# Avaliação de Linguagens de Programação do Plano de Dados em Redes Definidas por Software*

**Juliana Duarte Bol[1], Paula Duarte Bol[2], Rafael P. Esteves[1], Weverton Cordeiro[2], Roben C. Lunardi[1]**

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Campus Restinga
Porto Alegre – RS – Brasil

[2]Universidade Federal do Rio Grande do Sul (UFRGS) – Instituto de Informática
Porto Alegre, RS – Brasil.

{2020008028, rafael.esteves, roben.lunardi}@restinga.ifrs.edu.br,

{paula.bol, weverton.cordeiro}@inf.ufrgs.br

***Abstract.*** *The evolution of computer networking infrastructure and communication protocols has been restricted in the last decades. Some reasons for this are the difficulty in performing network infrastructure changes, the predominance of vendor-specific technologies and the limitations on available hardware. Software-Defined Networking (SDN) presents a new paradigm, helping the management of complex networks due to the possibility of programming the data plane. In order to provide an evaluation of available programming languages for the data plane, this work presents a preliminary study and an early analysis of the performance of programming languages POF, P4, $\mu$P4 and NPL.*

***Resumo.*** *A evolução das infraestruturas de redes de computadores e protocolos de comunicação tem sido restringida nas últimas décadas. Isso ocorreu devido às dificuldades em realizar mudanças na infraestrutura de rede, a predominância de tecnologias específicas do fornecedor e as limitações no hardware disponível. As Redes Definidas por Software (SDN) trazem um novo paradigma para as redes, auxiliando no gerenciamento e administração de redes complexas devido à possibilidade de programação do plano de dados. A fim de fornecer uma avaliação das linguagens de programação disponíveis para o plano de dados, este trabalho apresenta um estudo preliminar e uma análise inicial do desempenho das linguagens POF, P4, $\mu$P4, e NPL.*

## 1. Introdução

O uso da Internet está cada vez mais presente no dia a dia da sociedade. Porém, as redes de computadores tradicionais são muito complexas e de difícil gerenciamento, fazendo com que um novo protocolo de rede possa levar anos até a sua implementação. Diferente das redes atuais, que possuem o plano de controle e plano de dados agrupados, as Redes definidas por software (SDNs - *Software Defined Networks*) propõem a separação lógica do controle da rede, tornando mais simples a criação e introdução de novas abstrações sem exigir modificações significativas nos dispositivos de rede [Kreutz et al. 2015].

*O presente trabalho foi realizado com apoio do IFRS através do Edital IFRS Nº 12/2021.

As redes SDN podem ser descritas como uma composição de camadas, onde cada uma delas tem sua função específica. A camada do Plano de Aplicação, contém as aplicações responsáveis pela criação e implementação de novas funções que atuarão na rede, sendo responsável pelo gerenciamento da mesma. É através dela que o administrador pode interagir e alterar a atuação dos elementos que a compõem. A camada do Plano de Controle, por sua vez, é responsável pela parte inteligente da rede, por meio da implementação de um controlador, que facilita a modificação e adoção de novas políticas de rede. É nesse plano que também é implementada a interface responsável por permitir a realização das alterações pelo administrador. Por sua vez, a camada do Plano de Dados fica responsável somente pelo encaminhamento dos pacotes. Os elementos responsáveis por realizar este processo (como *switches* e roteadores) não utilizam protocolos de roteamento, pois estes são implementados no Plano de Controle [Farhad et al. 2014].

Redes Definidas por Software surgem como uma oportunidade de quebrar a ossificação das redes, promovendo a separação da lógica de controle e operação da rede e dos dispositivos nelas utilizados [Cordeiro et al. 2017]. A separação do plano de controle e de dados é realizada através de uma interface de programação (API) entre os *switches* e o controlador SDN, como por exemplo, OpenFlow [McKeown et al. 2008]. Portanto, SDN e OpenFlow oferecem uma interface aberta para construção de aplicações dinâmicas de redes, mudando a forma de configurar os dispositivos e determinar o comportamento da rede. Contudo, o protocolo OpenFlow possui algumas limitações, especialmente no que diz respeito a mudar o comportamento do dispositivo de forma dinâmica. Limitações estas que podem ser contornadas com a utilização do plano de dados programável, por meio de linguagens como POF (*Protocol-Oblivious Forwarding*) [Song 2013], NPL (*Network Programming Language*) [NPL 2019], P4 (*Programming Protocol-independent Packet Processors*) [Bosshart et al. 2014] e $\mu$P4 [Soni et al. 2020]. Tais linguagens possuem potencial para possibilitar um encaminhamento flexível e personalizável através da configuração dos *switches*.

