# NDN-ADAP: Uma Arquitetura para Encaminhamento Eficiente de Pacotes em Redes de Dados Nomeados

**André L. R. Madureira**[1], **Francisco R. C. Araújo**[1], **Lucas N. B. Prates**[1], **Leobino N. Sampaio**[1]

[1]Programa de Pós-Graduação em Ciência da Computação (PGCOMP)
Departamento de Ciência da Computação – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

{andre.romano, franciscorca, lucas.brito, leobino}@ufba.br

***Abstract.*** *Named Data Networking (NDN) is a new network architecture focused on the content itself, rather than the location of data, like today's IP networks do. NDN networks have advanced features and functions, which makes their data (forwarding) plane more complex, reducing packet switching performance. Thus, we propose a new network architecture based on the Fabric model, which uses the NDN-ADAP protocol proposed in this study. NDN-ADAP can be embedded in programmable data plane hardware, which allows multicast packet sending and link layer path steering. Our results show that NDN-ADAP can be more efficient in packet forwarding compared to IPv4, achieving an 8% reduction in average delay, 21% in network jitter and a gain of 3.6% in packet processing.*

***Resumo.*** *A Rede de Dados Nomeados (do inglês, Named Data Networking – NDN) é uma nova arquitetura de rede voltada para o conteúdo em si, em vez da localização dos dados como ocorre na rede IP atual. Redes NDN possuem recursos e funções avançadas, o que torna seu plano de dados (encaminhamento) mais complexo, reduzindo o desempenho da comutação de pacotes. Dessa forma, propomos uma nova arquitetura de rede baseada no modelo Fabric, que utiliza o protocolo NDN-ADAP, proposto neste estudo. O NDN-ADAP pode ser embarcado em hardwares com plano de dados programáveis, permitindo o envio de pacotes multicast e path steering em nível de camada de enlace. Nossos resultados mostram que o NDN-ADAP consegue ser mais eficiente no encaminhamento de pacotes, quando comparado ao IPv4, obtendo uma redução de 8% no atraso médio, 21% no jitter de rede e um ganho de 3,6% no processamento de pacotes.*

## 1. Introdução

As Redes de Dados Nomeados (do inglês, *Named Data Networking* – NDN) [Zhang et al. 2014] são arquiteturas projetadas para suportar *cache* na camada de rede, entrega *multicast*, apoio à mobilidade do consumidor, controle de tráfego e de fluxo, dentre outros benefícios, através das estruturas PIT (do inglês, *Pending Interest Table*), FIB (do inglês, *Forwarding Information Base*) e CS (do inglês, *Content Store*) [Zhang et al. 2014]. À medida que tais estruturas trazem robustez e flexibilidade à NDN, o encaminhamento do plano de dados torna-se mais complexo, contrapondo-se ao modelo minimalista da

arquitetura IP [Jahanian and Ramakrishnan 2019]. Em redes de núcleo, tais estruturas podem se tornar um fator limitante na obtenção de altas taxas de comutação de pacotes [Moiseenko and Oran 2017, Azgin et al. 2016], além de não apresentar ganhos significativos como nas redes de borda, caracterizadas pela alta mobilidade de nós, deslocamento de produtores de conteúdos e alta demanda por *caches* locais. Mesmo com o avanço e barateio das tecnologias de *hardware*, a implementação dos princípios da NDN em redes de núcleo esbarra nas limitações no tamanho da memória TCAM (do inglês, *Ternary content-addressable memory*) disponível em *switches* modernos [Moiseenko and Oran 2017].

Por causa dessas limitações, algumas propostas apresentadas na literatura sugerem melhorias no encaminhamento de pacotes NDN, de maneira a evitar buscas LNPM (do inglês, *Longest Prefix Match*) realizadas sobre as FIBs [Moiseenko and Oran 2017] e propõem modelos minimalistas para o núcleo de redes ICNs (do inglês, *Information-Centric Networks*). Buscas LNPM são essenciais para encaminhar pacotes de interesse, porém elas também são tarefas computacionalmente dispendiosas [Moiseenko and Oran 2017], pois os nomes dos pacotes NDN possuem tamanho variável, podendo ser consideravelmente longos. Já as propostas de modelos minimalistas [Azgin et al. 2016] consistem de soluções que não avançaram em termos de arquitetura, se restringindo à remoção de estruturas básicas da ICN, visando a obtenção de melhor desempenho com relação ao encaminhamento de pacotes. Além de tais iniciativas, outras propostas recentes buscaram integrar a NDN ao *hardware* dos comutadores das redes, visando obter melhor desempenho de encaminhamento. Em [Signorello et al. 2016, Miguel et al. 2018], os autores propuseram a integração da NDN à linguagem P4 (do inglês, *Programming Protocol-Independent Packet Processors*) [Bosshart et al. 2014]. Dessa forma, os comutadores baseados em *hardware* programável poderiam interpretar os pacotes NDN e possuir as estruturas PIT, FIB e CS. Tais pesquisas, contudo, apresentaram algumas limitações, as quais são da própria linguagem P4 e da arquitetura NDN. Por exemplo, a falta de recursos essenciais, tais como laços *for* e *while*, dificulta o processamento de cabeçalhos dos pacotes NDN, que possuem tamanho variável.

Diante das questões apresentadas, este artigo propõe uma arquitetura NDN baseada no modelo *Fabric* [Casado et al. 2012], que realiza o roteamento pela origem (do inglês, *source routing*) e é implementada em comutadores com plano de dados programável. A proposta mescla as ideias dos trabalhos anteriores ao construir comutadores NDN usando a linguagem P4 e em remover as estruturas NDN do núcleo da rede, porém mantém a pilha NDN completa nas bordas. Para isso, os comutadores do núcleo possuem *hardwares* programados com um novo protocolo de camada de enlace, chamado de NDN-ADAP. O NDN-ADAP é um protocolo que atua na camada de adaptação da pilha NDN [Zhang et al. 2018]. Assim, a mesma visa alcançar um melhor desempenho no encaminhamento de pacotes e oferece suporte à transmissões *multicast* adaptativas em nível de enlace. O mecanismo proposto atua no núcleo da rede e visa mitigar as limitações enfrentadas pelas pesquisas anteriores. Para comprovar a eficiência da proposta, realizamos uma avaliação analítica e testes experimentais no ambiente de emulação Mininet/BMv2 [Lantz et al. 2010]. Os resultados mostram que o NDN-ADAP supera os protocolos legados UDP/IPv4/Ethernet e obtém uma redução de 8% no atraso médio, 21% no *jitter* de rede e um ganho de 3,6% no processamento de pacotes.

