Garantias de QoS em Redes de Transporte 6G: Roteamento na Origem Orientado a Filas de Serviço

Adailton Saraiva[1], Cristina Klippel Dominicini[1], Ana J. Caetano Martins[1], Everson Scherrer Borges[1], Magnos Martinello[2], Matheus F. Martins[3], Marcos A. M. Vieira[3], Cristiano Bonato Both[4], Pedro Eduardo Camera[4]

[1] Instituto Federal do Espírito Santo (IFES), [2]Universidade Federal do Espírito Santo (UFES), [3]Universidade Federal de Minas Gerais (UFMG), [4]Universidade do Vale do Rio dos Sinos (UNISINOS)

adailton.saraiva@estudante.ifes.edu.br, cristina.dominicini@ifes.edu.br

Abstract. *This paper proposes SPolKA, an architecture for 6G transport network slicing that unifies source routing and quality of service provisioning with a stateless network core. The solution uses polynomial arithmetic to encode the route and the queuing policy into a single label at the network edge. P4-based prototypes using the Mininet emulator with the bmv2 software switch (enhanced with the Weighted Deficit Round Robin (WDRR) algorithm) and Tofino switches confirm traffic isolation, performance metrics, and agile migration.*

Resumo. *Este artigo propõe o SPolKA, uma arquitetura para fatiamento de redes de transporte 6G que unifica o roteamento na fonte e o provisionamento de QoS sem estados no núcleo da rede. A solução usa aritmética polinomial para codificar a rota e a política de filas em um único rótulo na borda da rede. Protótipos em P4 usando o emulador Mininet com software switch bmv2, aprimorado com o algoritmo Weighted Deficit Round Robin (WDRR) e switches Tofino comprovam o isolamento de tráfego, métricas de desempenho e migração ágil.*

1. Introdução

As redes de próxima geração exigem uma infraestrutura flexível e programável capaz de assegurar, de forma concomitante, demandas distintas, desde sistemas críticos até a escalabilidade massiva requerida pela Internet das Coisas. Nesse cenário, o fatiamento de redes, em inglês *Network Slicing* (NS), permite segmentar lógica os recursos físicos em múltiplas redes virtuais isoladas, garantindo que cada serviço atenda a seus requisitos específicos de desempenho [Afolabi et al. 2018].

Para viabilizar essas demandas, mecanismos robustos de Qualidade de Serviço, em inglês *Quality of Service* (QoS), tornam-se necessários para garantir o cumprimento dos Acordos de Nível de Serviço, do inglês *Service Level Agreement* (SLA), específicos de cada fatia na rede de transporte [Cominardi et al. 2018]. Por meio da escolha de rotas otimizadas na engenharia de tráfego (com menor atraso ou maior banda), aliada a políticas de priorização e escalonamento dinâmico de recursos, é possível conciliar demandas antagônicas, como a alta vazão do *enhanced Mobile Broadband* (eMBB) e a latência ultrabaixa do *Ultra-Reliable Low Latency Communications* (URLLC). Desse modo, evita-se a degradação mútua entre as fatias, assegurando o isolamento lógico e a eficiência operacional da infraestrutura compartilhada [Hamdi et al. 2025].

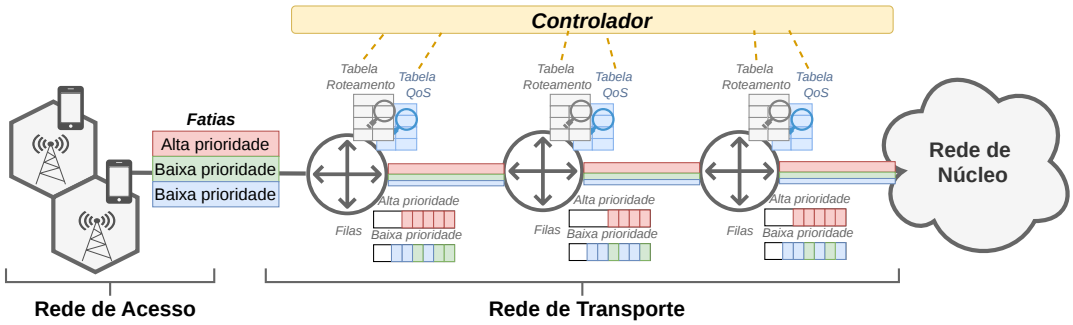


Figura 1. Fatiamento tradicional: tabelas de roteamento e de QoS em cada roteador.

