Provisionamento de QoS em NDN através da diferenciação de serviços e gerenciamento de cache

Francisco Renato C. Araújo[1,2], **Leobino N. Sampaio**[1*]

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

[2]Campus Avançado de Mombaça – Universidade Estadual do Ceará (UECE)

{franciscorca, leobino}@ufba.br

Abstract. *Named Data Networks (NDN) decouple data from its location through exclusive naming and operate on a best-effort model, limiting support for applications that require Quality of Service (QoS). The NDN cache expands manageable resources, but traditional cache policies do not meet QoS requirements. This work presents the DSPPC policy, which considers data popularity and its service class in cache management, along with a QoS encoding mechanism in packet names. Simulations demonstrate that the DSPPC proposal keeps priority data in the cache without compromising less-prioritized data, compared to the evaluated cache policies.*

Resumo. *As Redes de Dados Nomeados (NDN) desvinculam os dados de sua localização por meio de nomeação exclusiva e operam no modelo de melhor esforço, limitando o suporte a aplicações que exigem Qualidade de Serviço (QoS). O cache da NDN expande os recursos gerenciáveis, mas as políticas tradicionais de cache não atendem aos requisitos de QoS. Este trabalho apresenta a política DSPPC, que considera a popularidade dos dados e sua classe de serviço no gerenciamento do cache, além de um mecanismo de codificação de QoS no nome dos pacotes. Simulações demonstram que a proposta DSPPC mantém dados prioritários no cache sem comprometer os menos prioritários em comparação com as políticas de cache avaliadas.*

1. Introdução

As Redes de Dados Nomeados (*Named Data Networking* – NDN) adotam um modelo de comunicação orientado a dados, desvinculando a identificação do conteúdo de sua localização. O objetivo da NDN é atender às solicitações com base no conteúdo nomeado, em vez de depender da localização dos dispositivos que fornecem os dados. Para isso, os nós da NDN utilizam três estruturas principais: PIT (*Pending Interest Table*), para encaminhamento; FIB (*Forwarding Information Base*), para roteamento; e CS (*Content Store*), para o armazenamento temporário de dados [Zhang et al. 2014]. A CS representa um cache em cada nó e precisa ser gerenciada por alguma política (e.g., LRU e FIFO).

Embora promissora, a NDN trata os dados com base no modelo de melhor esforço [Ambalavanan et al. 2022]. Por esse motivo, aplicações que exigem Qualidade de

*Os autores agradecem o apoio da CAPES, do CNPq (nº 432064/2018-4, 316208/2021-3, 402854/2022-5), da FAPESB (nº TIC0004/2015) e do *Air Force Office of Scientific Research* (*award* FA9550-23-1-0631).

Serviço (*Quality of Service* – QoS), como serviços de *streaming* e aplicações em tempo real, não encontram suporte nativo na NDN. Mecanismos de diferenciação de serviços na NDN ainda necessitam de maior investigação [Gündoğan et al. 2020], especialmente porque o tráfego das aplicações modernas apresenta características diversificadas e requisitos distintos. Nesse contexto, há aspectos essenciais relacionados à QoS na NDN que permanecem em aberto [Ambalavanan et al. 2022]. A utilização de cache na rede expande o conjunto de recursos gerenciáveis, reduzindo tanto a latência quanto a carga de encaminhamento [Gündoğan et al. 2020]. No entanto, políticas de cache tradicionais podem não atender adequadamente aos requisitos de QoS.

Políticas de cache para NDN têm sido amplamente investigadas, porém, a maioria dessas pesquisas não foca nos requisitos de QoS. Algumas iniciativas de políticas de cache cientes de QoS foram propostas para cenários específicos, como redes veiculares [Dhakal et al. 2023] e Internet das Coisas [Gündoğan et al. 2020]. Já em cenários mais amplos, como a proposta de [Singh and Sarma 2021], as políticas propostas geralmente se baseiam em abordagens tradicionais, como LRU. O desenvolvimento de mecanismos eficazes para garantir QoS é essencial para que a NDN alcance ampla adoção e suporte às aplicações modernas [Ambalavanan et al. 2022].

Neste contexto, este trabalho apresenta uma política de cache ciente de QoS para NDN. A política proposta, DSPPC (*Differentiated Services and Popularity Probabilistic Cache*), considera a frequência de acesso aos dados e sua classe de serviço na tomada de decisão para a remoção de dados do cache. Para viabilizar o transporte da informação da classe de serviço nos pacotes NDN, foi desenvolvido o mecanismo de codificação de nome `nds` (abreviação de *Named-data Differentiated Services* – NDiffServ). O `nds` foi projetado como um componente de nome NDN [Team 2021]. Esse mecanismo estabelece uma convenção de nomeação que permite incorporar a classificação de classes de serviço nos nomes dos pacotes NDN. Assim, as principais contribuições deste trabalho residem na proposta da política de cache DSPPC e no mecanismo de codificação de nome `nds`. Ambas as contribuições abordam aspectos fundamentais da NDN, como o gerenciamento de cache e a nomeação de dados. A política DSPPC foi avaliada por meio de simulações realizadas no simulador ndnSIM [Mastorakis et al. 2017]. Os experimentos foram conduzidos em duas topologias de rede, Dumbbell e Abilene, com o objetivo de comparar o desempenho da proposta em relação às políticas ARC-QoS [Singh and Sarma 2021] e LRU [Mastorakis et al. 2017]. Os resultados demonstram que a DSPPC atinge seu objetivo de priorizar o armazenamento no cache de dados de classes de maior prioridade, sem prejudicar significativamente o atendimento de dados de classes menos prioritárias.

O restante do trabalho está organizado da seguinte forma: os trabalhos relacionados são apresentados na Seção 2. A proposta é apresentada na Seção 3. Os experimentos são discutidos na Seção 4. Por fim, o trabalho é concluído na Seção 5.

2. Trabalhos Relacionados

Nesta seção, são apresentados os trabalhos diretamente relacionados às políticas de substituição de cache em NDN, com foco na abordagem de QoS. Políticas voltadas para outros fins, fogem do escopo desta pesquisa. Quanto à codificação da QoS como componente de nome NDN, destacam-se dois trabalhos na literatura: [Moiseenko and Oran 2021] e [Jangam et al. 2020]. Ambos diferem na localização do componente de QoS nos nomes

dos pacotes e são modelos conceituais. [Moiseenko and Oran 2021] propuseram utilizar o prefixo de nome para representar a QoS, apresentando duas abordagens: EC3 (*Equivalence Class Component Count*) e ECNCT (*Equivalence Class Name Component Type*). A abordagem EC3 emprega o prefixo de maneira indireta, armazenando em um novo campo do pacote de dados a quantidade de componentes do prefixo que representam a QoS. Já a abordagem ECNCT utiliza o prefixo diretamente para codificar a QoS. Por outro lado, [Jangam et al. 2020] sugerem que a informação de QoS deve ser alocada no sufixo do nome. Apesar de apresentarem direções importantes de pesquisa, os trabalhos carecem de implementação e avaliação prática de suas propostas. Esses estudos serviram como base para o desenvolvimento da solução NDiffServ (`nds`), proposta neste trabalho.

