12], esse dispositivo oferece conectividade para comunicação sem fio e com fio, rede celular e rede de curto alcance, recursos de segurança, gerenciabilidade através de plataforma Web.

A Mozzila.org [13] também apresenta o *WebThings Gateway*, uma solução que pode utilizar um *raspberry* ou um roteador, como *gateways* residenciais inteligentes, que permitem aos usuários monitorar e controlar sua casa inteligente na Web sem um intermediário.

Os desafios e principais direcionamentos das pesquisas em IoT são apresentados por Z. Wen *et al.* [14], propondo uma estrutura que orquestra ambientes de computação na névoa, implementando uma estrutura baseada em algoritmo genético (GA-Par) no Spark para lidar com cenários de orquestração que envolvem a composição de um grande conjunto de aplicativos de IoT. Em nossa proposta trabalhamos com orquestração de recursos e utilizamos fatiamento de rede para comunicação de dispositivos heterogêneos. O uso de fatiamento de rede permite implementar políticas de segurança ou regras para controle de tráfego de rede específicas para cada tecnologia de comunicação, sendo uma alternativa que permite o gerenciamento da comunicação de dispositivos heterogêneos com maior eficiência visando um melhor desempenho.

Um *framework* para orquestração de carga de trabalho foi desenvolvido por D. Santoro *et al.* [15]. Este *framework* é multicamada, envolvendo a nuvem, a camada na névoa e os dispositivos IoT e utiliza tecnologias como *Kubernets* para a orquestração e *OpenStack* como camada de IaaS - *Infrastructure as a Service*. Ao final são apresentados três casos de uso para testar sua proposta. Esse estudo não apresenta o uso de fatiamento de rede e não é explicito sobre a integração de múltiplas tecnologias de comunicação.

A comunicação, considerando as múltiplas tecnologias, é um dos desafios das pesquisas aplicadas a IoT, visa atender a heterogeneidade de dispositivos e flexibilizar a infraestrutura da rede.

Recursos de orquestração, usando SDN em ambiente de *fog computing* levando em conta a comunicação *end-to-end* são implementados no estudo de R. Vilalta *et al.* [16]. A sua proposta estabelece caminhos de ponta a ponta em redes heterogêneas, no entanto, não emprega o fatiamento de rede para a comunicação usando tecnologias distintas, dessa forma não viabiliza a utilização de políticas de redes diferentes para cada fatia.

Um *framework*, denominado *FogFlow*,desenvolvido por B. Cheng *et al.*, para ambiente de *smart city*, integrando a *cloud e Fog Computing*, opera com recursos geodistribuídos, hierarquizados e heterogêneos [17]. Este *framework* possui três divisões logicamente separadas: 1) gerenciamento de serviços; 2) processamento de dados; e 3) gerenciamento de contexto. A proposta adota uma abordagem baseada em padrões e propõe um modelo de programação baseado em NGSI (*Next Generation Service Interface*) para permitir a programação fácil dos serviços de IoT na nuvem e nas bordas. Embora apresente uso de orquestração, ambiente *fog/cloud*, comunicação com múltiplas tecnologias, o trabalho não usa recursos de fatiamento de rede para a comunicação de múltiplas tecnologias.

O aumento da quantidade e da diversidade de dispositivos IoT conectados à Internet gera desafios para a arquitetura de rede tradicional, que não é projetada para suportar um alto nível de escalabilidade, entrega de dados em tempo real e mobilidade. Para abordar essas questões, S. Tomovic *et al.* [18] apresentam um novo modelo de arquitetura da Internet das Coisas, que combina os benefícios de duas tecnologias emergentes: redes definidas por *software* e computação na névoa. A estrutura do sistema proposto envolve dispositivos finais com várias soluções de comunicação sem fio, controladores SDN, infraestrutura na névoa heterogênea (servidores virtualizados, roteadores, pontos de acesso, etc.) e nuvem no núcleo da rede. Não identificamos o uso de fatiamento de rede para a comunicação de redes sem fio.

Uma estrutura para provisionamento de recursos em ambiente *fog* é apresentada por O. Skarlat *et al.* [19]. O modelo apresentado prevê uma arquitetura implantada usando o *framework CloudSim*, desenvolvido no laboratório da *University of Melbourne*, da Austrália. Este trabalho compara um modelo *baseline* com o modelo de provisionamento apresentado. O mesmo autor, em outro trabalho [20] apresenta um modelo de otimização para o mesmo ambiente descrito no trabalho anterior. Segundo estudo, é proposto um algoritmo genético com uma heurística de resolução de problemas e através de experimentos, demonstra-se que com o uso do algoritmo genético a execução do serviço pode alcançar uma redução dos atrasos na comunicação da rede e uma melhor utilização do neblina em comparação com a aplicação do método de otimização exato. Embora utilize orquestração de recursos com integração de ambiente *fog/cloud*, estes trabalhos não apresentam uso de SDN, o que permitiria o melhor controle e monitoramento da rede, não trata sobre a comunicação de múltiplas tecnologias e tampouco trata fatiamento de rede.

Um *gateway* IoT inteligente é apresentado por S. Guoqiang, *et al.* [21] com uso de cartões para interface de usuários, reconhecendo múltiplas tecnologias, suportando 3 (três) tipos de comunicação com a rede comum: Ethernet, 3G e RS485 Bus. No entanto, não apresenta orquestração de recursos, SDN/NFV, integração dos ambientes *fog/cloud* e fatiamento de rede. Este trabalho é voltado apenas para a conectividade de dispositivos, sem considerar fatores como a segurança e a qualidade do serviço.

A comunicação de dispositivos heterogêneos é um desafio para o avanço da IoT. Há uma necessidade latente de múltiplos *gateways* com suporte a múltiplas tecnologias, a necessidade de controle maior do fluxo de dados, o que pode ser implementado com uso de *containers*, orquestração, uso de virtualização para deixar as aplicações mais leves e escaláveis, com menor custo operacional. O uso de SDN pode viabilizar um controle mais eficaz sobre a rede, e.g. políticas de segurança.

Uma abordagem de gateway IoT é descrita por C. Mouradian *et al.* [22] para aplicação em cenário de desastre, sendo uma arquitetura distribuída baseado em SDN e NFV (*Network Function Virtualization*) para comunicação ad-hoc. Este *gateway* suporta múltiplas tecnologias de comunicação sem fio, no entanto, não apresenta uso de fatiamento de redes nem integração com ambiente *fog/cloud*. O fatiamento de rede é uma alternativa que permite tratar necessidades específicas com políticas específicas, dessa forma, conforme a demanda apresentada pode-se tratar de maneira específica apresentando uma solução pontual.

Os trabalhos relacionados e as tecnologias suportadas para ambientes IoT são listados na Tabela II. Nenhum desses estudos apresentam uso de fatiamento de rede, que permite fatiar a rede em partes que podem ser gerenciadas de maneira independente. O fatiamento de redes é empregado nas redes 5G. A tecnologia 5G aposta no fatiamento como forma de oferecer serviços à empresas ou clientes demandantes com tratamento de necessidades específicas. Os trabalhos [16] [17] [21] [22], em suas abordagens tratam da comunicação com tecnologias múltiplas, o que facilita a interoperabilidade de dispositivos em ambiente de IoT. O uso de SDN foi apresentado por [16] [17] [22] [14] [15] [18]. Apenas [21] não apresenta uso de orquestração de recursos. A integração com ambiente de *fog/cloud computing* não foi apresentado por [21] [22]. O uso de orquestração em ambiente *fog* visa o melhor aproveitamento dos recursos, implicando em redução de custos operacionais.

III. Fundamentação teórica

Esta seção apresenta os principais conceitos utilizados no desenvolvimento e avaliação deste trabalho.

Uma Rede Definda por *software* (*Software Defined Networking - SDN*) é uma arquitetura de rede em que o plano de controle e dados são separados, permitindo

TABELA II
Trabalhos relacionados

Art.	Sup. IoT	Múlt. rede	SDN	Serviço Orques-tração	Borda Nuvem	Slicing
[14]	Y	-	Y	Y	Y	-
[15]	Y	-	Y	Y	Y	-
[16]	Y	Y	Y	Y	Y	-
[19]	Y	-	-	Y	Y	-
[17]	Y	Y	Y	Y	Y	-
[18]	Y	-	Y	Y	Y	-
[20]	Y	-	Y	Y	Y	-
[21]	Y	Y	-	-	-	-
[22]	Y	Y	Y	Y	-	-

que regras de fluxos de encaminhamento sejam previamente programadas via *software* para tomadas de ações dinâmicas e eficientes. Com o uso de SDN é possível concentrar a lógica de controle em controladores SDN ou Sistemas Operacionais de Redes, sendo tal característica um diferencial das demais redes tradicionais [23] pois possibilita a criação de ambientes totalmente virtuais, como no caso implementado pelo projeto da UFG denominado SOFTWAY4IoT, que estabelece fatias de rede virtuais para uma determinada quantidade de aplicações também virtualizadas.

A Computação em Neblina (*Fog Computing*) consiste na alocação de poder de processamento intermediário próximo à borda de uma rede. *Fog computing* é um paradigma inovador que realiza computação distribuída, serviços de rede e armazenamento, além da comunicação entre *data centers* na nuvem até os dispositivos ao longo da borda da rede [24]. As principais características de *fog computing* são: heterogenidade, distribuição geográfica, suporte a mobilidade, interação em tempo real, escalabilidade, interoperabilidade entre outras. Para o segmento de IoT, computação em neblina se mostra promissora graças às possibilidades de redução de latência e implementação de políticas de segurança intermediárias entre usuários e nuvem.

Virtualização é uma tecnologia que combina ou divide recursos computacionais em um ou mais ambientes operacionais usando de metodologias específicas para simular ou emular completa ou parcialmente um sistema [25]. Além de baixar o custo da operação de diversos tipos de sistemas modernos, a virtualização também está em foco nas pesquisas que envolvem segurança em ambientes inteligentes como Campus ou Cidades Inteligentes.

