UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

ALLAN CARLOS FIGUEIREDO ECHEVERRIA

ANÁLISE COMPARATIVA DE ALGORITMOS DE RENOVAÇÃO DE CACHE EM REDES VEICULARES COM ESTRUTURA NDN

Joinville
2023

ALLAN CARLOS FIGUEIREDO ECHEVERRIA

ANÁLISE COMPARATIVA DE ALGORITMOS DE RENOVAÇÃO DE CACHE EM REDES VEICULARES COM ESTRUTURA NDN

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Prof. Dr. Ricardo Jose Pfitscher

Joinville
2023

Dedico este trabalho à minha família

AGRADECIMENTOS

Agradeço ao Prof. Dr. Ricardo José Pfitscher por me auxiliar durante o desenvolvimento do projeto e por saber escolher exatamente qual abordagem eu precisava em cada momento.

Agradeço à instituição da Universidade Federal de Santa Catarina, em específico ao Centro Tecnológico de Joinville e ao corpo docente do curso de Engenharia Mecatrônica.

Agradeço também ao Prof. Guilherme Braga Araújo, ao Prof. Francisco Renato Cavalcante Araújo e à Ma. Bruna Arruda Araujo por me auxiliarem em dúvidas que tive ao longo do caminho e fornecerem informações que foram fundamentais para o desenvolvimento do projeto.

Agradeço aos meus amigos por terem participado de toda a minha jornada pela graduação, e, principalmente, por terem me ouvido falar por horas sobre o desenvolvimento do meu TCC mesmo sem entender direito qual era o tema.

RESUMO

A atual arquitetura da internet é produto de um contexto de compartilhamento de recursos computacionais. Com a democratização do acesso à tecnologia, a internet hoje tem como principal função prover informações. Uma das estruturas de rede afetada por essa mudança de foco, devido à alta dinamicidade, é a estrutura de Redes Veiculares Ad Hoc (VANET). Uma solução proposta para esse problema é a Content-Centric Networking (CCN), que teve rápida adoção no contexto de VANETs por meio do Named Data Networking (NDN). Entretanto, alguns problemas persistem nessa nova proposta de solução, a exemplo da ineficiência na distribuição de conteúdo de forma homogênea ao longo da rede. Este estudo explora a análise e comparação de desempenho entre diversos algoritmos de caching em VANETs baseadas na arquitetura NDN. As análises realizadas indicam que a implementação do algoritmo de decisão CEE em conjunto aos algoritmos de decisão LRU ou MDMR pode trazer melhores desempenhos em contextos de alta criticidade dos dados em relação aos outros conjuntos de algoritmos analisados, enquanto o algoritmo de decisão pCASTING demonstra maior eficácia em situações que envolvem considerável diversidade de conteúdo na rede e uma elevada frequência de envio de interesses. Entretanto, a última suposição carece de confirmação em estudos futuros.

Palavras-chave: CCN; NDN; VANET; política de cache.

ABSTRACT

The current architecture of the internet is the product of a context of sharing computational resources. With the democratization of technology access, the internet's primary function today is to provide information. One of the network structures affected by this change of focus, due to high dynamism, is the structure of Vehicular Ad Hoc Networks (VANETs). A proposed solution to this issue is Content-Centric Networking (CCN), which quickly gained adoption in the context of VANETs through Named Data Networking (NDN). However, some issues persist in this new proposed solution, such as the inefficiency in distributing content uniformly across the network. This study explores the analysis and performance comparison of various caching algorithms in NDN-based VANETs. The analyses conducted suggest that implementing the CEE decision algorithm in conjunction with the LRU or MDMR decision algorithms may yield better performance in high data criticality contexts compared to other sets of analyzed algorithms. Meanwhile, the pCASTING decision algorithm demonstrates greater effectiveness in scenarios involving significant content diversity in the network and a high frequency of interest transmissions. However, the latter assumption requires confirmation in future studies.

