Implantação e Avaliação de um Protótipo para Filas Inteligentes utilizando um Dispositivo IoT Wi-Fi e um *Gateway* IoT Definido por Software

Divino Alves Ferreira Júnior[1,4], João Paulo Oliveira Cabral[1], Ciro Macedo[1,4], Tércio A. dos Santos Filho[2], Kleber Vieira Cardoso[1], Antonio C. Oliveira Jr[1,3]

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

[2]Instituto de Biotecnologia – Universidade Federal de Catalão (UFCAT)

[3]Fraunhofer AICOS Portugal

[4]Instituto Federal de Goiás - Campus Senador Canedo (IFG)
e-mail{divinoalves,joaocabral,ciro,kleber,antonio}@inf.ufg.br, tercioas@gmail.com

Resumo. *A Internet das Coisas (IoT) tem crescido de forma bem expressiva o que leva ao aumento de fluxo de dados nas redes de comunicação. Redes definidas por software teve grandes avanços e crescimento, viabilizando a utilização de recursos que otimizam os processos de comunicação. O conceito de Cidades Inteligentes e Campus inteligentes está diretamente relacionado ao uso de tecnologia nestes ambientes com objetivo de melhorar a qualidade de vida das pessoas. Neste contexto, diariamente, desenvolve-se aplicações que contribui diretamente na qualidade de vida. O objetivo deste artigo é implantar e avaliar a performance de um protótipo desenvolvido no âmbito de um campus inteligente, para ser aplicado na fila inteligente de um restaurante universitário.*

Abstract. *The internet of things has grown quite significantly which leads to increased data flow in communication networks. Software-defined networks have made great strides and growth, enabling the use of resources that optimize communication processes. The concept of Smart Cities and Smart Campuses is directly related to the use of technology in these environments to improve people's quality of life. In this context, daily, develops applications that directly contributes to the quality of life. The purpose of this paper is to deploy and evaluate the performance of a prototype developed within a smart campus, to be applied to the smart queue of a university restaurant.*

1. Introdução

Internet das Coisas (*Internet of Things* - IoT) é um tópico que colabora diretamente com o que chamamos de revolução tecnológica. IoT é um conceito em que objetos do cotidiano desde lâmpadas domésticas, relógios, aparelhos hospitalares até carros, denominados "coisa", ganham habilidades de se conectarem a uma rede e desempenharem uma ou mais funcionalidades que agregam inteligência. Segundo [Atzori et al. 2010] 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. Estudos indicam que até 2020 existirão cerca de 75 bilhões de dispositivos gerando tráfego de rede [Riggins and Wamba 2015], sendo esse, majoritariamente via protocolos de comunicação sem fio, gerando inúmeros desafios relativos às implementações eficientes de *gateways* IoT.

Um *gateway IoT* tem como função estabelecer a comunicação entre vários dispositivos para formar uma rede e compartilhar recursos e informações, permitindo ainda a intercomunicação entre os equipamentos com diferentes protocolos de rede conforme [Zhu et al. 2010]. Um elemento importante no *gateway* é a tecnologia de comunicação sem fio. Novos protocolos de comunicação sem fio são lançados no mercado periodicamente, dificultando a integração entre os milhares de *hardwares* e *softwares* pré-existentes no mercado.

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

Dentre as tecnologias mais utilizadas para lidar com grande variedade de dispositivos IoT se destacam o uso dos *containers*, que é um recurso utilizado para que processos rodem de forma isolada em um host com o mesmo sistema operacional. Esta tecnologia sobressai em questões de performance sobre técnicas de virtualização tradicionais como Máquinas Virtuais baseadas em *Hypervisor* [Morabito et al. 2015]. Provedores de serviços IoT e entusiastas que fazem uso de *containers* tem feito avaliações de performance majoritariamente com ênfase em consumo de *hardware* e eletricidade [Morabito et al. 2017].

Gateways IoT de mais alto nível são comuns, como por exemplo [Eclipse Kura 2019] e [Thingsboard 2019], assim como *gateways* de rede baseados em *hardware*. No entanto, não existem soluções bem consolidadas de *gateways* IoT empregando a tecnologia de de rede definda por *software*. Devido ao fato do grande uso de redes sem fio no cenário de IoT é de suma importância trabalhos que também visam considerar performance de rede em suas avaliações, como nos estudos [Quadros Júnior 2017] e [Krylovskiy 2015].

