Otimização de Tráfego IoT-LoRaWAN Usando Programação de Plano de Dados em P4

Alexandre Heideker[1], Dener Silva[1], João Henrique Kleinschmidt[1], Carlos Kamienski[1]

[1]Universidade Federal do ABC (UFABC)
Santo André – SP – Brazil

{alexandre.heideker,dener.silva,joao.kleinschmidt,cak}@ufabc.edu.br

Abstract. *The rapid adoption of Internet of Things (IoT) technologies generates a growing demand for solutions that guarantee the scalability of applications. The ever-increasing number of IoT devices generates a proportional increase in traffic at the network's edge. Data plane programming allows network devices, sometimes intended for specific and well-defined functions, to behave in a non-standard way, paving the way for new approaches to deal with traffic. This paper presents an IoT-LoRaWAN traffic optimization solution using a filter implemented in the P4 language for the data plane on the P4Pi platform. The experimental results show that the LoRaWAN P4 filter decreases the CPU load, delay, and packet loss, proving the viability of programming in the data plane to make the network edge more versatile.*

Resumo. *A rápida adoção de tecnologias de Internet das Coisas (IoT) gera uma demanda crescente por soluções que garantam a escalabilidade das aplicações. O número cada vez maior de dispositivos IoT gera um aumento proporcional no tráfego na borda da rede. A programação do plano de dados permite que os dispositivos de rede, ora destinados a funções específicas e bem definidas, possam se comportar de forma não padronizada, abrindo caminho para novas abordagens na solução de problemas. Este artigo apresenta uma solução de otimização de tráfego IoT com LoRaWAN, usando um filtro implementado na linguagem P4 para o plano de dados na plataforma P4Pi. Os resultados experimentais mostram que o filtro LoRaWAN P4 diminui a carga na CPU, o atraso e a perda de pacotes, comprovando a viabilidade do uso da programação no plano de dados para tornar a borda da rede mais versátil.*

1. Introdução

A massiva adoção de tecnologias relacionadas à Internet da Coisas (IoT) torna viável cenários cada vez mais ricos em informação e benefícios às aplicações e usuários. A enorme quantidade de dispositivos conectados, que aumenta a cada ano (em 2022 este número deve atingir 29 bilhões, dos quais 19 bilhões podem ser categorizados como IoT [Ericsson 2019]), já representa um grade desafio para garantir gerenciamento e escalabilidade [Zyrianoff et al. 2018].

Em muitos casos, o desafio de interpretar novos protocolos de rede propostos para solucionar estes e outros desafios [Resner et al. 2018] é confrontado com a falta de suporte dos dispositivos da infraestrutura de acesso à Internet [Bosshart et al. 2014]. Neste sentido, as redes definidas por software (SDN)[Singh and Jha 2017] surgem como

solução de gerenciamento desta demanda crescente e dinâmica. Apesar dos avanços obtidos nos últimos anos pelo uso de SDN, o conceito de separação do planos de controle do plano de dados não se mostrou suficientemente ágil para suportar os novos protocolos, bem como o desempenho exigido por esta demanda[Bosshart et al. 2014].

Para enfrentar este desafio e expandir o conceito de SDN, em 2014 é proposto o conceito de programação do plano de dados, através da linguagem *Programming Protocol-independent Packet Processors* (P4)[Bosshart et al. 2014]. A programação em P4 permite que o próprio plano de dados assuma comportamentos diferentes do tradicionalmente esperado, como em um switch de rede que apenas entrega os pacotes de rede recebidos por uma de suas portas para outra utilizando regras bem definidas. Em P4, o pacote de rede pode ser modificado, avaliado, novas informações podem ser agregadas, e novos protocolos podem ser suportados, tudo isso com o desempenho exigido para que o atraso na encaminhamento do pacote seja mínimo.

Apesar de promissora e bem explorada pela literatura, a tecnologia de programação de plano de dados ainda esbarra na ausência de dispositivos com preços competitivos, limitando seu uso à equipamentos com tecnologia FPGA [Ibanez et al. 2019] ou *switches* de auto custo, como o Intel Tofino [Agrawal and Kim 2020]. Mesmo com a futura redução no custo destes equipamentos, pesquisadores e o usuários finais não terão acesso a esta tecnologia em um curto prazo. Para atender à estas demandas e limitações na adoção da programação do plano de dados, em 2021 Laki et al. propõem o projeto P4Pi [Laki et al. 2021]. P4Pi representa um avanço no ensino e pesquisa em P4, permitindo que o amplamente difundido SBC (*Single Board Computer*) Raspberry Pi 4 seja transformado em um *switch* P4. Além dos aspectos didáticos defendidos pelos autores, o amplo uso do Raspberry Pi em cenários de IoT permite que os benefícios da programação do plano de dados seja utilizado em cenários de IoT.

Atualmente muitas aplicações IoT utilizam o LoRaWAN para conectar centenas ou milhares de dispositivos sem fio [Sornin et al. 2016]. Pos vezes, os aspectos tecnológicos que uma LPWAN (*Low Power Wide Area Networs*) como é o caso do LoRaWAN, implicam em baixos níveis de qualidade de serviço, como o aumento no atraso e na taxa de perda de pacotes, em certas condições [Queté et al. 2020]. Particularmente, a utilização de múltiplos *gateways* LoRaWAN, para expandir a cobertura geográfica da rede e/ou melhorar a garantia de entrega, replica múltiplas vezes as mensagens dos sensores e atuadores, prejudicando ainda mais o desempenho da rede.

Este artigo apresenta uma solução de otimização de tráfego IoT em uma rede LoRaWAN que utiliza os conceitos de programação do plano de dados em P4 com a plataforma P4Pi. Um código P4 implantado no plano de dados de um *switch* baseado em Raspberry Pi identifica as múltiplas mensagens replicadas, encaminhando apenas uma cópia de cada para o servidor LoRaWAN. Os resultados experimentais mostram que o filtro LoRaWAN P4 diminui a carga na CPU, o atraso e a perda de pacotes. Esses benefícios podem ser obtidos com o uso de soluções na borda da rede, não só pelo aproveitamento do poder computacional mas também da versatilidade da própria infraestrutura inteligente. A principal contribuição desse artigo é apresentar uma solução pioneira para otimização de tráfego LoRaWAN em aplicações IoT usando técnicas de programação para o plano de dados com a linguagem P4.