Keywords: CCN; NDN; VANET; caching policy.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com a democratização do acesso aos recursos computacionais, a integração destes recursos em tarefas cotidianas tornou-se cada vez mais comum, sendo um dos adventos recentes a comercialização de veículos completamente autônomos. A introdução de veículos inteligentes trouxe à tona discussões sobre a melhoria da segurança nos sistemas rodoviários, e o uso de redes veiculares para propagação de informações (posição, velocidade, acidentes) passou a ser uma proposta viável para estabelecer a segurança pró ativa no trânsito (Popescu-Zeletin; Radusch; Rigani, 2010).

A estrutura das redes da forma como é conhecida foi criada do contexto do projeto Arpanet, financiado pela Agência de Projetos de Pesquisa Avançada do Departamento de Defesa dos Estados Unidos (ARPA) na década de 1960 que tinha como objetivo compartilhar recursos de processamento interativo (Press, 1992). Entretanto, a realidade do acesso a recursos computacionais não é a mesma da década de 1960, e o principal recurso que a Internet pode fornecer, atualmente, é o conteúdo dos servidores.

Na década de 1990 surge o conceito de Content Delivery Network (CDN), que consiste em uma rede de servidores geograficamente distribuída, com o intuito de otimizar o fornecimento de conteúdo a partir do caching descentralizado. Apesar de ser utilizado na maior parte do tráfego de internet atualmente, o CDN tem seu lado negativo na necessidade de alocar novos recursos computacionais, além do host original (Leighton; Lewin, 2000).

Com a proposta de eliminar a necessidades de adaptações ao atual Protocolo de Internet (IP), surge em 2009 a proposta de um novo paradigma para a Internet, o Content-Centric Networking (CCN) (Jacobson *et al.*, 2009). Nessa estrutura, o foco é a solicitação de conteúdos nomeados, sendo responsabilidade dos nós da rede a procura por um servidor ou nó que possua o conteúdo em menor alcance. O projeto iniciado por Jacobson foi continuado por meio de um financiamento do Internet Research Task Force, por um grupo de colaboração de 12 campis e, devido às necessidades específicas levantadas por esse grupo, o projeto passou a ser uma bifurcação do projeto inicial, sendo renomeado para Named Data Networking (NDN) (NAMED DATA NETWORKING, 2023).

Devido à característica dinâmica das Redes Veiculares Ad Hoc (VANET), o Protocolo de Controle de Transmissão (TCP), complemento do IP, não é a estrutura ideal para suportar a execução de aplicações veiculares, pois a pilha de protocolos foi idealizada com uma organização hierárquica, composta por entidades estáticas provedoras de recursos e que mantêm conexões fixas. Por outro lado, a característica

intrínseca da VANET é que suas entidades (os veículos) têm conexões dinâmicas, com infraestrutura de comunicação móvel, o que dificulta a aplicação da mesma arquitetura TCP/IP (Wang *et al.*, 2012).

A implementação de uma CCN suprime esses problemas apresentados, todavia a implementação direta de NDN nessas redes traz alguns problemas, como o fato de não ter sido idealizada em um cenário onde todos os nós da rede estão se movendo, como em uma VANET, embora a arquitetura apresente suporte para usuários utilizando redes de dados móveis. Além disso, a distribuição de cópias em cache ao longos dos nós da rede é um desafio para a melhoria do desempenho das redes do futuro (Bian *et al.*, 2015).

Nesse contexto, propostas como a de Wang *et al.* (2012), que utiliza a estratégia de um algoritmo de decisão baseado em temporizadores para definir o encaminhamento de um pacote nomeado enquanto Chen *et al.* (2014) e Saltarin *et al.* (2019) propuseram soluções de política de caching baseadas em popularidade, buscam conceitos que não são utilizados em algoritmos convencionais para solucionar o problema apresentado.

Khelifi *et al.* (2020) sintetizaram o estado da arte do uso de NDN em redes veiculares, levando em consideração aspectos do cenário da época. Entretanto, alguns algoritmos promissores como o de Hail *et al.* (2015) e Hahm *et al.* (2017) não foram considerados, deixando uma lacuna a ser preenchida no âmbito de comparação dos algoritmos de gerenciamento de caching em NDN para redes veiculares.