Este artigo visa avaliar o impacto do uso de diferentes linguagens de programação de Plano de Dados em termos de desempenho, usabilidade, reuso e modularidade. Inicialmente serão apresentadas as principais características de cada uma das linguagens que serão estudadas. Por fim, será realizado um comparativo preliminar entre as mesmas.

## 2. SDN e Programabilidade do Plano de Dados

Em uma rede SDN, o plano de dados, o plano de controle e o plano de aplicação são separados e devem possuir interfaces para que seja realizada a comunicação (Figura 1). As aplicações (plano de aplicação) interagem com os controladores SDN (plano de controle) através da API *Northbound*. Os controladores SDN se comunicam com os dispositivos de encaminhamento (plano de dados) através de uma API *Southbound* (ex: OpenFlow) para definir regras de encaminhamento, além de extrair informações (como estatísticas e eventos) que serão enviadas para as aplicações [Kim and Feamster 2013]. Por exemplo, um controlador pode ser utilizado por diferentes aplicações, como depuradores [Tavares et al. 2017] e sistemas de prevenção de intrusões [Neu et al. 2018]. O plano de dados, por sua vez, é responsável por realizar o encaminhamento de pacotes entre os dispositivos, abstraindo o hardware utilizado [Farhad et al. 2014].

O protocolo OpenFlow se consolidou como o padrão de fato para comunicação

entre os controladores SDN e os dispositivos do plano de dados. O OpenFlow especifica os cabeçalhos dos protocolos operados por ele [McKeown et al. 2008]. Um *switch* OpenFlow é composto por tabelas de fluxos responsáveis por manter as informações sobre os fluxos e encaminhamentos a serem realizados pelo *switch*. Entretanto, o protocolo OpenFlow possui campos fixos de cabeçalhos, e limitações a protocolos existentes. Tais limitações podem ser contornadas com o plano de dados programável. Desta forma, com a possibilidade de utilizar um plano de dados programável, permite-se programar políticas de redes, bem como corrigir problemas de desempenho e uso mais adequado do hardware [Farhad et al. 2014]. Diversas linguagens foram propostas para a programabilidade do plano de dados. As linguagens discutidas no presente artigo são descritas a seguir.

A linguagem de programação P4 descreve como um pacote deve ser processado e implementado em *switches*, placas de rede, roteadores e dispositivos do tipo FPGA. A linguagem P4 auxilia a superar as limitações do OpenFlow, proporcionando a programabilidade das redes SDN [Garcia et al. 2018]. Os principais componentes da linguagem P4 associados ao modelo PISA *(Protocol-Independent Switch Architecture)* são *Headers*, *Parsers*, *Tables* e *Actions*. O *Parser* identifica os *Headers* presentes em cada pacote. Cada tabela *match-action* realiza uma busca em um subconjunto de campos de cabeçalho e aplica as ações correspondentes ao primeiro *matching* encontrado [Bosshart et al. 2014].

µP4 é uma linguagem de programação com uma arquitetura lógica que proporciona uma fina granularidade de abstrações para a composição do plano de dados. Deste modo, µP4 permite escrever os programas de forma modular, aproveitando funções definidas em simples programas e bibliotecas de código reutilizáveis. A linguagem µP4 é um *framework* do P4, com novas construções para oferecer modularidade e composição. Cada módulo µP4 processa um pacote completo ou parcial com metadados associados e gera um ou mais pacotes com metadados possivelmente modificados [Soni et al. 2020].

POF é uma linguagem com encaminhamento independente de protocolo, pois não precisa entender o formato do pacote. A análise do pacote é realizada pelo controlador, através de uma sequência de montagem da chave e instruções de uma tabela. Uma chave de pesquisa é definida como tuplas de deslocamento *(offset)* e extensão *(length)*, onde a primeira indica os *bits* ignorados, e a segunda o número de *bits* que deve ser incluído na chave a partir da posição de deslocamento. As tuplas são concatenadas para a formação da chave completa de pesquisa [Song 2013].