Na sequência do artigo, a Seção 2 apresenta uma revisão da literatura na área e alguns trabalhos relacionados e a Seção 3 apresenta a metodologia de avaliação de desempenho. A Seção 4 caracteriza os experimentos e apresenta os resultados. A Seção 5 discute os resultados e finalmente, a Seção 6 conclui o artigo e apresenta caminhos para trabalhos futuros.

2. Revisão de Literatura e Trabalhos Relacionados

Nesta seção serão apresentados os conceitos básicos e tecnologias utilizadas no desenvolvimento deste trabalho, além dos trabalhos relacionados.

2.1. IoT

A Internet das Coisas é formada por diferentes tipos de objetos inteligentes, como sensores, atuadores e smartphones, que coletam informações do ambiente, controlam dispositivos e interagem entre si para oferecer vários tipos de serviços aos usuários. Uma aplicação típica consiste de um grande número de sensores que transmitem periodicamente os valores de medição de uma grandeza física aos gateways, que por sua vez enviam os dados a servidores na nuvem. As aplicações de IoT incluem cidades inteligentes, agricultura inteligente, smart grids, transporte e saúde [Zyrianoff et al. 2018]. Estes cenários podem se beneficiar da programação no plano de dados, como por exemplo na agregação de dados e automação de serviços [Madureira et al. 2020, Hauser et al. 2021].

2.2. LoRaWAN

Para atender aos requisitos do ambiente de IoT, foram desenvolvidas várias tecnologias de comunicação sem fio conhecidas como LPWAN [Raza et al. 2017]. Entre elas, destaca-se a tecnologia LoRa, desenvolvido pela Semtech Corporation que opera nas faixas de frequência sub-GHz, com alcance de transmissão de até vários quilômetros e baixo consumo de energia. LoRa trata o enlace de rádio implementando a camada física. Para as camadas superiores é proposto o padrão LoRaWAN (padrão aberto desenvolvido pela LoRa Alliance) [Sornin et al. 2016].

A Figura 1 mostra a arquitetura implementada pelo projeto Chirpstack[1] baseada na especificação proposta pelo consórcio LoRaWAN[2] com seus principais elementos: o *transceptor* LoRa (LoRa HAT), responsável pela transmissão e recepção do sinal de rádio LoRa; o PACKET-FORWARDER, responsável por encapsular o *payload* LoRa em uma datagrama UDP; o GATEWAY-BRIDGE, que concentra a recepção dos datagramas dos diversos *gateways* e publica estes dados em um *broker* MQTT; o LORASERVER que decodifica os dados recebidos e gerencia chaves de autenticação dos sensores, publicando os valores obtidos; e finalmente o APPSERVER, que faz a integração entre a pilha LoRaWAN e aplicações de terceiros.

2.3. Programação do Plano de Dados

Dispositivos de rede tradicionais, como roteadores e switches, possuem em sua arquitetura o plano de controle (*control plane*), responsável pela tomada de decisões, e o plano

[1]Chirpstack - https://www.chirpstack.io/ - Acessado em 15/12/2022
[2]LoRa Aliance - LoRaWAN - https://lora-alliance.org/about-lorawan/ Acessado em 16/12/2022

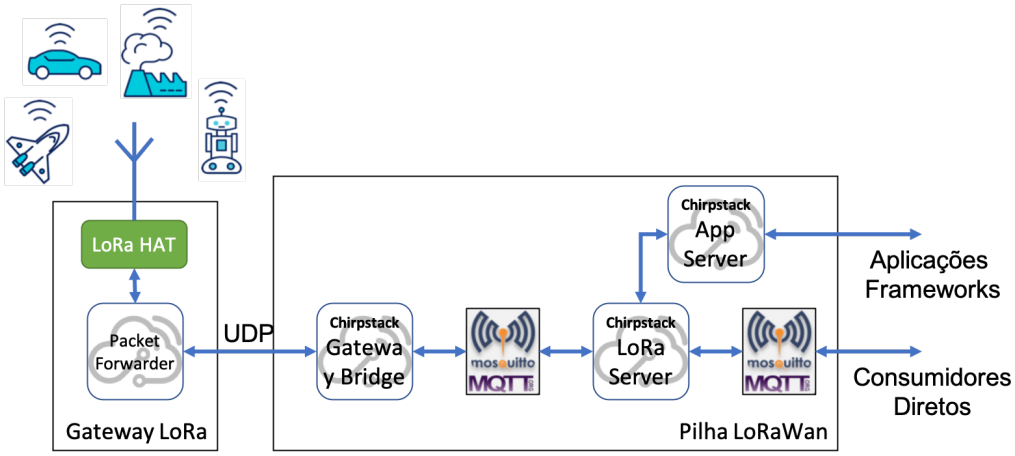

Figura 1. Arquitetura do LoRaWAN baseada na implementação de código aberto do Chirpstack, desde a captura do sinal pelo *gateway* até a disponibilização dos dados aos consumidores.

de dados (*data plane*), responsável pelo transporte da informação entre as diferentes portas destes dispositivos. Alguns modelos permitem definir o comportamento do plano de controle por meio de interfaces de linha de comando, web ou APIs fornecidas pelos fabricantes. Porém, os algoritmos que de fato eram utilizados poderiam ser alterados apenas pelos seus respectivos fornecedores [Hauser et al. 2021].

As Redes Definidas por Software (SDN) permitem que mudanças nos algoritmos do plano de controle desses dispositivos possam ser realizadas independentemente de seus fornecedores, promovendo novos comportamentos e maior versatilidade. O plano de dados mantinha seu comportamento padrão, muitas vezes ligado fortemente a protocolos bem conhecidos, limitando a inovação neste aspecto.

Expandindo o conceito de SDN, a programação do plano de dados, através da linguagem P4[3] (*Programming Protocol-independent Packet Processors*), permite a manipulação dos dados durante o encaminhamento dos mesmos, além de permitir o suporte a novos protocolos sem dependência de padronização e suporte pelo fabricante do equipamento[Bosshart et al. 2014].