A comunicação dos dispositivos IoT é um elemento fundamental. Normalmente, esses dispositivos utilizam uma comunicação baseada em tecnologia de redes sem fio para ter acesso a Internet. Alguns exemplos de tecnologias para comunicação em redes sem fio que temos disponível hoje são: WiFi Halow, BLE (Bluetooth), ZigBee, Z-Wave, NFC (Near Field Communication), LoRa/LoRaWAN, 2G/3G, LTE-M (Long Term Evolution - Machine), NB-LTE-M (NarrowBand-Long Term Evolution-Machine) e NB-IoT (Narrowband Internet of Things). Cada tecnologia destas tem uma característica própria, considerando inclusive o alcance de transmissão.

Uma solução de campus inteligente pode demandar o uso de mais de uma destas tecnologias, neste contexto, identifica-se a necessidade de *gateways* que suportam múltiplas tecnologias para que os dispositivos IoT tenham acesso a internet. Algumas das tecnologias de comunicação citadas ainda estão em padronização, desenvolvimento ou até mesmo em projeto.

Um projeto desenvolvido pela Universidade Federal de Goiás (UFG) e finan-

ciado pela Rede Nacional de Ensino e Pesquisa (RNP) desenvolveu uma solução de comunicação (*gateway*) totalmente aberto, virtualizado, definido por software, com suporte a tecnologia *fog computing* e integrado com a computação em núvem, denominado Softway4IoT [Cardoso et al. 2019]. Outra característica deste projeto é a integração de várias tecnologias de comunicação. Neste contexto 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 (I) sistema mais leve permitindo maior escalabilidade, (II) é um sistema aberto, (III) a migração de serviços para a borda da rede permite uma redução de custos operacionais, (IV) uso de SDN permite adicionar funcionalidades de tráfego, segurança, entre outras a um controlador centralizado, facilitando o gerenciamento da rede.

A solução é direcionada especialmente aos campus inteligentes que são vistos com várias características semelhantes a uma cidade inteligente. Ou seja, é um ambiente com alta densidade de pessoas tendo problemas relacionadas à mobilidade, estacionamento, segurança, alimentação, convívio social, etc. Por outro lado, comparado a uma cidade, o campus possui tamanho reduzido, a gestão mais simples e a 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.

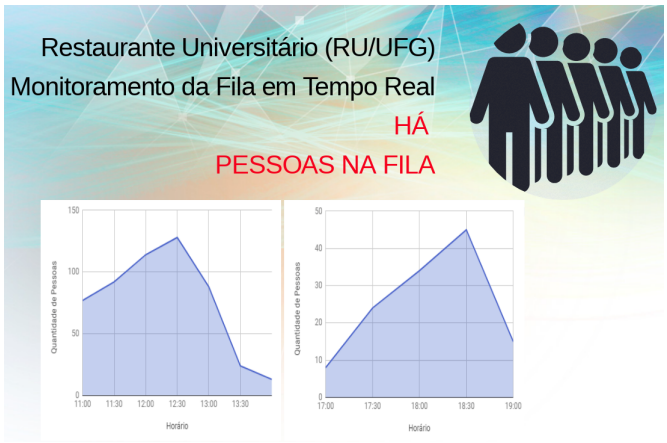


Figura 1. Monitoramento da Fila RU - UFG

Um bom exemplo de aplicação IoT é o uso de sensores para mensurar o tamanho da fila de um restaurante universitário (RU) Figura 1. O sensor coleta os dados da fila, envia via rede, alimentando uma base de dados, que poderá ser acessada via aplicação, por alunos ou demais interessados em verificar o status da fila. Por outro lado, serve para a administração do RU tomar decisões que minimizem o tamanho da fila.

Neste contexto, a proposta deste trabalho é utilizar o *gateway IoT* desenvolvido para implantar e avaliar o desempenho da comunicação estabelecida entre o dispositivo IoT e o *gateway* considerando a aplicação Fila-RU. Os experimentos são implementados em um ambiente real utilizando várias tecnologias modernas e integração com a computação em núvem.

Este artigo é organizado da seguinte forma. Após a Introdução, a seção 2 des-

creve os conceitos sobre as tecnologias utilizadas no desenvolvimento e avaliação deste trabalho. A Seção 3 dedica-se a implantação e avaliação do protótipo considerando o cenário de aplicação. Na Seção 4 são apresentados os resultados da avaliação considerando a transmissão, implantação e uso de máquina. As conclusões e trabalhos futuros deste trabalho são discutidas na Seção 5.