Uma técnica eficiente de virtualização, chamada de *container*, baseia-se no isolamento de processos em um sistema operacional hospedeiro, permitindo a execução de várias instâncias de sistemas distintos consumindo fatias de recursos de hardware e rede do mesmo [26]. *Containers* são máquinas virtuais cujo estado não é persistente, e são instanciados a partir de imagens. As imagens são pequenas, pois utilizam sistema de arquivos em camadas, por isto a criação de *containers* é muito rápida e o provisionamento de recursos torna-se mais eficiente em comparação às máquinas virtuais tradicionais [27]. Como os *containers* compartilham do mesmo *kernel* do Sistema Operacional Hospedeiro pode ser obtido pouco *overhead* de performance permitindo eficácia ao executar grandes números de *containers* de aplicações IoT.

Docker é uma tecnologia usada para virtualização de *containers* segundo [28]. *Docker* é como uma máquina virtual muito leve, além de criar *containers*, fornece o fluxo de trabalho do desenvolvedor. O *Docker* é um projeto de código aberto que se baseia em muitas tecnologias conhecidas da pesquisa de sistemas operacionais conforme [29]

Ainda no contexto de virtualização de *containers* contamos com o recurso do *Kata Containers*. Conforme [30], é um projeto de uma comunidade de *software* livre que trabalham para criar uma implementação padrão de máquinas virtuais (VMs) leves que se sentem e funcionam como *containers*, mas fornecem o isolamento da carga de trabalho e vantagens de segurança das VMs.

Neste projeto também foi utilizado o *Open vSwitch* - OVS, que é um *switch* virtual multicamada licenciado sob a licença Apache 2.0 de código aberto. Ele foi projetado para permitir automação de rede massiva por meio de extensão programática, enquanto ainda suporta interfaces e protocolos de gerenciamento padrão (por exemplo, NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag) conforme [31]. Esse recurso é direcionado para implantações de virtualização de vários servidores.

O *Kubernetes* é um recurso para orquestração de *containers*. É um produto *Open Source* utilizado para automatizar a implantação, o dimensionamento e o gerenciamento de aplicativos em *container*. Ele agrupa *containers* que compõem uma aplicação em unidades lógicas para facilitar o gerenciamento e a descoberta de serviço [32]

Contiv é um recurso que fornece *container* baseado em diretivas para rede. Ele facilita a implantação de microsserviços no seu ambiente, fornece um nível mais alto de abstração de rede para microsserviços e protege seu aplicativo usando uma rica estrutura de políticas [33]

Ansible é um mecanismo de automação de tecnologia da informação (TI) radicalmente simples que automatiza o provisionamento em nuvem, o gerenciamento de configurações, a implantação de aplicativos, a orquestração intra-serviço e muitas outras necessidades de TI conforme [34]. O *Ansible* modela sua infraestrutura, descrevendo como todos os seus sistemas se inter-relacionam, em vez de apenas gerenciar um sistema por vez.

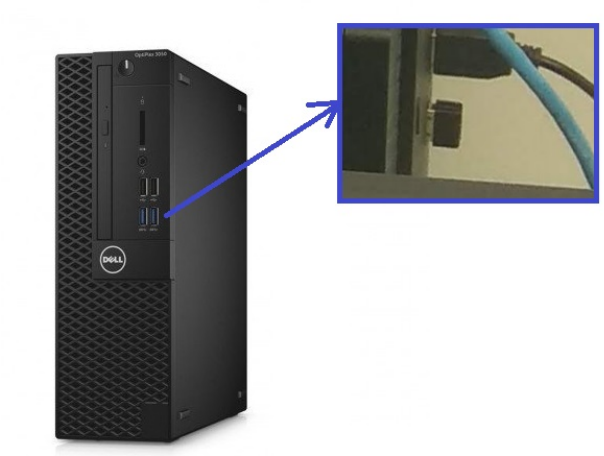
Uma técnica de fatiamento de recurso de rede vem sendo bastante empregada, denominado de *Network Slicing*. Conforme [35] cada fatia de rede representa uma rede ponta a ponta independente virtualizada, permitindo que os provedores de infraestrutura implantem arquiteturas diferentes em paralelo. Esta técnica permite executar várias redes lógicas independentes em uma infraestrutura física compartilhada.

IV. Protótipo SOFTWAY4IoT

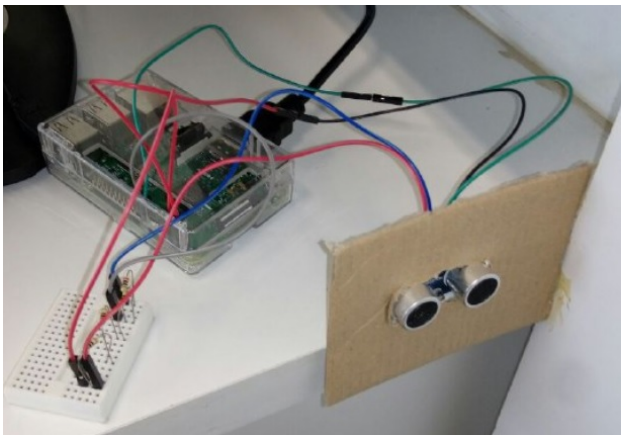
A aplicação IoT utilizada na implantação e avaliação do protótipo é denominada **Fila-RU**. Essa solução é proposta para ser implantada na entrada do Restaurante Universitário da UFG - Campus Samambaia com objetivo de monitoramento da fila do restaurante. A Figura 1 apresenta uma captura de tela da aplicação disponibilizada ao usuário. Esta aplicação foi desenvolvida em linguagem HTML-*HyperText Markup Language* (w3.org), usando JavaScript (*jQuery, Ajax*) (javascript.com) e PostgreeSQL (postgresql.org).

A. Implantação e avaliação

A implantação do protótipo consiste num dispositivo IoT (sensor ultrassônico) que encaminha os valores de presença constantemente à aplicação servidora presente no *gateway* IoT. Os dados encaminhados são armazenados em um banco de dados para consulta, permitindo aos interessados acessarem a aplicação disponibilizada ao usuário e consultar o estado da fila.



(a) Mini PC usado como *gateway*



(b) Sensor ultrassônico ligado ao Raspberry

Fig. 2. Equipamentos usados nos experimentos.

A Figura 2 mostra os equipamentos usados nos experimentos. O protótipo utiliza um *gateway* IoT virtualizado com uma interface sem fio para conexão dos dispositivos IoT. A comunicação é estabelecida por meio da tecnologia Wi-Fi 802.11n na frequência 2.4Ghz. Como *gateway* tem-se um *desktop* utilizando Sistema Operacional Debian 9.8 com processador Intel Core i5 7500 e 16Gb de RAM com um *Dongle* D-link Wi-Fi convencional como adaptador de rede Wi-Fi.

O dispositivo IoT utiliza um Raspberry Pi 3 Modelo B de processador Quad Core 1.2GHz com 1GB de Memória RAM. Um sensor ultrassônico plugado via GPIO (*General Purpose Input/Output*) faz leituras de distâncias relativas ao sensor quando uma pessoa se aproxima. Os dados das leituras feitas pelo sensor são enviados utilizando o protocolo MQTT, publicando no canal MQTT da aplicação que se situa num *container* da VM de aplicação do *gateway*.

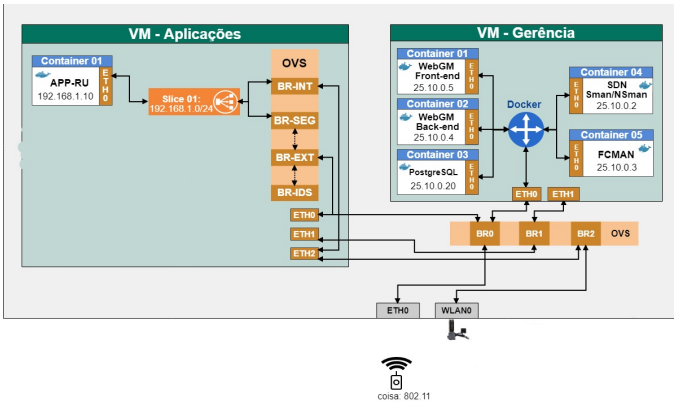


Fig. 3. Arquitetura do projeto SOFTWAY4IoT.

A Figura 3 representa a arquitetura disponibilizada para a implantação do SOFTWAY4IoT. O protótipo tem dois tipos de serviços computacionais virtualizados: VM (Máquina Virtual) de gerência e outra de aplicação. A VM de aplicação permite que as aplicações sejam implantadas em *containers*. A VM de gerência consiste nos serviços de infraestrutura de rede que estão relacionados a recursos que podem ser utilizados pelo próprio *gateway*.

Dentro da VM de gerência temos vários aplicativos. O módulo FCMan permite gerenciar os recursos computacionais virtualizados, oferecendo a capacidade de instanciar aplicações. O módulo SDN (OVS - *Open vSwtch*) é responsável por gerenciar a configuração dos componentes que devem formar a infraestrutura da rede. O módulo é dividido em dois componentes, SMan (*Slice Manager*) e NSMan (*Network Security Manager*), que são responsáveis por possibilitar o isolamento dos dispositivos IoT presentes na rede para cada aplicação, além de controlar as políticas de segurança. O componente SMan é responsável pelo fatiamento da rede em *slices* (*Network Slicing*), devendo isolar e agrupar as interfaces de comunicação sem fio e recursos computacionais virtualizados. NSMan (*Network Security Manager*) é usado para implementar políticas de segurança para cada fatia de rede. Este componente permite que sejam adicionadas ou modificadas regras de encaminhamento viabilizando o acesso adequado de/para cada fatia virtual de recursos conforme a demanda de

cada usuário ou aplicação. O componente WebGM (*Web-based Gateway Manager*) oferece ao administrador da infraestrutura, a interface para gerenciar todas as funções do *gateway* de comunicação e da computação em neblina. É através da interface do WebGM que são fornecidas e configuradas as informações/parâmetros necessárias para o funcionamento adequado dos demais componentes do sistema.