O restante do trabalho está organizado da seguinte forma: na Seção 2 são apresen-

tados os trabalhos relacionados. Os elementos da arquitetura e detalhes de implementação são apresentados na Seção 3. Em seguida, os experimentos e resultados são discutidos na Seção 4. Por fim, o trabalho é concluído na Seção 5.

## 2. Trabalhos Relacionados

Os trabalhos de [Signorello et al. 2016] e [Miguel et al. 2018] foram os primeiros a propor a integração das tecnologias P4 e NDN, porém essas pesquisas apresentaram várias limitações. Uma delas foi a tentativa de integrar todos os recursos das redes NDN dentro dos comutadores P4. Uma vez que a linguagem P4 não possui recursos de repetição, tais como *for* ou *while*, presentes na maioria das linguagens de programação, a compreensão e implementação de todos os recursos previstos na arquitetura de redes NDN não é viável. Além disso, os pacotes NDN utilizam um formato de tamanho variável, denominado TLV (do inglês, *Type-Length-Value*) [NFD 2019], o que também inviabiliza a compreensão de tais pacotes através da linguagem P4. Outro ponto importante a ser considerado é o fato dos *hardwares* modernos, utilizados em redes de nível comercial, possuírem limitações no tamanho da memória TCAM disponível [Moiseenko and Oran 2017, Ishimori et al. 2017]. Uma vez que as principais estruturas de dados das redes NDN (i.e., PIT, CS e FIB) demandam grandes espaços de armazenamento, a implementação dessas estruturas requer a utilização de módulos de memória mais lentos. Consequentemente, há uma redução na velocidade de comutação de pacotes da rede ao utilizar tais módulos de memória [Azgin et al. 2016].

Para contornar tais dificuldades, [Azgin et al. 2016] propôs que a utilização da PIT fosse limitada às bordas da rede, removendo-a dos comutadores NDN do núcleo. Dessa forma, o processamento de pacotes NDN se torna mais eficiente. Outra abordagem para o encaminhamento eficiente de pacotes NDN é o *ICN Path Switching*, proposto por [Moiseenko and Oran 2017]. Esta proposta permite o uso de *path discovery* e *path steering*, além de realizar o encaminhamento de pacotes NDN sem depender de pesquisas LNPM na FIB. Além disso, [Moiseenko and Oran 2017] propõe que os consumidores possam realizar *packet steering* e, com isso, seja possível obter melhores diagnósticos, medições de rede e controle de congestionamento (usando algoritmos de controle de congestionamento *multi-path*), além de obter melhor controle sobre o tráfego da rede, através de técnicas de engenharia de tráfego.

Diante do exposto, propomos uma arquitetura de rede que, ao contrário das propostas supracitadas, é capaz de encaminhar pacotes NDN eficientemente no núcleo das redes, mantendo a especificação oficial completa das redes NDN. Nossa proposta está descrita em detalhes na seção seguinte.

## 3. Arquitetura para o Núcleo de Redes de Dados Nomeados

A fim de alcançar os objetivos deste trabalho, projetamos a arquitetura de rede ilustrada na Figura 1. A arquitetura mantém a pilha NDN nas bordas da rede e aplica encaminhamento por segmento (do inglês, *Segment Routing*) no núcleo da rede. Os elementos da arquitetura são detalhados nas seções seguintes.

### 3.1. Modelo da rede e decisões de projeto

Neste estudo propomos uma arquitetura de rede centrada na informação baseada no modelo *Fabric* [Casado et al. 2012] e uma nova técnica de encaminhamento de pacotes ba-