Apesar do papel fundamental dos mecanismos de QoS, a gerência tradicional de redes sofre com problemas de escalabilidade e desempenho. Conforme exemplificado na Figura 1, a cada alteração de política, exige-se a atualização das tabelas de roteamento e QoS em cada roteador intermediário, o que sobrecarrega o plano de controle, eleva o tempo de convergência e compromete a agilidade operacional [Liberato et al. 2018, Froes et al. 2020]. Somado a isso, essas redes operam com um plano de dados rígido, restrito por gerenciadores baseados em funções fixas. Essa inflexibilidade impede a programabilidade de políticas complexas de escalonamento por classe de serviço e limita a modificação dinâmica de filas, prioridades e modelagem de tráfego [Elbediwy et al. 2025]. Consequentemente, tais restrições conjuntas nos planos de controle e de dados inviabilizam o atendimento aos rigorosos requisitos de fatiamento do 6G.

Para resolver esses problemas, propomos a arquitetura *SPolKA*, que estende o roteamento na origem PolKA (*Polynomial Key-based Architecture*) [Dominicini et al. 2020] para habilitar o fatiamento de redes de transporte 6G. Ao explorar o Sistema Numérico de Resíduos, do inglês *Residue Number System* (RNS), e a aritmética polinomial em campos de Galois de ordem 2 (GF(2)), o *SPolKA* permite à engenharia de tráfego selecionar simultaneamente caminhos e políticas granulares de filas para criar fatias de rede via um encaminhamento totalmente sem estado. As principais contribuições são:

- **Extensão do protocolo PolKA para fatiamento de redes com políticas de filas para atender aos requisitos de QoS das redes 6G:** Embora o protocolo PolKA permita a escolha de caminhos das fatias de rede, ele não trata classes de serviço nas filas de encaminhamento. Já o *SPolKA* trata no rótulo do pacote tanto a identificação do caminho quanto a política de filas de cada fatia.
- **Implementação eficiente do fatiamento no plano de dados em P4 (*Programming Protocol-independent Packet Processors*):** O *SPolKA* reusa o módulo CRC (*Cyclic Redundancy Check*) para garantir alto desempenho. A eficácia da solução RNS no fatiamento foi comprovada via emulação (Mininet/bmv2) e *hardware* (Tofino). Adicionalmente, a proposta viabiliza o fatiamento granular ao superar uma limitação nativa do bmv2 por escalonamento dinâmico de filas com pesos.
- **Estruturação do *testbed* de experimentação PORVIR [Nogueira et al. 2025]:** Este trabalho contribuiu ativamente para a criação de um *testbed* nacional equipado com *switches* Tofino programáveis de alto desempenho. Além disso, realizou-se a configuração do protocolo PolKA no *testbed*, operando em conjunto com o *framework* freeRtr, para a orquestração automatizada do plano de controle.

2. Fundamentos e Trabalhos Relacionados

As abordagens tradicionais de roteamento na origem, como *Segment Routing* e *Path Switching*, embora reduzam o estado no núcleo da rede, exigem manipulação de cabeçalhos a cada salto [Filsfils et al. 2018]. Evoluindo esse conceito, abordagens como RDNA [Liberato et al. 2018] introduziram um encaminhamento sem estados baseado em RNS e no Teorema Chinês do Resto, do inglês *Chinese Remainder Theorem* (CRT). Contudo, a aritmética modular inteira dessas soluções impõe severas restrições em *hardware* comercial, inviabilizando o processamento em taxa de linha. O protocolo PolKA [Dominicini et al. 2020] superou essa barreira ao adotar polinômios em GF(2), viabilizando o roteamento sem estado por meio do reuso de módulos CRC em *switches* programáveis. Apesar desse avanço, o PolKA original restringe-se à seleção da porta de saída, sem mecanismos nativos de gerenciamento de filas de prioridade.