Na arquitetura apresentada pela Figura 3 o sensor envia os dados (a partir do dispositivo IoT via Wi-Fi) para o *gateway*. Os dados são recebidos no *gateway* por meio do OVS. Ao receber os dados, a VM - Gerência fica responsável pela criação de *slices* pelo SMan, implementar as políticas de segurança para cada canal, pelo NSMan e o módulo FCMan ficará responsável pela instanciação de *containers Docker*. Os dados são armazenados em um *container* com o *PostgreSQL* [36]. A aplicação denominada **Fila-RU** fica armazenada em um *container* na VM - Aplicação, comunicando com o banco de dados através de um *slice* isolado. Neste caso, as duas VMs estão em uma mesma máquina/hardware.

Para os experimentos realizados com objetivo de testar o protótipo (*gateway)* foi desenvolvido um *script* em *Python3* [37], com uso de algumas bibliotecas e outras ferramentas, para coleta de dados como tráfego de rede e consumo de CPU/memória que serão usados nas avaliações do protótipo. A biblioteca *scapy* [38] foi usada para captura de arquivos do tipo .pcap, que contém informações sobre o tráfego de rede. A biblioteca *Psutil* [39] é usada para capturar as informações sobre uso de CPU e memória. A ferramenta *tshark* [40] analisa tráfego de rede a partir dos arquivos .pcap gerados pelo *script*. Ao final da execução, utiliza-se o *Matplotlib* [41] para gerar relatórios e gráficos preliminares.

Para gerar o tráfego, foi utilizado a ferramenta *tcpreplay* [42] que é configurada para reinjetar tráfego na rede, tendo como base uma amostra capturada da comunicação estabelecida entre o sensor e o *gateway* na rede Wi-Fi. O tráfego capturado foi retransmitido em *loop* por um período de 3 horas.

O *script* desenvolvido captura dados de tráfego na rede a cada 5 minutos e cria um arquivo .pcap com esses dados que serão analisados posteriormente pela ferramenta *tshark*. A cada hora temos 12 arquivos .pcap levando em conta que a captura se dá em intervalos de 5 minutos. A análise em questão, extrai dados que são usados para calcular os parâmetros de desempenho na comunicação entre o dispositivo IoT e o *gateway*. Os experimentos foram realizados durante 3 (três) horas tendo resultados definidos a cada intervalo de 1 (uma) hora e plotado nos gráficos.

No dispositivo IoT (Raspberry Pi), existe uma aplicação *Python* que publica mensagens no canal MQTT com tamanhos de aproximadamente 90 bytes nos tempos de 0,5s, 1s e 2s. Tal frequência, dita a intensidade de leituras no sensor ultrassônico, consequentemente gera mais pacotes de rede, servindo como base de comparação para aplicações de envio de dados mais intensos.

B. Análise dos Resultados Obtidos

Os resultados obtidos são analisados ponderando a perda de pacotes, vazão média, atraso médio, pacotes em função do tempo, consumo de CPU e RAM, *builds* de imagem da aplicação fila e tempo de instanciação de *containers*.

A análise do tráfego de rede permite uma avaliação do funcionamento da solução desenvolvida. Com os dados do tráfego de rede (vazão, perda de pacotes, atraso médio, pacotes x tempo) conseguimos avaliar a capacidade do *gateway* de comunicar com os dispositivos IoT. O consumo de CPU e RAM, *builds* de imagem da aplicação fila e tempo de instanciação de *containers* permite avaliar a capacidade do SOFTWAY4IoT para tratar os dados recebidos dos dispositivos IoT.

Após a coleta de dados, os cálculos são realizados pelo *script* desenvolvido. Os dados são coletados a partir do arquivo .pcap gerado pelo *script* que usa o *tcpreplay*.

Durante a transmissão há pacotes que são perdidos, os quais serão retransmitidos. Para analisar a perda de pacotes são coletados os dados referentes a quantidade de pacotes perdidos (qtd_{pctprd}), quantidade de pacotes total (qtd_{totpct}) e quantidade de arquivos .pcap (qtd_{pcap}) - usado para calcular a média aritmética conforme Equação 1. É calculada a razão entre os pacotes perdidos e pacotes total a cada 5 minutos. A razão é somada no intervalo de 1 hora e calculada a média aritmética, obtendo a média de perda de pacote a cada hora de tráfego.

$$perdadepacotes = \sum(\sum qtd_{pctprd} \div qtd_{totpct}) \div qtd_{pcap} \quad (1)$$

Onde qtd_{pctprd} representa a quantidade de pacotes perdidos, qtd_{totpct} representa a quantidde total de pacotes trafegados e qtd_{pcap} representa a quantidade de arquivos .pcap usados no intervalo de tempo considerado.

Os resultados apresentados na Figura 4, indicam uma conexão estável e com uma taxa de perda de pacotes aceitável, sendo o maior percentual de perda apresentado nos experimentos de 1,5%. Entende-se que a perda de pacote reflete em retransmissão, e neste caso, temos um percentual tolerável para a retransmissão.

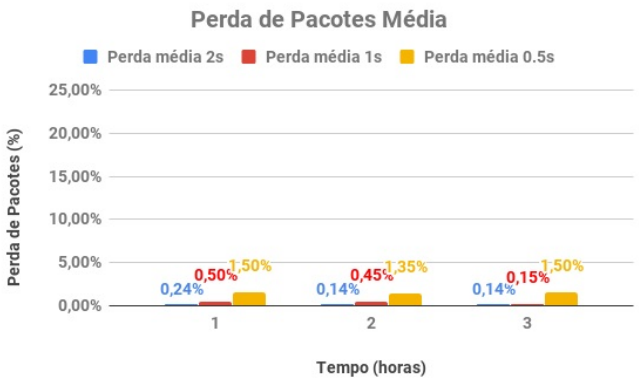


Fig. 4. Perda de Pacotes / Tempo.

A análise de vazão média é realizada a partir da quantidade de bits trafegados pela rede. É somada a quantidade total de bits (*bits*) e dividido pelo intervalo de tempo (tmp_{seg}), conforme Equação 2. Nesse caso, o intervalo de tempo considerado é de 5 minutos. Na sequência, somam-se os valores calculados num intervalo de 1 hora e obtém uma média aritmética. A vazão média é dada em bps (bits/seg). Quando o sensor envia dados a cada 2s temos uma intensidade de tráfego de rede. Se o intervalo de tempo diminui (frequência) para 0,5s esta intensidade aumenta, ou seja, mais dados são enviados ao *gateway*. Como esperado, conforme o aumento da intensidade de tráfego de rede, a vazão também aumenta. Com o aumento da intensidade de tráfego de rede teve-se uma variação da vazão média de 34,24 bps chegando até a 166 bps. O padrão de rede 802.11n pode chegar a uma vazão de até 600 Mbps. Observou-se que o sistema manteve estável gerenciando o fluxo de tráfego de rede.

$$vazao = \sum(\sum bits \div tmp_{seg}) \div qtd_{pcap} \quad (2)$$

Onde *bits* representa a quantidade de bits trafegados pela rede no intervalo de tempo considerado, tmp_{seg} representa o intervalo de tempo considerado no levantamento de dados e qtd_{pcap} representa a quantidade de arquivos .pcap usados no intervalo de tempo.

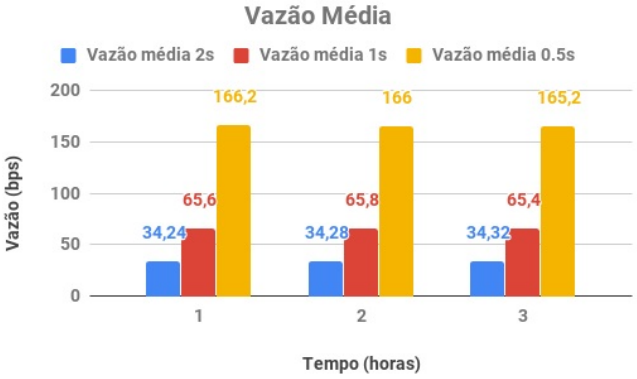


Fig. 5. Vazão média / Tempo.

O atraso médio foi levantado a partir do cálculo da diferença entre o tempo de recebimento do último pacote em relação ao pacote atual. Os tempos foram somados (tmp_{atrpct}) e divididos pelo tempo total (tmp_{totms}) a cada 5 minutos, conforme Equação 3. A razão calculada foi somada, e então, obtida a média do atraso num intervalo de 1 hora. O atraso médio é dado em milissegundos. Os resultados mostram que, com o aumento do fluxo de envio de pacotes o atraso médio foi menor, mas este atraso se mantém estável ao longo do tempo. No período de três horas, o atraso médio se manteve praticamente constante como mostrado na Figura 6. Com o sensor enviando dados a cada 2s, temos uma média entre 243,5ms e 241,75ms, uma variação pequena ao longo de 3 horas. Aumentando o envio de dados, ou seja, a cada 1s temos um atraso médio de aproximadamente 129ms e a cada 0,5s enviando dados o atraso médio é de aproximadamente 49,5ms.

$$atrasomedio = \sum(\sum tmp_{atrpct} \div tmp_{totms}) \div qtd_{pcap} \quad (3)$$

Onde tmp_{atrpct} representa o tempo de atraso de um pacote que é dado pela diferença do tempo de recebimento do último pacote em relação ao pacote atual, tmp_{totms} representa a soma dos tempos de atraso do pacote num intervalo de 5 minutos e qtd_{pcap} representa a quantidade de arquivos .pcap usados no intervalo de tempo.

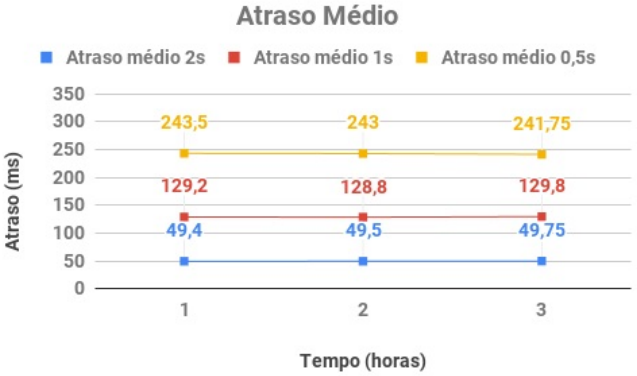


Fig. 6. Atraso Médio / Tempo.