Considerando isso, este trabalho apresenta uma análise comparativa dos algoritmos de gerenciamento de cache populares atualmente, como também de algoritmos promissores ainda não avaliados na literatura disponível, com o fim de estabelecer uma compreensão do desempenho dos algoritmos em diferentes cenários, estabelecendo o melhor desempenho entre estes.

A análise dos algoritmos é realizada a partir do uso de simuladores de redes veiculares, em um cenário comum de consumo de dados, com variação dos algoritmos selecionados em cada teste. Os índices de performance avaliados foram a taxa de cache hit, atraso médio na entrega de dados, taxa de retransmissão de interesse, a taxa total de despejo de cache, a carga no servidor e a média de saltos usptream por interesse transmitido.

1.1 OBJETIVOS

Para resolver a problemática da distribuição homogênea de conteúdo em redes VANET, com aplicação de NDN, propõe-se neste trabalho os seguintes objetivos.

1.1.1 Objetivo Geral

Realizar a análise comparativa dos algoritmos para gerenciamento de cache NDN em redes veiculares.

1.1.2 Objetivos Específicos

- Estabelecer um ambiente de VANET com NDN;
- Investigar mecanismos de gerenciamento de cache;
- Definir métricas de comparação de algoritmos;
- Avaliar comparativamente os mecanismos;
- Propor melhorias nos mecanismos de gerenciamento de cache.

2 FUNDAMENTAÇÃO TEÓRICA

Para atingir o objetivo de analisar algoritmos de gerenciamento de cache quanto à sua adequação ao uso em redes NDN veiculares, é necessário estabelecer algumas bases de conhecimento. Assim, neste capítulo serão apresentados os principais conceitos referentes as redes veiculares (Seção 2.1), redes de dados nomeados (Seção 2.2), algoritmos de decisão de cache (Seção 2.3), algoritmos de substituição de cache (Seção 2.4) e ferramentas para simulação das redes (Seção 2.5).

2.1 VANET

Assim como os semáforos foram uma tecnologia necessária para controle do tráfego no início da popularização dos veículos motorizados, as Redes Veiculares Ad Hoc (VANET) surgiram com o objetivo de amenizar o crescente trânsito e diminuir o número de acidentes em malhas viárias, a partir do compartilhamento de informações (Popescu-Zeletin; Radusch; Rigani, 2010).

As redes veiculares podem realizar comunicações por dois meios, a comunicação de forma direta entre veículos, conhecida como Vehicle-to-Vehicle (V2V), e a comunicação entre os veículos e os dispositivos atrelados à infraestrutura da rodovia, conhecida como Vehicle-to-Infrastructure (V2I) (Dua; Kumar; Bawa, 2014). A nomenclatura pode variar conforme a literatura, sendo também mencionada como Intervehicle Communication (IVC) e Roadside-to-Vehicle Communication (RVC) por Paul *et al.* (2017). Popescu-Zeletin et al. (2010) também utilizam o termo Vehicle-to-Roadside (V2R) como sinônimo da comunicação V2I.

Na prática, ambos os tipos de comunicação acontecem a todo instante, com os dispositivos de infraestrutura servindo como centralizadores da informação e responsáveis pelo roteamento adequado, caso o veículo destino da informação não esteja alcançável. A comunicação V2V age em situações críticas do tráfego, como possíveis colisões, transição de faixas, prevenção de direção no sentido oposto da via (Popescu-Zeletin; Radusch; Rigani, 2010). A ação híbrida resultante do funcionamento simultâneo das redes V2V e V2I é denominada Vehicle-to-Everything (V2X) (Yao *et al.*, 2018).

Devido à evolução da tecnologia e da forma de uso da internet, a estrutura das redes veiculares teve de expandir suas capacidades para que seja capaz de suportar transporte de streaming de áudio e vídeo, video chamadas e outros tipos de entretenimento. Para atingir esse objetivo foram necessárias adaptações em questões como otimização de armazenamento, alocação de banda e agendamento de envio de pacotes (Alaya *et al.*, 2021).

