# Posicionamento e Encadeamento em NFV: Como Lidar Quando o Plano de Dados é Programável e Multi-Tenant?

**Aline Fraga da Silva[1], Ivan Peter Lamb[1], Pedro Arthur Duarte[1], José Rodrigo Azambuja[1], Roben Lunardi[2], Luciano Paschoal Gaspary[1], Weverton Cordeiro[1]**

[1]Instituto de Informática – PPGC – Universidade Federal do Rio Grande do Sul
Porto Alegre – RS – Brazil

[2]Instituto Federal do Rio Grande do Sul – Campus Restinga
Porto Alegre – RS – Brazil

***Abstract.*** *Network Function Virtualization (NFV) has become key for scalable provisioning of in-network services, and network function placement and chaining (VNFPC) plays a critical role in this context. Despite the intense research activity in VNFPC, existing solutions do not consider that the data plane can be programmable and multi-tenant. On the one hand, some custom functions may require mechanisms like in-band telemetry or home-brewed protocols in programmable switches along the network path. On the other hand, the deployment of switches to meet specific network function requirements may affect other network flows. In this paper, we discuss the novel layer of complexity that data plane programmability introduces to VNFPC, and we propose a first solution to this problem, which leverages multi-tenancy in programmable switches to schedule virtual switches to process network function flows.*

***Resumo.*** *A Virtualização de Funções de Rede (NFV) tornou-se fundamental para o provisionamento escalável de serviços de redes, e o posicionamento e encadeamento de funções de rede (VNFPC) assume um papel crítico nesse contexto. Apesar da intensa atividade de pesquisa em VNFPC, as soluções existentes não consideram que o plano de dados pode ser programável e multi-tenant. Por um lado, algumas funções de rede customizadas podem necessitar de mecanismos como in-band telemetry ou de protocolos também customizados implementados em switches programáveis ao longo do caminho na rede. Por outro lado, a implantação de switches para atender às necessidades de funções de rede específicas pode afetar também os demais fluxos na rede. Neste artigo, discutimos a nova camada de complexidade que a programabilidade do plano de dados adiciona ao VNFPC, e propomos uma primeira solução a esse problema, a qual explora multi-tenancy em switches programáveis para escalonar switches virtuais que processam os fluxos de funções de rede.*

## 1. Introdução

A Virtualização de Funções de Rede (NFV) ganhou força significativa nos últimos anos com a consolidação de novas tecnologias de rede e paradigmas como 5G e *edge computing*. Neste contexto, NFV tem sido uma abordagem eficaz para virtualizar funções de rede – migrando-as de soluções de *hardware* especializadas e fechadas para soluções centradas em *software* que executam em *hardware* de propósito geral. Assim, provê-se maior flexibilidade para o processo de implantação, operação e gerenciamento de serviços de

rede [Luizelli et al. 2017a]. Um ponto chave nesse processo é o posicionamento e encadeamento de Funções de Rede Virtuais (VNFs) em pontos de presença de rede (N-PoPs). Tal deve ser feito de acordo com requisições chamadas de *Serviço de Rede* (NS) na terminologia NFV [ETSI 2023], também frequentemente referidas na literatura como *Service Function Chain* (SFC), de acordo com a terminologia do IETF [Halpern and Pignataro 2015]. Em resumo, cada NS especifica, entre outros, a) quais fluxos de rede devem ser processados, b) quais funções de rede (físicas e/ou virtuais) processarão tais fluxos, c) quais os pontos de origem e de destino dos fluxos, e d) como os fluxos serão processados pelas funções (em qual ordem, sob quais restrições de vazão e latência, etc.).

Em razão do papel crítico e da complexidade do problema de Posicionamento e Encadeamento de Funções de Rede (VFNPC) no provisionamento eficiente e escalável de NSes, a área tem recebido significativa atenção de pesquisa [Laghrissi and Taleb 2018, Sun et al. 2022]. No entanto, as soluções propostas não consideram que o plano de dados pode ser *programável* [Bosshart et al. 2014] e *multi-tenant* – ou seja, formado por *switches* programáveis os quais, por sua vez, podem hospedar instâncias virtuais de *switches* programáveis [Han et al. 2020]. A programabilidade do plano de dados pode trazer duas implicações para o VNFPC. Por um lado, algumas VNFs podem depender de que o caminho na rede, por onde seus fluxos irão transitar, ofereça suporte a protocolos de rede customizados ou inovadores (por ex., L3.5 NDN [Ribeiro et al. 2024]), ou mesmo ofereça funcionalidades específicas como *In-band Telemetry* (INT) [Kim et al. 2015], *Heavy-hitter Detection* [Sivaraman et al. 2017] ou *Key-value Store* [Jin et al. 2017]. Por outro lado, ao modificar o comportamento de um *switch* programável para atender às necessidades de uma NS que será implantada (incluindo as necessidades das VNFs que constituem a NS), os demais fluxos da rede que transitam pelo mesmo *switch* podem ser afetados.

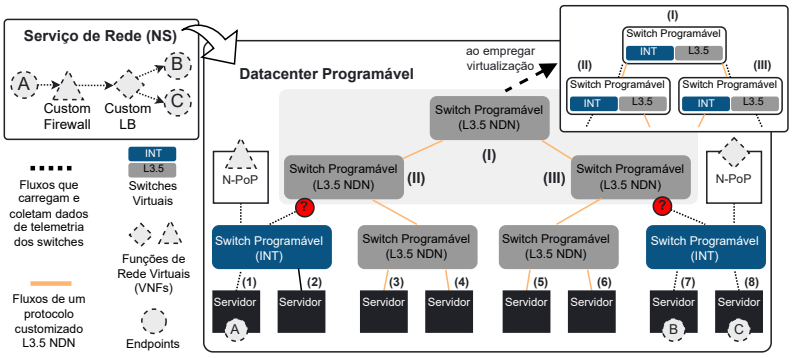


**Figura 1. Exemplo de um problema de VNFPC considerando o processamento customizado de fluxos de dados em um plano de dados programável.**