2. Tecnologias Utilizadas na Solução

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 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 [Kreutz et al. 2014] pois possibilita a criação de ambientes totalmente virtuais desde *hardware* até recursos de rede, como no caso implementado pelo Softway4IoT, que estabelece fatias de rede virtuais para uma determinada quantidade de aplicações também virtualizadas.

A tecnologia Wi-Fi (*Wireless Fidelity*) é o nome popular para protocolo de comunicação sem fio definido pelo padrão IEEE 802.11 (grouper.ieee.org/groups/802/11) amplamente utilizado em aplicações distribuídas modernas, sendo muitas delas IoT. A tecnologia Wi-Fi opera em faixas de frequências que não necessitam de licença para operação, fato este que a deixa mais atrativas. O IEEE - *Institute of Electrical and Electronic Engineers* - desenvolveu diversos padrões e subpadrões para tecnologia de WLAN - *Wireless Local Area Network* - entre eles, destacam-se os subpadrões 802.11a, 802.11b, 802.11g, 802.11n e 802.11ac. Esses padrões diferem em relação à frequência de operação, taxa de transmissão, largura de banda, à modulação utilizada para transmissão dos dados e recursos de segurança suportados [Figueiredo 2017]. No contexto de IoT seu uso vem se tornando comum em dispositivos domésticos devido à comodidade da grande variedade de *hardwares* que suportam Wi-Fi [Lehr and McKnight 2003].

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 [Machado et al. 2017]. Nesta arquitetura, a neblina se situa entre a fonte de dados oriundas de dispositivos e a tradicional nuvem a fim de obter maior eficiência no tráfego dos dados [Bonomi et al. 2012]. 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.

Uma técnica eficiente de virtualização, chamada de *Containers*, baseia-se no isolamento de processos em um sistema operacional hospedeiro permitindo a execução de várias instâncias de sistemas operacionais distintos consumindo fatias de recursos de hardware e rede do mesmo [Vögler et al. 2015]. *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 [Netto et al. 2016]. Como os *containers* compartilham do mesmo *kernel Linux* do Sistema Operacional Hospedeiro pode ser obtido pouco *overhead* de performance permitindo eficácia ao executar grandes números de *containers* de aplicações IoT.

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 [Chiueh and Brook 2005]. 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. Esse conceito foi amplamente utilizado na construção do protótipo deste trabalho com o intuito de intensificar a segurança, distanciando o acesso aos diversos tipos recursos computacionais físicos.

Uma técnica de fatiamento de recurso de rede vem sendo bastante empregada, denominado de *Network Slicing - Slice*. Conforme [Sciancalepore et al. 2017] cada fatia de rede representa uma rede ponta a ponta independente e 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.

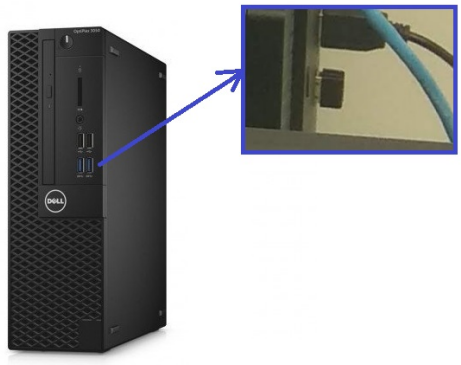
O protocolo de comunicação de mensagens MQTT - *Message Queuing Telemetry Transport* - é leve e foi criado pela IBM - *International Business Machines* - na década de 90 com foco em sistemas de supervisão e aquisição de dados. O protocolo evoluiu e encontrou seu espaço nesse amplo mercado de Internet das Coisas [Souza et al. 2018]. O protocolo MQTT foi concebido para conectar dispositivos, redes, aplicações, serviços e middlewares. O protocolo foi concebido visando utilizar a infraestrutura e realizar integração com os protocolos TCP e IP. Além disso, o MQTT foi projetado para aplicações que utilizam pouca banda de rede, com requisitos de *hardware* extremamente simples e leve. O sistema de troca de mensagens no protocolo MQTT é baseado no modelo publish/subscribe, caracterizando uma comunicação assíncrona, identificando as mensagens por meio de tópicos.