A Figura 7 apresenta os resultados obtidos para a quantidade média de pacotes trafegados a cada hora de tráfego (pct_{tmp}) conforme Equação 4. Esta coleta se deu num intervalo de 3 horas.

Para intervalo de envio de 0,5s causou um aumento significativo de pacotes enviados. A justificativa é devido uma maior intensidade de tráfego na rede e consequentemente um aumento na taxa de perda de pacotes.

$$pacotesvstempo = \sum pct_{tmp} \quad (4)$$

Onde pct_{tmp} representa a quantidade de pacotes trafegados pela rede a cada 5 minutos.

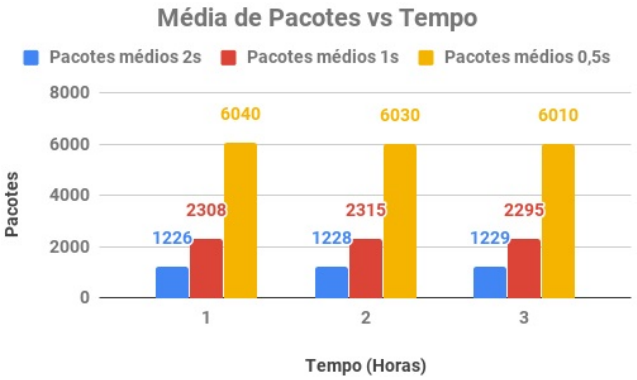


Fig. 7. Média de Pacotes / Tempo.

Os dados referentes ao consumo de CPU e memória são levantados pelo *script* a partir da biblioteca *psutil* do *Python*. Conforme exibido na Figura 8, nos relatórios de consumo de CPU foi possível observar os picos de processamento no dispositivo IoT num dado instante. Os picos são referentes a algum processo de sistema. Enquanto os relatórios de uso de Memória RAM apresentados na Figura 9 se mostraram estáveis.

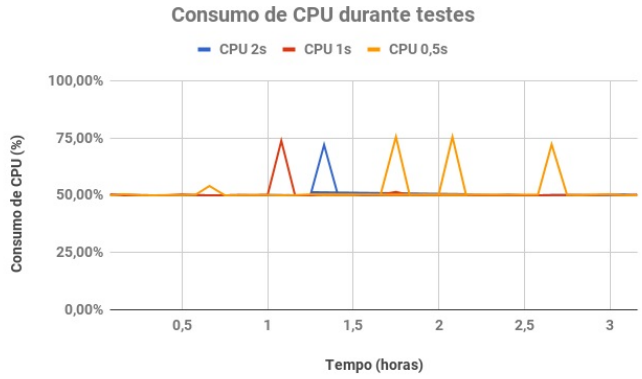


Fig. 8. Consumo de CPU ao longo dos experimentos.

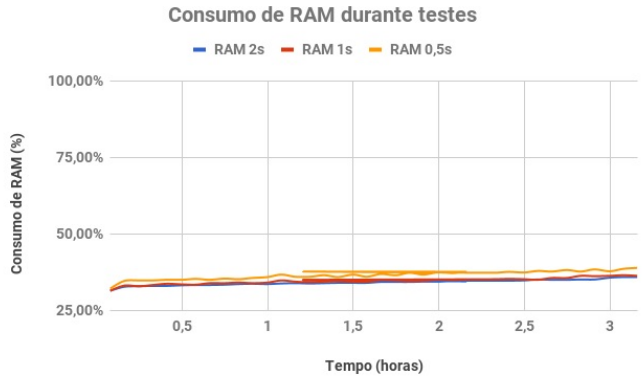


Fig. 9. Consumo de RAM ao longo dos experimentos.

A tecnologia de computação em neblina implementada no protótipo foi avaliada a partir de experimentos relativos à efetividade de instanciação de *containers* e construção de imagens (*build* de imagem) *Docker*. Quando tratamos de tempo de *build* de imagem refere-se ao tempo gasto para a construção da imagem pretendida. Uma imagem é um pacote leve e executável que engloba tudo que é preciso para executar um pedaço da aplicação. A instância de um *container* refere-se a um *container* que foi instanciado a partir de uma imagem já construída. Uma imagem pode ser usada para instanciar vários *containers*.

Foram feitos 10 experimentos sobre a aplicação Fila-RU, utilizando-se da API REST [43] do módulo FCMan, obtendo assim aproximadamente o mesmo tempo de execução em segundos que o acesso via interface web do *gateway*. O tempo *build* de imagem *Docker* variou entre 6s e 7,2s.

Sobre os mesmos experimentos foram extraídos os seguintes dados acerca da instanciação de *container*: tempo médio em segundos, desvio padrão em segundos e variância em segundos. O tempo (em segundos) de instanciação de *containers Docker* é exibido pela Figura 10, mostrando-se satisfatório, não sendo significativamente prejudicados em virtude da máquina virtual e alto isolamento de recursos. Seu tempo médio foi de 1.78s, desvio padrão de 0.10s e variância de 0.01s.

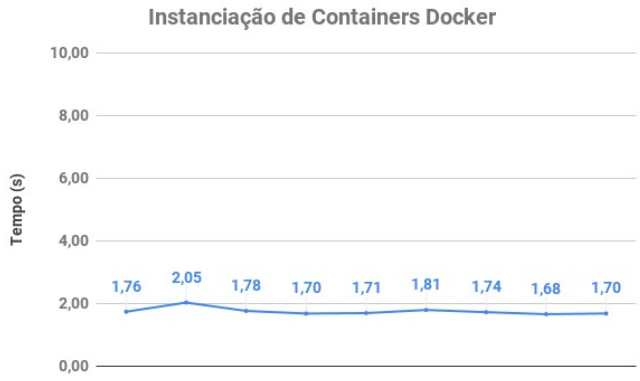


Fig. 10. Instanciação de *containers* da aplicação Fila.

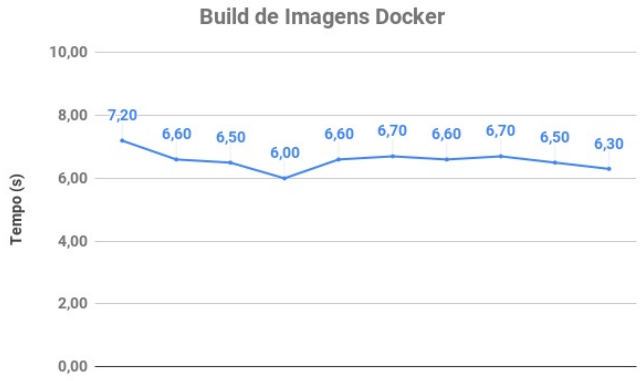


Fig. 11. *Builds* de Imagem da aplicação Fila.

Para a construção de imagens *Docker* [44] foram tomadas as precauções devidas para o não armazenamento em *cache* dos dados e a não reaquisição de dependências. Portanto, os dados coletados foram obtidos a partir de um *build* limpo e nenhum *download* foi realizado durante os experimentos. Conforme a Figura 11, o tempo de construção das imagens foram baixo levando em conta o fato da aplicação escolhida ser relativamente simples. Seu tempo médio foi de 6.57s, desvio padrão de 0.29s e variância de 0.08s.

V. Piloto - SOFTWAY4IoT

COMO apresentado na seção IV foram feitos experimentos que analisam o desempenho do SOFTWAY4IoT usando apenas um *gateway* para estabelecer a

comunicação. Na etapa do piloto o foco é a implementação de múltiplos *gateways*, visto que a implantação foca em ambiente de campus inteligente sendo uma área extensa que demanda um número maior de *gateways* para atender os serviços necessários.

O SOFTWAY4IoT foi projetado e desenvolvido de maneira modular, permitindo que diferentes conjuntos de componentes sejam usados. Além disso, os componentes podem ser executados em um único equipamento, como no protótipo apresentado, ou em múltiplos equipamentos como no piloto.

A. Arquitetura do sistema

Nesta fase, a implementação, avaliação e orquestração usaram múltiplos equipamentos. No piloto apresentado temos *gateway* genérico com módulos de rede, segurança, virtualização e interface, como mostra a Figura 12, e tem equipamento como *gateway manager* com módulos de segurança, rede, virtualização, orquestração, configuração e gerenciamento como mostra a Figura 13.

A interface de gerenciamento e configuração da infraestrutura do sistema é feita a partir do módulo WebSM. Esse módulo foi desenvolvido pelo grupo de trabalho GT-SOFTWAY4IoT e tem como objetivo administrar os principais componentes do sistema. Através dele é possível criar/editar *slices*; para cada *slice*, configurar os *gateways*, as aplicações, os dispositivos IoT; habilitar a tecnologia de comunicação sem fio e a conectividade do *slice* com a rede externa. O WebSM é basicamente a principal ferramenta a ser utilizada por parte de um administrador que faz uso do SOFTWAY4IoT.

O *Ansible* [1] é a ferramenta utilizada para automatizar todo o processo de implantação, permite descrever a infraestrutura como código, é uma ferramenta que trás a ideia de 'cultura ágil' largamente difundida em ambientes de engenharia de *software* para o ambiente de infraestrutura.

O módulo de orquestração do sistema é composto pelos componentes OrchSW4IoT, *Kubernetes* e o *Contiv*. O OrchSW4IoT é o componente de orquestração centralizada, responsável por configurar os recursos computacionais (com o *Kuberentes*) e de rede (com o *Contiv*), além de oferecer uma API Rest (*apiserver*) para realizar integração com outros módulos, como o PhySec e WebSM. O OrchSW4IoT é basicamente constituído de uma série de 'tarefas agendadas' que a cada n intervalos de tempo realiza uma série de verificações na infraestrutura controlada pelo SOFTWAY4IoT. Dentre as principais atividades executadas pelo elemento de orquestração estão: criação de aplicações, verificação de *status*, configuração de *slices*, entre outros.