Em relação às políticas de cache, [Singh and Sarma 2021] propuseram ARC-QoS (*Adaptive Replacement Cache with Quality of Service*), uma política focada no gerenciamento de cache para aplicações sensíveis ao atraso, priorizando dados com maior urgência. A política opera em duas etapas: primeiro, calcula periodicamente as taxas de acerto (popularidade) de conteúdos com alta e baixa prioridade nos nós NDN; em seguida, remove entradas de baixa prioridade para alocar espaço para dados mais críticos. Para isso, utiliza duas listas LRU: L1 para conteúdos de baixa popularidade e L2 para os de alta popularidade. A ARC-QoS busca evitar a inanição de serviços de baixa prioridade, mas não especifica como determina a prioridade nos pacotes. Assim, o mecanismo `nds` proposto foi adotado para esse fim.

Em [Kumar et al. 2021], foi proposta uma política de cache visando reduzir o atraso na recuperação de conteúdos e o consumo de recursos de rede. Essa política segmenta os nós da rede em *clusters*, permitindo que os dados sejam armazenados apenas no cache dos roteadores pertencentes ao *cluster* do consumidor solicitante. A remoção de dados do cache é baseada na popularidade do conteúdo e no número de saltos. A política busca armazenar conteúdos populares próximos aos dispositivos finais, utilizando uma `Tabela de Popularidade` para determinar a frequência de acesso em cada nó. Além disso, os pacotes de interesse e dados são alterados para carregar informações sobre a quantidade de saltos e o identificador do *cluster*. Em contraste, a política DSPPC proposta não altera os pacotes, transportando a informação necessária, a classe de serviço, diretamente no componente de nome `nds` proposto.

Em [Dhakal et al. 2023], foi proposta a PBDCV, uma política de cache ciente de QoS para redes NDN veiculares sem fio. A PBDCV classifica os dados por prioridade, particiona o cache e realiza a remoção de dados com base na temporalidade. A QoS é identificada no prefixo de nome dos pacotes, classificando-os em três níveis: alta, média e baixa prioridade. O cache é dividido conforme essas classes, e os dados são movidos entre as partições de acordo com sua temporalidade, com os dados de alta prioridade sendo transferidos para partições de menor prioridade conforme seu envelhecimento. A política DSPPC foi proposta para cenários amplos e utiliza o componente `nds` para QoS.

Em [Abdelaal et al. 2020], foi proposta a liteNDN, uma política de cache que integra uma estratégia de encaminhamento ciente de QoS. A principal característica da liteNDN é o compartilhamento de informações sobre os dados recebidos entre os nós vizinhos, através do envio de pacotes de dados virtuais, que são enviados mesmo na ausência de pacotes de interesse. Essa abordagem vai contra o modelo de comunicação da NDN, que é orientado pelo receptor. Além disso, a liteNDN também envia dados seg-

mentados sem a solicitação prévia de interesse, o que contraria o controle de fluxo nativo da NDN, que estabelece que um interesse deve recuperar apenas um dado específico.

3. Componente de Nome NDS e Política DSPPC

Nesta seção, são apresentadas a proposta de codificação da classe de serviço como um componente de nome, NDiffServ (`nds`), e a política de substituição de cache DSPPC.

3.1. Componente NDS

O componente de nome `nds` (abreviação para NDiffServ – *Named-data Differentiated Services*) tem como função principal transportar, no nome do pacote, a informação referente à classe de serviço. As classes de serviços do `nds` são derivadas das classes definidas pelo DiffServ IP, campo DSCP. O `nds` foi implementado como um componente de nome `NdsNameComponent`, criado no formato TLV (*Type-Length-Value*), com o TLV-TYPE 128 decimal[1] (0x80 hexadecimal), conforme mostrado na Tabela 1.

Tabela 1. Tipo de componente de nome atribuído ao NDS.

Tipo de componente	TLV-TYPE	Formato TLV-VALUE	Descrição	Formato URI
`NdsNameComponent`	128 (0x80)	`*OCTET`	Classes de serviço do NDS	`nds=<string>`

O `Type(T)` identifica o tipo `nds` com o valor fixo 128 (marcador 0x80), armazenado em 1 byte (`1 octet`). O `Length(L)` indica o tamanho do valor, por exemplo, o valor `AF11` (4 caracteres) possui tamanho 4, e esse número pode ser armazenado em 1 byte (`1 octet`). O `Value(V)` representa o valor real da classe de serviço, como `AF11`, que ocupa 4 bytes (1 byte por caractere). A Tabela 2 sumariza a codificação TLV do componente `nds`, com o exemplo da classe de serviço `AF11`, resultando em `nds=AF11`. O tamanho total da codificação TLV neste exemplo seria: 1 (`T`) + 1 (`L`) + 4 (`V`) = 6 bytes. Assim, para a classe `BE` (`V=2` bytes), seriam necessários 4 bytes no total, incluindo `1(T)+1(L)`. O mesmo se aplica à classe de serviço `EF`.

Tabela 2. Exemplo de codificação do componente NDS com a classe `AF11`.

Conteúdo	Campo	Conteúdo codificado	Tamanho (Bytes)
`nds`	`T`	128 (valor fixo)	1
	`L`	4	1
`AF11`	`V`	‘A’, ‘F’, ‘1’, ‘1’	4

No nome `/ufba/ic/nds=AF11/seq=8` o componente de nome `nds=AF11` é anexado ao nome principal do interesse `/ufba/ic` e anterior ao componente sequência (`seq`). A posição do componente `nds` após o nome principal do interesse se baseia na ideia de adicionar a informação da classe de serviço como sufixo de nome, pelas razões discutidas adiante. A decisão de projeto do componente `nds` vir antes do componente `seq` se fundamenta no fato dessa abordagem seguir as especificações definidas no formato

[1]O tipo decimal 128 $\in$ ao subconjunto de intervalos reservados TLV-TYPE da especificação do pacote NDN. A faixa [128, 252] é reservada para futuras atribuições de uso da aplicação (codificação de 1 byte). Disponível em: `docs.named-data.net/NDN-packet-spec/current/types.html` e `redmine.named-data.net/projects/ndn-tlv/wiki/NameComponentType`.

de pacote NDN e na política de atribuição de componentes de nome [Team 2021]. Além de causar menor impacto no código base da NDN.