3. Implantação e Avaliação do Protótipo

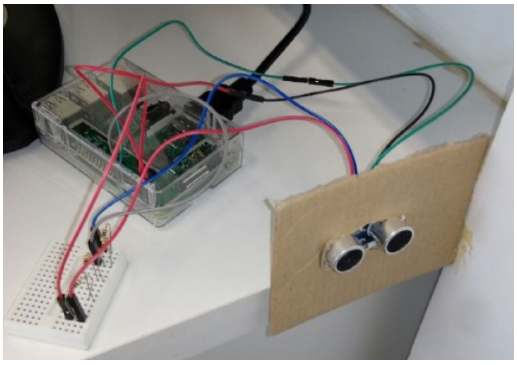
A aplicação IoT utilizada na implantação e avaliação do protótipo e incluída no plano piloto do projeto chamada **Fila-RU**. Esta aplicação é proposta para ser implantada na entrada do Restaurante Universitário da UFG - Campus Samambaia. A Figura 1 apresenta o *front end* da aplicação disponibilizada para os usuários que foi desenvolvida em linguagem HTML-*HyperText Markup Language* (w3.org), usando JavaScript (*jQuery, Ajax*) (javascript.com) e PostgreeSQL (postgresql.org).

A implantação do protótipo e da aplicação consiste num dispositivo IoT com um sensor ultrassônico que encaminha os valores de presença constantemente à aplicação servidora presente no gateway IoT. Os dados encaminhados são armazenados para consulta, permitindo aos interessados acessarem a aplicação e consultar o status da fila.

Conforme a Figura 2(a), o protótipo é implementado considerando um *gateway*



(a) *Gateway* IoT



(b) *Raspeberry*

Figura 2. Equipamentos usados nos experimentos

IoT virtualizado com uma interface sem fio para conexão dos dispositivos IoT. A comunicação é por meio da tecnologia WiFi 802.11n na frequência 2.4Ghz. Para o *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 Wifi convencional como adaptador de rede WiFi.

A Figura 2(b) apresenta o dispositivo IoT, o qual é utilizado 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 são enviados a uma aplicação utilizando o protocolo MQTT, publicando no canal MQTT da aplicação que se situa num *container* da VM de aplicação do *gateway*.

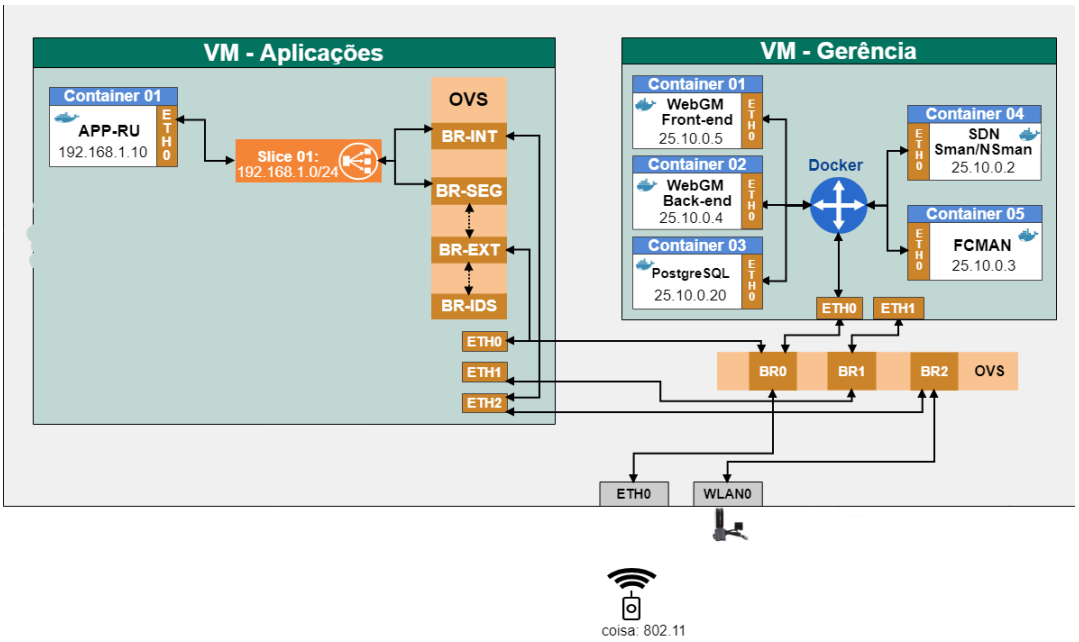


Figura 3. Arquitetura do projeto Softway4IoT

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*.