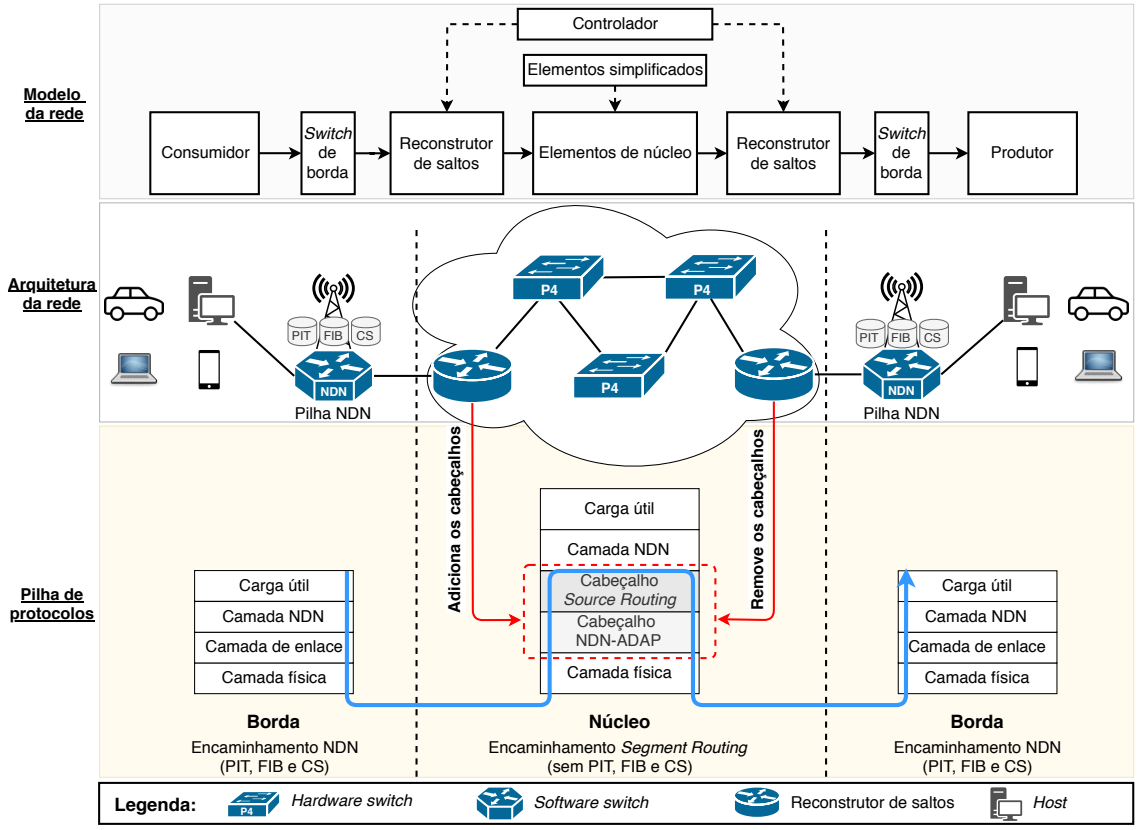



**Figura 1. Arquitetura e pilha de protocolos propostos para o núcleo das Redes de Dados Nomeados.**

seada em segmentos, a qual permite o envio *multicast* de pacotes e *path steering*. Ao utilizar o encaminhamento por segmentos, o plano de dados dos comutadores se torna mais simples e o encaminhamento de pacotes mais eficiente.

Na arquitetura proposta, o núcleo da rede NDN deixa de usar estruturas de dados complexas (e.g., PIT, FIB e CS), presentes na arquitetura original da NDN, devido às seguintes razões: (i) ao contrário da borda, redes de núcleo não apresentam mudanças constantes de topologia, em que a FIB auxilia nós consumidores a encontrar produtores de conteúdo, por meio da difusão de pacotes de interesse pelas interfaces de saída; (ii) a quantidade de conteúdos em um rede de núcleo e a distância para nós consumidores, frequentemente localizados na borda, não justifica a utilização de *caches* em nós do núcleo; (iii) de acordo com [Azgin et al. 2016, Moiseenko and Oran 2017], a utilização das estruturas PIT e FIB da NDN, impacta negativamente no desempenho do encaminhamento de pacotes.

Diante dos pontos apresentados, na arquitetura proposta, o núcleo da rede mantém a mesma simplicidade das redes IP, ficando encarregado somente do encaminhamento de pacotes, como ilustrado na Figura 1. Este encaminhamento é realizado através de uma camada de adaptação [Zhang et al. 2018], programada para encaminhar eficientemente os pacotes NDN através do plano de dados.

### 3.2. Arquitetura da rede e compatibilidade com os princípios da NDN

Neste trabalho propomos uma arquitetura que visa obter maior eficiência no encaminhamento de pacotes mantendo a compatibilidade com os princípios da NDN, conforme ilustrado na Figura 1. Para isso, as estruturas PIT, FIB e CS, previstas pela arquitetura NDN, são removidas dos comutadores do núcleo da rede e mantidas nos comutadores da borda. Dessa forma, os comutadores da borda encaminham pacotes seguindo as especificações da NDN (e.g., roteamento por nomes e uso de *cache*). Portanto, a arquitetura proposta não viola os princípios das redes NDN, uma vez que os comutadores de borda atendem a todas as características previstas na especificação oficial da rede. É importante ressaltar que a implementação prevê a convergência inicial do mecanismo de encaminhamento, que consiste em mapear os nomes de conteúdo para as interfaces de saída.

Nos comutadores do núcleo das redes NDN-ADAP, o encaminhamento é realizado usando a técnica *Segment Routing*. No entanto, apesar das técnicas de encaminhamento *Segment Routing* proporcionarem múltiplos benefícios, tais como o alto desempenho e aplicações de engenharia de tráfego, as mesmas possuem restrições no que tange a escalabilidade. Isto é consequência da utilização de parte do MTU (do inglês, *Maximum Transfer Unit*) do pacote para armazenamento de instruções de encaminhamento. Ao armazenar essas instruções no pacote, o espaço disponível reservado para a transmissão da carga útil e a quantidade de saltos do pacote se tornam limitados.

A fim de mitigar as limitações de espaço das técnicas *Segment Routing*, propomos o uso de comutadores especiais de pacotes, chamados de reconstrutores de instruções *Segment Routing*, conforme ilustrado na Figura 1. Os reconstrutores de instruções devem ser estrategicamente localizados na transição do núcleo da rede com a borda. Além disso, esses reconstrutores também devem estar dispostos a uma certa distância de saltos entre si. Esses comutadores especiais, ao receberem pacotes diretamente da borda, constroem o ID de conteúdo através dos nomes NDN, ao aplicar a função $H$ nos componentes de nome e encapsulam o pacote NDN recebido em um pacote NDN-ADAP. Os demais reconstrutores de instruções da rede apenas usam o ID de conteúdo criado pelo primeiro reconstrutor de instrução para criar as novas rotas *Segment Routing*, dando continuidade ao encaminhamento do pacote. Assim, o ID de conteúdo é utilizado apenas pelos reconstrutores de instruções, a fim de construir a rota *Segment Routing* que o pacote percorrerá. Dessa forma, o nome do conteúdo NDN não é usado para encaminhamento pelos comutadores de núcleo. Quando o pacote alcança o último reconstrutor de instruções, o pacote NDN-ADAP é desencapsulado e a partir de então, o encaminhamento de pacotes ocorrerá utilizando a especificação completa das redes NDN.

Conforme ilustrado na Figura 1, os reconstrutores de instruções são gerenciados pelo controlador da SDN (do inglês, *Software Defined Networking*), que informa detalhes da topologia da rede para os mesmos e mantém o estado da FIB e PIT para permitir o roteamento dos pacotes de interesse e dados. Uma vez que os reconstrutores estejam espalhados, a intervalos regulares de saltos, o encaminhamento *Segment Routing* ocorre de forma escalável. Dessa forma, a arquitetura proposta permite que o encaminhamento dos pacotes NDN-ADAP ocorra prioritariamente através do *Segment Routing*, evitando as consultas LNPM nas tabelas FIB dos comutadores do núcleo da rede. Consequentemente, nossa arquitetura é capaz de obter um desempenho de rede superior à arquitetura original da NDN [Zhang et al. 2014], ainda mantendo as vantagens da NDN na borda da rede.