O *Kubernetes* foi originalmente criado com o intuito de orquestrar e gerenciar *clusters* de *containers*, eliminando a maior parte dos processos manuais que eventualmente são necessários para implantar e escalar aplicações containerizadas. Em outras palavras, quando se tem a necessidade de agrupar em *clusters* os *hosts* executados em *containers* (que podem estar em *clouds* públicas, privadas, hibridas, etc), utiliza-se o *Kubernetes* para facilitar o trabalho. O *Kubernetes* não executa containers diretamente, ele envolve um ou mais containers em uma estrutura chamada *pod*. Qualquer container no mesmo *pod* compartilhará os mesmos recursos. Através dessa abordagem é possível solucionar a maioria dos problemas associadas a proliferação de *containers*. Os *pods* não são lançados diretamente em um *cluster*, eles tem uma camada de abstração chamada *deployment* que tem como objetivo gerenciar os *pods* em execução.

A virtualização é um recurso com uso expressivo neste projeto. O módulo de virtualização conta com os componentes *Open vSwitch* (OVS), *Kata Container*, e *Docker*. O OVS é um *switch* virtual que pode, por exemplo, conectar os '*pods*' do *Kubernetes*, possibilitando isolar ou redimensionar recursos de rede. O propósito é o de transportar dados entre portas, em que cada uma delas pode estar conectada a outros componentes da rede (VMs, *switch*, etc). Tudo é feito através da utilização de tabelas de fluxo, cada uma das quais pode ser caracterizada como uma lista de regras ordenadas conforme uma prioridade específica.

No contexto do projeto, o *Contiv,* além de entregar IP para cada *container*, vai entregar IP também para os dispositivos IoT. Foram adicionados funções ao *Contiv* com a finalidade de entregar IP para os dispositivos e *containers*. A solução é integrada com uma aplicação que gerencia um *switch* OVS.

O *Docker* possibilita empacotar uma aplicação ou ambiente inteiro dentro de um *container*, tornando o respectivo ambiente portável a qualquer outro *host* que tenha uma versão *Docker* instalada. A utilização de *containers* reduz de forma considerável o tempo de *deployment* de alguma infraestrutura ou aplicação; uma vez containerizado, não há necessidade de ajustes no ambiente para o correto funcionamento do serviço, pois o ambiente será sempre o mesmo; configura-se uma vez e utilize quantas vezes for necessário. No escopo do SOFTWAY4IoT os principais artefatos de *software* foram containerizados (WebSM, ApiServer, Apps, etc).

O componente *Kata Container* é utilizado para realizar o isolamento das instâncias de *containers* em execução. Este componente permite a criação de máquinas virtuais leves e coloca *containers* dentro conectando à plataforma de orquestração.

AgentSW4IoT é o módulo executado em cada *gateway* IoT. Ele é composto de drivers IoT (IoTDrivers), de aplicações SDN (SDNApps) e do componente de configuração e monitoramento dos *slices* (NetSlice). Os Drivers IoT são os componentes que habilitam os *gateways* a se comunicarem com as interfaces sem fio instaladas neles, como, Zigbee, LoRa, Wi-Fi ou NRF24. Desta forma, para dar suporte a novas tecnologias de comunicação sem fio (e.g., Z-Wave), basta implementar um novo Driver IoT. As aplicações SDN são de DHCP (integrada ao *Contiv*) e de *Learning Switch* para conectar os dispositivos IoT com a rede de computação do *Kubernetes*.

[1] https://www.ansible.com/

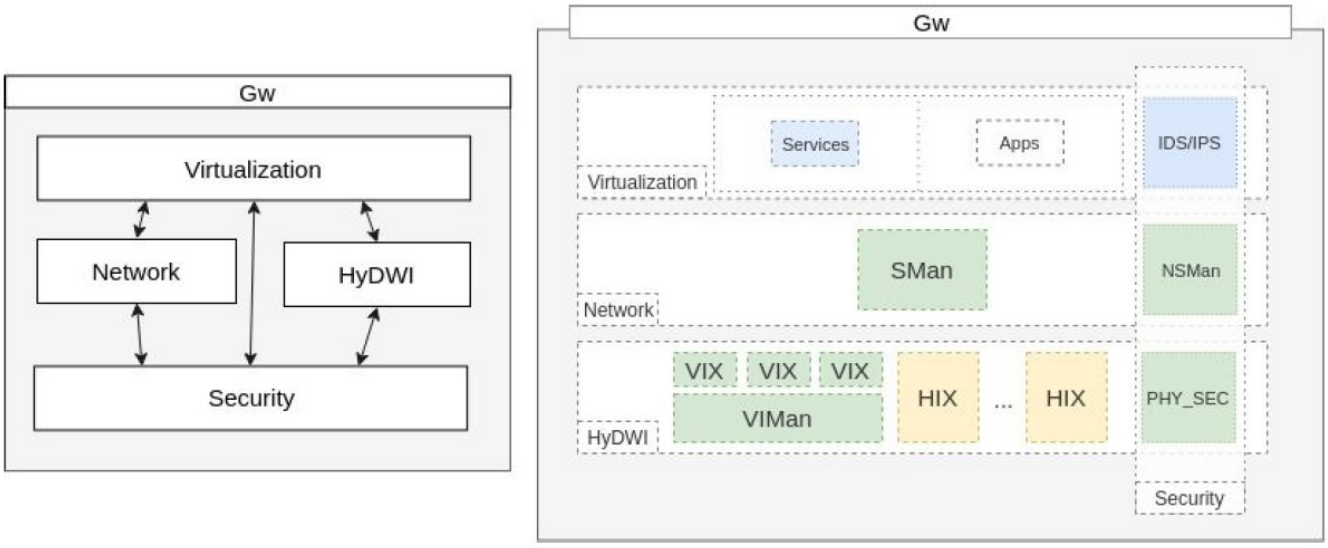


Fig. 12. Arquitetura do *Gateway* genérico usado no SOFTWAY4IoT.

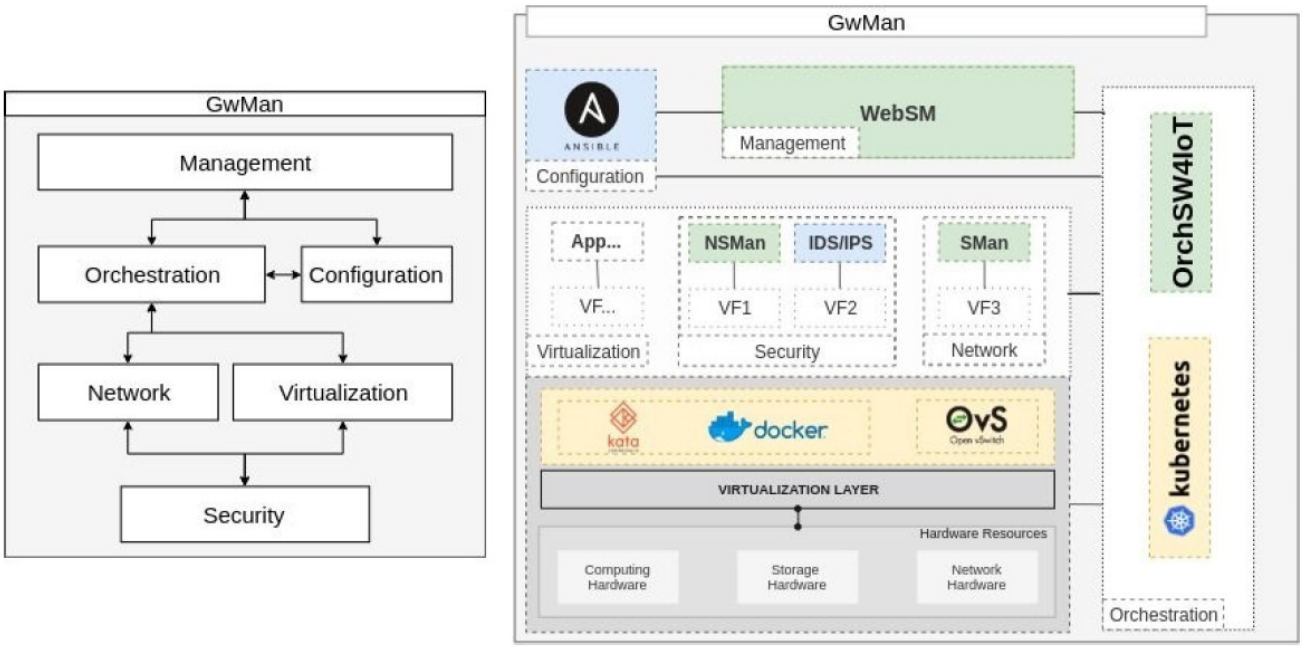


Fig. 13. Arquitetura do *Gateway* manager usada no SOFTWAY4IoT.

No módulo de rede temos o componente SMan que é o responsável pelo fatiamento da rede em *slices*, devendo isolar e agrupar as interfaces de comunicação sem fio e recursos computacionais virtualizados.

No módulo de segurança temos os componentes NSMan, IDS/IPS e Phy-Sec. NSMan (*Network Security Manager*) é o componente usado para implementar políticas de segurança para cada fatia de rede. Este componente permite que sejam adicionadas ou modificadas regras de encaminhamento permitindo o acesso adequado de/para cada fatia virtual de recursos conforme a demanda de cada usuário ou aplicação. O componente IDS/IPS tem a função de analisar o tráfego da rede e detectar uma possível intrusão. Neste recurso utilizou-se a ferramenta Snort como IDS/IPS.

O Phy-Sec é um componente do módulo de segurança que atua de maneira inteligente para a detecção de intrusão. Ele é responsável por realizar a captura, processamento, treinamento e classificação dos sinais eletromagnéticos para identificar os dispositivos IoT. Este componente analisa o sinal através do SDR criando um padrão de reconhecimento de cada dispositivos. Assim, um intruso na rede que tenha clonado o endereço MAC de algum dispositivo será detectado por este recurso, uma vez que, o clone não terá o mesmo sinal padrão do dispositivo real.