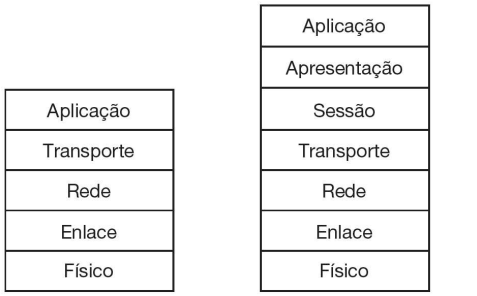
As VANETs possuem características que as diferem das demais Mobile Ad-Hoc

Networks (MANETs), como a previsibilidade da movimentação de seus nós, a ausência de limitações relacionadas ao fornecimento de energia para o sistema e variações rápidas na topologia da rede (Al-Sultan *et al.*, 2014). Essas especificidades levam a restrições na idealização de uma rede veicular, como a necessidade do alcance de um single hop em longas distâncias, alta taxa de entrega de pacotes, baixa latência e alta frequência de atualização dos dados (Popescu-Zeletin; Radusch; Rigani, 2010).

2.2 REDES DE DADOS NOMEADOS

Em 1994 a Organização Internacional para Padronização (ISO) publicou uma atualização da norma ISO 7498-1 com o objetivo de estabelecer um padrão para os protocolos de comunicação de rede. Essa norma definiu uma estrutura de sete camadas (Figura 1), com tarefas específicas para cada nível de abstração, possibilitando a interoperabilidade entre diferentes protocolos (Kurose; Ross, 2013).

Figura 1 – Modelo simplificado e modelo completo da estrutura de camadas OSI



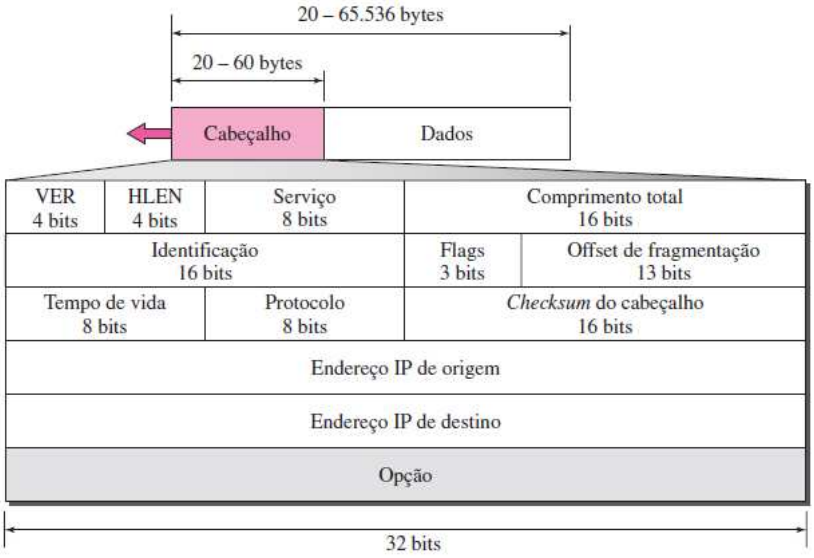
a. Pilha de protocolos da Internet de cinco camadas

b. Modelo de referência ISO de sete camadas

Fonte: Kurose (2013, p. 38).

A camada de Redes é responsável pela transmissão dos datagramas, mensagens com estrutura dividida em duas partes: um cabeçalho, com as informações essenciais para que a informação seja entregue ao destino, e uma seção com os dados em si. Atualmente a Internet tem como protocolo de rede o IP, cuja estrutura do datagrama pode ser vista na Figura 2 (Forouzan; Griesi; Fegan, 2010).

Figura 2 – Formato do datagrama no IPv4



Fonte: Forouzan et al. (2010, p. 583).

A estrutura do protocolo IP foi idealizada em um período no qual o foco das comunicações era o de compartilhamento de recursos, e ao longo do tempo foi adaptado para ser compatível com a atual estrutura de busca por conteúdo. Dessa forma, atualmente os conteúdos são transmitidos na internet pelos datagramas, que tem como princípio a conexão entre um endereço de origem e um endereço de destino (Forouzan; Griesi; Fegan, 2010).