Para integrar informações de fatiamento, tais como *Network Slice Selection Assistance Information* (NSSAI), especificamente o S-NSSAI [3GPP 2023], às tecnologias de transporte, é necessário mapear identificadores do plano de controle 5G/6G para mecanismos de encapsulamento e de QoS no plano de dados. A ausência de um mapeamento semântico unificado entre o núcleo 5G/6G e o transporte frequentemente causa a perda de granularidade do S-NSSAI ao ser encapsulado em túneis genéricos [Afolabi et al. 2018]. No *Segment Routing* (SRv6 ou SR-MPLS) [Filsfils et al. 2018], o S-NSSAI é vinculado a *Segment Identifiers* (SIDs) que definem caminhos com restrições de latência e banda [3GPP 2022]. Já em redes VLAN ou VXLAN, o mapeamento utiliza IDs virtuais e prioridades *p-bits* (IEEE 802.1Q). Contudo, a implementação dessas especificações permanece em aberto, carecendo de garantias robustas de isolamento lógico e de desempenho.

No provisionamento de QoS em redes programáveis, destacam-se o Gerenciamento Ativo de Filas [Iqbal and Chen 2023] e escalonadores PIFO (*Push-In First-Out*) e derivados [Elbediwy et al. 2025]. Embora expressivas, essas arquiteturas mantêm estados por fluxo no núcleo, limitando a escalabilidade. Para prover QoS, o ProgLab [Froes et al. 2020] codifica porta e fila num único rótulo RNS. Contudo, a divisão de inteiros longos impede o processamento em taxa de linha no plano de dados e não resolve as restrições de escalonamento do bmv2. Superando essas lacunas, o *SPolKA* alia o controle de filas à aritmética polinomial do PolKA, viabilizando o fatiamento e QoS fim a fim com alto desempenho em *hardware* comercial, com um núcleo sem estado.

3. Arquitetura *SPolKA*

Os cálculos polinomiais baseados em RNS [Shoup 2009, Dominicini et al. 2020] na abordagem *SPolKA* funcionam conforme a descrição a seguir. Considere um pacote a ser roteado por um caminho selecionado, representado por N nós de núcleo e suas respectivas portas de saída (*portID*s) e filas de serviço (*qID*s). Seja S = $s_1(t), s_2(t), \ldots, s_N(t)$ o conjunto de polinômios que representam os *nodeID*s dos nós neste caminho. O conjunto S deve ser composto por polinômios coprimos entre si.

Além disso, seja O = $o_1(t), o_2(t), \ldots, o_N(t)$ o conjunto de N polinômios, onde $o_i(t)$ representa a concatenação dos identificadores da porta de saída (*portID*) e da fila de prioridade (*qID*) para um determinado pacote no nó de núcleo $s_i(t)$, para $i = 1, 2, \ldots N$, satisfazendo a condição de que $degree(s_i) > degree(o_i)$. Com base na definição do caminho representado por S e O, o Controlador calcula o *routeID* usando o CRT polinomial

