Interoperabilidade e Automação de Serviços em Redes de Internet das Coisas usando P4

Bruna Carvalho[1], **Dener Silva**[1], **Lucas Trombeta**[1]
Alexandre Heideker[2], **Carlos Kamienski**[1], **João Henrique Kleinschmidt**[1]

[1] Universidade Federal do ABC (UFABC)
Santo André – SP – Brazil
[2]Alma Mater Studiorum – Università di Bologna (UNIBO)
Bologna – BO – Italy

{bruna.carvalho, dener.silva, lucas.trombeta

joao.kleinschmidt, carlos.kamienski}@ufabc.edu.br

alexandre.heideker@unibo.it

Abstract. *The P4 programming language has been widely used for data plane programming in network devices, allowing packet processing customization in physical or virtual switches and routers. This work explores its application in the Internet of Things (IoT), investigating its feasibility for automation and interoperability. The research addresses the handling of different protocols and data formats in IoT applications with P4. Using P4 Flow Forge and P4Docker, a multiprotocol gateway was developed, capable of identifying different protocols and data types and generating alerts via MQTT. The results demonstrate P4's potential to optimize IoT network management.*

Resumo. *A linguagem de programação P4 tem sido amplamente utilizada na programação do plano de dados em dispositivos de rede, permitindo a personalização do processamento de pacotes em switches e roteadores físicos ou virtuais. Este trabalho explora sua aplicação na Internet das Coisas (IoT), investigando sua viabilidade para automação e interoperabilidade. A pesquisa aborda o tratamento de diferentes protocolos e formatos de dados em aplicações IoT com P4. Utilizando P4 Flow Forge e P4Docker, foi desenvolvido um gateway multiprotocolo capaz de identificar diferentes protocolos e tipos de dados, além de gerar alertas via MQTT. Os resultados demonstram o potencial do P4 para otimizar o gerenciamento de redes IoT.*

1. Introdução

O crescimento da Internet das Coisas (IoT) impõe desafios que demandam abordagens inovadoras [Tahsien et al. 2020]. As redes definidas por software (SDNs) revolucionam a arquitetura de redes ao separar controle e plano de dados, aumentando flexibilidade e eficiência. Nesse contexto, P4 surge como solução para a programação do plano de dados e o processamento de pacotes em SDN, superando limitações de programabilidade e adaptabilidade [BUDIU and DODD 2017]. O uso de SDN em redes IoT é amplamente explorado, e P4 pode trazer benefícios adicionais.

A IoT envolve diversos dispositivos que se comunicam por protocolos como MQTT (*Message Queuing Telemetry Transport*) e CoAP (*Constrained Application Protocol*). *Gateways* IoT são fundamentais para integrar dispositivos e enviar dados à nuvem, melhorando a interoperabilidade e automatizando serviços [Heideker et al. 2025]. Apesar dos benefícios, a aplicação de P4 enfrenta desafios, como implementação em hardware específico e suporte limitado. Estudos propõem seu uso em agregação de dados, automação de serviços e segurança [MADUREIRA et al. 2020, QIN et al. 2020], mas sua aplicação na IoT ainda é pouco explorada.

Este trabalho investiga a aplicação de P4 em *gateways* IoT, analisando dois cenários: um *gateway* multiprotocolo para interoperabilidade entre dispositivos e sua extensão para *smart health*, explorando IoT em contextos médicos.

1.1. Objetivos e Contribuições

O objetivo geral é analisar a aplicação da linguagem P4 na IoT, explorando sua integração com dispositivos e redes de comunicação. Especificamente, busca-se investigar como a programação do plano de dados aprimora a automação de serviços e a interoperabilidade em um *gateway* IoT, possibilitando a classificação e o encaminhamento eficiente de pacotes em tempo real. A principal contribuição é demonstrar como P4 resolve desafios da IoT, como a classificação e manipulação de pacotes em tempo real. Os resultados mostram que seu uso melhora a automação de serviços e a inspeção profunda de pacotes, permitindo classificar protocolos como HTTP, MQTT e UDP, além de gerar alertas automáticos no cenário de *smart health* através da análise de *payload* em tempo real.