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 o 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 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 do WebGM que são fornecidas as informações necessárias para o funcionamento adequado dos demais componentes.

Considerando a arquitetura apresentada na Figura 3 o sensor enviará os dados a partir do dispositivo IoT, via WiFi, para o *gateway*. Ao receber os dados, a VM - Gerência fica responsável pela criação de *slices* pelo SMan, implementar as políticas de segurança, considerando cada canal, pelo NSMan e o módulo FCMan ficará responsável pela instanciação de *containers Docker*. Os dados são armazenados em um *container* com o Postgreesql [Postgreesql 2019]. 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.

Para o levantamento de dados usados na avaliação foi desenvolvido um *script* em *Python3* [Python 2019b] que faz uso de bibliotecas como *scapy* [Scapay 2019] usada para captura de arquivos do tipo .pcap no *python*, que contém informações sobre o tráfego de dados. A biblioteca *Psutil* [Python 2019a] é usada para capturar as informações sobre uso de CPU e memória. A ferramenta *tshark* [Wireshark 2019] analisa tráfego de rede a partir dos arquivos .pcap gerados pelo *script*. Ao final da execução, gerou relatórios de uso de recursos e gráficos preliminares utilizando *Matplotlib* [Matplotlib 2019].

Para fins de gerar o tráfego, foi utilizado a ferramenta *tcpreplay* [TCPReplay 2019] que é configurada para reinjetar tráfego na rede, considerando uma amostra capturada da comunicação estabelecida entre o sensor e o *gateway* na rede Wi-Fi. O tráfego capturado foi retransmitidos em *loop* até o fim dos experimentos, considerando o intervalo de tempo de 3 horas. O *script* desenvolvido captura dados a cada 5 minutos de tráfego na rede, criando um arquivo .pcap. A cada hora temos 12 arquivos .pcap que serão analisados pelo *tshark*. A análise em questão extrai dados que são usados para calcular os parâmetros 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 gerando mais pacotes de rede, servindo como base de comparação para aplicações de envio de dados mais intensos.

4. Análise dos Resultados Obtidos

Os resultados obtidos são analisados considerando 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 permite uma avaliação do funcionamento da solução desenvolvida. Com os dados do tráfego (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 de 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, estes pacotes serão retransmitidos. Para analisar a perda de pacotes, os dados são coletados referente 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 calculado a razão entre os pacotes perdidos e pacotes total a cada 5 minutos. Esta razão é somada considerando o intervalo de 1 hora e calculado a média aritmética, obtendo assim a média de perda de pacote a cada hora de tráfego.

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

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.

A análise de vazão média é realizada a partir da quantidade de bits trafegados. Neste caso somamos a quantidade total de bits ($bits$) e dividimos pelo intervalo de tempo (tmp_{seg}), neste caso 5 minutos conforme Equação 2. Na sequência somamos os valores calculados num intervalo de 1 hora e obtivemos uma média aritmética. A vazão média é dada em bps (bits/seg). Como esperado, conforme o aumento da intensidade de tráfego, a vazão também aumenta ao longo de 3 horas. Conforme o aumento da intensidade tivemos 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.