Para melhor ilustrar o problema acima, considere o exemplo da Figura 1. Ela ilustra uma topologia *fat-tree* simplificada de *datacenter* programável, com *switches*, servidores e pontos de presença (N-PoPs) para execução de VNFs. Os servidores 3 a 6 trocam fluxos entre si usando um protocolo de rede customizado chamado L3.5 NDN [Ribeiro et al. 2024]. Para permitir a comunicação entre esses servidores, alguns dos *switches* do *datacenter* foram inicializados com o programa que implementa o L3.5 NDN. A figura também ilustra uma requisição de Serviço de Rede (NS) a ser implantada no *datacenter* programável. A NS é composta por duas VNFs customizadas (*firewall* e balanceador de carga), três *endpoints* A, B, e C, e o grafo de encaminhamento de fluxos entre *endpoints* e VNFs. Para operarem, as VNFs customizadas requerem a coleta de dados de

telemetria *in-band* de cada *switch* por onde passarem os fluxos relacionados à NS.

O problema que surge ao implantar a NS no *datacenter* programável, conforme ilustrado na figura, considerando os *endpoints* A, B e C associados aos servidores 1, 7 e 8, respectivamente, é que alguns dos *switches* que farão parte do caminho dos fluxos da NS não suportam telemetria *in-band* (INT). No entanto, carregar nesses *switches* os programas que implementam INT fará com que os demais fluxos da rede, os quais trafegam entre os servidores 3, 4, 5 e 6 e dependem do protocolo L3.5 NDN, sejam afetados. Assim, torna-se evidente uma limitação das soluções de VNFPC existentes na literatura [Laghrissi and Taleb 2018, Sun et al. 2022]: ao definir o posicionamento de VNFs e o encadeamento de fluxos entre endpoints de uma NS passando pelas VNFs, as soluções de VNFPC não levam em consideração se os caminhos ao longo de planos de dados programáveis por onde passarão os fluxos possuem as capacidades de interpretar e processar os pacotes desses fluxos. Este não é um problema simples de resolver pois, caso o VNFPC também leve em consideração a troca dos programas dos switches programáveis ao longo do caminho, os demais fluxos na rede não relacionados à NFV serão afetados e sofrerão disrupção, afetando usuários de aplicações diversas.

Para preencher essa lacuna, neste artigo propomos uma solução de escalonamento e implantação de *switches* virtuais programáveis [Han et al. 2020] que atendam aos requisitos de processamento de pacotes de diferentes serviços sem afetar os demais fluxos da rede. Em resumo, recebemos um conjunto de requisições de *switches* exigidos por NSes posicionados e encadeados, bem como de *switches* que processam outros fluxos da rede, e determinamos quais instâncias virtuais devem ser implantadas para satisfazer as especificações de processamento de cada serviço sem afetar o processamento de outros fluxos da rede. Em seguida, agendamos a implantação desses *switches* programáveis virtuais através do escalonamento baseado nos recursos disponíveis da rede. A solução combina o Problema de Empacotamento *Two-Dimensional Level Strip Packing* (2LSP), a meta-heurística *Best-Fit Decreasing Height* (BFDH) e, para virtualização do *switch* físico programável, utiliza uma arquitetura *multi-tenant* composicional apresentada em trabalho anterior. Em resumo, o presente artigo apresenta duas contribuições para o estado da arte:

- A definição de uma nova perspectiva de gerenciamento de ambientes baseados em NFV, considerando *switches* programáveis virtuais para processar fluxos de NSes com diferentes especificações sem afetar o processamento dos demais fluxos da rede, aspecto não abordado anteriormente na literatura até onde sabemos;
- Como nossa primeira iteração sobre este novo problema, apresentamos uma solução para receber e agendar requisições de *switches* virtuais e implantá-las em dispositivos físicos. Uma avaliação de nossa solução forneceu evidências de que a mesma pode agendar e compor *switches* virtuais com uma baixa sobrecarga em termos de latência de processamento, reduzindo o *makespan* (tempo total de execução de todas requisições) em uma média de 55%.

O restante deste artigo é organizado como segue. Na Seção 2, fornecemos um histórico dos conceitos principais que formam a base da nossa pesquisa. A Seção 3 analisa os trabalhos relacionados. Na Seção 5, apresentamos 2LSP, a meta-heurística BFDH e nossa modelagem para o problema de escalonamento. Em seguida, apresentamos a implementação da nossa solução na Seção 4. Na Seção 6, discutimos os resultados obtidos de uma avaliação experimental. Por fim, fechamos o artigo na Seção 7 com uma discussão sobre as limitações do trabalho, considerações finais e direções para pesquisas futuras.

## 2. Fundamentação Teórica

Descrevemos a seguir os conceitos principais que formam a base da nossa pesquisa, dentre eles Virtualização de Funções de Rede (NFV) (2.1), Rede Definida por Software (SDN) (2.2), Planos de Dados Programáveis (PDP) e Virtualização de PDP (2.3) e finalmente, Problemas de Empacotamento Bidimensional (2.4). Por conveniência, o leitor familiarizado com qualquer um dos conceitos pode seguir para subseção correspondente.