1.2. Organização do Trabalho

O artigo é organizado em cinco seções: a Seção 1 introduz o tema e os objetivos, a Seção 2 aborda a linguagem P4 e suas ferramentas, a Seção 3 descreve os cenários de simulação, a Seção 4 analisa os resultados, e a Seção 5 apresenta as conclusões.

2. Revisão de Literatura

A IoT oferece uma vasta gama de possibilidades e diversas aplicações, mas enfrenta desafios como escalabilidade, gerenciamento de grandes volumes de dados, segurança e privacidade [MAKHSHARI and MESBAH 2021]. Para lidar com esses desafios, especialmente no que diz respeito à flexibilidade e ao controle da infraestrutura de rede, arquiteturas baseadas em SDN vêm sendo amplamente adotadas. Nesse contexto, a linguagem P4 se destaca pela programação de processadores de pacotes independentes de protocolo, possibilitando a personalização do plano de dados e a interoperabilidade entre diferentes protocolos.

2.1. Redes Definidas por Software (SDN)

As SDNs se baseiam na separação dos planos de controle e encaminhamento, diferentemente da arquitetura tradicional. Nessa abordagem, o plano de controle é centralizado em um controlador que gerencia vários dispositivos de rede, proporcionando vantagens de orquestração, automação e gestão dinâmica do tráfego. A arquitetura SDN possui três camadas: aplicação, controle e dados. A primeira abriga soluções como monitoramento e gestão de tráfego. O controlador, na camada de controle, desempenha um papel central na orquestração da rede, enquanto a camada de dados inclui dispositivos como *switches* e roteadores [Farris et al. 2018].

2.2. Programação de Processadores de Pacotes Independentes de Protocolo (P4)

A linguagem P4 permite programar a análise e o processamento de pacotes no plano de dados de forma independente do protocolo, garantindo flexibilidade e reconfiguração. Apesar de suas possibilidades, P4 possui limitações, como a ausência de funções recursivas e alocação dinâmica de memória. Contudo, sua flexibilidade permite explorar novas abordagens de pesquisa, especialmente em contextos como a IoT [BUDIU and DODD 2017].

2.3. Internet das Coisas (IoT)

A IoT é uma rede de dispositivos conectados à internet, como *smartphones*, sensores e atuadores, que facilita a interação entre os mundos físico e digital. Sua arquitetura clássica é dividida em três camadas: a camada de aplicação, que contém as interfaces de uso; a camada de rede, responsável pela troca de informações; e a camada de dispositivos, composta pelos sensores e atuadores. Com o crescimento da IoT, surgem desafios relacionados ao processamento de grandes volumes de dados, escalabilidade e interoperabilidade entre dispositivos [Tahsien et al. 2020].

2.4. *P4Docker*

O *P4Docker* é uma ferramenta baseada em *containers* Docker para emulação de topologias de rede em testes com P4, incluindo um switch BMv2 e uma interface gráfica (GUI) [P4Docker] [Silva et al. 2024]. Desenvolvida no projeto PROFISSA [PROFISSA] sua arquitetura é composta por um *frontend* em JavaScript e um *backend* que gera *scripts* para gerenciar as topologias e integrar o compilador P4. Ela permite o armazenamento persistente e o compartilhamento de conteúdo entre *containers*, além de gerar *scripts* para desmontar ambientes.

2.5. P4 *Flow Forge*

O *P4 Flow Forge* é uma ferramenta para geração e análise de tráfego, que suporta protocolos como TCP, UDP, HTTP, MQTT e personalizados via JSON [P4FlowForge]. Desenvolvida em *Python*, adota uma arquitetura cliente-servidor, onde o cliente gera o tráfego e o servidor analisa atraso e latência. No MQTT, um *broker* facilita a comunicação.

2.6. Trabalhos Relacionados

Os principais estudos sobre P4 em IoT exploram automação de serviços, agregação de dados e segurança, além de comparações tecnológicas. Carrascal et al. analisaram P4 e *Express Data Path* (XDP), indicando P4 para *gateways* IoT e XDP para dispositivos com restrições de memória e processamento [CARRASCAL et al. 2020]. Na automação de serviços, Uddin et al. propuseram o BLESS, um *switch* programável para *Bluetooth Low Energy* (BLE), e o Muppet, uma arquitetura multiprotocolo para BLE e ZigBee [UDDIN et al. 2017, UDDIN et al. 2018]. Atutxa et al. usaram P4 em um *SmartNIC* para otimizar alarmes via MQTT [Atutxa et al. 2021].