### 3.3. Protocolo NDN-ADAP

Neste trabalho propomos a implementação de um protocolo de adaptação de camada de enlace (nível 2) para redes NDN. Este protocolo provê suporte à tecnologia NDN, possibilitando encaminhamento, identificação e verificação de pacotes de interesse e dados. Dessa forma, os pacotes NDN podem ser encaminhados de maneira eficiente no núcleo da rede.

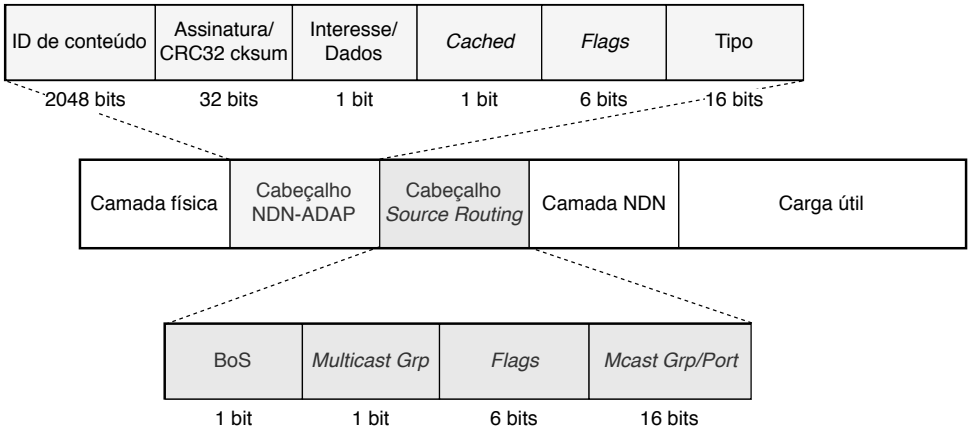



**Figura 2. Estrutura do protocolo NDN-ADAP.**

O NDN-ADAP está localizado na camada de enlace e possui cabeçalho de tamanho fixo, o que fornece um bom desempenho de encaminhamento para o protocolo. A Figura 2 ilustra a proposta, a parte inferior da figura representa o método de encaminhamento que será discutido na Seção 3.4 e a parte superior da figura ilustra o NDN-ADAP que possui os seguintes campos de cabeçalho:

- **ID de conteúdo:** Identificador de conteúdo, semelhante ao nome dos pacotes NDN, obtido após a aplicação de uma função $H$ nos componentes de nome dos pacotes. Dessa forma, o NDN-ADAP consegue acelerar pesquisas LNPM que ocorrem na FIB dos reconstrutores de instruções (comutadores especiais) da rede.
- **Assinatura:** Faz o *checksum* do pacote inteiro, identificando erros na transmissão dos dados e assinando o pacote NDN-ADAP na camada de enlace.
- **Interesse/Dados:** Indica se o pacote NDN é do tipo interesse ou dados.
- ***Cached*:** Indica se o pacote de dados foi encontrado na *cache* de algum nó da rede.
- ***Flags*:** *Flags* reservados para uso de aplicações futuras, como o envio de pacotes NACK para detecção de ataques DoS baseados em nomes de conteúdos inválidos.
- **Tipo:** Indica qual o *payload* do pacote NDN-ADAP. Se o pacote ainda não chegou a seu destino, este campo indica o mecanismo de encaminhamento que será usado para enviar o pacote ao próximo salto.

### 3.4. Encaminhamento baseado em roteamento por segmento

O NDN-ADAP pode utilizar diversos mecanismos de encaminhamento para envio de pacotes na rede. Entre eles é possível citar o Ethernet, IPv4 e *Segment Routing*. Neste trabalho propomos o uso do mecanismo de encaminhamento baseado em um protocolo *Segment Routing* que atua acima da camada de enlace, conforme ilustrado na parte inferior da Figura 2 e cujos campos estão descritos abaixo:

- ***Bottom of Stack* (BoS):** Indica que esta é a última instrução da pilha *Segment Routing*. Se este *flag* estiver atribuído, então o nó atual é o penúltimo salto do pacote antes dele alcançar o próximo reconstrutor de instruções da rede.
- ***Multicast Grp:*** Indica qual instrução *Segment Routing* o comutador deve realizar. Se este *flag* estiver atribuído, o pacote é enviado usando *multicast*. Caso contrário, o encaminhamento *unicast* é realizado.
- ***Flags*:** *Flags* reservados para uso de aplicações futuras, como melhorias na qualidade de serviço, visando minimizar atrasos.
- ***Mcast Grp/Port:*** Utilizado para identificar a porta por onde o pacote é encaminhado, quando o *flag Multicast Grp* não está atribuído. Se este *flag* estiver atribuído, o campo "*Mcast Grp/Port*" representa o grupo *multicast* a ser utilizado para enviar o pacote.

Conforme um pacote NDN-ADAP chega a um comutador do núcleo da rede, o mesmo executa o Algoritmo 1. Primeiro ocorre uma verificação da assinatura/*checksum* do pacote com o objetivo de: (i) identificar falhas na transmissão do pacote para o destinatário (serviço de *checksum*); e (ii) identificar se o pacote sofreu alguma alteração indevida (serviço de assinatura NDN). Em seguida, o comutador irá executar a primeira instrução da pilha *Segment Routing*. Se esta instrução contiver o *flag Multicast Grp* atribuído, então a informação contida no campo *Mcast Grp/Port* indica o grupo *multicast* para transmissão do pacote. Caso contrário, o campo *Mcast Grp/Port* é interpretado como sendo uma porta específica do comutador, para a qual o mesmo enviará o pacote (envio *unicast*). Por fim, o comutador irá verificar se ainda restam instruções *Segment Routing* no pacote NDN-ADAP. Se não houver nenhuma outra instrução, então os dados que vêm logo após o NDN-ADAP são o próprio pacote NDN completo.