A codificação da classe de serviço como prefixo no nome do interesse [Moiseenko and Oran 2021] pode sobrecarregar tanto as entradas da FIB quanto da PIT [Jangam et al. 2020]. Isso ocorre porque a inclusão da informação de QoS no prefixo do nome do interesse resulta em entradas que diferem apenas pelo componente de identificação da classe de serviço. Consequentemente, a agregação de entradas na FIB é comprometida, levando a uma sobrecarga dessa estrutura. Além disso, a PIT, estrutura fundamental para o funcionamento da NDN, é responsável por manter as entradas necessárias para assegurar a entrega dos dados solicitados. Contudo, a sobrecarga na PIT pode ampliar a superfície de ataques por interesses falsos [Araújo et al. 2023]. Para mitigar esses problemas, o componente `nds` foi codificado como sufixo do nome. Então, as aplicações podem definir a classe de serviço desejada como componente de nome no pacote.

3.2. Política de Substituição de Cache

A política de substituição de cache proposta, DSPPC (*Differentiated Services and Popularity Probabilistic Cache*), se baseia na classe de serviço e na popularidade dos dados para gerenciar as entradas no cache. A Tabela 3 mostra as classes de serviço adotadas e seus respectivos pesos. A política DSPPC utiliza o `nds` presente no nome dos dados para atribuir um peso (W_c) de importância ao dado de acordo com a classe de serviço: BE (*Best Effort*), AF (*Assured Forwarding*) e EF (*Expedited Forwarding*).

Tabela 3. Classes de serviço e seus respectivos pesos.

Classe[a]	Valor DSCP	Peso (W_c)	Descrição
BE	0	0.1	Melhor esforço, sem garantias de QoS.
AF11-AF13	10, 12, 14	0.3, 0.25, 0.2	Baixa prioridade dentro de AF.
AF21-AF23	18, 20, 22	0.5, 0.45, 0.4	Média prioridade dentro de AF.
AF31-AF33	26, 28, 30	0.7, 0.65, 0.6	Alta prioridade dentro de AF.
AF41-AF43	34, 36, 38	0.9, 0.85, 0.8	Máxima prioridade dentro de AF.
EF	46	1.0	EF, prioridade máxima.

[a] `iana.org/assignments/dscp-registry/dscp-registry.xhtml`.

A popularidade $P(d)$ de um dado d é determinada com base em sua frequência de acesso $A(d)$ e no peso da classe W_c, este último definido conforme a classe de serviço do próprio dado, conforme a Equação 1. O objetivo é atribuir maior popularidade a dados mais frequentemente acessados ou pertencentes a classes de maior prioridade.

$$P(d) = \frac{e^{W_c} \cdot A(d)^{\beta}}{\max_{i \in \text{Cache}} \left(e^{W_{c_i}} \cdot A(i)^{\beta} \right)}; \qquad \text{onde:} \tag{1}$$

- $P(d)$: Popularidade do dado d;
- e: Número de Euler ($e \approx 2.718$), utilizado para garantir crescimento exponencial e favorecer classes de maior prioridade;
- W_c: Peso da classe de serviço do dado d (prioridade com valores de 0.1 a 1.0);
- $A(d)$: Frequência de acesso ao dado d (quantidade de solicitações);
- β: Parâmetro de suavização para ajustar o impacto de $A(d)$;
- $\max_{i \in \text{Cache}}$: O máximo valor de popularidade bruta entre todos os dados i armazenados no cache.

A probabilidade de remoção $R(d)$ de um dado d do cache é definida como: $R(d) = 1 - P(d)$. Essa probabilidade é inversamente proporcional à popularidade $P(d)$. Dessa forma, quanto menor a popularidade de um dado, maior será sua probabilidade de remoção. A estratégia de decisão para substituição de um dado d no cache utiliza o valor de $R(d)$ como critério probabilístico para determinar sua remoção. Assim, dados com valores mais altos de $R(d)$ têm maior probabilidade de serem removidos. Essa abordagem garante que dados de menor prioridade ainda possuam uma chance, embora reduzida, de permanecer no cache. Consequentemente, reduz-se o risco de inanição de classes com pesos menores, promovendo maior diversidade de dados no cache entre diferentes classes.

3.2.1. Função de β

O parâmetro β controla a sensibilidade à frequência de acesso $A(d)$ no cálculo da popularidade dos dados. Para $\beta > 1$, o impacto da frequência é ampliado, tornando dados com frequências mais altas significativamente mais populares, mesmo que pertencem a classes de menor prioridade. Quando $\beta = 1$, a popularidade é determinada de forma proporcional e linear pela combinação da classe e da frequência de acesso. Já para $\beta < 1$, o impacto da frequência é reduzido, e o peso da classe (W_c) passa a ter maior influência, favorecendo dados de classes mais altas, independentemente da frequência de acesso. Dessa forma, β ajusta o equilíbrio entre o peso da classe e o peso de acesso.

3.2.2. Gerenciamento de Dados

O gerenciamento dos dados armazenados no cache é descrito no Algoritmo 1. Se o tamanho do cache ultrapassar o limite, o algoritmo calcula a popularidade dos dados com base na Equação 1 e, a partir dessa popularidade, determina a probabilidade de remoção $R(d)$ de cada item. O algoritmo então itera sobre as entradas do cache, gerando um número aleatório para cada dado e comparando-o com a sua probabilidade de remoção. Se o número gerado for menor que a probabilidade associada ao dado, ele é removido do cache, interrompendo o processo. Caso nenhuma entrada seja removida durante a iteração, uma entrada aleatória é excluída para garantir que o tamanho do cache seja reduzido. O custo computacional do algoritmo, no pior caso, é $O(n)$, onde n é o número de entradas no cache, pois é necessário iterar sobre todas as entradas para verificar a condição de remoção. Assim, o algoritmo assegura que o cache não ultrapasse o limite, realizando a remoção de forma probabilística, priorizando dados mais populares e permitindo a permanência dos dados de menor prioridade.