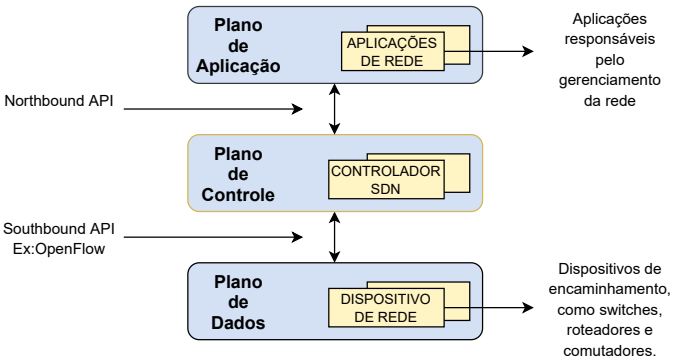


**Figura 1. Arquitetura SDN.**

A linguagem NPL tem como objetivo principal o comportamento do processamento de pacotes utilizando um conjunto de construções. De acordo com as especificações da linguagem, um aplicativo NPL inclui construções de alto nível para análise, combinação de tabelas e edição de pacotes, podendo incluir também construções específicas para outros recursos [NPL 2019].

## 3. Avaliação Preliminar

Nosso estudo comparativo das linguagens de programação do plano de dados inicia com a análise da compatibilidade de tais linguagens no que diz respeito ao suporte a modularização, reuso, e manutenção/atualizações disponíveis para usuários. Este comparativo pode auxiliar na escolha da linguagem a ser utilizada, de forma a atender as necessidades do administrador da rede.

### 3.1. Critérios para Avaliação

Para realizar a comparação, utilizamos três critérios qualitativos. Vale destacar, que em trabalhos futuros, critérios quantitativos serão analisados. Os critérios qualitativos utilizados neste trabalho são:

- **Modularização**: Capacidade de criar módulos funcionais que se comunicam uns com os outros. Este critério pode auxiliar na escolha de uma linguagem para a elaboração de programas que necessitem encapsulamento e reuso de bibliotecas.
- **Reuso**: Para que possibilite o uso de trechos de código para implantação em diferentes *switches* e diferentes arquiteturas. Este critério, auxilia na avaliação de uma linguagem que seja capaz de ser utilizada independente da arquitetura e infraestrutura rede, gerando código otimizado para cada dispositivo de encaminhamento.
- **Manutenção/Atualizações**: Atualizações de código disponíveis, tutoriais da linguagem, códigos disponíveis para o usuário. Este critério, permite avaliar a continuidade e suporte da linguagem a ser adotada.

### 3.2. Discussão

Conforme apresentado na Tabela 1, a linguagem de programação P4 não possui as formas de abstração e encapsulamento necessárias para programas altamente modulares. No entanto admite a adição de novas funções, o que permite um nível médio de modularização. Os códigos de programas em P4 geralmente são escritos em estilo monolítico, tendo um nível baixo no que diz respeito ao reuso de código. Como programas P4 são dependentes do modelo do *pipeline* para qual foi escrito, portar o programa para outra arquitetura sem mudanças relevantes é extremamente difícil. A linguagem P4 possui atualizações e tutoriais disponíveis em seu repositório no GitHub[1], com última atualização em 13/09/2021.

**Tabela 1. Comparação das linguagens de programação do plano de dados**

| Linguagem | Modularização | Reuso | Manutenção/Atualizações |
|---|---|---|---|
| **P4** | Médio | Baixo | Alto |
| **$\mu$P4** | Alto | Alto | Médio |
| **POF** | Baixo | Baixo | Baixo |
| **NPL** | Médio | Baixo | Baixo |

[1]Disponível em https://github.com/p4lang .

Por outro lado, o *framework* μP4 oferece construções para que os usuários escrevam programas com suporte de alto nível de modularização e composição. Com base em funções e bibliotecas reutilizáveis e não dependentes das demais funções e hardware utilizado, o μP4 apresenta um alto nível de reuso. Quanto a manutenção da linguagem, o μP4 não possui atualizações desde 30/07/2020 em seu repositório no GitHub[2].

Semelhante ao P4, o POF (Protocol-Oblivious Forwarding) tenta desacoplar os protocolos de encaminhamento tornando o plano de encaminhamento reconfigurável e programável, possuindo um conjunto de instruções independente do protocolo. A arquitetura de rede POF possui um controlador centralizado para gerenciar o encaminhamento de pacotes, sendo dependente da utilização do OpenFlow [Song 2013]. Sem atualizações desde 09/02/2015, a linguagem possui repositório no GitHub[3].