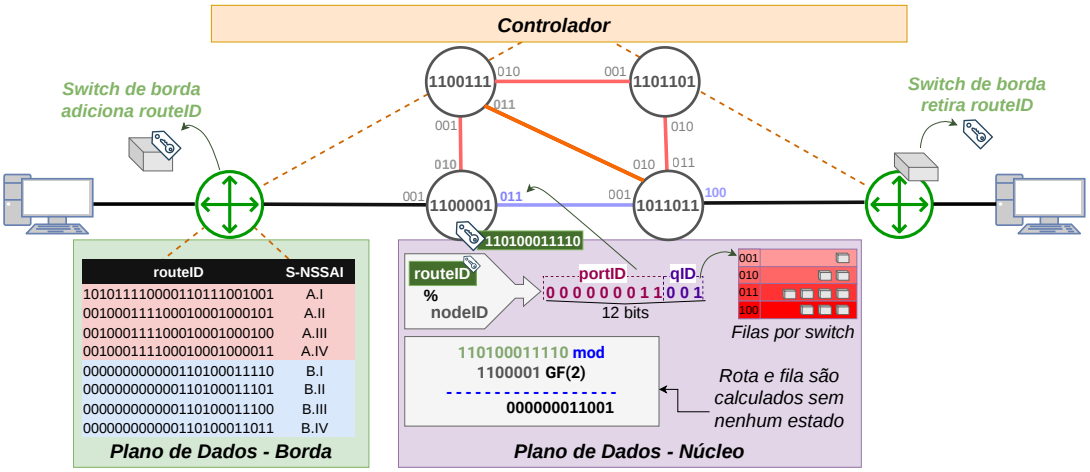


Figura 2. Arquitetura *SPolKA*.

[Shoup 2009, Dominicini et al. 2020] como o polinômio $R(t)$ que satisfaz a seguinte condição: $R(t) \equiv o_i(t) \mod s_i(t), \quad para \quad i = 1, 2, \ldots N$

O controlador associa um rótulo de rota (*routeID*) e uma classe de serviço para selecionar caminhos específicos e políticas de prioridade para diversos fluxos. As propriedades no GF(2) permitem que o *routeID* permaneça inalterado ao longo de todo o caminho. Posteriormente, esse *routeID* é embutido em cada pacote pela borda e cada nó de núcleo calcula a porta de saída e a fila de prioridade como o resto da divisão polinomial (módulo) do *routeID* no pacote pelo seu *nodeID*: $o_i(t) = < R(t) >_{s_i(t)}$.

A Figura 2 ilustra um caminho composto por dois nós de núcleo, que receberam seus *nodeID*s do Controlador durante a fase de configuração da rede: $s_1(t) = t^6 + t^5 + 1 = 1100001$ e $s_2(t) = t^6 + t^4 + t^3 + t + 1 = 1011011$. Considerando o caminho $s_1 \rightarrow s_2$, o *routeID* calculado de acordo com o CRT polinomial é $R(t) = 110100011110$. Para provisionar o fatiamento, o Controlador instala regras nas tabelas de fluxo do nó de ingresso na borda. É nessa etapa que a tabela associa o identificador S-NSSAI (das redes 5G/6G) ao seu respectivo *routeID*, embutindo-o no pacote. No núcleo da rede, o encaminhamento ocorre sem estados: cada nó extrai simultaneamente a porta de saída (*portID*) e a fila de serviço (*qID*) ao calcular o resto da divisão polinomial do *routeID* pelo seu próprio *nodeID*. Por exemplo, ao dividir $R(t) = 110100011110$ por $s_1(t) = 1100001$, obtém-se o resto $o_1(t) = 000000011001$. Esse resultado de 12 bits é então particionado: os 9 bits mais significativos (000000011) determinam o identificador da porta física, enquanto os 3 bits menos significativos (001) definem o identificador da fila.

4. Avaliação

Para validar a eficácia do fatiamento baseado em RNS, desenvolveram-se dois protótipos complementares. O primeiro, em *hardware* Tofino sob o *framework* freeRtr[1], atesta a capacidade do PolKA original em prover QoS apenas pela seleção de caminhos fim a fim. O segundo foca no *SPolKA*, elevando a granularidade ao codificar rota e escalonamento num único rótulo. As operações no plano de dados do *SPolKA* são as mesmas do PolKA original, que já foi validado em *hardware* Tofino [Dominicini et al. 2021]. Para este trabalho,

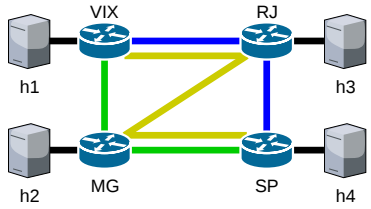
[1] `http://www.freertr.org/`

a solução foi implementar o *SPolKA* no *switch* de *software* BMv2 (emulado no Mininet), pois o *framework* freeRtr ainda não suporta as extensões de cabeçalho do *SPolKA*. Para superar a inanição causada pela prioridade estrita nativa do BMv2, aprimorou-se a arquitetura com o algoritmo *Weighted Deficit Round-Robin* (WDRR), que distribui a banda proporcionalmente aos pesos das fatias [Elbediwy et al. 2025]. Baseando-se em [Elbediwy et al. 2025][2], integrou-se um *Traffic Manager* (TM) customizado em C++ via *software externs*, que transferem os pacotes ao TM no final do *pipeline* de *Ingress*. Isso alinha a emulação ao comportamento de *hardwares* de produção (como o Tofino).