**Algoritmo 1:** Encaminhamento NDN-ADAP no núcleo da rede.

```
1  Função NDN−ADAP (pkt) :
2      sig = pkt.assinatura
3      tipo = pkt.tipo
4      size = pkt.src_route.size
5      se VERIFICARPACOTE(sig) e tipo == SourceRouting e size > 0 então
6          se pkt.src_route[0].multicast_grp == 1 então
7              ENVIARPACOTEGRUPOMULTICAST(pkt.src_route[0].mcast_grp_port)
8          senão
9              ENVIARPACOTEPORTAUNICAST(pkt.src_route[0].mcast_grp_port)
10         se pkt.src_route[0].bottom_of_stack == 1 então
11             pkt.tipo = NDN
12         pkt.src_route.pop(0)
13     senão
14         DESCARTARPACOTE(pkt)
```

### 3.5. Implementação do NDN-ADAP

No planejamento inicial, a implementação do NDN-ADAP seria realizada a partir da classe *NDN Faces*, presente na biblioteca oficial utilizada nas redes NDN (ndn-cxx)

[NDN-CXX 2019]. No entanto, essa classe faz uso de chamadas do sistema operacional, criando *sockets* para enviar os pacotes NDN. Dessa forma, precisaríamos construir um módulo dentro do kernel Linux para o NDN-ADAP poder ser utilizado como um *socket* pela biblioteca ndn-cxx. No entanto, realizar tal implementação implicaria em modificar o código fonte do kernel Linux, uma vez que protocolos da camada de enlace não podem ser implementados no kernel como módulos externos. Além disso, esta abordagem não permitiria que o NDN-ADAP pudesse ser utilizado em outros sistemas operacionais, como o Mac OSX, Solaris, BSD ou Windows. Assim, visando tornar o NDN-ADAP uma escolha viável para redes NDN, independente de sistema operacional, escolhemos realizar a implementação do *packet crafter* e *sniffer* do NDN-ADAP utilizando a biblioteca Python Scapy [Biondi et al. 2019].

Com intuito de obter métricas de desempenho mais precisas para avaliação do NDN-ADAP, utilizamos pacotes NDN como *payload* para os nossos experimentos. Para isso, foi necessário implementar os campos dos pacotes NDN dentro do Scapy, seguindo o formato TLV descrito na especificação oficial da NDN [NFD 2019]. Em seguida, utilizamos o Wireshark associado ao *plugin* oficial da NDN para validarmos a implementação do pacote NDN no Scapy. Para isso, enviamos pacotes NDN utilizando a pilha de rede UDP/IP/Ethernet. Uma vez que o Wireshark conseguiu interpretar corretamente os pacotes NDN enviados pelo Scapy, atestamos a corretude da implementação do NDN no Scapy.

Por fim, desenvolvemos o protocolo NDN-ADAP na linguagem P4 e utilizamos o ambiente de emulação Mininet/BMv2 para avaliar o desempenho da proposta. A próxima subseção detalha essas ferramentas.

### 3.5.1. Linguagem P4

O P4 foi utilizado na implementação do NDN-ADAP por se tratar de uma linguagem de alto nível que permite a programação do plano de dados de comutadores baseados em *hardware*. Ao utilizar a linguagem P4 foi possível descrever como os cabeçalhos dos pacotes são interpretados e quais ações podem ser executadas na chegada dos pacotes às interfaces de rede. Ademais, o controlador SDN da rede tem a função de fornecer detalhes acerca da topologia da rede para os reconstrutores de instruções. Com essas informações, os reconstrutores de instruções conseguem construir a rota a ser percorrida pelo pacote através de uma pilha de instruções *Segment Routing*. Dessa forma, o *switch* P4 analisa cada pacote NDN-ADAP recebido e executa uma ação específica, descrita pelas instruções *Segment Routing*.

### 3.5.2. Conversão de nomes hierárquicos em nomes planos

Para acelerar o encaminhamento de pacotes no núcleo da rede, os nomes hierárquicos dos pacotes NDN são convertidos em nomes planos, conforme Figura 3. Na figura, o nome `/ufba/dcc/video1` é separado em prefixo `/ufba/dcc` e sufixo `/video1`, e estes são passados a uma função $H()$, que extrai o código ASCII de cada caractere. Então as saídas de $H$ são concatenadas para obter o nome plano $1696 \cdots 8070$. A conversão ocorre quando os pacotes de interesse da NDN ingressam no primeiro reconstrutor de instrução

da borda, construindo o ID de conteúdo (i.e., nome plano) a partir dos nomes dos pacotes.

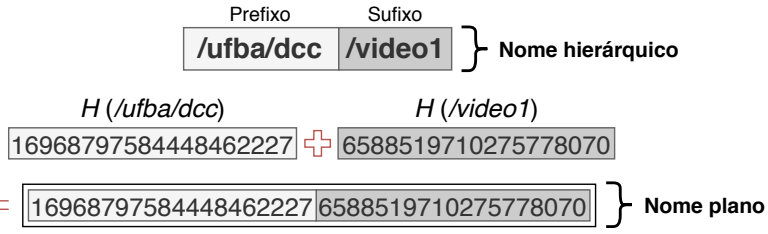



**Figura 3. Conversão de nomes hierárquicos em nomes planos (ID de conteúdo).**

## 4. Avaliação de Desempenho

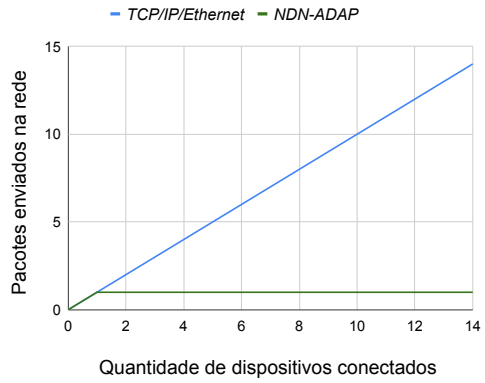
Analisamos o desempenho da arquitetura de rede e do protocolo propostos através de uma avaliação analítica e um estudo experimental baseado em emulação no ambiente Mininet/BMv2. As próximas subseções descrevem em detalhes a metodologia adotada e os resultados obtidos.