4. Estudo Experimental e Análise dos Resultados

4.1. Ambiente de Experimentação

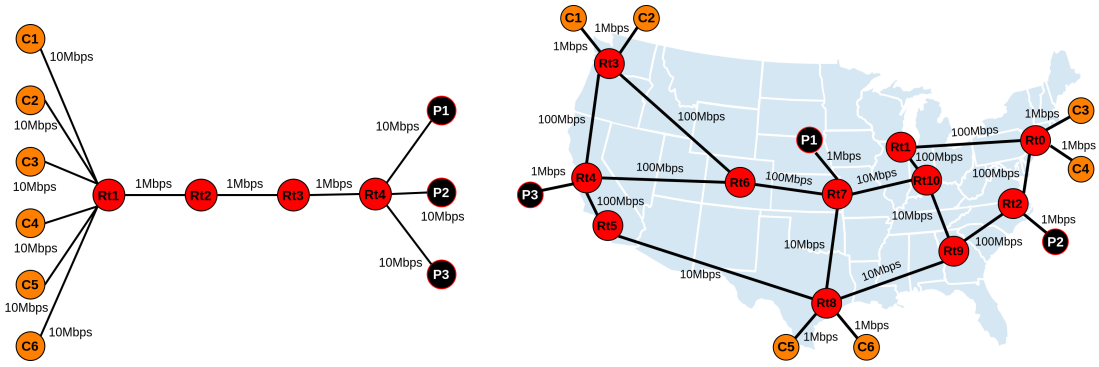
A proposta foi implementada no simulador oficial de redes NDN, ndnSIM [Mastorakis et al. 2017], na versão mais recente[2]. Os experimentos utilizaram as topologias da Figura 1. A Figura 1(a) apresenta a topologia Dumbbell, onde os consumidores C1 a C6 estão conectados aos produtores P1, P2 e P3 via roteadores Rt1 a Rt4, com

[2]ndnSIM v2.9. Disponível em: `https://ndnsim.net/current/`

Algoritmo 1: Gerenciamento de dados no cache.

```
1  Função Remoção():
2      Tamanho do cache = GETCS()->SIZE()
3      Limite = GETLIMIT()
4      se Tamanho do cache > Limite então
5          Calcular a popularidade P(d) normalizada com a Equação 1
6          Calcular a probabilidade de remoção R(d)
7          removido = não
8          para cada entrada it no cache faça
9              Gerar um número aleatório r no intervalo [0, 1]
10             se r < probabilidade de remoção da entrada então
11                 Remover a entrada it do cache // Remoção baseada em R(d).
12                 removido = sim
13                 Interromper o laço
14         se não removido então
15             Remover uma entrada aleatória do cache // Remoção aleatória.
```

gargalos nos enlaces de 1 Mbps entre os roteadores. Já a Figura 1(b) ilustra a topologia Abilene, com múltiplos caminhos possíveis entre consumidores e produtores, e gargalos nos enlaces de 1 Mbps nos dispositivos finais. Em ambas topologias, C1 solicita dados da classe BE ao produtor P1, enquanto C6 solicita dados da classe EF ao P3. O produtor P2 fornece dados nas classes AF12, AF22, AF32 e AF42 para os consumidores C2 a C5, respectivamente. As topologias foram inspiradas no *dataset Topology Zoo*[3] e permitem avaliar cenários simples e complexos de teste.



(a) Topologia Dumbbell. (b) Topologia Abilene.

Figura 1. Em ambas as topologias utilizadas, os consumidores C1 (BE), C2 (AF12), C3 (AF22), C4 (AF32), C5 (AF42) e C6 (EF) requisitaram dados aos produtores P1 (BE), P2 (AF12-AF42) e P3 (EF), de acordo com a classe de serviço.

A aplicação dos consumidores utilizou a distribuição Zipf-Mandelbrot, estendida para incluir a codificação da classe de serviço no nome dos pacotes NDN, conforme a proposta `nds` (Seção 3.1). A frequência da requisição de pacotes de interesse por segundo (pps) foi configurada da seguinte forma: C1 (`nds=BE`) enviou 5 pps; C2, 10 pps, com incrementos de 5 pps até C5 (25 pps). C6 (`nds=EF`) enviou 50 pps. Cada classe de serviço foi configurada com um catálogo de dados específico: `BE` com 100 conteúdos, `AF12-AF42` com 300, e `EF` com 50 pacotes. Esses parâmetros, apresentados na Tabela 4,

[3] *The Internet Topology Zoo*. Disponível em: `https://topology-zoo.org/index.html`.

foram definidos com base em trabalhos relacionados para criar um ambiente experimental com concorrência entre as classes de serviço.

Todos os roteadores foram configurados com capacidade de cache para 50 pacotes, enquanto o cache dos dispositivos finais foi desabilitado para forçar a busca de dados nos roteadores e produtores. Cada simulação durou 90 segundos e foi replicada 10 vezes com diferentes sementes, calculando-se a média com um intervalo de confiança de 95% (distribuição *t*-Student, $\alpha = 5\%$). Os fatores e níveis utilizados nos experimentos estão na Tabela 5. A política de cache DSPPC foi avaliada nas topologias Dumbbell e Abilene, comparada às políticas ARC-QoS [Singh and Sarma 2021] e LRU [Mastorakis et al. 2017]. Além disso, foram utilizados os parâmetros q e s da distribuição Zipf-Mandelbrot, que modelam padrões de acesso: s regula a concentração de acessos aos itens mais populares, enquanto q ajusta a magnitude inicial dessa concentração.

Tabela 4. Parâmetros da Simulação.

Parâmetro	Valor
Capacidade do cache de cada roteador	50 pacotes de dados
Política de cache nos dispositivos finais	Desabilitada
Duração de cada simulação	90 s
C1: {`nds`, frequência, # de conteúdos}	{`BE`, 5, 100}
C2: {`nds`, frequência, # de conteúdos}	{`AF12`, 10, 300}
C3: {`nds`, frequência, # de conteúdos}	{`AF22`, 15, 300}
C4: {`nds`, frequência, # de conteúdos}	{`AF32`, 20, 300}
C5: {`nds`, frequência, # de conteúdos}	{`AF42`, 25, 300}
C6: {`nds`, frequência, # de conteúdos}	{`EF`, 50, 50}

Tabela 5. Fatores e Níveis.

Fatores	Níveis
Topologias	Dumbbell e Abilene
Zipf $\{q, s\}$	{0,6, 0,8}, {0,8, 1,0}
Políticas	DSPPC$_{(\beta=1)}$, ARC-QoS e LRU

A Tabela 4 resume os parâmetros utilizados nos experimentos. Os parâmetros não listados assumiram os valores padrão do ndnSIM. Destaca-se ainda o uso do protocolo de roteamento LFID (*Loop-Free Inport-Dependent*) [Schneider et al. 2020] e da estratégia de encaminhamento *Best-route* [Mastorakis et al. 2017].

4.2. Métricas de Desempenho

As métricas utilizadas para a análise de desempenho foram: (i) Atraso médio (s): tempo médio de resposta para os consumidores recuperarem os dados, incluindo possíveis retransmissões de interesses; (ii) Taxa de acerto de cache (pps): proporção de dados encontrados no cache dos roteadores; e (iii) Taxa de interesses satisfeitos (pps): média de pacotes de interesse emitidos pelos consumidores que resultaram na recuperação dos dados solicitados.