Na agregação de dados, o protocolo IoTP, implementado em P4, melhorou a eficiência da rede em 78% [MADUREIRA et al. 2020]. Wang et al. desenvolveram métodos para agregação e desagregação de pacotes, atingindo taxas de 100 Gbps [Wang et al. 2020]. Na segurança, Basil et al. analisaram dispositivos domésticos inteligentes [Basil et al. 2018], enquanto P4ANIS foi proposto para evitar espionagem

[Liu et al. 2021]. Quin et al. usaram aprendizado profundo para detectar ataques em redes IoT [QIN et al. 2020]. Estes trabalhos mostram que aplicações IoT podem se beneficiar da programação do plano de dados com P4, mas questões de como abordar diferentes protocolos e formatos de dados em aplicações IoT com P4 não tem sido exploradas na literatura.

3. Cenários de Simulação

Nesta etapa, foram definidos dois cenários para explorar a tecnologia P4 na interoperabilidade e automação de serviços em redes IoT. Utilizou-se o *P4Docker* para criar os *testbeds* e o *P4 Flow Forge* para gerar fluxos de pacotes. Os testes analisaram como P4 auxilia no processamento de pacotes entre os dispositivos IoT. Os cenários simulam a interoperabilidade entre dispositivos com diferentes protocolos e formato de dados e a automação de serviços, como análise de dados de sensores e geração de alertas, visando melhorar o monitoramento de pacientes e a resposta a emergências.

Os testes foram realizados em um ambiente que simula a comunicação entre dispositivos IoT, representados como estações sem fio (h1, h2 etc.), utilizando protocolos como MQTT e HTTP. Para estabelecer uma linha de base de latência, também foram feitas simulações com um *switch OVS* convencional, sem tecnologia P4 ou alterações na análise de pacotes.

3.1. Cenário 1 - *Gateway* Multiprotocolo

O primeiro cenário, ilustrado na Figura 1, simula um *gateway* multiprotocolo para interoperabilidade entre dispositivos IoT com diferentes protocolos e formatos de dados.

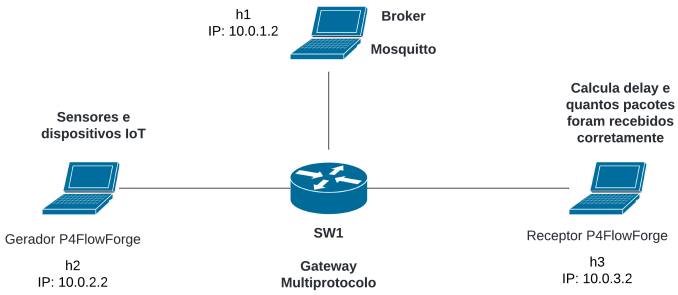


Figura 1. Diagrama do cenário 1 - *Gateway* Multiprotocolo

Neste cenário, um *switch* P4 (SW1) atua como *gateway* multiprotocolo, com três *hosts*: um gera pacotes, outro os recebe (Figura 2) e calcula métricas, e o terceiro funciona como *broker* MQTT. O *P4 Flow Forge* gera pacotes com diferentes formatos e protocolos, que são classificados e processados pelo *parser* do *switch*, com registros em *log* para análise de desempenho. Para isso, cabeçalhos específicos foram mapeados para cada protocolo. No MQTT, a parte fixa permite identificar pacotes *publish*; no HTTP, a definição de uma porção fixa do cabeçalho possibilita acessar os dois primeiros *bytes* do *payload* e identificar o tipo de dado.