### 2.1. Virtualização de Funcões de Rede (NFV)

NFV surgiu como uma proposta da indústria de telecomunicações [Schardong et al. 2021] para mudar funções de rede típicas de *hardware* dedicado e especializado, que frequentemente contribui para uma fração significativa de capital e despesas operacionais em provedores de nuvem e *datacenters* [Sherry et al. 2012], para soluções centradas em *software* executadas em *hardware* de propósito geral, usando a virtualização como tecnologia facilitadora. Assim, a NFV permite a implantação flexível e escalável de funções de rede virtual por demanda, reduz o tempo de lançamento no mercado de novas soluções de rede e promove o fornecimento de serviços inovadores e personalizados. A estrutura da NFV, conforme proposta pelo Instituto Europeu de Padrões de Telecomunicações [ETSI 2023, ETSI 2024], inclui diversos componentes. A seguir são destacados os principais componentes, incluindo suas respectivas definições conforme apresentadas nos documentos oficiais do ETSI [ETSI 2023, ETSI 2024]:

- **Função de Rede Virtual (VNF).** Uma implementação de função de rede centrada em *software* e que pode ser executada sobre uma infraestrutura de virtualização de função de rede, usando hardware de prateleira (ex., servidores);
- **Serviço de Rede (NS).** Um serviço composto de uma ou mais funções de rede. Ele também inclui uma topologia do serviço, com pontos de extremidade de rede; um conjunto de *links* virtuais conectando os componentes do serviço; um grafo de encaminhamento de fluxos; e os requisitos, considerações e restrições de implantação, requisitos e objetivos de nível de serviço, etc.
- **Ponto de Presença de Rede (N-PoP).** O local na infraestrutura em que as funções de rede são implantadas e executadas como parte de um NS.

Vários aspectos são considerados no processo de implantação de um serviço de rede (NS), incluindo (i) onde colocar VNFs entre N-PoPs candidatos e (ii) como encadear essas funções de rede entre N-PoPs e pontos de extremidade com *links* virtuais de acordo com os grafos de encaminhamento. Esses dois aspectos, conhecidos por Posicionamento e Encadeamento de Funções de Rede Virtuais (VNFPC) [Moens and De Turck 2014, Luizelli et al. 2015, Cohen et al. 2015, Luizelli et al. 2017b], atraíram atenção significativa da pesquisa, dada a natureza desafiadora do problema, que provou ser NP-completo [Luizelli et al. 2017b].

### 2.2. Redes Definidas por Software (SDN)

Redes Definidas por Software (SDN) é um paradigma que ajudou a quebrar a ossificação da rede [Turner and Taylor 2005, McKeown et al. 2008] e promoveu uma revolução na maneira como operamos e gerenciamos redes [Cordeiro et al. 2017]. Em resumo, a SDN remodelou a estrutura de rede ao desacoplar o controle de rede (lógica e algoritmos para lidar com fluxos) do encaminhamento (feito por dispositivos como *switches* e roteadores) [Kreutz et al. 2014]. A SDN, portanto, reorganizou essa estrutura em:

- Um **plano de aplicação**, hospedando aplicativos que encapsulam e materializam a inteligência global da rede (roteamento, árvore de abrangência, engenharia de tráfego, distribuição de rótulos, etc.);
- Um **plano de controle**, uma entidade logicamente centralizada que faz a ponte entre os planos de aplicativo e dados. Ele faz isso fornecendo uma interface que as aplicações no plano de aplicativo usam para operações e gerenciamento de rede global (recebendo status e estatísticas de rede e enviando regras de configuração) e uma interface para acessar funções fornecidas por dispositivos de rede;
- Um **plano de dados**, hospedando dispositivos de rede que agora realizam encaminhamento de pacotes entre *links*, com base em regras de encaminhamento definidas pelo plano de controle.

Ao habilitar a reconfiguração flexível de encaminhamento de fluxo entre *switches* e roteadores, a SDN se tornou uma aliada natural e conveniente para a NFV, possibilitando (re)configurar os *links* virtuais entre funções de rede, *endpoints*, etc., cumprindo assim os grafos de encaminhamento especificados em solicitações de NSes.

### 2.3. Planos de Dados Programáveis (PDPs) e Virtualização de PDPs

Paralelamente à SDN, os Planos de Dados Programáveis (PDPs) trouxeram uma mudança significativa no relacionamento entre fornecedores de rede e operadores/gerentes. Os PDPs permitem que os gerentes e operadores de rede redefinam livremente como os dispositivos de encaminhamento analisam e processam fluxos de pacotes e permitem a inovação de rede diretamente no plano de dados. Em resumo, os PDPs trouxeram ao mercado uma nova geração de SmartNICs e dispositivos de encaminhamento. Esses dispositivos são reprogramáveis por meio de Linguagens Específicas de Domínio (DSLs) como P4 [Bosshart et al. 2014] e eBPF [Vieira et al. 2020].

Considerando os PDPs, a comunidade de rede começou a trabalhar em soluções para trazer a virtualização de rede [Anderson et al. 2005] para *switches* programáveis. Isso permite redes virtuais completas para uma variedade de casos de uso, como fatiamento de rede, multilocação em infraestruturas de nuvem, etc. [Bueno et al. 2022]. A virtualização do PDP, portanto, se concentrou em conceber soluções arquitetônicas e centradas em *software* para permitir que um único *switch* físico (*host*) execute vários *switches* programáveis virtuais. Estas soluções focam em (i) composição de código para executar programas de vários *switches* virtuais programáveis em um programa único [Lamb et al. 2024b, Lamb et al. 2024a], (ii) emulação de *switches* virtuais programáveis por meio de um único programa de uso geral [Hancock and Van Der Merwe 2016] e (iii) fatiamento de recursos de *hardware* para fornecer várias instâncias de *switches* virtuais programáveis em cima de um único dispositivo físico [Saquetti et al. 2020].