Como base comportamental de um *switch* típico, o consórcio P4 propôs o *Protocol Independent Switch Architecture* (PISA), exibida na Figura 2, onde o tráfego segue em uma única direção. O primeiro elemento é o *Parser*, responsável por decodificar o pacote de dados de acordo com seu respectivo protocolo. Aqui é possível definir cabeçalhos para novos protocolos. Em seguida, os conjuntos *Ingress Match-Action* e *Egress Match-Action* são responsáveis por extruturar as tabelas de fluxo que serão posteriormente alimentadas pelo controlador SDN. O Gerenciador de Tráfego permite a inclusão de algoritmos específicos para manipulação dos dados e tomada de decisão. Finalmente, o *Deparser* é responsável por ”montar”novamente o pacote de dados para encaminhá-lo para o meio de transmissão ou porta de rede específica.

Apesar de sua semelhança com a linguagem C, há particularidades da linguagem P4 que devem ser levadas em consideração pelo programador durante o desenvolvimento

[3]P4 - https://p4.org - Acessado em 22/10/2022

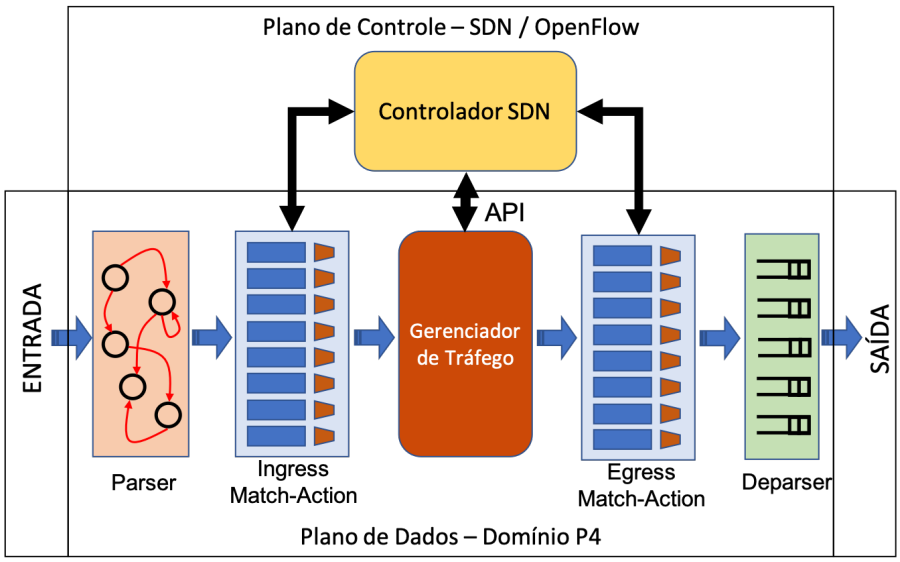

Figura 2. Arquitetura PISA mostrando o trafego em uma única direção através dos diversos blocos funcionais do *switch* e a interação do controlador SDN através das tabelas de fluxo.

de algoritmos, como por exemplo a impossibilidade de criação de laços, chamadas recursivas e operações matemáticas não suportadas pelo hardware. Muitos equipamentos sequer permitem a realização da operação matemática de divisão. Esta limitação deve-se ao fato que há uma janela de tempo limitada para o processamento do pacote, para que a fluidez da rede não seja comprometida.

2.4. Trabalhos Relacionados

Um tema recorrente na literatura envolvendo programabilidade do plano de dados é a agregação de tráfego. Em [Wang et al. 2019], os autores propõe a agregação de tráfego IoT, considerando o tamanho reduzido dos pacotes típicos deste tipo de tráfego, com a posterior desagregação, utilizando switches P4. Protocolos como Sigfox e LoRa são típicos casos de uso para este tipo de abordagem - Sigfox tem payload de 8 a 12 bytes, por exemplo. Da mesma forma, o protocolo LoRa também possui uma baixa relação entre header e payload, gerando um desperdício de até 78% da banda de rede disponível. Os autores implementam o processo de agregação e desagregação de tráfego utilizando switches baseados no chip Tofino, avaliando diversas combinações entre tamanho de *payload* e o limite de pacotes agregados. Apesar da importante contribuição do artigo, o uso de equipamentos deste porte limitam o benefício auferido apenas ao núcleo da rede, em *datacenters* ou em *backbones*.

No mesmo sentido da agregação de tráfego utilizando a programabilidade do plano de dados, em [Madureira et al. 2020] os autores propõe um protocolo específico para auxiliar neste processo, com o objetivo de tornar a solução independente do protocolo de IoT utilizado na borda da rede. O uso de informações sobre as características do enlace, como por exemplo o MTU, permite um maior desempenho no processo de agregação, assim como permitir a priorização de tráfego e redução do atraso. Assim como neste trabalho, os autores propõe o tratamento do tráfego já nos dispositivos da borda da rede que podem encapsular o tráfego IoT no protocolo proposto (IoTP). Os autores, com o uso

da ferramenta Mininet, apresentam como resultado uma melhora de 78% na eficiência do uso da rede.

Apesar da mesma temática da agregação de tráfego, em [Sapio et al. 2017] os autores afirmam que o que se espera da programabilidade do plano de dados, muitas vezes está além da capacidade prática destes dispositivos. Há um *tradeoff* claro entre o que se pode fazer em P4 e o tipo de dispositivo, além da consequente disponibilidade do mesmo. Muito do que se propõe em P4 está apenas disponível em equipamentos de *datacenter*, com custo e requisitos ambientais exigentes.

Esta limitação imposta pelo que é desejado e o que é possível, e principalmente qual é o custo computacional associado, fica claro em [Scholz et al. 2019]. Muitas aplicações em potencial envolvendo a programabilidade da rede envolvem operações de *hash* e criptografia. O desafio é como tornar possível este tipo de operação, sem impor mais atraso ou até congestionar a rede como um todo. A linguagem P4 não permite laços ou recursões por exemplo, legando estas operações à chamadas externas, ou a implementações personalizadas em plataformas escolhidas para este fim, recursos aos quais os autores fazem uso para avaliar o impacto destas operações na vazão e no atraso imposto ao tráfego experimental.

3. Metodologia

O objetivo do trabalho é aplicar a tecnologia de programação do plano de dados para otimizar o tráfego IoT, considerando um cenário típico no qual uma grande quantidade de sensores envia dados periodicamente para sua respectiva infraestrutura de nuvem. Este cenário fim a fim utiliza tecnologias de comunicação sem fio específicas para IoT (LoRa, LoRaWAN), Wi-Fi (IEEE 802.11) e cabeada, compondo esta solução típica.