Na borda, a classificação usa o campo DSCP (*Differentiated Services Code Point*) do *DiffServ* para mapear fluxos em rotas e classes de serviço num único *routeID*. Em trabalhos futuros, planeja-se adotar os parâmetros SST (*Slice/Service Type*) e SD (*Slice Differentiator*) do identificador S-NSSAI. No núcleo, a operação de módulo em taxa de linha contorna as restrições matemáticas do P4 reusando o *hardware* de CRC. Conforme detalhado na análise de escalabilidade em [Dominicini et al. 2020], o tamanho do cabeçalho do *routeID* é dado pela multiplicação do número máximo de saltos da topologia por 2 bytes (CRC16 com até 4.080 nós de núcleo). Neste artigo, adotou-se o número máximo de 10 saltos, um *routeID* de 20 bytes, um *portID* de 13 bits e um *qID* de 3 bits (8 filas).

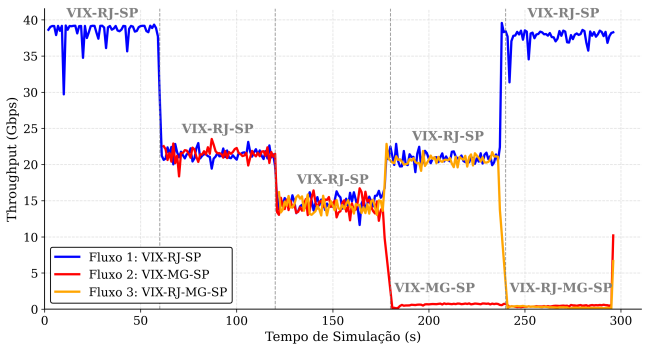
O diferencial do *SPolKA* é a migração ágil de rotas e políticas de filas, alterando apenas um rótulo na borda, sem reconfigurar o núcleo. Os experimentos evidenciam como essa adaptação dinâmica torna o *SPolKA* particularmente eficaz para o fatiamento de redes.

4.1. Protótipo 1: PolKA original: Fatiamento com seleção de caminhos



Caminho	Latência média	Vazão média	Retransmissão (pacotes)
VIX-RJ-SP	13 ms	38,610 Gbps	998
VIX-MG-SP	17 ms	0,829 Gbps	186327
VIX-RJ-MG-SP	26 ms	0,787 Gbps	244680

(a) Métricas por caminho.



(b) Mudança de caminho.