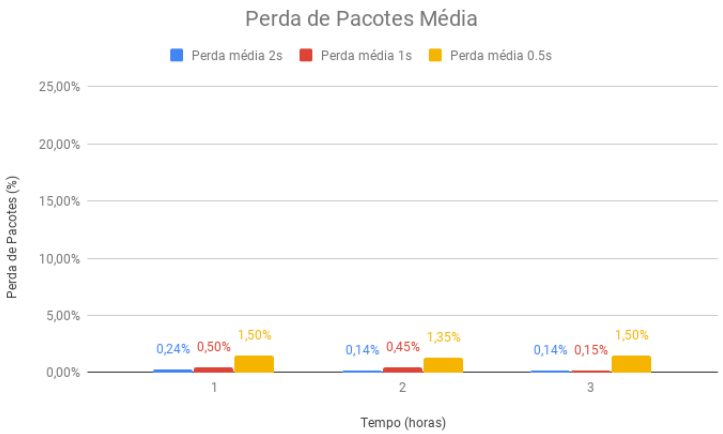


Figura 4. Perda de Pacotes / Tempo

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

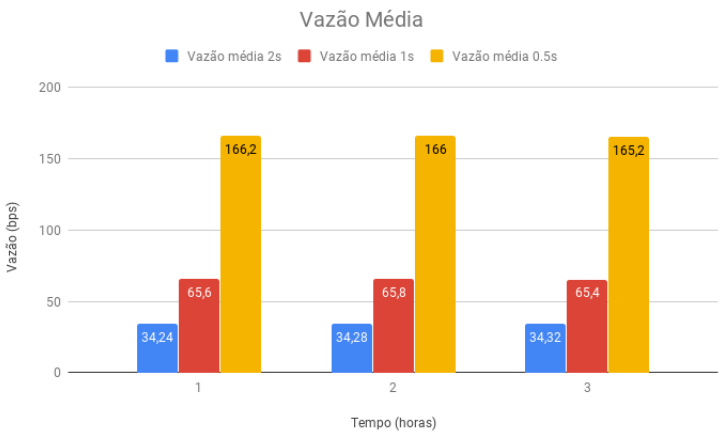


Figura 5. Vazão média / Tempo

O atraso médio foi levantado a partir do cálculo da diferença entre o tempo de recebimentos do último pacote em relação ao pacote atual. Estes 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 obtido a média do atraso num intervalo de 1 hora. O atraso médio é dado em milissegundos. Os resultados mostram que, aumentando o fluxo de envio de pacotes diminui o atraso médio, mas este atraso se mantém estável ao longo do tempo. No período de três horas o atraso médio se manteve praticamente constante. 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.

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

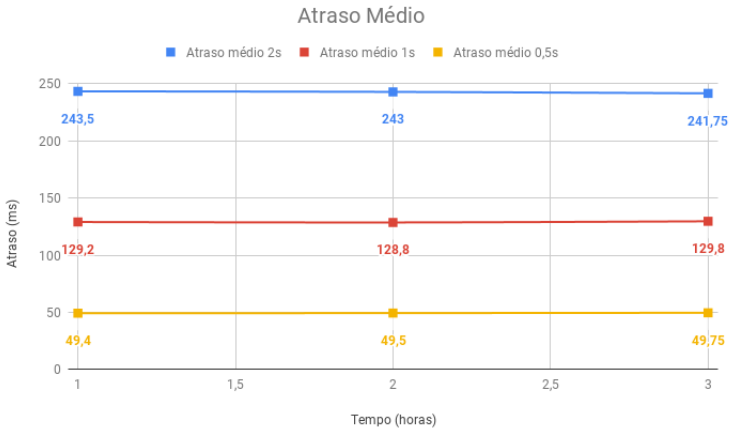


Figura 6. Atraso Médio vs Tempo

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 frequências de envios maiores como a cada 0,5s causaram um aumento significativo de envio de pacotes se comparado às demais frequências.

$$pacotesvstempo = \sum pct_{tmp} \tag{4}$$

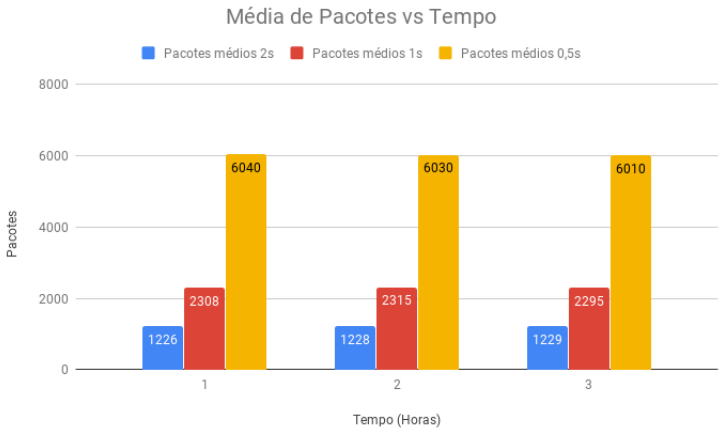


Figura 7. Média de Pacotes vs Tempo

Os dados referente 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 do teste devido ao alto número de pacotes a serem processados, especialmente com a frequência de 0,5s. Já os relatórios de uso de Memória RAM apresentados na Figura 9 se mostraram estáveis.