3.1. Cenário

A Figura 3 apresenta o cenário explorado nos experimentos. Considera-se uma área onde há uma concentração de sensores que enviam informação periodicamente. Esta área é coberta por um número adequado de *gateways* LoRa com o objetivo de garantir a maior confiabilidade possível para o sistema. A transmissão dos dados pelos sensores é realizado por *broadcast*, ou seja, todos os *gateways* que estiverem ao alcance do sinal receberão o mesmo sinal.

Após receber o sinal LoRa, o *gateway* encaminha o dado recebido e processado pelo PACKET-FORWARDER para a módulo GATEWAY-BRIDGE, instanciado na infraestrutura de névoa ou nuvem, de acordo com a arquitetura implementada. Este trajeto entre o *gateway* LoRa e a infraestrutura de névoa/nuvem utiliza, em geral, um enlace tradicional de rede, como o Wi-Fi (IEEE 802.11). No cenário da Figura 3, a mensagem (Ms) é captada por todos os *gateways* LoRa, e é encaminhada múltiplas vezes (Ms.1-4), criando um tráfego redundante neste enlace. As múltiplas mensagens são finalmente publicadas no *broker*, que nesta configuração típica é responsável por unificar estas mensagens, notificando o LORASERVER apenas uma vez.

Para otimizar este cenário, reduzindo o tráfego redundante, este trabalho propõe a utilização da tecnologia de programação do plano de dados no enlace entre os *gateways* LoRa e a infraestrutura de névoa/nuvem, reduzindo a utilização deste enlace e a sobrecarga da infraestrutura de névoa/nuvem, sem o uso de equipamentos com alto poder de

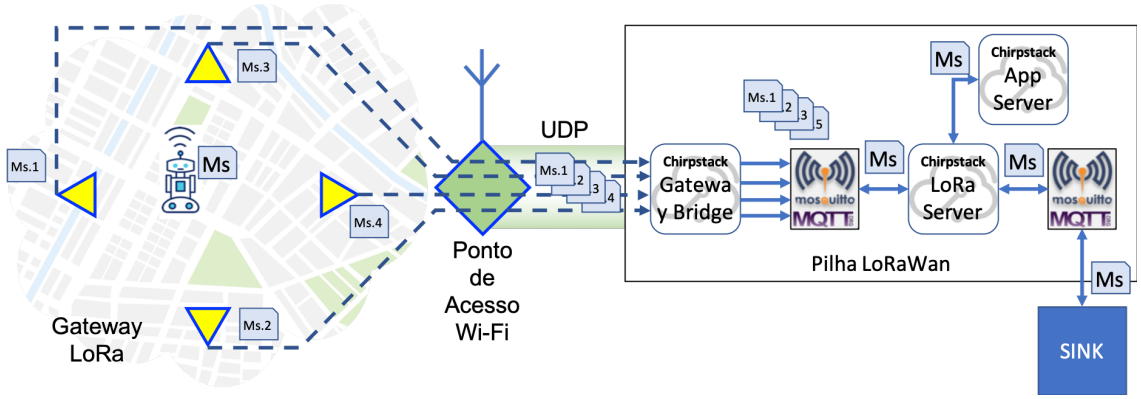

Figura 3. Cenário utilizando tecnologias tradicionais direcionando todo o tráfego para a infraestrutura na nuvem.

processamento neste ponto da rede, muitas vezes em ambientes inóspitos e/ou de difícil acesso. A Figura 4 mostra a nova configuração deste cenário, utilizando um ponto de acesso Wi-Fi com seu plano de dados programável. Nota-se que a redução no tráfego redundante permite um melhor aproveitamento ou até a redução do enlace entre o ponto de acesso e a infraestrutura de névoa/nuvem, além da própria dimensão desta última, considerando que haverá uma redução na carga de trabalho sobre os módulos de software da pilha LoRaWAN.

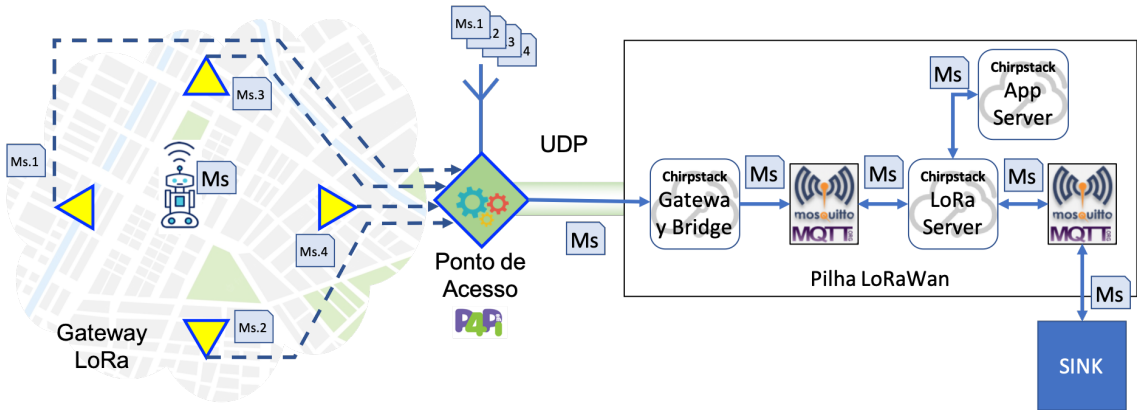

Figura 4. Cenário utilizando programação do plano de dados para reduzir o tráfego e uso de infraestrutura de suporte na nuvem.

3.2. P4Pi

A tecnologia de programação do plano de dados ainda não possui dispositivos dedicados de custo acessível, estando disponível atualmente em dispositivos como NetFPGA e switches utilizando o chip Intel Tofino, por exemplo. Com o objetivo de popularizar seu acesso, proporcionando uma plataforma didática para a linguagem P4, foi criado o projeto P4Pi [Laki et al. 2021]. O projeto P4Pi utiliza o hardware RaspberryPi 4, juntamente com o *framework* T4P4S e a biblioteca DPDK[4], traduzindo o código P4 para arquiteturas computacionais tradicionais e de forma otimizada para o dispositivo alvo[Vörös et al. 2018].