4.3. Resultados e Discussões

Os resultados foram analisados com as bibliotecas Pandas e Matplotlib da linguagem Python para análise e visualização dos dados, respectivamente. O primeiro segundo de simulação foi descartado por corresponder ao tempo de configuração do simulador com mensagens de controle internas. Os dados restantes foram avaliados em dois cenários: o cenário simples, com a topologia Dumbbell, e o cenário complexo, com a topologia Abilene, conforme apresentado a seguir.

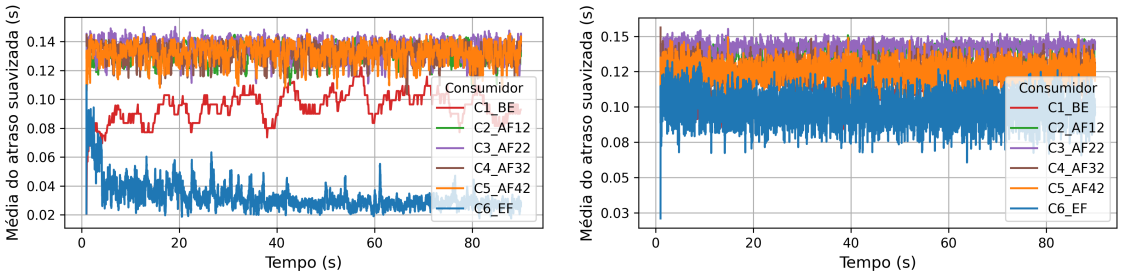
4.3.1. Cenário Simples: Topologia Dumbbell

Neste cenário, com a topologia Dumbbell, existe apenas um caminho entre os consumidores e os produtores. Os quatro roteadores no caminho possuem um gargalo, com um enlace compartilhado de 1 Mbps, conforme ilustrado na Figura 1(a). A Figura 2 apresenta a média suavizada do atraso observado pelos consumidores ao recuperar dados dos produtores. A suavização foi realizada com uma média móvel de janela 20 para melhorar a visualização dos resultados. Após a primeira recuperação, o dado pode ser recuperado do cache de algum roteador nas requisições subsequentes. Quando o dado é obtido do cache de um roteador, o tempo de resposta é reduzido. Assim, quanto mais próximo do consumidor o dado for recuperado, menor o atraso.

A Figura 2(a) apresenta a média suavizada do atraso obtido pelos consumidores com a política de cache proposta DSPPC implementada nos roteadores. Observa-se que o consumidor C6, responsável por requisitar dados com `nds=EF`, inicialmente obteve um atraso médio de $\approx$ 0,10 segundos. Nos primeiros 5 segundos, o atraso caiu para menos de 0,06 segundos, estabilizando em torno de 0,04 segundos após 40 segundos de simulação e permanecendo nesse patamar até o final. A classe EF possui maior peso (W_c) no cálculo da popularidade de dados pela política DSPPC, favorecendo a predominância desses dados no cache e reduzindo o atraso de recuperação. O consumidor C1, embora solicite dados com `nds=BE`, apresentou o segundo menor atraso geral devido à baixa frequência de requisição (5 interesses por segundo), resultando em menor impacto na média de atraso. Já os consumidores C2 a C5, que requisitam dados das classes AF12 a AF42, registraram atrasos médios próximos a 0,14 segundos, influenciados por uma maior frequência de requisição. As Figuras 2(b) e 2(c) mostram os resultados das políticas ARC-QoS e LRU, que apresentaram desempenhos similares, já que a ARC-QoS utiliza duas listas baseadas no princípio LRU. De forma geral, a política DSPPC alcançou um atraso médio geral de $\approx$ 0,09 s, ligeiramente inferior ao das políticas ARC-QoS e LRU, ambas com um atraso médio geral de cerca de 0,11 s, conforme ilustrado na Figura 2(d).

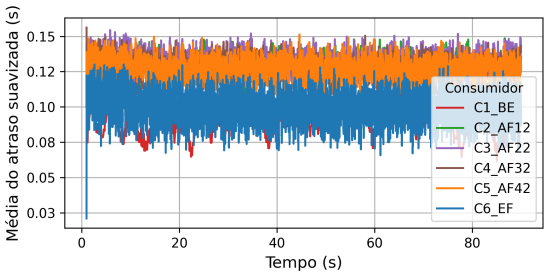
A Figura 3 apresenta a taxa de acerto de cache nos roteadores da rede. Na Figura 3(a), observa-se que o roteador Rt1, localizado na borda próxima aos consumidores, alcançou um acerto de cache de $\approx$ 45 pacotes de dados por segundo, destacando a eficiência da política DSPPC em armazenar e recuperar dados localmente. Já o roteador Rt2 obteve um acerto próximo de 5 pacotes por segundo (pps), refletindo a influência do gargalo nos enlaces e da distância em saltos em relação à borda. Nas políticas ARC-QoS (Figura 3(b)) e LRU (Figura 3(c)), o desempenho do roteador Rt1 foi inferior, com acertos de cache de $\approx$ 25 pacotes por segundo, evidenciando a menor eficiência dessas políticas em comparação com a DSPPC. A Figura 3(d) resume os resultados gerais de acerto de cache por política. A política DSPPC obteve um total de aproximadamente 12 pacotes por segundo, superando as políticas ARC-QoS e LRU, que registraram resultados de cerca de 7,5 pps.

A Figura 4 apresenta a taxa de interesses satisfeitos, i.e., os interesses que resultaram na recuperação de dados. As três políticas avaliadas exibiram resultados semelhantes, com um leve destaque para a política DSPPC, que alcançou uma taxa de satisfação de $\approx$ 49 pps para o consumidor C6, classe de serviço EF (Figura 4(a)). Em comparação, as políticas ARC-QoS (Figura 4(b)) e LRU (Figura 4(c)) obtiveram cerca de 47 pps. Os demais consumidores apresentaram taxas de satisfação proporcionais às suas respectivas

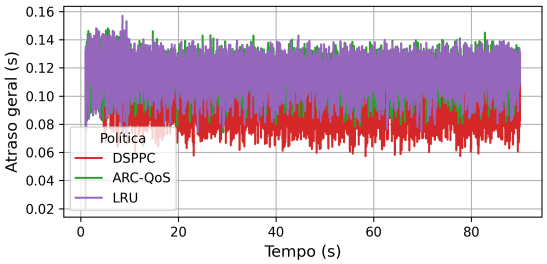


(a) Média de atraso com a política DSPPC.

(b) Média de atraso com a política ARC-QoS.