Para avaliar a tecnologia de computação em neblina implementado no protótipo, realizou-se experimentos relativos à efetividade de instanciação de *containers* e

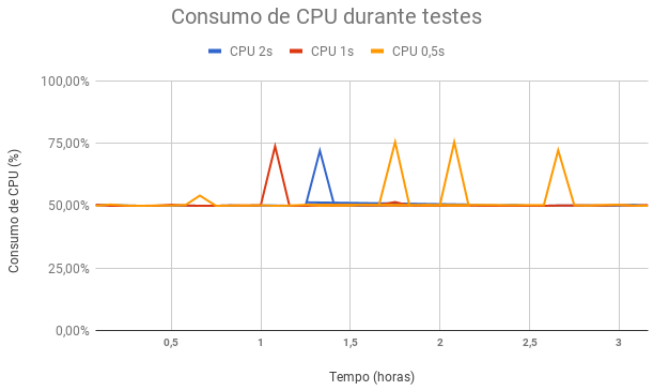


Figura 8. Consumo de CPU ao longo dos experimentos

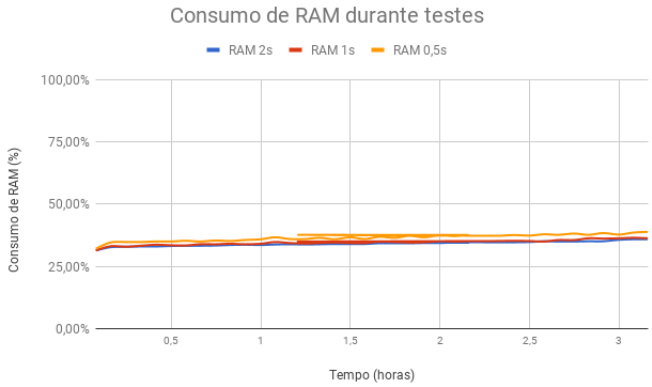


Figura 9. Consumo de RAM ao longo dos experimentos

construção de imagens *Docker*. Foram feitos 10 experimentos sobre a aplicação Fila-RU utilizando-se da API REST [Fielding 2000] do módulo FCMan, obtendo assim aproximadamente o mesmo tempo de execução em segundos que o acesso via interface web do *gateway*. O build das imagens Docker variou entre 6s e 7,2s

Sobre os mesmos experimentos foram extraídos os seguintes dados acerca da variação de tempo dos experimentos: Tempo médio em segundos, Desvio Padrão em segundos e Variância em segundos. Os tempos de criação de instâncias de *containers Docker* em segundos exibidos na Figura 10 foram satisfatórios, 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.

Para a construção de imagens *Docker* [Docker.com 2019] 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 os valores se mostraram dentro do esperado considerando 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.

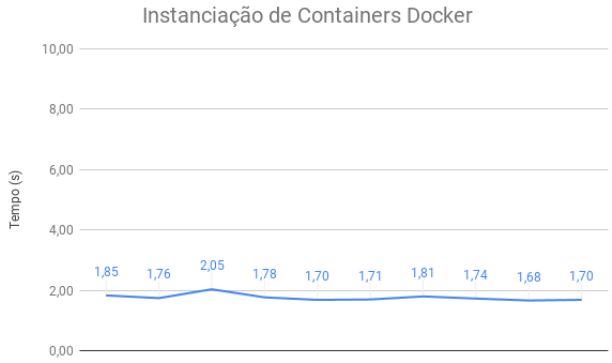


Figura 10. Instanciação de *Containers* da aplicação Fila

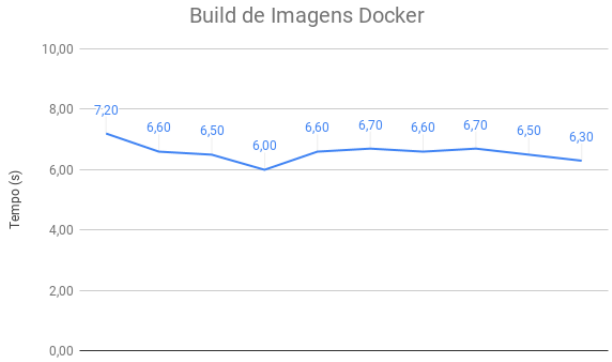


Figura 11. *Builds* de Imagem da aplicação Fila

5. Conclusão

Neste trabalho foi apresentado uma implantação e avaliação de performance de um protótipo utilizado para fins de controlar a fila de um restaurante universitário. O projeto contou com a implantação de um *gateway* totalmente aberto, virtualizado, definido por software, com suporte a tecnologia *fog computing* e integrado com a computação em núvem, responsável por estabelecer a comunicação com o dispositivo IoT contendo um sensor ultrassônico da fila do restaurante. Os dados provenientes desta comunicação são usados para alimentar uma aplicação chamada Fila-RU que tem o propósito de apresentar o status da fila aos interessados.