[4]DPDK - (*Data Plane Development Kit*) - https://www.dpdk.org/ - Acessado em 27/10/2022

Para obter a velocidade de processamento necessária ao encaminhamento dos pacotes, produzindo o mínimo de atraso, dois núcleos físicos da CPU do RaspberryPi 4 são reservados para uso exclusivo do plano de dados. A Figura 5 apresenta a arquitetura da solução proposta em [Laki et al. 2021], contando com a porta Ethernet Gigabit e a Interface Wi-Fi disponíveis na placa do RaspberryPi 4.

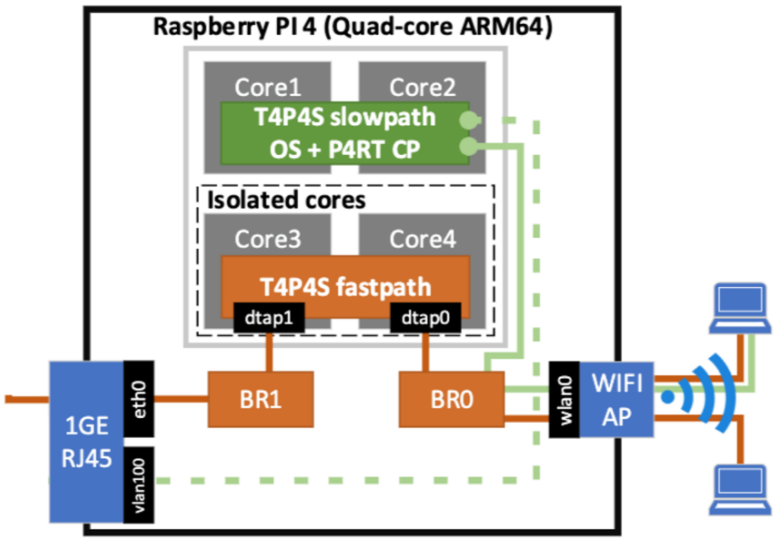

Figura 5. Arquitetura do P4Pi utilizando o hardware RaspberryPi 4. Adaptado de [Laki et al. 2021].

3.3. Filtragem de Tráfego

Em uma arquitetura computacional tradicional, diversos algoritmos estão disponíveis para implementar técnicas sofisticadas de filtragem do tráfego. Para este trabalho foi considerada as limitações impostas pela linguagem $P4_{16}$, além das características do próprio hardware disponível, o RaspberryPi 4.

Por definição, a linguagem P4 manipula apenas os cabeçalhos dos pacotes, sem acesso ao *payload*. Para acessar o *payload*, em P4 utiliza-se a possibilidade de definição de seus próprios cabeçalhos para suporte à novos protocolos. Neste caso, o *payload* passa a fazer parte do cabeçalho que será manipulado. A Figura 6(a) mostra um datagrama UDP com conteúdo LoRa capturado entre o PACKET-FORWARDER e o GATEWAY-BRIDGE, com destaque para as estruturas de interesse.

O PARSER P4 utiliza então a estrutura definida pela Figura 6(b), tornando o *payload* LoRaWAN parte do cabeçalho para o protocolo definido. Com o datagrama decodificado, utiliza-se um registrador de 608 bits de largura para armazenar o ultimo *payload* recebido. Caso o novo datagrama possua um *payload* igual, o datagrama é descartado (DROP). Os experimentos utilizaram apenas um registrador, porém, a utilização de mais de um registrador pode melhorar o desempenho do algoritmo mas o *trade-off* entre desempenho e acurácia deve ser considerada.

4. Experimentos e Resultados

Os experimentos foram executados em um *testbed* composto por um gerador de tráfego LoRa em uma máquina física com interface Wi-Fi, um RaspberryPi 4 com a distribuição

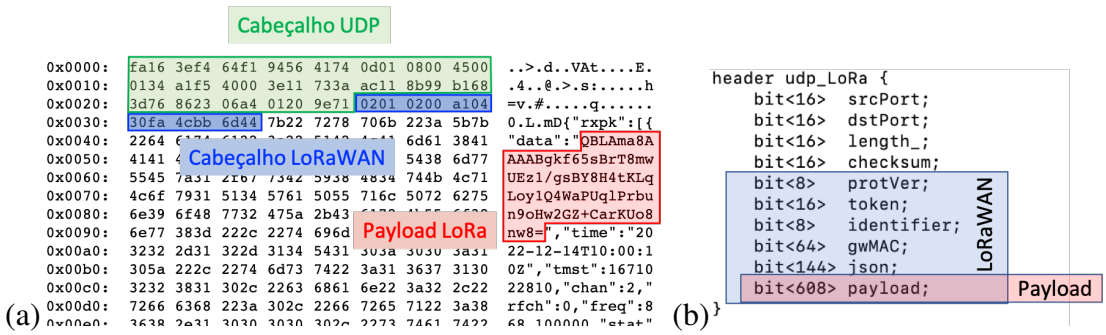

Figura 6. Pacote UDP com conteúdo LoRaWAN e a identificação dos principais elementos estruturais utilizados no algoritmo de filtragem de pacotes em P4.

Tabela 1. Configurações do Gerador de Tráfego.

Sensores	Taxa de Mensagens
500	8,3 mensagens/s
1.000	16,7 mensagens/s
2.000	33,3 mensagens/s
6.000	100,0 mensagens/s
12.000	200,0 mensagens/s
30.000	500,0 mensagens/s

P4Pi versão 0.0.4, e uma infraestrutura de nuvem OpenStack versão Rocky. Foi utilizada uma máquina virtual com 1 vCPU e 2 Gigabytes de RAM para hospedar a pilha Chirpstack LoRaWAN. A conexão Wi-Fi entre o gerador de tráfego LoRa e o P4Pi utiliza o padrão IEEE 802.11g com um nível de sinal de -55 dBm.

O cenário foi configurado para que os sensores enviassem, a cada minuto, um *payload* contendo o *timestamp* em micro segundos seguido de caracteres aleatórios com o objetivo de completar o *payload* em 50 bytes. O gerador de tráfego produz pacotes LoRa com origem em 4 (quatro) *gateways* distintos para representar a recepção do sinal oriundo do sensor. A Tabela 1 apresenta as configurações avaliadas com suas respectivas cargas de trabalho.