Outra linguagem alvo do nosso estudo é o NPL (*Network Programming Language*), que permite ao usuário especificar detalhes de tabelas e outros objetos para definir o comportamento da rede. NPL oferece construções que permitem a inclusão de bibliotecas de componentes, mas fortemente dependente do hardware. É uma linguagem projetada para fornecer todos os blocos de construção para implementar uma solução de comutação de rede [NPL 2019]. Mais informações podem ser obtidas na página oficial[4].

Pode-se observar que as linguagens avaliadas possuem diferentes características no que diz respeito aos critérios utilizados nesta comparação inicial. Em comum, todas visam suprir as limitações do OpenFlow, porém variam a forma de promover encaminhamento dos pacotes e gerenciamento do comportamento da rede. Em especial, destacamos a possibilidade de modularização e reuso da linguagem $\mu$P4. Devido a estas características, o $\mu$P4 pode ser usado em arquiteturas diversas para a implementação de políticas específicas em cada *switch* em uma arquitetura OneBigSwtich [Bol et al. 2021].

## 4. Conclusões e Trabalhos Futuros

Com a evolução das pesquisas em SDN, diferentes linguagens de programação do plano de dados foram propostas para contornar limitações do protocolo OpenFlow. Para auxiliar no entendimento e escolha, este trabalho apresentou uma avaliação de quatro linguagens para programação do plano de dados: P4, $\mu$P4, POF e NPL. Para avaliar essas linguagens, foram estabelecidos 3 critérios qualitativos. Como resultado da avaliação, percebeu-se que a linguagem $\mu$P4 possui maior nível de modularização e reuso, enquanto o P4 é a linguagem com atualizações mais recentes e recorrentes. Desta forma, o $\mu$P4 parece mais adequado em cenários que necessitem de otimização do código em cada *switch*. Por outro lado, o P4 é a linguagem com mais atualizações, e por consequência, com maior suporte e manutenabilidade.

Como trabalhos futuros, pretendemos realizar a avaliação das linguagens de programação do plano dados com a utilização de critérios quantitativos, como medidas de desempenho, uso da memória, dentre outros. Além disso, pretende-se avaliar estas linguagens em diferentes cenários, como por exemplo, com o particionamento de políticas em cada *switch*, conforme apresentado por *Bol et* al. [Bol et al. 2021], assim como a análise do nível de portabilidade das linguagens em diferentes arquiteturas.

[2]Disponível em https://github.com/cornell-netlab/MicroP4 .

[3]Disponível em https://github.com/ProtocolObliviousForwarding .

[4]Disponível em https://nplang.org/ .

## Referências


Bol, P. D., Lunardi, R., de França, B., and Cordeiro, W. (2021). Modular switch deployment in programmable forwarding planes with switch(de)composer. In *Proceedings of the SIGCOMM '21 Poster and Demo Sessions*, SIGCOMM '21, page 30–32.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and Walker, D. (2014). P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95.

Cordeiro, W., Marques, J., and Gaspary, L. (2017). Data plane programmability beyond openflow: Opportunities and challenges for network and service operations and management. *Journal of Network and Systems Management*, 25(4):784–818.

Farhad, H., Lee, H., and Nakao, A. (2014). Data plane programmability in sdn. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 583–588.

Garcia, L. F. U., Villaça, R. S., N., M. R., Martins, R. F. T., Verdi, F. L., and Marcondes, C. (2018). Introdução à linguagem p4 - teoria e prática. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) - Minicursos*.

Kim, H. and Feamster, N. (2013). Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119.

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Neu, C. V., Tatsch, C. G., Lunardi, R. C., Michelin, R. A., Orozco, A. M. S., and Zorzo, A. F. (2018). Lightweight IPS for port scan in OpenFlow SDN networks. In *IEEE/IFIP Network Operations and Management Symposium Workshops*, pages 1–6.

NPL (2019). Npl specifications. Disponível em: ≪https://nplang.org/npl/specifications/≫. Acesso em 13/09/2021.

Song, H. (2013). Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, page 127–132, New York, NY, USA. Association for Computing Machinery.

Soni, H., Rifai, M., Kumar, P., Doenges, R., and Foster, N. (2020). Composing dataplane programs with $\mu$p4. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 329–343.

Tavares, T. N., da Rocha, A. M., da Cruz Marcuzzo, L., da Silva, N. C. B., Garcia, V. F., and dos Santos, C. R. P. (2017). Estudo comparativo entre depuradores para SDN. In *15a ESCOLA REGIONAL DE REDES DE COMPUTADORES (ERRC)*, pages 73–79.