### 2.4. Problemas de Empacotamento Bidimensional

Neste artigo, o problema de escalonamento de *switches* programáveis virtuais com base nos recursos disponíveis da rede é modelado como um Problema de Empacotamento Bidimensional (2BP), problema de minimização que envolve o empacotamento de um dado conjunto de retângulos pequenos de altura e largura variadas (*items*), em um número mínimo de retângulos grandes de altura e largura idênticas (*bins*), sem sobreposição (cuja modelagem será detalhada em seção posterior). Este problema tem inúmeras aplicações no mundo real, dentre elas indústria e redes de computadores [Pietrobuoni 2015]. Em

resumo, uma instância 2BP recebe um conjunto de n *items* retangulares, cada um com uma largura $w$ e uma altura $h$ associadas. Esses *items* são empacotados em um número finito ilimitado de *bins* retangulares idênticas de largura $W$ e altura $H$. O objetivo é alocar todos os *items* no menor número de *bins* possível, sem sobreposições.

Dependendo da aplicação, 2BP pode ter diferentes requisitos práticos, como orientação, níveis e cortes de guilhotina, o que resulta em muitas variantes do problema. Existem variantes do 2BP em que, ao invés de diversas *bins* idênticas, é utilizada uma única *bin* que cresce em uma ou duas dimensões à medida que os *items* são empacotados. Dentre elas está o *2D Level Strip Packing Problem* (2LSP) [Bezerra et al. 2020], que analogamente recebe um conjunto de *items* com largura $w$ e uma altura $h$ associadas, para serem empacotados nesta única *bin*, chamada *strip*, de largura $W$ fixa e altura $H$ variável. 2LSP empacota todos os *items*, separando-os por níveis, em uma *strip* de menor altura possível. Cada nível possui uma altura $H'$, definida pela altura $h$ do *item* mais alto.

## 3. Trabalhos Relacionados

Houve uma investigação substancial na literatura de NFV e de soluções VNFPC, considerando funções de rede virtuais executadas em servidores de *hardware* de propósito geral. No entanto, até onde sabemos, a literatura não leva em consideração a implantação de *switches* programáveis virtuais ao longo do caminho da rede SDN para processamento de pacotes específico feito por *switches* programáveis exigido pelo NS, o que permitiria a operação destas NSes sem afetar o processamento de outros fluxos da rede.

Trabalhos anteriores se concentram na topologia de rede e como ela impacta o VNFPC, uma vez que, dependendo da topologia e dos recursos da rede, certos algoritmos de VNFPC têm melhor desempenho. O trabalho anterior [Hsieh et al. 2016] apresenta o posicionamento e o encadeamento de VNFs em uma topologia em forma de árvore, modelando a solução como um problema de empacotamento multicamadas. Ele minimiza o número de servidores de uso e implanta VNFs na camada inferior para reduzir o custo da rede, usando como base as restrições dos NSes e propondo dois algoritmos gananciosos. O trabalho em [Li and Yang 2018] apresenta o problema de VNFPC e o modelo de programação de antenas em redes de *datacenter* híbridos. Ele representa o problema utilizando Mixed-Integer Programming (MIP) e fornece um algoritmo heurístico em cenários *offline* e *online*. O trabalho em [Dietrich et al. 2017] decompõe o VNFPC em duas partes: (i) particionamento de *NF-graph* realizado por um *coordinator* centralizado e (ii) mapeamento de *NF-subgraph* em redes de *datacenter*. Ele também apresenta formulações de programação linear para encontrar soluções quase ótimas para ambas as partes. Este trabalho propõe um orquestrador de incorporação de serviços de rede chamado Nestor.

Outros trabalhos consideram tecnologias relacionadas a infraestrutura física, como redes ópticas. [Xia et al. 2015] modelou VNFPC em Binary Integer Programming (BIP) e propôs uma heurística para resolvê-lo. O objetivo é minimizar conversões entre *links* ópticos e elétricos. Os trabalhos em [Miyamura and Misawa 2023] e [Wang et al. 2023] também modelaram VNFPC para considerar *links* ópticos em seus algoritmos.

## 4. Arquitetura para Escalonamento de Switches Programáveis Virtuais

Em nosso primeiro esforço para abordar o problema de VNFPC considerando switches programáveis virtuais, neste artigo propomos uma arquitetura para escalonamento de

*switches* programáveis virtuais com a suposição simplificadora de que o VNFPC já foi resolvido para um determinado conjunto de NSes, que devem ser implantados. Na Seção 5 apresentamos um modelo preliminar de programação linear inteira para o escalonamento de switches virtuais, enquanto que na Seção 7 discutimos as limitações da solução e direções prospectivas para seguimento de pesquisas na área.

Nosso objetivo é receber requisições de *switches* programáveis com as funcionalidades requisitadas pelos serviços, bem como dos *switches* e as funcionalidades necessários para processar os demais fluxos da rede, e determinar *slots* de tempo em que as requisições serão executadas simultaneamente. Adicionalmente, implantar esses *switches* (no *slot* de tempo no qual foram escalonados), por meio da virtualização do PDP. Focamos nas técnicas necessárias para habilitar esse cenário, permitindo que NSes com requisições específicas de processamento sejam implantados e compartilhem os recursos de um único PDP físico. Fluxos de NSes processados por plano de dados não programáveis ou que não compartilham PDP com outros fluxos independentes não fazem parte do escopo deste trabalho. Um trabalho extenso na literatura cobre esses casos [Luizelli et al. 2015].

Na implementação da nossa solução, recebemos todas as descrições necessárias dos *switches* programáveis, por exemplo, códigos P4, requisitos de recursos e tempo de atividade do *switch* virtual. Transformamos as requisições nas entradas do problema de escalonamento, conforme descrito na Seção 5. Após obter o cronograma, empregamos uma estratégia de virtualização composicional para executar simultaneamente os *switches* programáveis escalonados no dispositivo físico. Para materializar a implantação de *switches* virtuais, adotamos uma arquitetura de virtualização *multi-tenant* proposta em trabalho anterior. Neste artigo, não abordaremos com profundidade a arquitetura, extensivamente detalhada em [Lamb et al. 2024b, Lamb et al. 2024a].