B. Implantação do piloto

O SOFTWAY4IoT foi implantado no Campus Samambaia da Universidade Federal de Goiás, mais especificamente, no Instituto de Informática (INF), a Biblioteca (BC) e o Restaurante Universitário (RU). O caso de uso envolve o gerenciamento inteligente de resíduos orgânicos no RU, gerenciamento de níveis de lixo sólidos em lixeiras seletivas e monitoramento por vídeo no INF e de medições de umidade, temperatura e nível de gás na BC.

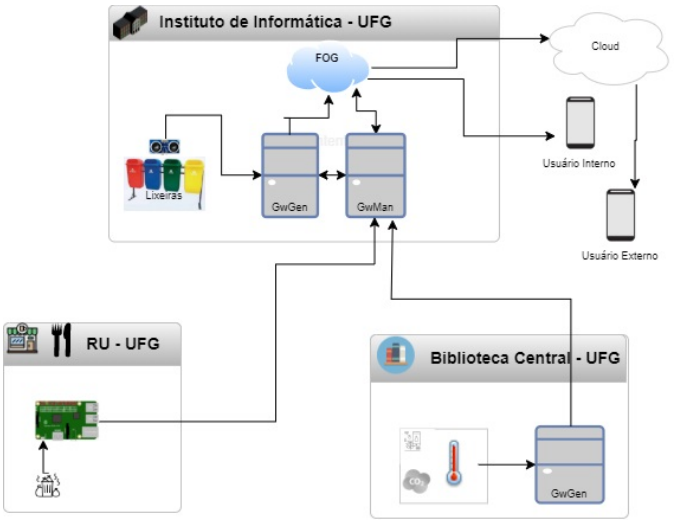


Fig. 14. Cenário de avaliação do SOFTWAY4IoT.

No laboratório de computação (Computer Networks and Distributed Systems Laboratory - LABORA) do INF foi instalado e configurado o *Gateway Manager* - GwMan. O GwMan foi configurado em dois computadores (PC completo) com processador Intel Core I7 com 16 Gb de RAM, sendo um master e outro de virtualização. Na entrada (área externa) do INF foram disponibilizadas 4 (quatro) lixeiras (vidro, papel, plástico e metal) com sensores ultrassônicos para verificar o nível de lixo. Na parte interna da porta tem-se uma câmera IP para monitoramento das lixeiras, como mostra a Figura 15. Para estes dispositivos disponibilizou-se um PC com processador Intel Core I7 de 16 Gb de RAM como *gateway* genérico usando sinal Wi-Fi para comunicação.

Fig. 15. Lixeira com sensor e monitorada por câmera.

Fig. 16. Balança com células de carga para pesagem de lixo orgânico do RU

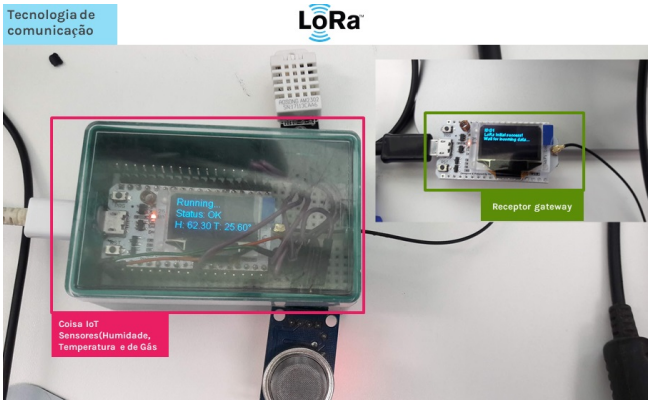


Fig. 17. Sensor de umidade, temperatura e gás da Biblioteca Central - Campus Samambaia

No restaurante universitário, verificou-se a necessidade de mensurar a quantidade de resíduos orgânicos que são descartados diariamente. Para este fim, foi disponibilizado uma balança com 4 células de carga que coletam o peso das lixeiras utilizadas para descarte de resíduo orgânico como mostra a Figura 16. No restaurante universitário foi colocado um *gateway* genérico usando um Raspberry Pi3 B ARM 1Gb, que recebe os dados dos sensores e encaminha para uma aplicação na *Fog*, estabelecendo a comunicação através de um dispositivo Lora instalado no Raspberry.

Foram instalados sensores de temperatura, umidade e gás, que se comunicam através da tecnologia Lora, no ambiente da Biblioteca Central do Campus Samambaia. Estes dispositivos foram ligados a um computador (Mini PC) com processador Intel Core I3 com 8 Gb de RAM. A Figura 17 apresenta os dispositivos utilizados na biblioteca.

Os dados coletados pelos sensores são enviados a uma aplicação na *Fog* por meio de comunicação sem fio. Nesta implantação foi utilizado as tecnologias de comunicação Wi-Fi e Lora.

C. Testes realizados

Os experimentos realizados analisam a implantação do *gateway*, a comunicação entre os componentes, a criação

de *slices*, a conectividade dentro do *slice* e a implantação da aplicação piloto.

O teste de implantação do *gateway* visa avaliar a implantação de todos os módulos necessários para deixar um *gateway* totalmente operacional. Através do WebSM é executado a implantação do *gateway* e analisado a conectividade com o GwMan. Como resultado, espera-se que todos os módulos e serviços estejam instalados no *gateway* sendo este controlados pelo GwMan. Toda implantação foi realizada com a execução de um *playbook Ansible* de instalação, via terminal. Obtiveram-se os resultados esperados com todos os módulos instalados no PC Completo (INF), no Mini PC (BC) e no Raspberry Pi (RU).

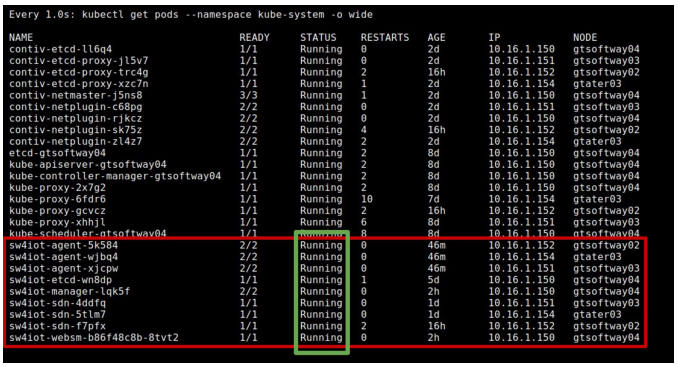


Fig. 18. Captura de tela mostrando os módulos implantados em execução.

A Figura 18 apresenta um teste de checagem básica cujo objetivo é verificar se os módulos de orquestração, gerenciamento, configuração, virtualização, rede e segurança estão sendo executados no *gateway* e no GwMan. A imagem mostra os módulos com o *status* em execução (*Running*). Os testes foram realizados e constatado que todos os módulos foram instalados e estão sendo executados corretamente. O GwMan (Master) é o nó (NODE) gtsoftway04, o GwMan (virtualização) é o nó gtsoftway03, e os nós gtater03 e gtsoftway02 são os *gateway* genéricos instalados na secretaria do INF e na biblioteca central do campus respectivamente.

Um teste visando a criação e configuração básica de um *slice* também foi realizado. Através da interface do componente WebSM foi feita a criação do *slice*, e a adição dos *gateways*, dispositivos IoT e conectividade do *slice*. O teste foi bem sucedido e seu intuito era averiguar a comunicação do componente WebSM com o orquestrador para solicitar a configuração de todos os componentes envolvidos. O WebSM se comunica com o orquestrador via API REST, e adiciona: o slice, os gateways (no slice piloto), os dispositivos; habilita a conectividade do slice com comunicação Wi-Fi e seleciona os gateways para que os dispositivos IoT se conectem, como ilustra a Figura 19.

Verificou-se a conectividade de um dispositivo IoT de um determinado *slice* com uma aplicação que está sendo executada no *gateway* mais próximo e no GwMan. Ao adicionar o gateway no *Slice* Piloto (id = d96d47) a *bridge* "br-d96d47" é criada para esse *slice* e a interface

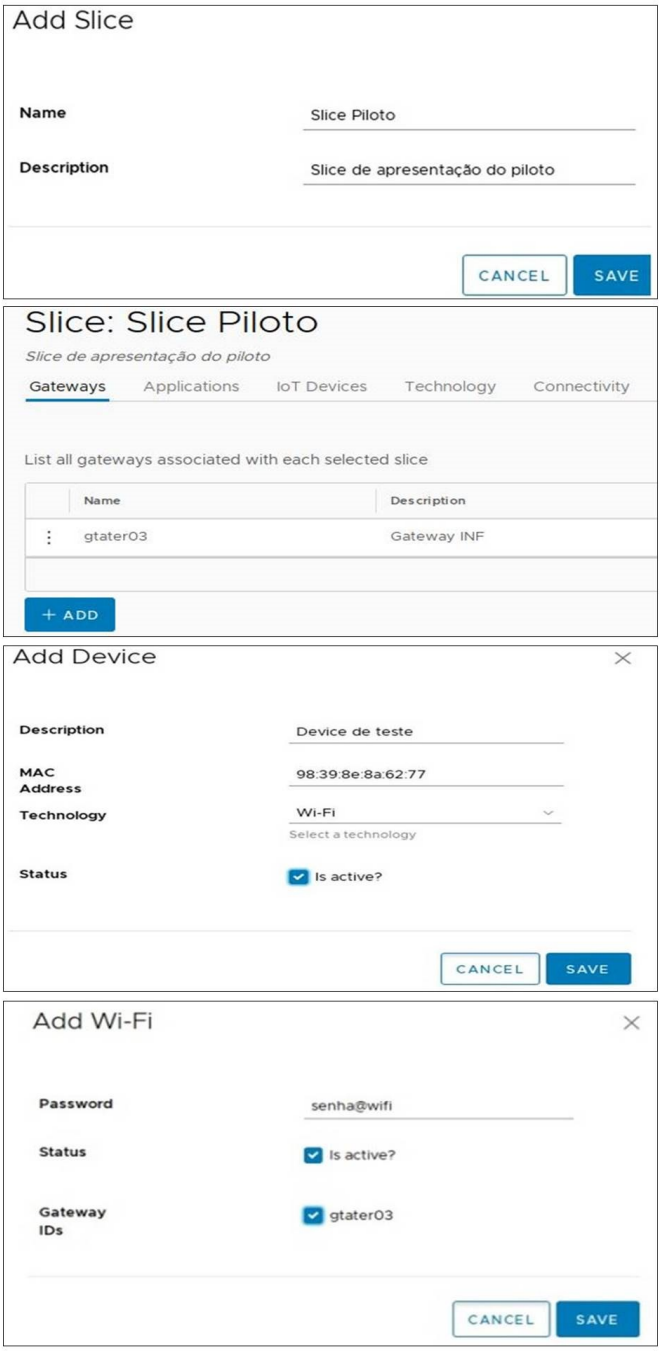


Fig. 19. Criação e configuração básica de slice a partir da aplicação WebSM - 1º - adiciona/cria *slice*, 2º - adiciona *gateway* ao *slice*, 3º - adiciona um dispositivo IoT, 4º - autentica/ativa a tecnologia de comunicação.