As mesmas cargas de trabalho são submetidas ao *testbed* com o P4Pi apenas encaminhando os pacotes entre as interfaces Wi-Fi e cabeada, e em uma segunda variação com o P4Pi executando o programa P4 para realizar a filtragem do tráfego redundante entre as suas interfaces. Adicionalmente, foi realizado um experimento preliminar com a mesma carga de trabalho, porém, contabilizando a perda de pacotes entre o gerador de tráfego e o P4Pi para determinar esta taxa no enlace Wi-Fi. Foram realizadas 30 (trinta) replicações em cada configuração do cenário e respectiva carga de trabalho, obtendo um intervalo de confiança assintótico de 99%.

Além da perda de mensagens fim-a-fim, foram avaliados o atraso entre o envio da mensagem LoRa ao *gateway* até a notificação realizada pelo Chirpstack a um software subscrito no tópico do sensor, bem como da carga de CPU utilizada pelos módulos da pilha do Chirpstack e da máquina virtual como um todo.

O gráfico da Figura 7 mostra o resultado do experimento preliminar para avaliar a perda de mensagens no enlace Wi-Fi entre o gerador de tráfego LoRa e o ponto de acesso utilizando o P4Pi. Este resultado mostra que para todas as cargas de trabalho utilizadas nos experimentos a perda de mensagens ocasionadas pelas características da comunicação IEEE 802.11n estão em torno de 1%.

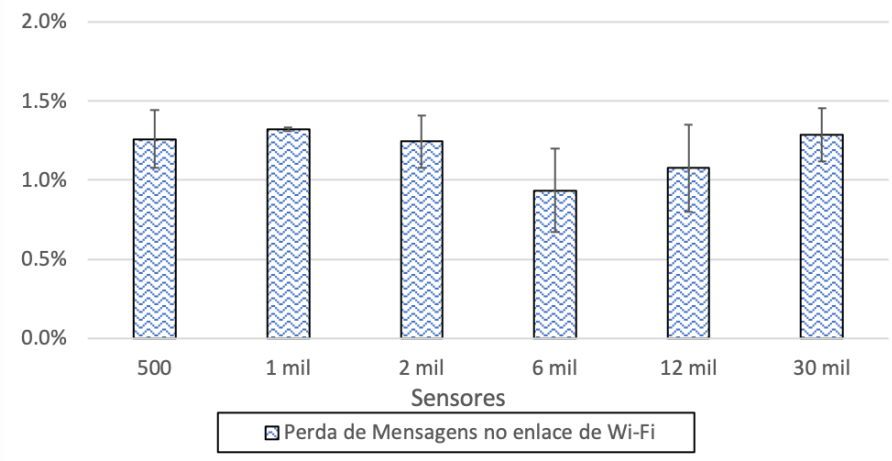

Figura 7. Perda de mensagens no enlace Wi-Fi entre o gerador de tráfego LoRa e o ponto de acesso P4Pi.

Na Figura 8 a perda de mensagens observada no experimento preliminar exibido na Figura 7 apresenta seu reflexo nas primeiras cargas de trabalho, com 500 e 1 mil sensores. Porém, a partir de 2 mil sensores fica clara a influência da capacidade da infraestrutura de névoa/nuvem em absorver este tráfego. A utilização do filtro P4 no ponto de acesso reduz a carga de trabalho sobre esta infraestrutura reduzindo significativamente a perda de mensagens. Importante ressaltar que a avaliação considera que a mensagem enviada pelo sensor foi recebida pelo consumidor (*sink*), ou seja, pelo menos uma das mensagens captadas pelos quatro *gateways* chegou até o *sink*.

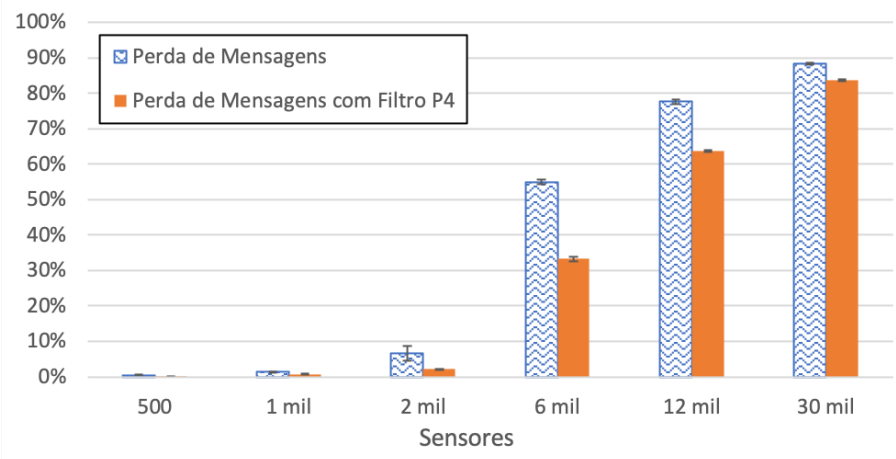

Figura 8. Perda de mensagens enviadas pelo sensor em relação às notificações recebidas pelo usuário (*sink*) sem a filtragem de pacotes e com a filtragem de pacotes em P4.

O argumento da sobrecarga da infraestrutura quando há mensagens redundantes é confirmado na avaliação do uso de CPU, exibido na Figura 9. Esta redução de uso de CPU obtida pela filtragem do tráfego não só gera economia de recursos computacionais na névoa/nuvem, como também produz um efeito direto no atraso do recebimento das mensagens, visto na Figura 10.

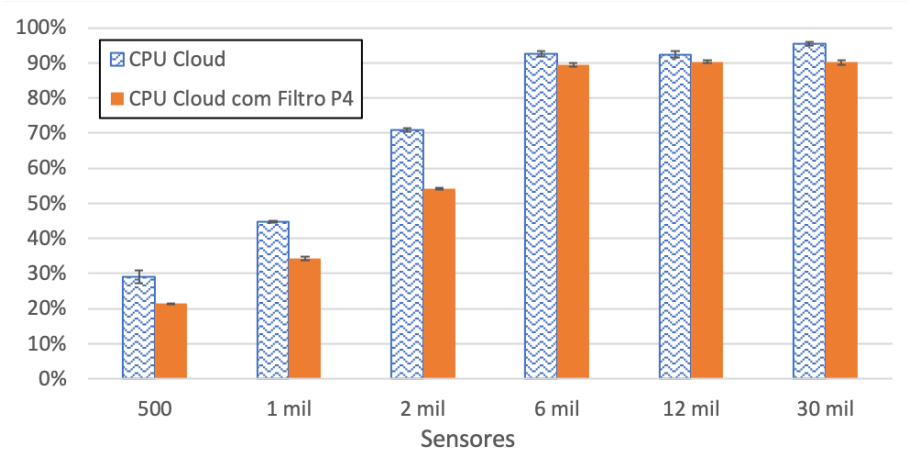

Figura 9. Uso de CPU na máquina virtual utilizada para hospedar os módulos da pilha do chirpstack, sem a filtragem de pacotes e com a filtragem de pacotes em P4.