A solução pode ser dividida em duas fases: **planejamento** (Subseção 4.1) e **execução** (Subseção 4.2), conforme mostrado na Figura 2. Esta figura também apresenta uma visão completa da nossa arquitetura modular, composta por três módulos: *Scheduler Module*, *Virtualization Composition Module* e *Virtualization Runtime Module*. Esses módulos podem atuar em uma ou mais fases, conforme detalhado a seguir. A solução foi implementada em C++, utilizando o *framework* de comunicação *open-source* gRPC. Os módulos ilustrados na figura têm as seguintes funções:

- **Scheduler Module:** Responsável por receber as requisições de *switches* programáveis e agendá-las usando o algoritmo de escalonamento adequado. Ele também envia os programas P4 dos *switches* escalonados em um mesmo *slot* para Virtualization Composition Module e envia o programa composto gerado para o PDP do dispositivo físico.
- **Virtualization Composition Module:** Responsável por compor diferentes programas P4, cada um representando um *switch* virtual. Ele gera um novo programa P4 que implementa a lógica de todos os programas fornecidos isoladamente.
- **Virtualization Runtime Module:** Fornece uma camada de abstração, via servidor P4Runtime, permitindo aos operadores programar e gerenciar seus *switches* virtuais de forma independente, como se estivessem lidando com *switches* físicos.

### 4.1. Fase de Planejamento

A fase de planejamento é responsável por agendar as requisições de *switches* virtuais recebidas e preparar a configuração do plano de dados com base nestas requisições. Em

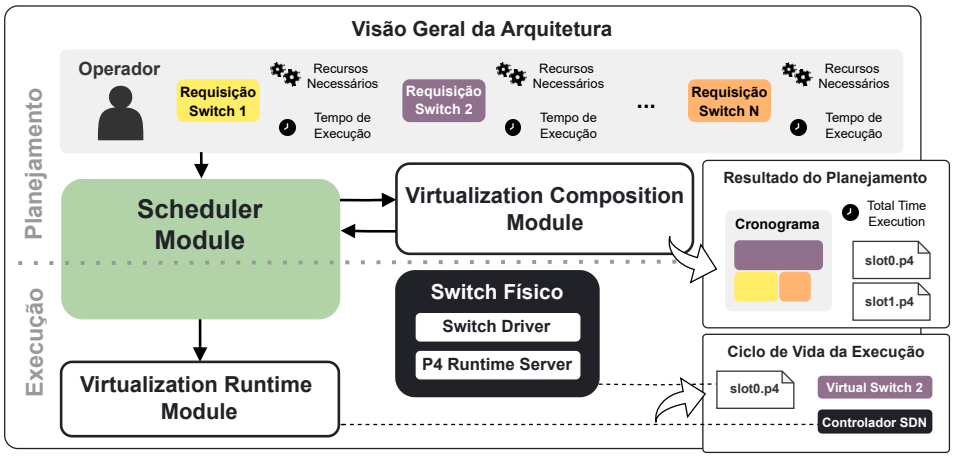


**Figura 2. Visão completa da arquitetura proposta.**

um ambiente integrado, nossa solução pode ser acoplada a outra solução de VNFPC e receber como entrada a saída do algoritmo que considera NSes e a topologia da rede, então escalonar os *switches* necessários para processar de forma personalizada os fluxos das NSes sem afetar o processamento dos demais fluxos da rede.

Toda requisição de *switch* inclui as seguintes informações: (i) nome do dispositivo, (ii) código P4, (iii) número de portas físicas necessárias e (iv) tempo total de execução. Os parâmetros (iii) e (iv) são usados pelo algoritmo de escalonamento, enquanto o parâmetro (ii) é enviado ao módulo de composição. O recebimento das requisições é, até o momento, realizado de forma *offline* devido a uma limitação de reconfigurabilidade dos *switches* virtuais da arquitetura de virtualização utilizada na solução, que será discutida na Subseção 4.2. Adicionalmente, o número de portas físicas do *switch* é o único recurso considerado atualmente para o escalonamento e a virtualização, pois a arquitetura provê o isolamento dos *switches* virtuais diferenciando os pacotes pela porta de ingresso. Outros recursos compartilhados, como a memória, certamente serão considerados futuramente.

## 4.2. Fase de Execução

A fase de execução é responsável por configurar o PDP, implantando os programas compostos associados a cada *slot*, de acordo com o cronograma definido na fase de planejamento. O módulo Scheduler compila e implanta o programa composto. Nesta seção, discutimos as limitações do trabalho relacionadas à execução dos *switches* virtuais.

Como abordado nas seções anteriores, adotamos uma arquitetura de virtualização baseada em composição para executar instâncias virtuais de *switches* programáveis em *switches* físicos. No entanto, há restrições quanto à reconfigurabilidade das instâncias virtuais durante sua execução no PDP: quando um programa composto é implantado no plano de dados, os *switches* virtuais não podem ser reconfigurados sem afetar a execução dos demais *switches*. Assim, até o momento, nossa solução não pode escalonar novos *switches* virtuais em um *slot* durante sua execução, mesmo se houver recursos disponíveis.

Para lidar com essas limitações, nossa implementação não permite remover uma requisição quando ela está escalonada no *slot* em execução ou no *slot* seguinte a ser executado. Apesar das limitações impostas pela arquitetura de virtualização, a composição é realizada diretamente no códigos-fonte (linguagem P4), permitindo obter uma estratégia

*target independent*. Assim, nossa solução cobre diferentes *targets* de forma abrangente, desde *software switches* até dispositivos físicos programáveis de última geração. Como mencionado, nossa solução pode ser integrada em soluções de VNFPC para fornecer os *switches* programáveis necessários para processar de forma personalizada fluxos de diferentes NSes sem afetar o processamento dos demais fluxos da rede.