Três simulações avaliaram o atraso na entrega de pacotes, com taxas de 5, 10 e 24 mensagens por segundo durante 10 minutos. Foram testados os protocolos HTTP (HTML, JSON, XML), UDP e MQTT, gerando em média 3.000, 6.000 e 14.400 pacotes

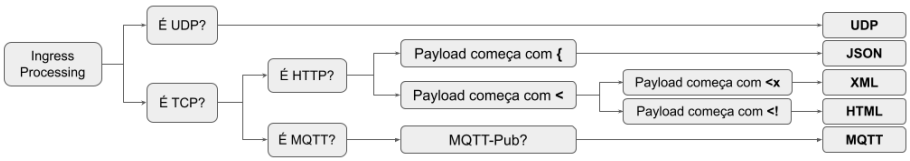


Figura 2. Fluxograma do cenário 1

por teste. Compararam-se cenários com e sem P4: no segundo, o *Docker* usou um *switch OVS* sem classificação, servindo como linha de base para medir o atraso.

3.2. Cenário 2 - *Gateway* Multiprotocolo no Contexto de *Smart Health*

O segundo cenário expande o primeiro, implementando um *gateway* multiprotocolo em *smart health* para explorar a IoT em contextos médicos. A tecnologia é aplicada na coleta de dados vitais, otimização da resposta a emergências e automação hospitalar, melhorando eficiência e qualidade do atendimento.

A topologia inclui um *switch* P4 (SW1) e dez *hosts* (h1 a h10). Nos testes com MQTT, h1 funcionou como *broker*, três *hosts* simularam dispositivos médicos gerando dados de sensores, três atuaram como receptores e os demais receberam alertas direcionados a médico, enfermeira e equipe de manutenção, conforme a Figura 3.

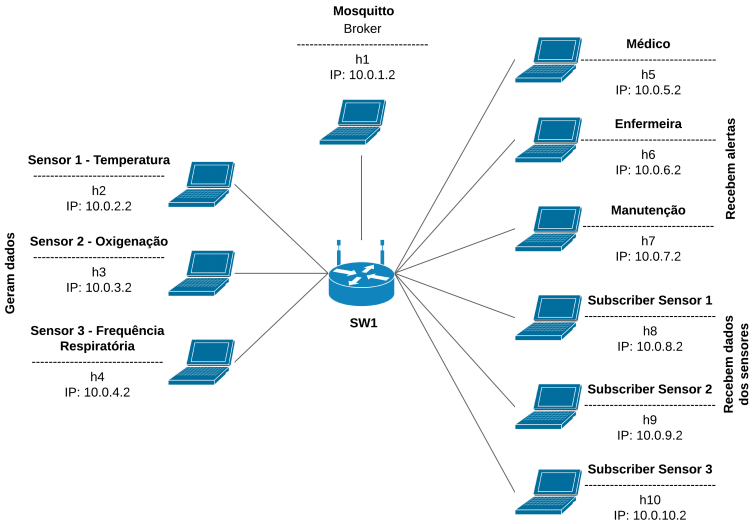


Figura 3. Diagrama do Cenário 2 - *Gateway* Multiprotocolo em *Smart Health*

A ferramenta P4 *Flow Forge* gerou pacotes MQTT com valores aleatórios de sensores, que foram monitorados em tempo real pelo *switch* P4 por meio do *parser* e da análise do *payload*. Com a definição de uma parte fixa do cabeçalho MQTT, foi possível identificar pacotes do tipo *publish* e extrair os *bytes* correspondentes ao tópico e a mensagem. Com base em faixas de valores pré-definidas no código P4, pacotes inválidos foram descartados antes de chegarem ao *broker*, evitando sobrecarga de dados inválidos no sistema. Os pacotes válidos foram classificados em duas categorias: aqueles que apresentaram valores dentro das faixas normais, os quais seguiram para o *broker* sem gerar alertas, e aqueles com valores fora dessas faixas, que também seguiram para o *broker*, mas

geraram alertas, sendo direcionados a médicos, enfermeiras ou para manutenção. Os alertas foram gerados por pacotes clonados, criados diretamente no código P4. A clonagem de pacotes no *switch* P4 possibilitou a criação de alertas específicos de forma automatizada através da manipulação do *payload* dos pacotes MQTT, permitindo a modificação dos tópicos para facilitar a identificação das notificações.

Testes com UDP e HTTP seguiram o cenário 1, com SW1 como *gateway* multiprotocolo, dois *hosts* e pacotes HTTP em HTML, JSON e XML. O cenário 2 segue o mesmo fluxo, com análise adicional de pacotes MQTT, gerando alertas conforme intervalos de referência, como mostrado na Figura 4.