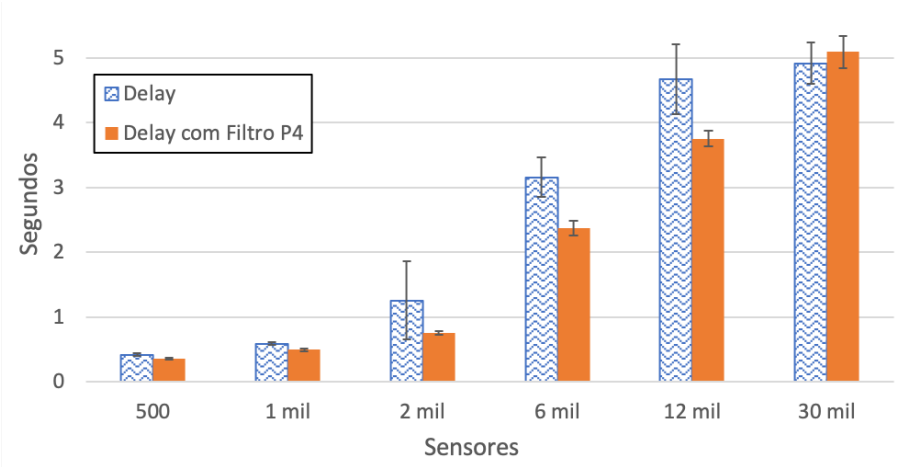

Figura 10. Atraso desde o envio do sensor até a notificação ao usuário (*sink*) sem a filtragem de pacotes e com a filtragem de pacotes em P4.

Finalmente, a Figura 11 mostra a carga de CPU nos principais módulos que compõem a pilha do LoRaWAN Chirpsatck, o GATEWAY-BRIDGE na Figura 11(a), o CHIRPSTACK-SERVER na Figura 11(b) e o MOSQUITTO MQTT-BROKER na Figura 11(c). Novamente o efeito da filtragem de tráfego apresenta um redução substancial na carga de CPU dos módulos de software.

5. Discussão

Diante dos resultados experimentais é possível observar que grande parde da responsabilidade pela escalabilidade do sistema está sobre a infraestrutura de software na

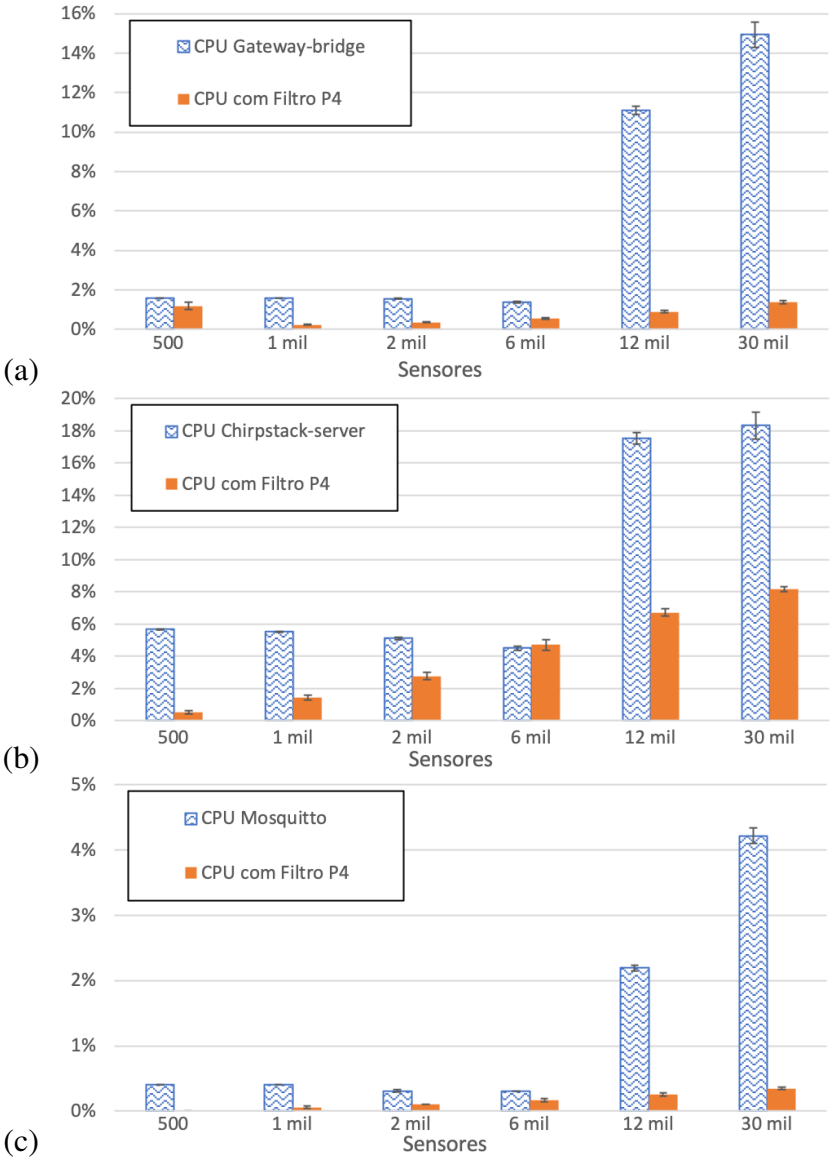

Figura 11. Uso de CPU pelos módulos da pilha Chirpstack: (a)gateway-bridge; (b)chirpstack-server; (c)MQTT broker Mosquitto.

névoa/nuvem. Apesar de intuitivamente ser apenas este o ponto de intervenção de gerenciamento para aumentar a capacidade de absorção de novos sensores e atuadores, os resultados mostram que a manipulação do tráfego é aliada neste processo.