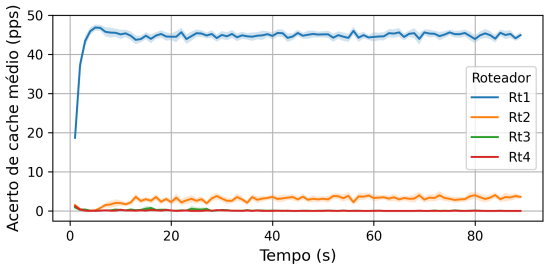


(c) Média de atraso com a política LRU.

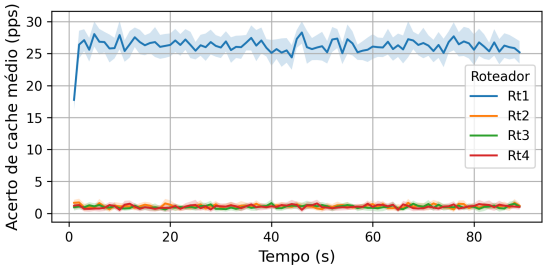


(d) Atraso geral por política.

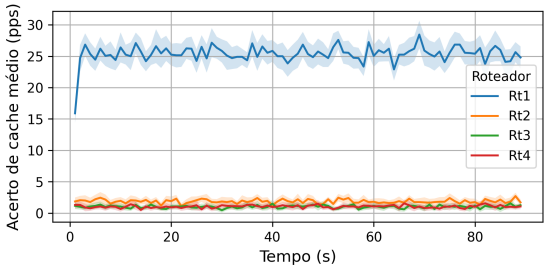
Figura 2. Média do atraso suavizada dos consumidores na topologia Dumbbell.



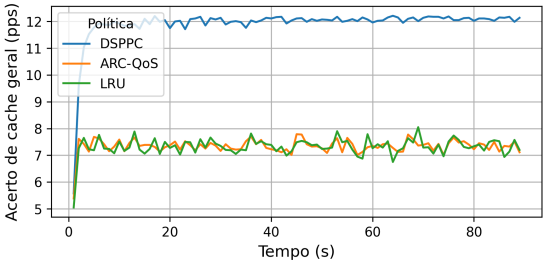
(a) Média de acerto de cache na política DSPPC.



(b) Média de acerto de cache na política ARC-QoS.



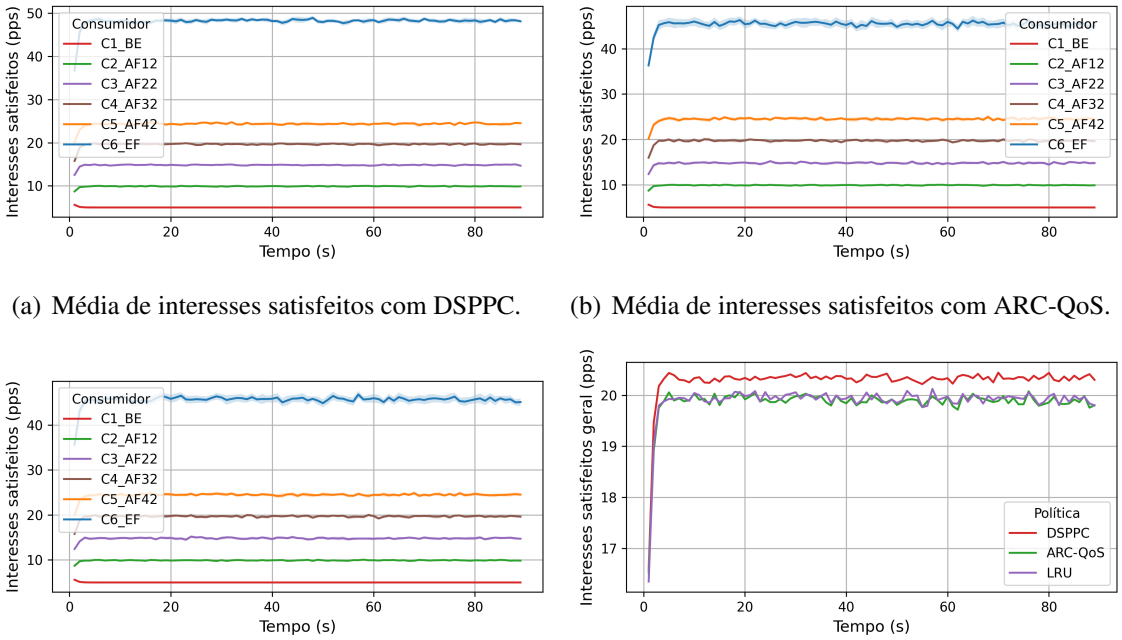
(c) Média de acerto de cache na política LRU.



(d) Acerto de cache por política.

Figura 3. Taxa de acerto de cache nos roteadores com a topologia Dumbbell.

frequências de solicitação de interesse. A Figura 4(d) resume a taxa geral de interesses satisfeitos por política. Nota-se que a política DSPPC apresentou desempenho ligeiramente superior, alcançando uma taxa média de ≈ 21 pps, enquanto as políticas ARC-QoS e LRU registraram valores próximos de 20 pps.



(a) Média de interesses satisfeitos com DSPPC.

(b) Média de interesses satisfeitos com ARC-QoS.

(c) Média de interesses satisfeitos com LRU.

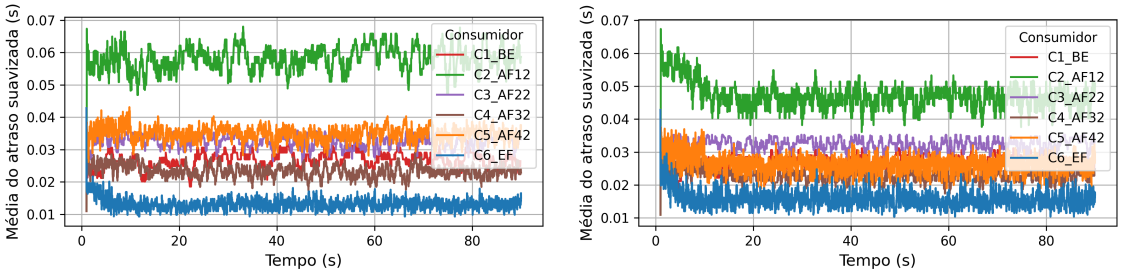
(d) Interesses satisfeitos por política.

Figura 4. Interesses satisfeitos nos consumidores na topologia Dumbbell.