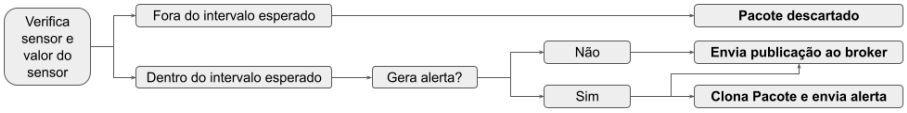


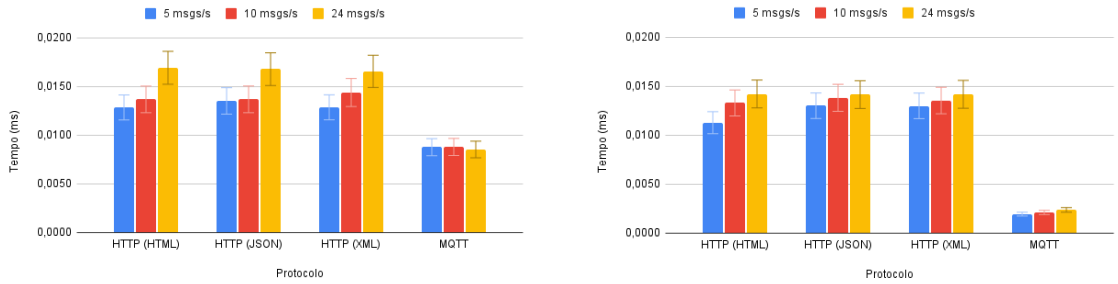
Figura 4. Fluxograma de funcionamento do cenário 2

O atraso foi medido com base no tempo de chegada dos pacotes ao destino. Os testes, realizados por 10 minutos com taxas de 5, 10 e 24 mensagens por segundo, geraram em média 3.000, 6.000 e 14.400 pacotes por protocolo e tipo de dado.

4. Resultados

4.1. Resultados do Cenário 1

Os resultados indicam uma discrepância mínima no atraso com P4, especialmente para MQTT, conforme mostrado nas Figuras 5a e 5b.



(a) Atraso dos pacotes com P4

(b) Atraso dos pacotes sem P4

Figura 5. Comparação do atraso dos pacotes com e sem P4.

No cenário sem P4, os pacotes são encaminhados sem classificação, enquanto com P4 são categorizados por tipo de dado, principal vantagem da abordagem. O protocolo MQTT apresentou a maior variação no atraso entre os cenários. Pacotes UDP, devido ao baixo atraso, não foram representados nos gráficos. A identificação dos pacotes com P4 atingiu 100% de acurácia em todas as simulações.

4.2. Resultados do Cenário 2

Para o protocolo MQTT, além do atraso dos pacotes regulares, mediu-se o atraso dos pacotes de alerta, verificando sua correta geração e entrega. Os pacotes MQTT foram distribuídos entre três sensores, mantendo o mesmo volume nas simulações. A acurácia

dos alertas foi confirmada por *logs*, atingindo 100%. Os tempos de atraso, ilustrados na Figura 6, mostraram vantagens com P4, com variações mínimas entre os testes, especialmente para MQTT. A implementação demonstrou a viabilidade da automação em saúde com P4 e IoT.

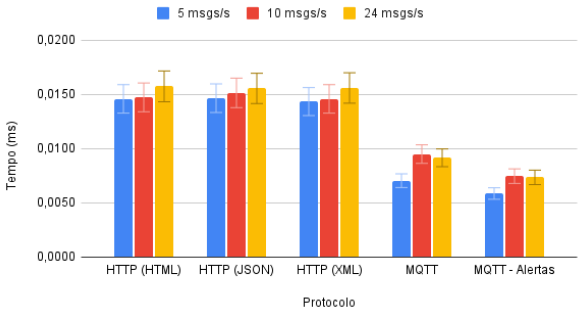


Figura 6. Atraso dos pacotes no Cenário 2

5. Conclusão

A integração da IoT com a linguagem P4 demonstra grande potencial, especialmente na automação e no gerenciamento de redes heterogêneas. Este estudo apresenta implementações que comprovam sua eficiência na classificação e manipulação de pacotes em tempo real. As simulações evidenciaram vantagens significativas, como automação de serviços e inspeção profunda de pacotes. A tecnologia permitiu a classificação de protocolos como HTTP, MQTT e UDP, além da geração de alertas automáticos no cenário de *smart health*.