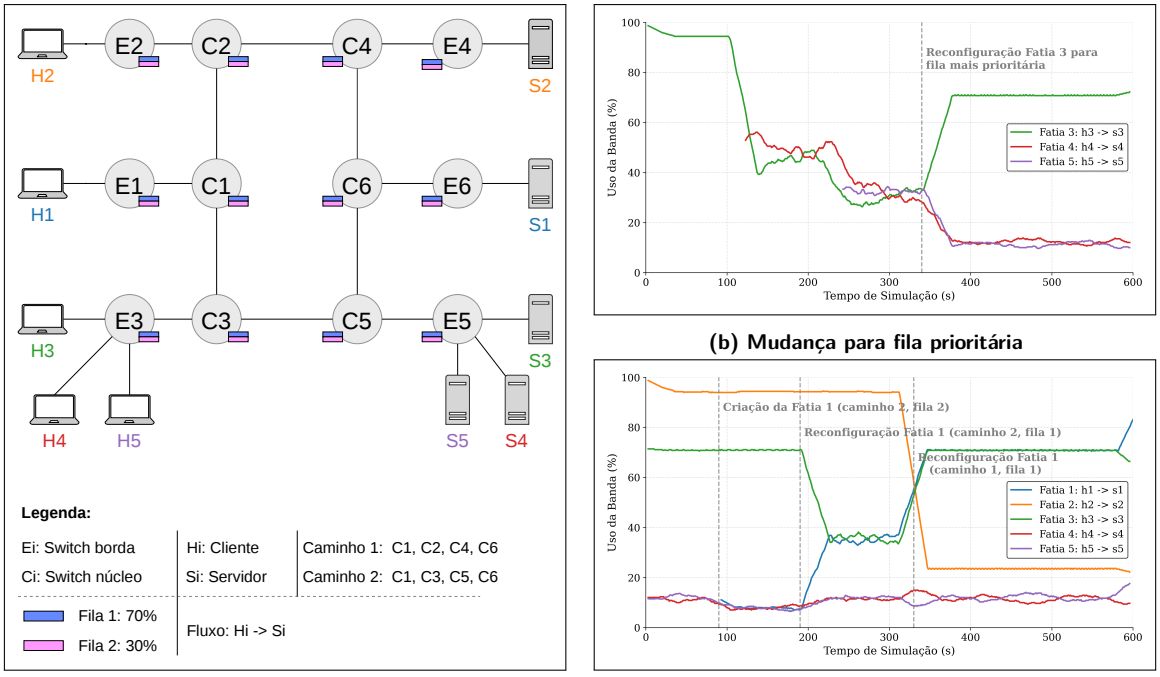
Figura 3. Resultados dos experimentos no testbed nacional Porvir.

A topologia PORVIR (Figura 3a) é composta por quatro nós implementados em *switches* programáveis Tofino. Para avaliar individualmente os caminhos, configuraram-se três túneis PolKA. Inicialmente, caracterizou-se o desempenho de cada caminho da rede em termos de latência, vazão média e retransmissões, conforme a Tabela 3a. As medições de tráfego TCP ocorreram via iperf3 por 60 minutos para cada caminho, com amostragens a cada 30 s. Adicionalmente, o tempo de resposta médio (RTT) foi aferido com a ferramenta ping (amostras de 1 s) por 60 minutos. Os resultados mostram que o caminho VIX–RJ–SP apresenta desempenho significativamente superior, com vazão 46 vezes maior, menor atraso e menos retransmissões.

[2]`https://github.com/Elbediwy/DR-PIFO_hierarchical_TM_SW.git`

Após a caracterização, um experimento com três fluxos TCP (via *iperf3*) no enlace de alta capacidade (VIX–RJ–SP) representou a coexistência de um tráfego de maior prioridade com tráfegos de menor prioridade. Nesse cenário, cada fluxo atua como uma fatia de rede independente, que recebe recursos isolados e é configurada dinamicamente conforme seus requisitos específicos de QoS. A Figura 3b apresenta a vazão coletada ao longo dos eventos, espaçados a cada 60 s. Na fase de competição (0–180 s), a fatia do Fluxo 1 inicia isolada no caminho principal. Aos 60 s e 120 s, as fatias dos Fluxos 2 e 3 são ativadas na mesma rota, compartilhando a banda do enlace. Na fase de migração (180–300 s), evidenciando o isolamento e a adaptação dinâmica, as fatias concorrentes são realocadas para rotas alternativas de menor capacidade: o Fluxo 2 para VIX–MG–SP (aos 180 s) e o Fluxo 3 para VIX–RJ–MG–SP (aos 240 s). Com isso, a fatia do Fluxo 1, mais prioritário, restabelece o uso integral do enlace principal.

4.2. Protótipo 2: *SPolKA*: Fatiamento com seleção de caminhos e políticas de filas



(a) Topologia da rede

(b) Mudança para fila prioritária

(c) Mudança de caminho e fila

Figura 4. Alteração de políticas de QoS para fatias de rede com SPolKA.