Os resultados da avaliação da vazão média, perda de pacotes, quantidade de pacotes trafegados, atraso médio, consumo de CPU e memória, mensurados no protótipo, foram satisfatórias. Relativo a instanciação de *containers* tanto o tempo da instanciação quanto do *build* foram satisfatórios e comprovam que o uso da máquina virtual e isolamento de recursos não ofereceram prejuízos significativos ao processo.

Para trabalhos futuros, pretende-se avaliar o desempenho da solução com uso de outras tecnologias de comunicação sem fio voltados para IoT, como por exemplo LoRa. Pretende-se ainda, colocar o piloto em operação no restaurante universitário a fim de avaliar a robustez e escalabilidade do protótipo.

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

[Atzori et al. 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805.

[Bonomi et al. 2012] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM.

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

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

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

[Eclipse Kura 2019] Eclipse Kura (2019). Eclipse kura — the eclipse foundation. online.

[Fielding 2000] Fielding, R. T. (2000). *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine.

[Figueiredo 2017] Figueiredo, D. A. (2017). Análise de vulnerabilidades e ameaças presentes em redes wi-fi (ieee 802.11) de instituições de ensino superior de minas gerais. *Projetos e Dissertações em Sistemas de Informação e Gestão do Conhecimento*, 5(2).

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

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

[Lehr and McKnight 2003] Lehr, W. and McKnight, L. W. (2003). Wireless internet access: 3g vs. wifi? *Telecommunications Policy*, 27(5-6):351–370.

[Machado et al. 2017] Machado, J. d. S., Moreno, E. D., and Ribeiro, A. d. R. L. (2017). Uma revisão da fog computing e seus desafios de pesquisas. *Journal on Advances in Theoretical and Applied Informatics*, 3(2):32–39.

[Matplotlib 2019] Matplotlib (2019). Matplotlib - documentation. https://matplotlib.org/. September 30, 2019.

[Morabito et al. 2017] Morabito, R., Farris, I., Iera, A., and Taleb, T. (2017). Evaluating performance of containerized iot services for clustered devices at the network edge. *IEEE Internet of Things Journal*, 4(4):1019–1030.

[Morabito et al. 2015] Morabito, R., Kjällman, J., and Komu, M. (2015). Hypervisors vs. lightweight virtualization: a performance comparison. In *2015 IEEE International Conference on Cloud Engineering*, pages 386–393. IEEE.

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

[Postgreesql 2019] Postgreesql (2019). About. https://www.postgresql.org/about/. September 30, 2019.

[Python 2019a] Python (2019a). Psutil - project description. https://pypi.org/project/psutil/. September 30, 2019.

[Python 2019b] Python (2019b). Python 3.7.4 documentation. https://docs.python.org/3/. September 30, 2019.

[Quadros Júnior 2017] Quadros Júnior, E. M. d. (2017). 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.

[Riggins and Wamba 2015] Riggins, F. J. and Wamba, S. F. (2015). Research directions on the adoption, usage, and impact of the internet of things through the use of big data analytics. In *2015 48th Hawaii International Conference on System Sciences*, pages 1531–1540. IEEE.

[Scapay 2019] Scapay (2019). Scapy project. https://scapy.net/. September 30, 2019.

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

[Souza et al. 2018] Souza, D. S. d. et al. (2018). Estudo da aplicação de um sistema iot baseado no protocolo de comunicação mqtt a área da robótica industrial.

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

[Thingsboard 2019] Thingsboard (2019). Thingsboard open-source iot platform. online.

[Vögler et al. 2015] Vögler, M., Schleicher, J., Inzinger, C., Nastic, S., Sehic, S., and Dustdar, S. (2015). Leonore–large-scale provisioning of resource-constrained iot deployments. In *2015 IEEE Symposium on Service-Oriented System Engineering*, pages 78–87. IEEE.

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

[Zhu et al. 2010] Zhu, Q., Wang, R., Chen, Q., Liu, Y., and Qin, W. (2010). Iot gateway: Bridgingwireless sensor networks into internet of things. In *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pages 347–352. Ieee.