### 4.1. Avaliação Analítica

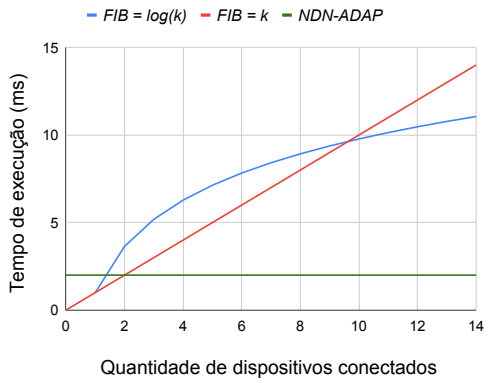
A avaliação analítica teve como objetivo verificar a redução do número de pacotes replicados enviados na rede, o desempenho da comutação de pacotes e a escalabilidade da técnica de encaminhamento *Segment Routing*. Com relação à analise de pacotes replicados enviados pela rede, o NDN-ADAP permite o envio *multicast* de pacotes em camada de enlace. Dessa forma, aplicações NDN podem fazer uso desta funcionalidade, visando melhor desempenho de comutação na rede. Com o NDN-ADAP, ao invés de se enviar $n$ pacotes idênticos para $n'$ destinatários, é possível enviar apenas $1$ pacote com uma instrução *multicast* para que o mesmo seja replicado $n$ vezes e enviado para os destinatários correspondentes. Consequentemente, comutadores NDN-ADAP processarão menos pacotes que comutadores que operam com protocolos sem suporte a *multicast* nativo, como o IPv4 e o Ethernet (ver Figura 4(a)).

#### 4.1.1. Desempenho da comutação de pacotes

Com relação ao desempenho da comutação de pacotes, o NDN-ADAP é capaz de utilizar técnicas de encaminhamento de alto desempenho, como o *Segment Routing*. O *Segment Routing* é uma técnica de encaminhamento de pacotes na qual a origem do pacote especifica um caminho completo ou parcial para o destino [Lee et al. 2015]. O caminho escolhido é codificado nos cabeçalhos dos pacotes, ao contrário de outras técnicas de roteamento nas quais são tomadas decisões de encaminhamento distribuído em cada dispositivo de encaminhamento [Lee et al. 2015, Moiseenko and Oran 2017]. Consequentemente, a complexidade de estratégias de encaminhamento *Segment Routing* em termos de tempo de execução é $O(n)$ (complexidade linear), enquanto que estratégias de encaminhamento baseadas em tabelas FIB possuem complexidade $O(n \log k)$ ou $O(n.k)$, onde $n$ é o número de saltos que o pacote deve realizar para alcançar seu destino, e $k$ é a quantidade de dispositivos conectados diretamente a cada comutador da rede (número de entradas na tabela FIB) (ver Figura 4(b)). Esta diferença na complexidade das estratégias de encaminhamento baseadas em FIB varia de acordo com a implementação do algoritmo de encaminhamento e das estruturas de dados utilizadas por ele.




(a) Envio de pacotes replicados (protocolos *multicast* versus *unicast*).




(b) Protocolo NDN-ADAP versus protocolos baseados em FIB.

**Figura 4. Quantidade de pacotes replicados e complexidade de execução dos protocolos NDN-ADAP e IP.**

### 4.1.2. Número máximo de saltos

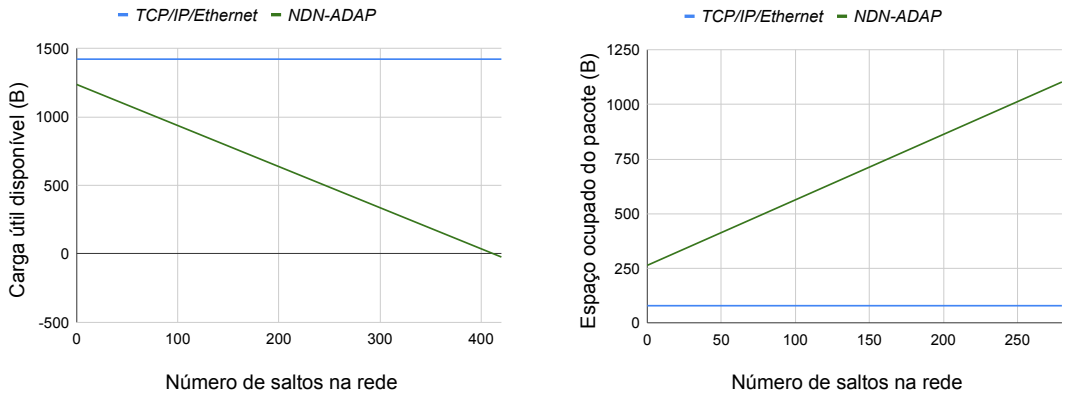
O desempenho do *Segment Routing* do NDN-ADAP acompanha uma perda da capacidade de transmissão da carga útil do pacote, uma vez que parte do MTU da rede será utilizado para envio de instruções de encaminhamento *Segment Routing*, conforme ilustrado pela Equação 1 e Figura 5(a). Isto ocorre, pois estratégias de encaminhamento *Segment Routing* utilizam uma pilha de instruções de encaminhamento [Moiseenko and Oran 2017] e, consequentemente, possuem complexidade espacial $O(n)$, onde $n$ é o número de saltos entre comutadores até que o pacote alcance seu destino [Lee et al. 2015]. Já as estratégias de encaminhamento baseadas em FIB possuem complexidade espacial $O(1)$, pois usam um campo de tamanho fixo do cabeçalho do pacote (ver Figura 5(b)).

$$P = MTU - c - \sum_{k=1}^{n} s_k \tag{1}$$

Onde $P$ é a carga útil (*payload*) do pacote, $MTU$ é a unidade máxima de transmissão da rede, $c$ é o tamanho do cabeçalho fixo NDN-ADAP, $s_k$ é o tamanho da $k$-ésima instrução *Segment Routing* contida no pacote NDN-ADAP e $n$ é o número de saltos que o pacote NDN-ADAP realizará até alcançar seu destino.