4.3.2. Cenário Complexo: Topologia Abilene

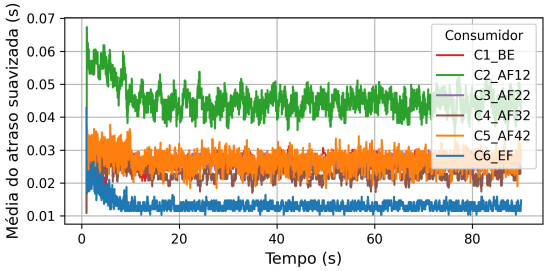
Na topologia Abilene, há múltiplos caminhos para os interesses dos consumidores alcançarem os produtores, mantidos pelo protocolo de roteamento LFID [Schneider et al. 2020]. Entretanto, com a estratégia de encaminhamento *Best-route*, os interesses tendem a seguir a melhor rota ao produtor. A Figura 5 apresenta a média suavizada do atraso alcançada pelas três políticas. A DSPPC (Figura 5(a)) manteve o atraso da classe EF (C6) próximo a 0,01 s, similar à LRU (Figura 5(c)), enquanto a ARC-QoS (Figura 5(b)) apresentou maior variação, chegando a 0,02 s. As três políticas priorizaram o tráfego EF. O consumidor C2 teve o maior atraso, devido C2 estar a pelo menos 8 saltos do produtor P2, enquanto os demais consumidores mantiveram atrasos entre 0,02 s e 0,04 s. No atraso total por política (Figura 5(d)), todas alcançaram $\approx$ 0,03 s.

A Figura 6 apresenta a taxa de acerto de cache nos roteadores no cenário complexo. O roteador Rt8 registrou o maior acerto de cache entre as três políticas avaliadas. Na política DSPPC (Figura 6(a)), Rt8 alcançou uma taxa de acerto de $\approx$ 45 pps, correspondendo a 90% da capacidade de seu cache. Em comparação, Rt8 obteve $\approx$ 38 pps na política LRU (Figura 6(c)) e $\approx$ 33 pps na política ARC-QoS (Figura 6(b)). Este desempenho pode ser atribuído à posição de Rt8 como roteador de borda do consumidor C6, que requisitou dados da classe EF, e do consumidor C5 (`nds=AF42`), ambas classes de maior peso nos experimentos. O roteador Rt5 apresentou o segundo maior acerto de cache, devido à sua posição como próximo salto de Rt8 em direção ao produtor P3, responsável por atender às requisições da classe EF. Rt5 alcançou $\approx$ 20 pps com a política ARC-QoS, $\approx$ 18 pps com a política LRU e apenas $\approx$ 5 pps com a política DSPPC. Na prática, Rt5 complementou Rt8 ao armazenar os 10% restantes do catálogo de dados de C6 (50 pacotes de dados). Assim, Rt8 acomodou quase totalmente a carga de interesses de C6 (50 pps) direcionados ao produtor P3, utilizando sua capacidade de cache de 50

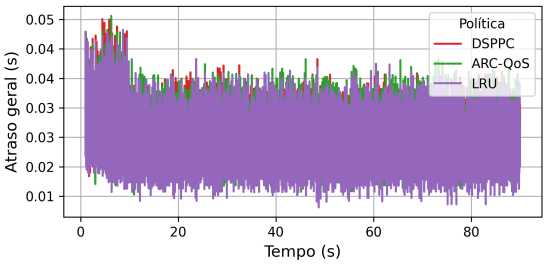


(a) Média de atraso com a política DSPPC.

(b) Média de atraso com a política ARC-QoS.



(c) Média de atraso com a política LRU.



(d) Atraso geral por política.

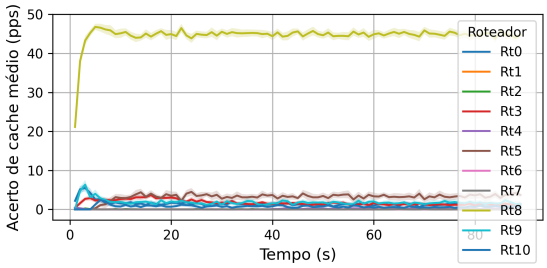
Figura 5. Média do atraso suavizada nos consumidores com a topologia Abilene.

pacotes. Vale destacar que a política DSPPC pode ser ajustada com valores de $\beta > 1$ para aumentar a sensibilidade à frequência de acesso aos dados no cache, o que pode influenciar o desempenho em cenários como este. Os demais roteadores mantiveram taxas de acerto inferiores a 8 pps em todas as políticas avaliadas. De modo geral, com $\beta = 1$, a política DSPPC alcançou uma média global de acerto de cache de ≈ 5 pps, enquanto as políticas ARC-QoS e LRU registraram valores ligeiramente superiores, próximos de 6 pps, conforme mostrado na Figura 6(d).

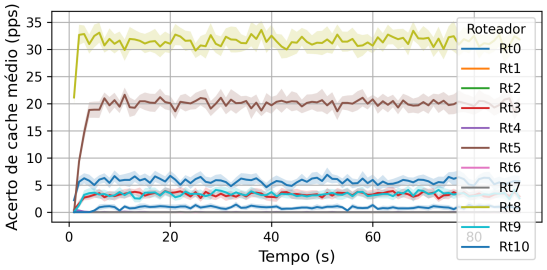
A Figura 7 apresenta a taxa de satisfação de interesses alcançada pelos consumidores no cenário complexo. Devido à menor concorrência por recursos de rede na topologia Abilene, as três políticas obtiveram taxas semelhantes, alinhadas à frequência de requisições de interesses enviadas pelos consumidores. As Figuras 7(a), 7(b) e 7(c) destacam as taxas de satisfação específicas para as políticas DSPPC, ARC-QoS e LRU, respectivamente. Essa similaridade de desempenho entre as políticas também é evidenciada na Figura 7(d), que ilustra a taxa global de satisfação de interesses dos consumidores no cenário avaliado.

5. Conclusão e Trabalhos Futuros

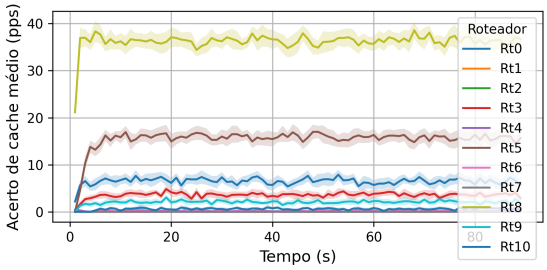
A arquitetura NDN se apresenta como uma solução promissora para a Internet do futuro, oferecendo benefícios significativos no modelo de comunicação orientado a dados. No entanto, seu tratamento de tráfego com base no melhor esforço revela a necessidade de mecanismos robustos para lidar com os desafios de QoS. Este trabalho propôs um mecanismo inovador para tratar QoS nas redes NDN, introduzindo o esquema de codificação da informação de QoS nos nomes dos pacotes, denominado NDiffServ (`nds`). Adicionalmente, foi desenvolvida a política de cache DSPPC, que utiliza a informação `nds` e a popularidade dos dados para otimizar o gerenciamento do espaço de armazenamento. A DSPPC aplica uma substituição probabilística de dados no cache, levando em considera-



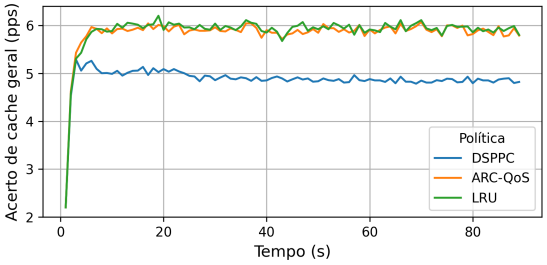
(a) Média de acerto de cache na política DSPPC.