O segundo protótipo foi avaliado em dois cenários de acordo com a topologia da Figura 4a: no primeiro, investigou-se exclusivamente o impacto da alteração das políticas de filas, enquanto o segundo analisou a reconfiguração concomitante de rotas e filas. Os enlaces têm capacidades de 10 Mbps com duas filas de prioridade por interface em cada roteador: a *Fila 1*, com reserva de 70% da largura de banda, e a *Fila 2*, com os 30% remanescentes. Os fluxos de dados foram gerados usando *iperf3* com protocolo TCP.

No primeiro cenário (Figura 4b), analisou-se o desempenho de três fatias de rede encaminhadas pela rota $\{C_3, C_5\}$: uma fatia de alta prioridade, representada pelo fluxo $H_3 \rightarrow S_3$, e duas fatias de menor prioridade, representadas pelos fluxos $H_4 \rightarrow S_4$ e $H_5 \rightarrow S_5$. O objetivo foi validar a eficácia do isolamento e da priorização de recursos em condições de saturação de rede. Em $t = 0$ s, o fluxo $H_3 \rightarrow S_3$ foi alocado na *Fila 2*

em todos os saltos. A aplicação do algoritmo WDRR garante que, na ausência de fluxo na fila prioritária, a banda total disponível pode ser compartilhada entre as demais filas. Posteriormente, em $t = 90$ s e $t = 180$ s, os fluxos $H_4 \rightarrow S_4$ e $H_5 \rightarrow S_5$ foram configurados, na mesma fila, resultando no compartilhamento equitativo da banda disponível entre as três fatias. Finalmente, em $t = 270$ s, procedeu-se à reconfiguração do fluxo $H_3 \rightarrow S_3$ para a *Fila 1*, que possui uma reserva estrita de 70% da largura de banda. Esta reconfiguração visou assegurar o isolamento de desempenho para o tráfego prioritário. Os fluxos remanescentes na Fila 2 (origem H_4 e H_5) passaram a competir pelos 30% restantes da capacidade alocada, validando a capacidade do mecanismo de fatiamento em assegurar requisitos de QoS e mitigar interferências entre diferentes classes de serviço.

O segundo cenário (Figura 4c) analisa o impacto da reconfiguração dinâmica concomitante de filas e rotas. Em $t = 0$ s, foi instanciada uma fatia de rede na rota $\{C_2, C_4\}$ referente ao fluxo $H_2 \rightarrow S_2$ e três fatias na rota $\{C_3, C_5\}$ referentes aos fluxos $H_3 \rightarrow S_3$, $H_4 \rightarrow S_4$ e $H_5 \rightarrow S_5$. A fatia com origem H_2 foi alocada na *Fila 2*, ocupando toda a banda disponível, devido à ausência de tráfego concorrente em sua rota. No outro caminho, o fluxo $H_3 \rightarrow S_3$ foi designado à *Fila 1* (prioritária) operando em conjunto com os fluxos $H_4 \rightarrow S_4$ e $H_5 \rightarrow S_5$, alocados na *Fila 2*. Em $t = 60$ s, introduziu-se o fluxo $H_1 \rightarrow S_1$ via $\{C_1, C_3, C_5, C_6\}$ na *Fila 2*. Nesse cenário, o tráfego passou a competir pela reserva de 30% da largura de banda disponível nesta fila com os fluxos H_4 e H_5. Posteriormente, em $t = 180$ s, procedeu-se à reconfiguração do fluxo $H_1 \rightarrow S_1$ para a *Fila 1*, sem alteração de caminho, passando este a compartilhar a reserva de 70% da banda com o fluxo $H_3 \rightarrow S_3$. Finalmente, em $t = 300$ s, realizou-se a migração de rota do fluxo $H_1 \rightarrow S_1$ para o caminho $\{C_1, C_2, C_4, C_6\}$, mantendo-se a configuração na *Fila 1*. Esta reconfiguração de rota mitigou a disputa por recursos críticos: a fatia $H_2 \rightarrow S_2$ passou a ocupar, isoladamente, a reserva da *Fila 2* em seu trajeto, enquanto as fatias prioritárias $H_1 \rightarrow S_1$ e $H_3 \rightarrow S_3$ atingiram a vazão de 70% da banda, operando sem concorrência em suas respectivas instâncias da *Fila 1* em cada caminho.