Tomando como base para discussões a carga de trabalho de 6 mil sensores, há uma redução de 40% no taxa de perda de mensagens, mostrando que a replicação de mensagens com o uso de múltiplos *gateways* por si só não garante a entrega, causando um efeito negativo com o aumento da carga de trabalho do sistema.

Também pode ser observada uma redução de 25% no atraso do recebimento das mensagens com a carga de trabalho de 6 mil sensores e de 20% com 12 mil sensores. Já com 30 mil sensores há um aumento de 3% no atraso, causado pelo tempo de processamento do datagrama pelo P4Pi. Este aumento é esperado considerando as limitações de desempenho oferecida pelo RaspberryPi comparado com uma solução nativa P4, como o

chip Intel Tofino, por exemplo. No entanto, mesmo com o aumento do atraso, houve uma redução de 5% na taxa de perda de mensagens nesta carga de trabalho.

A análise do uso de CPU pelos módulos de software da pilha Chirpstack apresentam uma redução substancial em praticamente todas as cargas de trabalho experimentadas, com exceção à carga de trabalho de 6 mil sensores no módulo CHIRPSTACK-SERVER. Além do aumento na capacidade da infraestrutura de absorver o aumento no número de sensores, esta redução também pode influenciar no custo da operação (OPEX), seja pela redução da configuração da máquina virtual utilizada, ou pela quantidade de instâncias utilizadas em uma arquitetura *server-less*, por exemplo.

Finalmente, a redução no tráfego redundante pode permitir a redução na capacidade do enlace em pontos críticos do trajeto, por vezes representando o maior custo da solução. Áreas remotas como regiões agrícolas podem exigir enlaces de alto custo [Zyrianoff et al. 2018, Queté et al. 2020].

6. Conclusão

Este trabalho abordou um dos desafios de Internet das Coisas, a absorção de uma quantidade cada vez maior de dispositivos e seu reflexo no tráfego de rede e infraestrutura computacional adjacente. Para tratar este problema, o uso da técnica de programação do plano de dados com P4 mostrou-se eficaz na redução do uso de CPU, atraso e perda de mensagens. Outra importante contribuição foi o uso da plataforma P4Pi para prover uma solução viável em P4 na borda da rede, tornando acessível o uso desta tecnologia em pesquisas e em soluções de baixo custo. Espera-se em trabalhos futuros explorar ainda mais o potencial da linguagem P4 com melhoria no algoritmo utilizado para filtragem de tráfego, além de novas intervenções no tráfego, como telemetria, novos protocolos, entre outras possibilidades.

Agradecimentos

À Fundação de Apoio a Pesquisa do Estado de São Paulo(FAPESP) pelo suporte ao projeto PROFISSA nº 20/05152-7.

Referências

Agrawal, A. and Kim, C. (2020). Intel tofino2–a 12.9 tbps p4-programmable ethernet switch. In *2020 IEEE Hot Chips 32 Symposium (HCS)*, pages 1–32. IEEE Computer Society.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., et al. (2014). P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95.

Ericsson, T. L. (2019). Growth of the internet of things and in the number of connected devices is driven by emerging applications and business models, and supported by standardization and falling device costs. In *Internet of Things - number of connected devices worldwide 2015-2025*. Ericsson Co. [Online; accessed 25-October-2021].

Hauser, F., Häberle, M., Merling, D., Lindner, S., Gurevich, V., Zeiger, F., Frank, R., and Menth, M. (2021). A survey on data plane programming with p4: Fundamentals, advances, and applied research. *arXiv preprint arXiv:2101.10632*.

Ibanez, S., Brebner, G., McKeown, N., and Zilberman, N. (2019). The p4-¿ netfpga workflow for line-rate packet processing. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 1–9.

Laki, S., Stoyanov, R., Kis, D., Soulé, R., Vörös, P., and Zilberman, N. (2021). P4pi: P4 on raspberry pi for networking education. *ACM SIGCOMM Computer Communication Review*, 51(3):17–21.

Madureira, A. L. R., Araújo, F. R. C., and Sampaio, L. N. (2020). On supporting iot data aggregation through programmable data planes. *Computer Networks*, 177:107330.

Queté, B., Heideker, A., Zyrianoff, I., Ottolini, D., Kleinschmidt, J. H., Soininen, J.-P., and Kamienski, C. (2020). Understanding the tradeoffs of lorawan for iot-based smart irrigation. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 73–77. IEEE.

Raza, U., Kulkarni, P., and Sooriyabandara, M. (2017). Low power wide area networks: An overview. *ieee communications surveys & tutorials*, 19(2):855–873.

Resner, D., de Araujo, G. M., and Fröhlich, A. A. (2018). Design and implementation of a cross-layer iot protocol. *Science of Computer Programming*, 165:24–37.

Sapio, A., Abdelaziz, I., Aldilaijan, A., Canini, M., and Kalnis, P. (2017). In-network computation is a dumb idea whose time has come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pages 150–156.

Scholz, D., Oeldemann, A., Geyer, F., Gallenmüller, S., Stubbe, H., Wild, T., Herkersdorf, A., and Carle, G. (2019). Cryptographic hashing in p4 data planes. In *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 1–6. IEEE.

Singh, S. and Jha, R. K. (2017). A survey on software defined networking: Architecture for next generation network. *Journal of Network and Systems Management*, 25(2):321–374.

Sornin, N., Luis, M., Eirich, T., Kramp, T., and Hersent, O. (2016). Lorawan specification. lora alliance, version 1.0. 2. 2016. `https://lora-alliance.org/resource_hub/lorawan-specification-v1-0-2/`. [Online; acessado em 21-Dezembro-2022].

Vörös, P., Horpácsi, D., Kitlei, R., Leskó, D., Tejfel, M., and Laki, S. (2018). T4p4s: A target-independent compiler for protocol-independent packet processors. In *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–8. IEEE.

Wang, S.-Y., Wu, C.-M., Lin, Y.-B., and Huang, C.-C. (2019). High-speed data-plane packet aggregation and disaggregation by p4 switches. *Journal of Network and Computer Applications*, 142:98–110.

Zyrianoff, I., Heideker, A., Silva, D., and Kamienski, C. (2018). Scalability of an internet of things platform for smart water management for agriculture. In *2018 23rd Conference of Open Innovations Association (FRUCT)*, pages 432–439. IEEE.