Para redes Ethernet com MTU de 1500 *bytes*, considerando que o tamanho do cabeçalho NDN-ADAP ($c$) é de 263 *bytes* e que o tamanho de cada instrução *Segment Routing* ($s_k$) é de 3 *bytes*, obtemos que o número máximo de saltos possíveis ($n$) é 412 saltos, supondo que o *payload* ocupe a menor porção possível do pacote, tendo apenas 1 *byte* de tamanho (ver Figura 5(a)).

Uma vez que há um limite máximo de saltos previsto no NDN-ADAP, é necessário que um reconstrutor de instruções *Segment Routing* seja colocado a intervalos regulares de saltos na rede, a fim de permitir que o NDN-ADAP seja utilizado em redes de grande porte. Supondo uma rede Ethernet com MTU de 1500 *bytes*, o NDN-ADAP precisará de,




(a) Número de saltos versus carga útil disponível.

(b) Número de saltos versus complexidade espacial do protocolo (NDN-ADAP versus IP).

**Figura 5. Carga útil disponível e complexidade espacial por pacote.**

pelo menos, um reconstrutor de instruções *Segment Routing* a cada 412 saltos, supondo um *payload* de 1 *byte*. Para *payloads* maiores, precisaremos utilizar mais reconstrutores de instruções ao longo da rede.

### 4.2. Avaliação Experimental

Realizamos experimentos no ambiente de emulação Mininet [Lantz et al. 2010] em conjunto com o BMv2[1], visando ratificar os resultados encontrados na avaliação analítica e avaliar o comportamento do núcleo de uma rede NDN-ADAP. Para isso, geramos pacotes NDN-ADAP dentro dos *hosts* $H_1$ e $H_2$ do Mininet e utilizamos dados NDN como *payload* desses pacotes. Em seguida, os *hosts* encaminharam esses pacotes para a rede usando a técnica *Segment Routing*.

#### 4.2.1. Ambiente de experimentação

Para criar e avaliar a solução proposta neste estudo, uma máquina virtual Ubuntu 16.04 LTS com kernel 4.4.0-141-generic x86_64 SMP foi configurada no VirtualBox 5.2.26 PUEL através da ferramenta vagrant e das instruções contidas no repositório P4 Tutorials[2]. Essa máquina foi executada sobre o sistema operacional hospedeiro Debian 9 e configurada com duas CPUs virtuais *Intel Core i7 4500U* com *clock* de 2 GHz e com 4 GB de RAM DDR3 de 1666 MHz.

No ambiente de emulação Mininet/BMv2, executado na máquina virtual, conectamos dois *hosts* $H_1$ e $H_2$ nos *switches* das extremidades do núcleo da rede NDN-ADAP. No núcleo da rede, conectamos cinco *switches* de rede utilizando a topologia estrela. Desta forma, foi possível mimetizar o cenário presente nos núcleos das redes. Os *hosts* $H_1$ e $H_2$ operam como reconstrutores *Segment Routing*, sendo conectados aos *switches* do núcleo da rede por meio da tecnologia de enlace cabeado IEEE 802.3ab (Gigabit Ethernet). Para

[1] `https://github.com/p4lang/behavioral-model`
[2] `https://github.com/p4lang/tutorials`

configurar o ambiente de emulação e os enlaces supracitados, utilizamos os parâmetros de emulação descritos na Tabela 1. Todos os demais parâmetros de emulação não contidos nessa tabela foram adotados a partir dos padrões do Mininet.

**Tabela 1. Parâmetros da Emulação.**

| | | |
|---|---|---|
| **Geral** | Tempo de emulação | 10 segundos |
| | Número de execuções | 100 replicações |
| **Fluxo de envio de pacotes (carga)** | Velocidade máxima de envio | Carga linear |
| **Tecnologia dos enlaces** | *Hosts* e comutadores | IEEE 802.3ab |
| **Capacidade dos enlaces** | *Hosts* e comutadores | 1000 Mbps |

### 4.2.2. Metodologia de avaliação

Realizamos experimentos com o NDN-ADAP comparando-o com a pilha de protocolos UDP/IPv4/Ethernet. Para realizar os experimentos de maneira mais realista possível, assumimos que o *payload* dos pacotes NDN-ADAP e UDP/IP são pacotes NDN, que utilizam o formato TLV descrito na especificação oficial da rede NDN [NFD 2019]. Além disso, selecionamos as seguintes métricas de desempenho para avaliar a solução proposta:

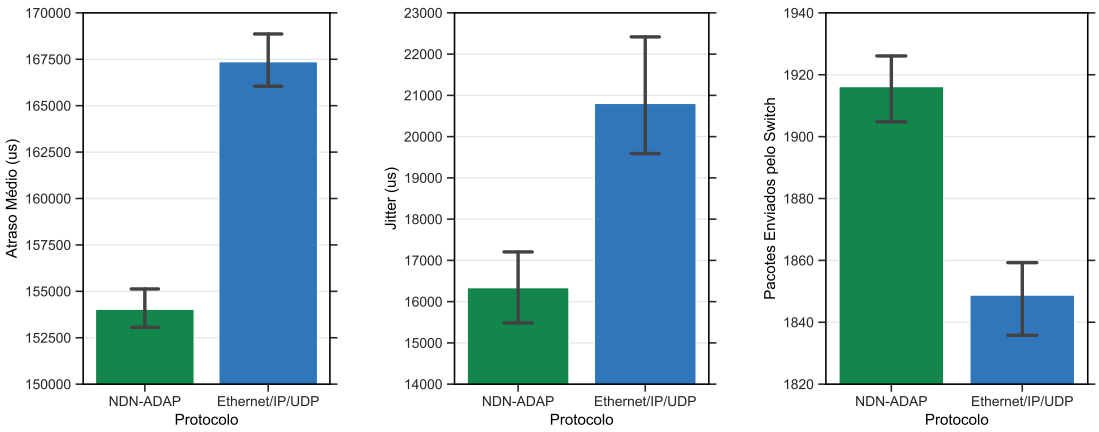
- **Total de pacotes enviados pelo *switch* (pps)**: Número total de pacotes enviados pelo *switch* para o *host* durante cada execução da emulação;
- **Atraso médio (μs)**: Intervalo de tempo associado ao envio e recepção de pacotes NDN a partir dos *hosts* $H_1$ e $H_2$, respectivamente;
- ***Jitter* (μs)**: Variação do atraso associado aos envios de pacotes NDN na rede.