"wlans_d96d470" do AP Wi-Fi é criada no gateway, indicando assim que o AP Wi-Fi está executando e que os dispositivos IoT Wi-Fi podem se conectar. Constata-se através do comando *ping* que a comunicação foi estabelecida e o dispositivos IoT está acessando uma rede externa como mostra a Figura 20.

Logo após, utilizando o mesmo *slice*, adicionou-se um *container* de serviço no GwMan verificando a conectividade do dispositivo IoT com o mesmo. O teste, cujo resultado esperado era a comunicação efetiva do dispositivo

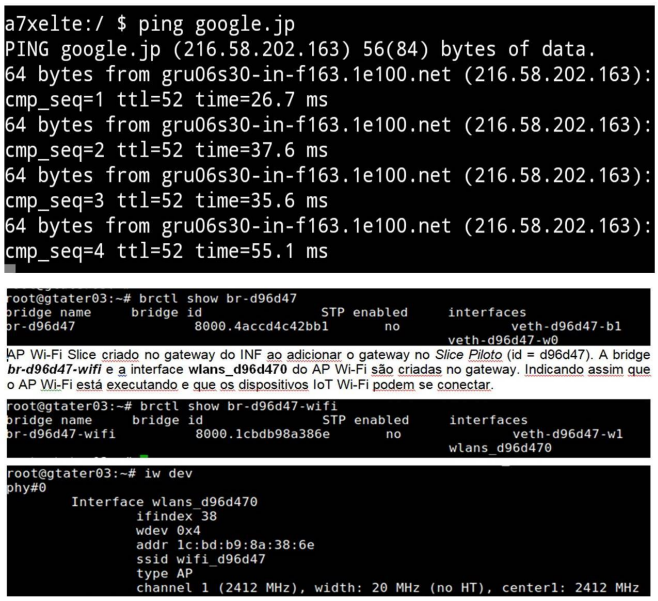


Fig. 20. A figura mostra a *bridge* (br-d96d47) e a interface do AP Wi-Fi (wlans_d96d470) criadas no *gateway* e constata através do ping a comunicação com a rede externa.

IoT de forma isolada com *containers* de serviço, sendo executado nos *gateways* foi realizado. O dispositivo IoT se comunica com a aplicação (*container*) de serviço que está em execução no *slice*, conforme Figura 21.

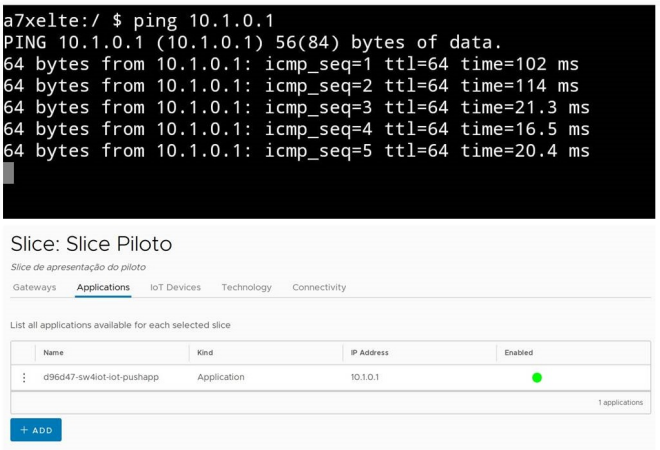


Fig. 21. Teste de comunicação do dispositivo IoT de forma isolada com *containers* de serviço, sendo executado no *gateway*

O piloto foi implantado e testado com sucesso. Na implantação do piloto foram usados 3 *gateways* genéricos, um *gateway manager* e os módulos do sistema. Usou-se o WebSM para a implantação dos *gateways* e instalar a aplicação de gestão inteligente de resíduos sólidos.

A Figura 22 apresenta uma aplicação sendo executada dentro do *container* no *slice* selecionado. Temos os dados que foram enviados pelos sensores da Biblioteca Central e do Restaurante Universitário para o banco de dados. Os dados enviados são referentes umidade e temperatura da biblioteca central e umidade, temperatura e peso dos lixos

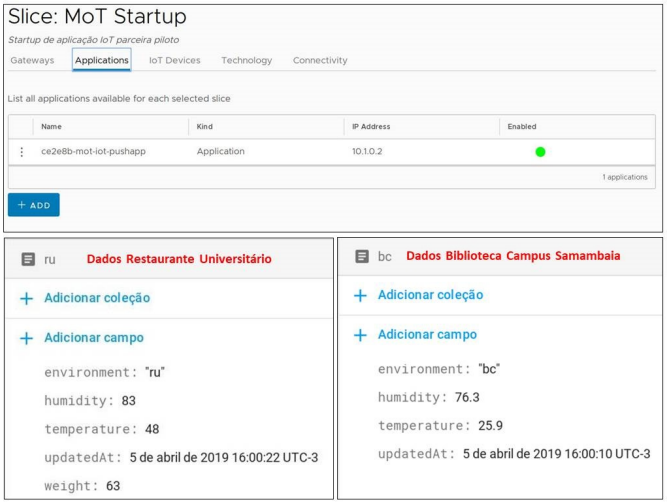


Fig. 22. Aplicação do *slice* selecionado sendo executada no *container* e dados enviados do dispositivo IoT da biblioteca central e do restaurante universitário do Campus Samambaia para o banco de dados

Fig. 23. Gráficos exibindo a variação nos níveis de lixo da lixeira seletiva com os dados enviados pelo sensor ultrassônico instalado na parte superior das lixeiras.

orgânicos do restaurante universitário. A informação de data e hora do envio dos dados também ficam registradas.

A Figura 23 ilustra o gráfico gerado com os dados enviados dos sensores ultrassônicos das lixeiras seletivas (papel, metal, plástico e vidro) que foram instaladas na porta de entrada do Instituto de Informática - UFG. Os sensores enviam dados ao sistema com a variação nos níveis de lixo. As oscilações mostram uma oscilação não comum, nesse caso as variações do nível de lixo foram provocadas para fins de teste de funcionamento.

Por último, a Figura 24 mostra uma imagem capturada da câmera IP instalada na entrada do INF (lado interno) para fim de monitoramento. O *streaming* de vídeo é enviado ao servidor usando a mesma infraestrutura que estabelece a comunicação das lixeiras. O monitoramento por vídeo pode detectar a movimentação na porta do INF e também pode monitorar a presença de macacos na lixeira, visto que nesta área há muitos macacos que circulam a

procura de alimento.

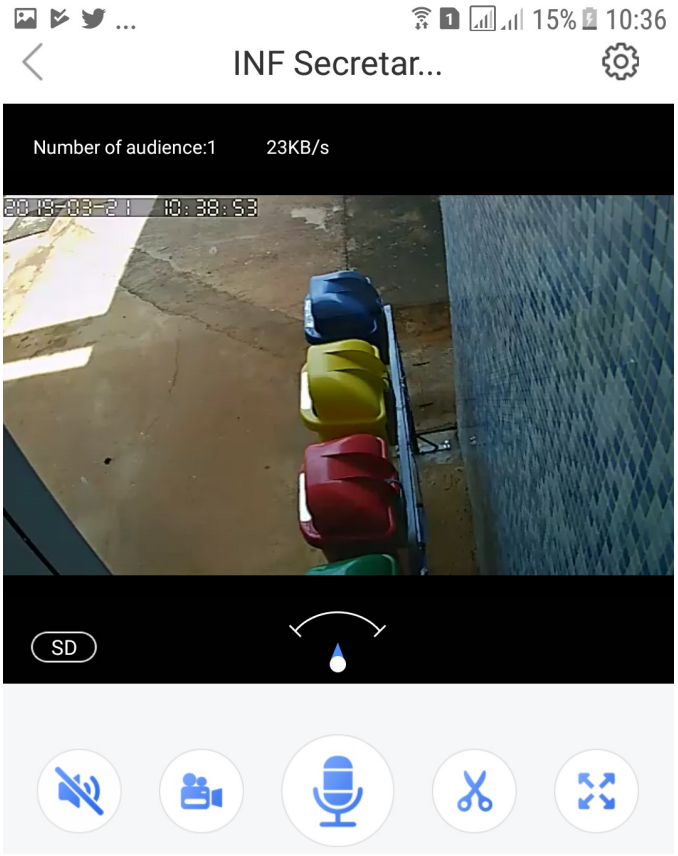


Fig. 24. Dados enviados da lixeira-INF

VI. Considerações Finais

ESTE trabalho propôs uma solução de comunicação para dispositivos IoT, denominada SOFTWAY4IoT, ponderando a heterogeneidade, a distribuição geográfica e a escalabilidade. A abordagem apresenta um único gateway IoT definido por software com uso de virtualização, orquestração, reconhecimento de múltiplas tecnologias de comunicação sem fio, aplicando o fatiamento de rede para separar as tecnologias de comunicação integrando *fog/cloud*. Um protótipo com base na aplicação **Fila RU** foi desenvolvido e avaliado o seu desempenho com as seguintes métricas: vazão média, perda de pacotes, quantidade de pacotes trafegados, atraso médio, consumo de CPU e memória, instanciação de containers e *builds* de imagem. A avaliação do protótipo mostrou resultados favoráveis.