5. Conclusão

Este artigo apresentou a arquitetura *SPolKA* para o fatiamento de redes e o provisionamento de QoS em infraestruturas 6G. Como um núcleo sem estados, o *SPolKA* estende o roteamento na fonte PolKA ao usar RNS polinomial para embutir, em um único rótulo fixo, o caminho e a política de filas de cada fatia de rede, delegando a complexidade de classificação à borda. A eficácia da solução foi validada por meio de protótipos desenvolvidos em P4. No *switch* de *software* BMv2, implementou-se o algoritmo de escalonamento WDRR para superar as restrições de prioridade estrita nativas e evitar a inanição dos fluxos. Adicionalmente, os experimentos em *hardware* com *switches* Tofino no *testbed* nacional PORVIR demonstraram a alta eficiência do PolKA em cenários realistas de transporte. Os resultados confirmam a agilidade da arquitetura na migração e no isolamento de caminhos, sem sobrecarregar o plano de controle. Dessa forma, o *SPolKA* se destaca como uma tecnologia pioneira que busca garantir a qualidade do transporte em redes 6G. Como trabalhos futuros, planeja-se investigar a sobrecarga da integração do gerenciador de tráfego, além de implementar algoritmos de escalonamento em conjunto com o roteamento RNS no *hardware* Tofino. Por fim, planejamos comparar a solução com trabalhos como Segment Routing ou DiffServ, bem como avaliar a solução com diferentes métricas de QoS em casos de uso das redes 6G, como baixa latência e confiabilidade.

Agradecimentos

Este trabalho foi realizado com apoio financeiro parcial da FAPES (PIBICES, 732/2024, 1003/2025), do CNPq (312058/2023-3) e do MCTIC/CGI.br/FAPESP, (PORVIR-5G) nº 2020/05182-3.

Referências

3GPP (2022). Management and orchestration; concepts, use cases and requirements. Technical Specification TS 28.530, 3rd Generation Partnership Project (3GPP). Rel-17.

3GPP (2023). System architecture for the 5G system (5GS). Technical Specification TS 23.501, 3rd Generation Partnership Project (3GPP). Rel-18.

Afolabi, I., Taleb, T., et al. (2018). Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):2429–2453.

Cominardi, L. et al. (2018). Understanding qos applicability in 5G transport networks. In *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5.

Dominicini, C. et al. (2020). Polka: Polynomial key-based architecture for source routing in network fabrics. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 326–334. IEEE.

Dominicini, C. et al. (2021). Deploying polka source routing in p4 switches. In *2021 International Conference on Optical Network Design and Modeling (ONDM)*. IEEE.

Elbediwy, M. et al. (2025). Enabling rank-based p4 programmable schedulers: Requirements, implementation, and evaluation on bmv2 switches. *IEEE Transactions on Networking*, 33(1):299–310.

Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and Shakir, R. (2018). *Segment Routing Architecture*. IETF.

Froes, W. et al. (2020). Proglab: Programmable labels for QoS provisioning on software defined networks. *Computer Communications*, 161:99–108.

Hamdi, W., Dağdeviren, O., and Bulut, H. (2025). Qos-aware network slicing and resource management for internet of vehicles in 5G networks. *Ad Hoc Networks*, page 103976.

Iqbal, M. S. and Chen, C. (2023). P4-mlfq: A p4 implementation of multi-level feedback queue scheduling using a coarse-grained timer for data center networks. In *2023 IEEE 12th International Conference on Cloud Networking*, pages 120–125. IEEE.

Liberato, A. et al. (2018). Rdna: Residue-defined networking architecture enabling ultra-reliable low-latency datacenters. *IEEE Transactions on Network and Service Management*, 15(4):1473–1487.

Nogueira, J. M. et al. (2025). Porvir-5G: Programmability, orchestration, and virtualization of next-generation networks. *Annals of Telecommunications*.

Shoup, V. (2009). *A computational introduction to number theory and algebra*. Cambridge university press.