### 4.2.3. Resultados e discussões

Na Figura 6 constatamos os atrasos alcançados por ambas as soluções avaliadas (NDN-ADAP e UDP/IP). O NDN-ADAP obteve um atraso médio inferior ao UDP/IP, como mostra a Figura 6(a). Isso demonstra que a solução proposta (NDN-ADAP) é mais eficiente que a utilizada em redes legadas (UDP/IP). Ao analisarmos a métrica *jitter*, observamos que o NDN-ADAP apresenta menor variação nos atrasos médios dos pacotes NDN, como mostra a Figura 6(b). Isto se deve à complexidade envolvida na busca de uma entrada na FIB, que requer um tempo de processamento maior do comutador, quando comparado a uma mera execução de instrução *Segment Routing*. Além disso, um *switch* que utiliza o NDN-ADAP consegue processar um fluxo maior de pacotes do que os *switches* baseados em UDP/IP, como ilustrado pela Figura 6(c). Novamente, isto é um reflexo do tempo de processamento necessário para encaminhar um pacote através de uma busca pelas entradas das tabelas FIB, que é significativamente superior ao tempo necessário para a execução de uma instrução *Segment Routing* simples.

## 5. Conclusão e Trabalhos Futuros

Neste estudo, propomos uma arquitetura de rede baseada no modelo *Fabric* para redes centradas na informação, denominada de NDN-ADAP. Além disso, propomos um protocolo de enlace que atua na camada de adaptação da NDN para encaminhar pacotes nesta




(a) Atraso médio.

(b) *Jitter*.

(c) Quantidade de pacotes enviados pelo *switch*.

**Figura 6. Atraso fim a fim e quantidade de pacotes enviados pelo *switch*.**

arquitetura, também denominado de NDN-ADAP. Através deste protocolo, a rede pode encaminhar pacotes utilizando diferentes mecanismos de encaminhamento (e.g., Ethernet, IPv4, *Segment Routing*, Bluetooth, etc.). A arquitetura de rede proposta busca viabilizar a implantação eficiente e escalável de redes NDN em larga escala, de maneira a melhorar o encaminhando de pacotes NDN no núcleo. Para analisar e validar o NDN-ADAP, utilizamos um mecanismo de encaminhamento baseado na técnica de roteamento por segmentos. Nossos resultados experimentais indicam que os comutadores do núcleo da rede, baseados no protocolo NDN-ADAP, apresentam melhorias significativas em relação ao atraso médio da rede (de até 8%) e em relação ao *jitter* da rede (de até 21%), quando comparados aos comutadores baseados no protocolo IPv4. Com relação ao número de pacotes enviados, o *switch* NDN-ADAP conseguiu enviar (processar) 3,6% mais pacotes que o *switch* UDP/IP/Ethernet.

Como sugestão de trabalhos futuros, pretende-se avaliar o desempenho global da arquitetura de rede NDN-ADAP utilizando um *testbed* real. Dessa forma, será possível avaliar com maior precisão o impacto sobre o encaminhamento dos pacotes fim a fim, de maneira a identificar gargalos no encaminhamento, derivados tanto do núcleo da rede quanto da pilha NDN, presente nas bordas da rede.

## Agradecimentos

Os autores agradecem o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB).

## Referências

Azgin, A., Ravindran, R., and Wang, G. (2016). pit/LESS: Stateless Forwarding in Content Centric Networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7.

Biondi, P., Lalet, P., Potter, G., Valadon, G., Raynal, F., Kacherginsky, P., Loss, D., and Scapy Community (2019). Scapy. GitHub Repository.

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.

Casado, M., Koponen, T., Shenker, S., and Tootoonchian, A. (2012). Fabric: A Retrospective on Evolving SDN. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 85–90, New York, NY, USA. ACM.

Ishimori, A., Cerqueira, E., and Abelém, A. (2017). Tag-and-Forward: A source-routing enabled data plane for OpenFlow Fat-Tree Networks. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 923–928.

Jahanian, M. and Ramakrishnan, K. K. (2019). Name Space Analysis: Verification of Named Data Network Data Planes. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN '19, pages 44–54, New York, NY, USA. ACM.

Lantz, B., Heller, B., and McKeown, N. (2010). A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA. ACM.

Lee, T., Pappas, C., Basescu, C., Han, J., Hoefler, T., and Perrig, A. (2015). Source-Based Path Selection: The Data Plane Perspective. In *The 10th International Conference on Future Internet*, CFI '15, pages 41–45, New York, NY, USA. ACM.

Miguel, R., Signorello, S., and Ramos, F. M. V. (2018). Named Data Networking with Programmable Switches. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 400–405.

Moiseenko, I. and Oran, D. (2017). Path Switching in Content Centric and Named Data Networks. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, pages 66–76, New York, NY, USA. ACM.

NDN-CXX (2019). ndn-cxx: NDN C++ library with eXperimental eXtensions. Disponível em: `https://named-data.net/doc/ndn-cxx/current/`. Último acesso em: 29 de novembro de 2019.

NFD (2019). Nfd developer's guide. Disponível em: `https://named-data.net/publications/techreports/ndn-0021-6-nfd-developer-guide/`. Último acesso em: 29 de novembro de 2019.

Signorello, S., State, R., François, J., and Festor, O. (2016). NDN.p4: Programming information-centric data-planes. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 384–389.

Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. (2014). Named Data Networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73.

Zhang, Z., Lu, E., Li, Y., Zhang, L., Yu, T., Pesavento, D., Shi, J., and Benmohamed, L. (2018). NDNoT: A Framework for Named Data Network of Things. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*, ICN '18, pages 200–201, New York, NY, USA. ACM.