Com resultados positivos neste primeiro momento, propôs-se a implementação de um piloto para o SOFTWAY4IoT em cenário de uso massivo de IoT, com possibilidade de múltiplos *gateways*, múltiplas tecnologias de comunicação em um ambiente de campus inteligente, tendo como base o Campus Samambaia da UFG. Nesse ambiente foram utilizados aplicações de controle de resíduos sólidos, monitoramento de vídeo, controle de níveis de lixo em lixeiras seletivas, coleta de dados de temperatura, umidade e gás.

Os testes realizados com o piloto usou 3 *gateways* genéricos e 1 *gateway master*, para comunicação usou-se os protocolos Lora e Wi-Fi. Os experimentos tiveram o objetivo de avaliar a implantação, checagem básica dos módulos do sistema, a criação/configuração de *slices*, a conectividade do dispositivo IoT com a aplicação que está sendo executada nos *gateways*, e o funcionamento global da solução apresentada. Os resultados mostraram que todos os módulos foram executados de maneira correta e eficiente, viabilizando a comunicação de múltiplas tecnologias com uso de fatiamento de rede, orquestrando os recursos com uso dos 4 *gateways*. A aplicação das tecnologias SDN, virtualização, orquestração, para a comunicação e integração dos ambiente *fog/cloud* permitiu a heterogeneidade, a distribuição geográfica e escalabilidade de aplicações.

Como proposta para trabalhos futuros, sugere-se aplicar técnicas de otimização para melhorar a escalabilidade das aplicações e técnicas de *Machine Learning* para auxiliar no dimensionamento da infraestrutura, considerando o número de dispositivos conectados e para auxiliar na alocação de recursos da *fog/cloud*.

Agradecimentos

Os autores agradecem a Rede Nacional de Ensino e Pesquisa (RNP) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo fomento da chamada Universal 01/2016, Nº do Processo: 431552/2016-9.

Referências

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "Iot gateway: Bridgingwireless sensor networks into internet of things," in *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, Ieee, 2010, pp. 347–352.

[3] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight virtualization: A performance comparison," in *2015 IEEE International Conference on Cloud Engineering*, IEEE, 2015, pp. 386–393.

[4] R. Morabito, I. Farris, A. Iera, and T. Taleb, "Evaluating performance of containerized iot services for clustered devices at the network edge," *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 1019–1030, 2017.

[5] Eclipse Kura, *Eclipse kura | the eclipse foundation*, online, Jul. 2019. [Online]. Available: https://www.eclipse.org/kura/.

[6] Thingsboard, *Thingsboard open-source iot platform*, online, Jul. 2019. [Online]. Available: https : / / thingsboard.io/.

[7] E. M. d. Quadros Júnior, "Avaliação de um modelo para o gerenciamento de recursos em um ambiente iot usando virtualização baseada em contêineres," Master's thesis, Universidade Federal de Pernambuco, 2017.

[8] A. Krylovskiy, "Internet of things gateways meet linux containers: Performance evaluation and discussion," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, IEEE, 2015, pp. 222–227.

[9] K. Cardoso, A. O. Jr, and S. Correa, "Softway4iot: Software-defined gateway and fog computing for iot (internet of things)," *WRNP 2019*, vol. 1, 2019.

[10] D. A. Ferreira Jr, J. P. O. Cabral, C. Macedo, T. A. Santos Filho, K. V. Cardoso, and A. C. Oliveira Jr, "Implantação e avaliação de um protótipo para filas inteligentes utilizando um dispositivo iot wi-fi e um gateway iot definido por software," in *VII Edição da Escola Regional de Informática de Goiás (ERI – GO).*, Goiânia - GO, Nov. 2019, pp. 277–290.

[11] KAYO, Galem, *Turning your raspberry pi 4 into an edge gateway*, online, Jan. 2019. [Online]. Available: https://ubuntu.com/blog/turning-your-raspberry-pi-4-into-an-edge-gateway-part-i.

[12] Intel, *Intel iot gateway*, online, Jan. 2019. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/gateway-solutions-iot-brief.pdf.

[13] Mozilla, *Webthings gateway*, online, Jan. 2019. [Online]. Available: https://iot.mozilla.org/gateway/.

[14] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," *IEEE Internet Computing*, vol. 21, no. 2, pp. 16–24, 2017.

[15] D. Santoro, D. Zozin, D. Pizzolli, F. De Pellegrini, and S. Cretti, "Foggy: A platform for workload orchestration in a fog computing environment," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, IEEE, 2017, pp. 231–234.

[16] R. Vilalta, A. Mayoral, D. Pubill, R. Casellas, R. Martınez, J. Serra, C. Verikoukis, and R. Muñoz, "End-to-end sdn orchestration of iot services using an sdn/nfv-enabled edge node," in *2016 Optical Fiber Communications Conference and Exhibition (OFC)*, IEEE, 2016, pp. 1–3.

[17] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "Fogflow: Easy programming of iot services over cloud and edges for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, 2017.

[18] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for iot," *Wireless Personal Communications*, vol. 92, no. 1, pp. 181–196, 2017.

[19] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource provisioning for iot services in the fog," in *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*, IEEE, 2016, pp. 32–39.

[20] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized iot service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, 2017.

[21] S. Guoqiang, C. Yanming, Z. Chao, and Z. Yanxu, "Design and implementation of a smart iot gateway," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, IEEE, 2013, pp. 720–723.

[22] C. Mouradian, N. T. Jahromi, and R. H. Glitho, "Nfv and sdn-based distributed iot gateway for large-scale disaster management," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4119–4131, 2018.

[23] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *arXiv preprint arXiv:1406.0440*, 2014.

[24] J. d. S. Machado, E. D. Moreno, and A. d. R. L. Ribeiro, "Uma revisão da fog computing e seus desafios de pesquisas," *Journal on Advances in Theoretical and Applied Informatics*, vol. 3, no. 2, pp. 32–39, 2017.

[25] S. N. T.-c. Chiueh and S. Brook, "A survey on virtualization technologies," *Rpe Report*, vol. 142, 2005.

[26] M. Vögler, J. Schleicher, C. Inzinger, S. Nastic, S. Sehic, and S. Dustdar, "Leonore–large-scale provisioning of resource-constrained iot deployments," in *2015 IEEE Symposium on Service-Oriented System Engineering*, IEEE, 2015, pp. 78–87.

[27] H. Netto, L. C. Lung, M. Correia, and A. F. Luiz, "Replicação de máquinas de estado em containers no kubernetes: Uma proposta de integração," *Anais do XXXIV Simpósio Brasileiro de Redes de Computadores-SBRC*, 2016.

[28] C. Anderson, "Docker [software engineering]," *IEEE Software*, vol. 32, no. 3, pp. 102–c3, May 2015. DOI: 10.1109/MS.2015.62.

[29] C. Boettiger, "An introduction to docker for reproducible research," *SIGOPS Oper. Syst. Rev.*, vol. 49, no. 1, pp. 71–79, Jan. 2015, ISSN: 0163-5980. DOI: 10.1145/2723872.2723882. [Online]. Available: http://doi.acm.org/10.1145/2723872.2723882.

[30] Katacontainers, *About kata containers*, https://katacontainers.io/, october 22, 2019, 2019. (visited on 10/22/2019).

[31] Openvswitch, *What is open vswitch?* https://www.openvswitch.org, october 24, 2019, 2019. (visited on 10/24/2019).

[32] Kubernetes, *Kubernetes documentation*, https://kubernetes.io, october 24, 2019, 2019. (visited on 10/24/2019).

[33] Contiv, *What is contiv?* https://contiv.io/documents/gettingStarted/, october 24, 2019, 2019. (visited on 10/24/2019).

[34] R. H. Ansible, *Overview. how ansible works*, https://www.ansible.com/overview/how-ansible-works, october 24, 2019, 2019. (visited on 10/24/2019).

[35] V. Sciancalepore, F. Cirillo, and X. Costa-Perez, "Slice as a service (slaas) optimal iot slice resources orchestration," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, IEEE, 2017, pp. 1–7.

[36] Postgreesql, *About*, https://www.postgresql.org/about/, September 30, 2019, 2019. (visited on 09/30/2019).

[37] Python, *Python 3.7.4 documentation*, https://docs.python.org/3/, September 30, 2019, 2019. (visited on 09/30/2019).

[38] Scapay, *Scapy project*, https://scapy.net/, September 30, 2019, 2019. (visited on 09/30/2019).

[39] Python, *Psutil - project description*, https://pypi.org/project/psutil/, September 30, 2019, 2019. (visited on 09/30/2019).

[40] Wireshark, *Tshark - description*, https://www.wireshark.org/docs/man-pages/tshark.html, September 30, 2019, 2019. (visited on 09/30/2019).

[41] Matplotlib, *Matplotlib - documentation*, https://matplotlib.org/, September 30, 2019, 2019. (visited on 09/30/2019).

[42] TCPReplay, *Tcpreplay - pcap editing and replaying utilities*, http://tcpreplay.appneta.com/, September 30, 2019, 2019. (visited on 09/30/2019).

[43] R. T. Fielding, "REST: architectural styles and the design of network-based software architectures," Doctoral dissertation, University of California, Irvine, 2000. [Online]. Available: http://www.ics.uci.edu/˜fielding/pubs/dissertation/top.htm.

[44] Docker.com, *What is a container?* https://www.docker.com/resources/what-container/, September 30, 2019, 2019. (visited on 09/30/2019).