## 5. Visão Geral e Modelagem do Problema de Escalonamento de Switches

Neste artigo, argumentamos que o escalonamento de *switches* programáveis virtuais é um componente crucial, porém ausente, na realização de VNFPC em ambientes programáveis, devido às especificidades de processamento que podem ser requeridas pelos serviços de rede atuais (que não podem afetar o processamento de outros fluxos). Portanto, a partir da arquitetura apresentada na seção anterior, nesta seção fornecemos uma descrição detalhada da modelagem do problema de escalonamento de *switches* programáveis virtuais como uma instância do 2LSP, problema de empacotamento bidimensional introduzido na seção anterior.

Considerando que um *datacenter* programável é capaz de executar um número significativo de NSes simultaneamente, argumentamos que é possível implantá-las com o uso de estratégias de virtualização do PDP, implantando os *switches* virtuais necessários que realizem o processamento de pacotes requisitado juntamente com os *switches* necessários para processar os demais fluxos da rede. Para que isso seja possível, é preciso (i) considerar os recursos disponíveis e (ii) escalonar os *switches* virtuais necesários. Entendemos a complexidade adicionada ao considerar um ambiente programável para realização de VNFPC. Nossa modelagem inicialmente considera o tempo e os recursos do *switch*. Os recursos considerados serão detalhados na seção seguinte.

Para modelar nosso problema de escalonamento como uma instância do 2LSP, consideramos a largura e altura da *strip* como os recursos do *switch* e o tempo de execução do NS, respectivamente, e as requisições de *switches* como *items*. A altura de cada *item* é o tempo de execução do NS e a largura do *item* é o número de recursos necessários do dispositivo físico. Cada nível da *strip* será um *slot* de tempo, sendo que todas requisições escalonadas em um mesmo nível executarão simultaneamente no *switch* físico.

Para entradas de maior custo computacional, utilizamos uma abordagem conhecida como Best-Fit Decreasing Height (BFDH), heurística clássica usada para resolver problemas de empacotamento como 2LSP. Seus passos envolvem, resumidamente, classificar *items* por altura decrescente e empacotá-los nesta ordem. Para cada *item*, BFDH determina qual nível da *strip*, dentre os níveis existentes, ao empacotá-lo, terá menor largura restante. Se nenhum nível puder acomodar o *item*, um novo nível será criado. Como, mencionado anteriormente, o *item* com maior altura determina a altura do nível.

Para entradas de baixo custo computacional, utilizamos um modelo de Integer Linear Programming (ILP), derivado de [Lodi et al. 2004]. Seguindo a definição do problema, assumimos que a *strip* tem uma largura fixa $W$ e cada *item* $i$ possui uma altura $h_i$ e uma largura $w_i$. Primeiramente, *items* são classificados em altura crescente. Para cada item $i$, a variável binária $y_i$ significa se $i$ inicia um nível (ou seja, é o mais alto do nível). Além disso, para cada par distinto de itens $i$ e $j$, a variável binária $x_{ij}$ indica se $j$ está dentro do nível iniciado por $i$. A função objetivo visa minimizar a altura total enquanto leva em conta os inicializadores de nível. O primeiro conjunto de restrições garante que cada *item* inicialize um nível ou seja colocado em um nível já inicializado por um *item*

mais alto. O segundo conjunto de restrições garante que o processo de empacotamento respeite a largura da *strip*. Consequentemente, o modelo ILP proposto no escopo deste artigo compreende a função objetivo e restrições conforme descrito a seguir:

$$
\begin{aligned}
\textbf{min} \quad & \sum_{i=1}^{n} h_i y_i \\
\textbf{subject to} \quad & \sum_{i=1}^{j-1} x_{ij} + y_j = 1, j = 1, ..., n \\
& \sum_{j=i+1}^{n} w_j x_{ij} \leq (W - w_i) y_i, i = 1, ..., n-1 \\
& y_i \in \{0, 1\}, i = 1, ..., n \\
& x_{ij} \in \{0, 1\}, i = 1, ..., n-1, j = i+1, ..., n
\end{aligned}
$$

## 6. Avaliação

Nesta seção discutimos os resultados da avaliação da nossa solução a partir de três experimentos para avaliar o escalonamento e a composição de *switches* virtuais. Usamos um *switch* de *datacenter* programável Edgecore Wedge100BF-32X (32 portas, chip Intel Tofino) com uma taxa de transferência agregada de 6,4 Tbps para realização dos experimentos. O objetivo dos nossos experimentos é responder as seguintes perguntas de pesquisa: 1) Quanto podemos reduzir o *makespan* da implantação do NSes que têm fluxo processado por *switches* programáveis, considerando a virtualização do PDP? e 2) Qual é a sobrecarga (em termos de latência de processamento) adicionada pela fase de planejamento? Os experimentos foram definidos da seguinte forma:

- **Experimento #1:** 15 requisições de *switches*, cada uma com um programa adaptado dos tutoriais P4. O número de portas necessárias e tempos de execução definidos varia de 1 a 16 e de 2 a 20, respectivamente.
- **Experimento #2:** 5 requisições de *switches*, cada uma com um programa de exemplo do Tofino SDE. O número de portas necessárias e tempos de execução definidos varia de 12 a 20 e de 8 a 25, respectivamente.
- **Experimento #3:** 25 requisições de *switches*, cada uma com programas dos Experimentos 1 e 2. O número de portas necessárias e tempos de execução solicitados variam de 1 a 20 e de 2 a 25, respectivamente.