Entretanto, essa adaptação traz alguns problemas, como a baixa segurança nas comunicações, a alta dependência de localização geográfica e a necessidade de mecanismos auxiliares para contornar esses problemas. Com o objetivo de resolver esses problemas do protocolo IP, Jacobson *et al.* (2009) propôs a arquitetura de Rede Centrada em Conteúdo (CCN), substituindo a busca pelo endereço de destino pela busca de um arquivo nomeado.

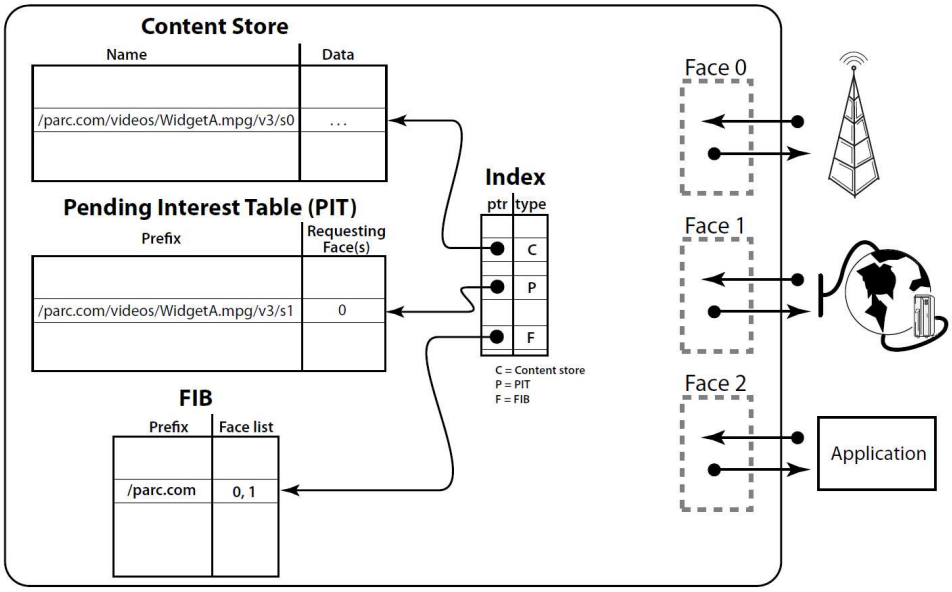
A proposta do protocolo, posteriormente nomeado NDN, é de manter a simplicidade e a baixa sobrecarga da camada de enlace que o protocolo IP têm. Ao mesmo tempo, busca trazer melhorias, como o máximo proveito de conexões simultâneas (Ethernet, Bluetooth, 3G e 802.11), dinâmicas de otimização na camada de estratégia, e segurança do conteúdo em si, evitando as vulnerabilidades que a proteção à conexão traz no protocolo IP (Jacobson *et al.*, 2009).

O NDN tem como base dois tipos de pacote, pacotes de Interesses e pacotes de Dados. Um nó consumidor que deseja receber um conteúdo transmite o pacote de Interesse a todas as conexões disponíveis. Caso algum nó possua a informação que satisfaz a requisição, responde com o pacote de Dados, consumindo o pacote de Interesse (Jacobson *et al.*, 2009).

Para gerenciar a transmissão de pacotes, um nó NDN tem três estruturas

básicas (Figura 3): Forwarding Information Base (FIB), Pending Interest Table (PIT) e Content Store (CS). A FIB é responsável por determinar potenciais fontes que satisfaçam um interesse, permitindo uma transmissão para múltiplas faces, ou seja, para múltiplas conexões estabelecidas por cada nó. A PIT é responsável por manter um controle dos pacotes de interesse transmitidos para que os dados possam ser retornados aos devidos solicitantes. A CS serve como um buffer de dados que, diferente do protocolo IP, utiliza políticas de caching que preservam um potencial conteúdo pelo maior tempo possível (Jacobson *et al.*, 2009).