(b) Média de acerto de cache na política ARC-QoS.

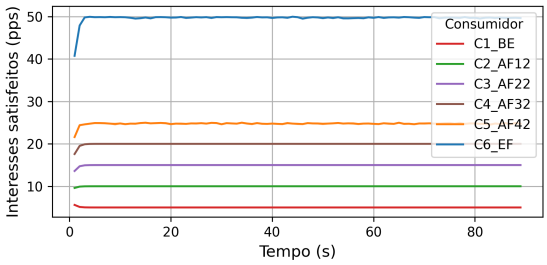


(c) Média de acerto de cache na política LRU.

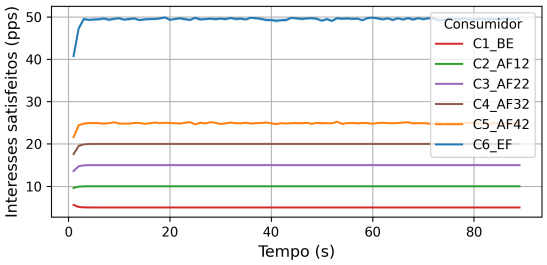


(d) Acerto de cache por política.

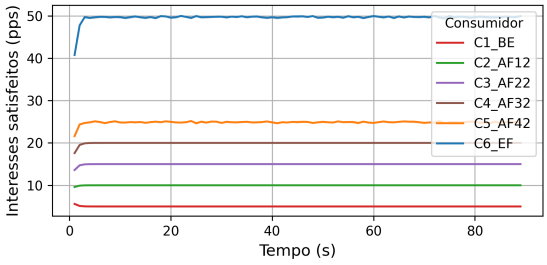
Figura 6. Taxa de acerto de cache nos roteadores com a topologia Abilene.



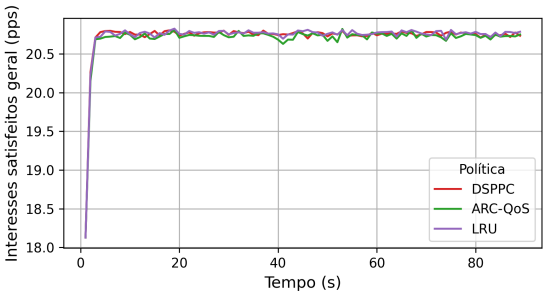
(a) Média de interesses satisfeitos com DSPPC.



(b) Média de interesses satisfeitos com ARC-QoS.



(c) Média de interesses satisfeitos com LRU.



(d) Média de interesses satisfeitos por política.

Figura 7. Interesses satisfeitos nos consumidores na topologia Abilene.

ção o peso associado à classe de serviço e a frequência de acesso dos dados. Os resultados obtidos evidenciam o potencial da proposta, demonstrando ganhos no desempenho da rede em relação às políticas comparadas. Como trabalho futuro, planeja-se investigar o impacto da variação do parâmetro β na política DSPPC para ajustar sua sensibilidade à frequência de acesso dos dados, além de desenvolver um módulo de gerenciamento de encaminhamento, baseado na solução proposta em [Araújo et al. 2021], a fim de ampliar os benefícios do esquema `nds` e da política DSPPC.

Referências

Abdelaal, M., Karadeniz, M., Dürr, F., and Rothermel, K. (2020). litendn: Qos-aware packet forwarding and caching for named data networks. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–9.

Ambalavanan, U., Nayak, N., Grewe, D., and Mohan, N. (2022). Resource reservation in information centric networking. In *Proceedings of the 9th ACM Conference on Information-Centric Networking*, ICN '22, page 165–167, New York, NY, USA. ACM.

Araújo, F. R. C., Madureira, A. L. R., and Sampaio, L. N. (2021). Balanceamento de carga dinâmico, distribuído e centrado no conteúdo para redes de dados nomeados. In *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 99–112, Porto Alegre, RS, Brasil. SBC.

Araújo, F. R. C., Madureira, A. L. R., and Sampaio, L. N. (2023). A multicriteria-based forwarding strategy for interest flooding mitigation on named data wireless networking. *IEEE Transactions on Mobile Computing*, 22(12):7000–7013.

Dhakal, D., Dey, S., Dey, P., Rathor, M., and Sharma, K. (2023). Investigating the impact of barrier adjustment and priority caching on improving quality of service in named data vehicular sensor networks. In *2023 4th International Conference on Computing and Communication Systems (I3CS)*, pages 1–8.

Gündoğan, C., Pfender, J., Kietzmann, P., Schmidt, T. C., and Wählisch, M. (2020). On the impact of qos management in an information-centric internet of things. *Computer Communications*, 154:160–172.

Jangam, A., Suthar, P., and Stolic, M. (2020). QoS Treatments in ICN using Disaggregated Name Components. Internet-Draft icnrg-dnc-qos-icn-02, IETF. Work in Progress.

Kumar, S., Tiwari, R., and Hong, W.-C. (2021). Qos improvement using in-network caching based on clustering and popularity heuristics in ccn. *Sensors*, 21(21).

Mastorakis, S., Afanasyev, A., and Zhang, L. (2017). On the Evolution of ndnSIM: An Open-Source Simulator for NDN Experimentation. *ACM SIGCOMM Computer Communication Review*, 47(3):19–33.

Moiseenko, I. and Oran, D. R. (2021). Flow Classification in Information Centric Networking. Internet-Draft icnrg-flowclass-07, IETF. Work in Progress.

Schneider, K., Zhang, B., and Benmohamed, L. (2020). Hop-by-hop multipath routing: Choosing the right nexthop set. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 2273–2282.

Singh, P. and Sarma, N. (2021). Adaptive replacement cache with quality of service for delay sensitive applications in named data networking. In *2021 IEEE 18th India Council International Conference (INDICON)*, pages 1–6.

Team, N. P. (2021). Ndn technical memo: Naming conventions. Technical Report. NDN-0022, revision 3, NDN.

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.