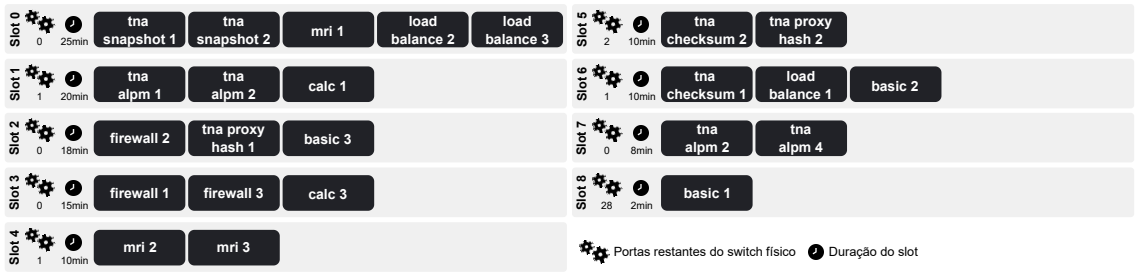


**Figura 3. Resultado do Experimento #3.**

No geral, os experimentos mostraram que é possível realizar o escalonamento e a composição de *switches* virtuais para reduzir o número de dispositivos físicos necessários na topologia de rede. A Tabela 1 mostra o *makespan* do agendamento com e

sem nossa solução, para cada experimento. Observe que o *makespan* foi reduzido em 55%, com desvio padrão de 13,73s. Também avaliamos o desempenho do agendamento e da composição separadamente. A Tabela 2 mostra a sobrecarga em termos de latência adicionada pela fase de planejamento. Os resultados apresentados são uma média de 10 repetições. A média do tempo de escalonamento tem um desvio padrão nos Experimentos #1, #2 e #3 de 24,31us, 23,35us e 25,75us, respectivamente. A média do tempo de composição de slot tem um desvio padrão de 5,32ms no Experimento #1 e 4,98ms no Experimento #2. A Figura 3 apresenta o resultado do Experimento #3, para o qual se obteve uma média de tempo de composição de slot de 2,34ms.

**Tabela 1. *Makespan* com e sem o escalonamento de *switches* virtuais**

| | Sem nossa solução | Com nossa solução |
|---|---|---|
| **Experimento #1** | 158min | 51min |
| **Experimento #2** | 79min | 51min |
| **Experimento #3** | 299min | 118min |

**Tabela 2. Tempo para escalonar e compor os *switches* virtuais**

| | Escalonamento | Composição |
|---|---|---|
| **Experimento #1** | 921us | 728ms |
| **Experimento #2** | 470us | 289ms |
| **Experimento #3** | 2026us | 537ms |

Em resumo, nossa solução foi capaz de agendar e compor corretamente os *switches* virtuais, reduzindo substancialmente o *makespan* de implantação. A sobrecarga resultante dos experimentos mostra que a fase de planejamento tem um bom desempenho para o cenário escolhido. No entanto, não podemos concluir se nossa solução adiciona ou não adiciona sobrecarga significativa em termos de latência de processamento de forma generalizada, pois depende das particularidades de cada cenário.

## 7. Considerações Finais

Neste artigo, argumentamos que o VNFPC deve considerar *switches* programáveis ao longo do caminho da rede SDN, uma vez que NSes podem exigir processamento diferenciado de pacotes realizado por esses dispositivos, que não podem afetar o processamento de outros fluxos não relacionados. Adicionalmente, propomos uma solução de escalonamento e implantação de *switches* virtuais programáveis que pode ser executada em conjunto com soluções de VNFPC para lidar com este problema. Até onde sabemos, a literatura não leva em consideração a implantação desses *switches*, por meio da virtualização do PDP, para processar de maneira independente os diferentes fluxos de serviços da rede.

Nossa solução combina o Problema de Empacotamento Bidimensional Two-Dimensional Level Strip Packing (2LSP), a meta-heurística Best-Fit Decreasing Height (BFDH) e, para virtualização dos *switches* programáveis, utiliza uma arquitetura composicional apresentada em trabalho anterior. A partir de avaliação experimental, fornecemos evidências da viabilidade da modelagem de escalonamento, bem como da capacidade de compor e implantar *switches* virtuais em um *switch* físico, reduzindo o *makespan* (tempo total de execução de todas requisições) em uma média de 55%.

No entanto, até o momento, nossa solução recebe o resultado do VNFPC de algoritmos que não consideraram o problema apresentado neste artigo, o que pode gerar um

resultado subótimo. Com isso em mente, temos como futura direção de pesquisa incorporar a nossa modelagem no processo de VNFPC e, então, virtualizar *switches* programáveis que processam fluxos de NSes e também fluxos independentes na rede. Outras direções de pesquisa incluem i) estender nossa solução para lidar com escalonamento centralizado de *switches* virtuais (para múltiplos *switches* físicos) e ii) lidar com mudanças nas condições de rede, ou seja, receber as requisições de forma *online*.

## Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e Grant #88887.954253/2024-00. Este estudo foi financiado em parte pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) - Grants #444978/2024-0, #314506/2023-3, e #405940/2022-0. O trabalho também recebeu apoio da Fundação de Amparo a Pesquisa de São Paulo (FAPESP) Grant #20/05183-0 (Skynet), #21/00199-8 (SMARTNESS), #23/00673-7 (IDRCIC), e #23/00816-2 (Low End Networks).

## Referências

Anderson, T., Peterson, L., Shenker, S., and Turner, J. (2005). Overcoming the internet impasse through virtualization. *Computer*, 38(4):34–41.

Bezerra, V. M., Leao, A. A., Oliveira, J. F., and Santos, M. O. (2020). Models for the two-dimensional level strip packing problem–a review and a computational evaluation. *Journal of the Operational Research Society*, 71(4):606–627.

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 CCR*, 44(3):87–95.

Bueno, G., Saquetti, M., Rodrigues, P., Lamb, I., Gaspary, L., Luizelli, M. C., Zhani, M. F., Azambuja, J. R., and Cordeiro, W. (2022). Managing virtual programmable switches: Principles, requirements, and design directions. *IEEE Communications Magazine*, 60(2):53–59.

Cohen, R., Lewin-Eytan, L., Naor, J. S., and Raz, D. (2015). Near optimal placement of virtual network functions. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1346–1354. IEEE.