Sua aplicação transforma *switches* em componentes fundamentais da solução IoT, como *smart gateways*, melhorando a interoperabilidade entre dispositivos. Além disso, alcançou 100% de precisão na identificação dos pacotes, garantindo maior eficiência e confiabilidade no processamento de dados. A solução multiprotocolo mostrou-se viável para segmentação de serviços e automação sem depender de processamento em nuvem, reduzindo a latência. Este estudo reforça o potencial da linguagem na IoT e sua aplicabilidade em cenários mais complexos, oferecendo soluções para processamento de dados na borda da rede.

Acknowledgements

This work was supported by the São Paulo Research Foundation (FAPESP), Brazil, under Grant 20/05152-7.

Referências

Atutxa, A., Franco, D., Sasiain, J., Astorga, J., and Jacob, E. (2021). Achieving low latency communications in smart industrial networks with programmable data planes. *Sensors*, 21(15):5199.

Basil, A. O., Mu, M., Sharman, J., and Goldsney, J. (2018). P4-assisted network security for future smart homes.

BUDIU, M. and DODD, C. (2017). The p416 programming language. *SIGOPS Operating Systems Review*, 51(1):5–14.

CARRASCAL, D., ROJAS, E., ÁLVAREZ-HORCAJO, J., LOPEZ-PAJARES, D., and MARTÍNEZ-YELMO, I. (2020). Analysis of p4 and xdp for iot programmability in 6g and beyond. *IoT*, 1(2):605–622.

Farris, I., Taleb, T., Khettab, Y., and Song, J. (2018). A survey on emerging sdn and nfv security mechanisms for iot systems. *IEEE Communications Surveys & Tutorials*, 21(1):812–837.

Heideker, A., Silva, D., Kamienski, C. A., and Trotta, A. (2025). Achieving seamless iot interoperability through data plane programmability. In *2025 IEEE 22st Consumer Communications Networking Conference (CCNC)*, pages 1–6.

Liu, G., Quan, W., Cheng, N., Gao, D., Lu, N., Zhang, H., and Shen, X. (2021). Softwarized iot network immunity against eavesdropping with programmable data planes. *IEEE Internet of Things Journal*, 8(8):6578–6590.

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.

MAKHSHARI, A. and MESBAH, A. (2021). Iot bugs and development challenges. In *IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 460–472.

P4Docker. P4docker. `https://https://dnredsons-organization.gitbook.io/p4docker`. Accessed: 2024-01-05.

P4FlowForge. P4flowforge. `https://dnredsons-organization.gitbook.io/p4-flow-forge`. Accessed: 2024-01-05.

PROFISSA, P. Projeto profissa. `https://profissa.rnp.br`. Accessed: 2024-01-25.

QIN, Q., POULARAKIS, K., and TASSIULAS, L. (2020). A learning approach with programmable data plane towards iot security. In *IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 410–420.

Silva, D., Heideker, A., Trombeta, L., Carvalho, B., Kleinschmidt, J., and Kamienski, C. (2024). P4docker: Enabling efficient p4 switch testbeds with docker integration. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Niterói-RJ.

Tahsien, S. M., Karimipour, H., and Spachos, P. (2020). Machine learning based solutions for security of internet of things (iot): A survey. *Journal of Network and Computer Applications*, 161:102630.

UDDIN, M., MUKHERJEE, S., CHANG, H., and LAKSHMAN, T. V. (2017). Sdn-based service automation for iot. In *IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–10.

UDDIN, M., MUKHERJEE, S., CHANG, H., and LAKSHMAN, T. V. (2018). Sdn-based multi-protocol edge switching for iot service automation. *IEEE Journal on Selected Areas in Communications*, 36(12):2775–2786.

Wang, S. Y., Li, J. Y., and Lin, Y. B. (2020). Aggregating and disaggregating packets with various sizes of payload in p4 switches at 100 gbps line rate. *Journal of Network and Computer Applications*, 165:102676.