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

Dietrich, D., Abujoda, A., Rizk, A., and Papadimitriou, P. (2017). Multi-provider service chain embedding with nestor. *IEEE Transactions on Network and Service Management*, 14(1):91–105.

ETSI (2023). Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV. `https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV003v1.8.1-GR-Terminology.pdf`.

ETSI (2024). Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Architectural Framework Specification. `https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV006v4.5.1-GS-MANOArchFwk.pdf`.

Halpern, J. and Pignataro, C. (2015). Service function chaining (sfc) architecture. request for comments: 7665. Technical report.

Han, S., Jang, S., Choi, H., Lee, H., and Pack, S. (2020). Virtualization in programmable data plane: A survey and open challenges. *IEEE Open Journal of the Communications Society*, 1:527–534.

Hancock, D. and Van Der Merwe, J. (2016). Hyper4: Using p4 to virtualize the programmable data plane. In *CoNEXT'16*, pages 35–49. ACM.

Hsieh, C.-H., Chang, J.-W., Chen, C., and Lu, S.-H. (2016). Network-aware service function chaining placement in a data center. In *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–6. IEEE.

Jin, X., Li, X., Zhang, H., Soulé, R., Lee, J., Foster, N., Kim, C., and Stoica, I. (2017). Netcache: Balancing key-value stores with fast in-network caching. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 121–136.

Kim, C., Sivaraman, A., Katta, N., Bas, A., Dixit, A., Wobker, L. J., et al. (2015). In-band network telemetry via programmable dataplanes. In *ACM SIGCOMM*, volume 15, pages 1–2.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Laghrissi, A. and Taleb, T. (2018). A survey on the placement of virtual resources and virtual network functions. *IEEE Communications Surveys & Tutorials*, 21(2):1409–1434.

Lamb, I. P., Duarte, P. A. P. R., Luizelli, M. C., Gaspary, L. P., Azambuja, J. R., and da Costa Cordeiro, W. L. (2024a). Multi-tenant programmable switch virtualization leveraging explicit resource sharing. In *20th International Conference on Network and Service Management (CNSM 2024)*, pages 1–8. IFIP/IEEE.

Lamb, I. P., Facen, T., Duarte, P., Azambuja, J. R., and Cordeiro, W. (2024b). Multi-tenant programmable switch virtualization architecture. In *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pages 1–5. IEEE.

Li, Z. and Yang, Y. (2018). Placement of virtual network functions in hybrid data center networks. *IEEE Transactions on Multi-Scale Computing Systems*, 4(4):861–873.

Lodi, A., Martello, S., and Vigo, D. (2004). Models and bounds for two-dimensional level packing problems. *Journal of Combinatorial Optimization*, 8.

Luizelli, M. C., Bays, L. R., Buriol, L. S., Barcellos, M. P., and Gaspary, L. P. (2015). Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 98–106. IEEE.

Luizelli, M. C., da Costa Cordeiro, W. L., Buriol, L. S., and Gaspary, L. P. (2017a). A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. *Computer Communications*, 102:67–77.

Luizelli, M. C., da Costa Cordeiro, W. L., Buriol, L. S., and Gaspary, L. P. (2017b). A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. *Computer Communications*, 102:67–77.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2):69–74.

Miyamura, T. and Misawa, A. (2023). Joint optimization of optical path provisioning and vnf placement in vcdn. *Optical Switching and Networking*, 49:100740.

Moens, H. and De Turck, F. (2014). Vnf-p: A model for efficient placement of virtualized network functions. In *10th international conference on network and service management (CNSM) and workshop*, pages 418–423. IEEE.

Pietrobuoni, E. (2015). Two-dimensional bin packing problem with guillotine restrictions.

Ribeiro, G., Pedrosa, L., Signorello, S., and Ramos, F. M. V. (2024). Internet architecture evolution: Found in translation. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, HotNets '24, page 300–307, New York, NY, USA. Association for Computing Machinery.

Saquetti, M., Bueno, G., Cordeiro, W., and Azambuja, J. R. (2020). P4vbox: Enabling p4-based switch virtualization. *IEEE Communications Letters*, 24(1):146–149.

Schardong, F., Nunes, I., and Schaeffer-Filho, A. (2021). Nfv resource allocation: A systematic review and taxonomy of vnf forwarding graph embedding. *Computer Networks*, 185:107726.

Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., and Sekar, V. (2012). Making middleboxes someone else's problem: Network processing as a cloud service. *ACM SIGCOMM Computer Communication Review*, 42(4):13–24.

Sivaraman, V., Narayana, S., Rottenstreich, O., Muthukrishnan, S., and Rexford, J. (2017). Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*, SOSR '17, page 164–176, New York, NY, USA. Association for Computing Machinery.

Sun, J., Zhang, Y., Liu, F., Wang, H., Xu, X., and Li, Y. (2022). A survey on the placement of virtual network functions. *Journal of Network and Computer Applications*, 202:103361.

Turner, J. S. and Taylor, D. E. (2005). Diversifying the internet. In *GLOBECOM'05. IEEE Global Telecommunications Conference, 2005.*, volume 2, pages 6–pp. IEEE.

Vieira, M. A., Castanho, M. S., Pacífico, R. D., Santos, E. R., Júnior, E. P. C., and Vieira, L. F. (2020). Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications. *ACM Computing Surveys (CSUR)*, 53(1):1–36.

Wang, Y., Nguyen, L., and Hu, Q. (2023). Network function virtualization in elastic optical networks. *Journal of Lightwave Technology*, 41(16):5183–5192.

Xia, M., Shirazipour, M., Zhang, Y., Green, H., and Takacs, A. (2015). Network function placement for nfv chaining in packet/optical datacenters. *Journal of Lightwave Technology*, 33(8):1565–1570.