Figura 3 – Estrutura de gerenciamento de encaminhamento de pacotes de um nó NDN



Fonte: Jacobson et al. (2009, p. 3).

2.2.1 NDN em redes veiculares

A estrutura do NDN apresenta desempenho significativamente superior à pilha TCP/IP na transmissão de conteúdos como áudio, vídeo e dados em tempo remoto. Devido à crescente demanda de consumo de conteúdo em VANETs, trabalhos como Bian *et al.* (2015), Chen *et al.* (2014), Khelifi *et al.* (2020) e Wang *et al.* (2012) têm considerado a adoção do protocolo NDN como a melhor alternativa para o desenvolvimento escalável dessas redes.

Porém, a implementação do protocolo NDN nas VANETs traz alguns desafios. Em seu trabalho, Wang *et al.* (2012) propõem uma solução para dois grandes problemas, a alta dependência de conexões 3G, que nem sempre são gratuitas, e a possibilidade de longos atrasos, ocasionados por colisões de dados, na transmissão de informações críticas pela internet. Como proposta de solução do problema, Wang *et al.* (2012) apresentam uma camada NDN responsável pelas regras de negócio da transmissão

de informações, que utilizam um conjunto de temporizadores para minimizar colisões, compensando as limitações da transmissão sem fio.

Chen *et al.* (2014) sugerem a utilização de roteamento reativo, em que não haveria uma FIB estática, além da minimização da contagem de saltos entre as Roadsite Units (RSU) e o veículo solicitante de dados. Essa proposta seria capaz de reduzir a sobrecarga das unidades de infraestrutura, que, na pilha TCP/IP, centralizam o controle de requisições.

Devido à natureza dinâmica das VANETs, em que todos os nós estão em movimento constante, não é possível contar com tabelas de encaminhamento estáticas, resultando na necessidade frequente de recorrer à técnica de flooding, que consiste em transmitir os dados para todos os dispositivos alcançáveis. No entanto, essa abordagem resulta em uma alta sobrecarga nos nós da rede devido à transmissão indiscriminada dos pacotes (Bian *et al.*, 2015).

Como proposta para solução desse problema, Bian *et al.* (2015) sugerem uma estratégia de encaminhamento de dados baseada em geolocalização. A partir da inserção de características de posição geográfica inseridas no nome do pacote e dos pacotes de interesse padronizados, os veículos seriam capazes de gerar uma FIB dinâmica com baixa sobrecarga e utilizar a informação de posição para tomar a decisão do encaminhamento do pacote.

Além do problema de encaminhamento, no NDN os nós alocam cópias de todos os dados recebidos por padrão, resultando em armazenamento redundante, indesejado por diminuir o desempenho em casos de urgência da rede. Bian *et al.* (2015) são responsáveis por elaborar uma das primeiras propostas relevantes para solução dos problemas de caching.

Para solucionar esse problema é sugerida a combinação de três estratégias heurísticas na estrutura de caching das redes NDN. A primeira é a de caching aleatório, onde um determinado nó será atribuído como responsável por alocar o conteúdo em cache de forma probabilística. A estratégia de consciência de densidade é responsável por diminuir a probabilidade se o veículo estiver em um local muito povoado. Por fim, é proposta a estratégia de centralização do encaminhamento, que aproveita a proposta de encaminhamento baseado em geolocalização para garantir que os nós selecionados estejam espaçados geograficamente (Bian *et al.*, 2015).

No trabalho de Khelifi *et al.* (2020) são apresentados os esquemas populares na decisão de caching intra rede, que são os esquemas probabilísticos, esquemas cooperativos, esquemas baseado em popularidade e esquemas baseados em geolocalização.

2.3 ALGORITMOS DE DECISÃO DE CACHE

Caching é um método para armazenar, de forma temporária, um conjunto de informações na memória interna de um dos nós da rede, com o objetivo de agilizar as futuras requisições que sejam encaminhadas para este nó (Forouzan; Griesi; Fegan, 2010). O cache transforma um nó em um cliente e servidor ao mesmo tempo, característica essencial para o funcionamento das redes veiculares de forma dinâmica (Leira; Luís; Sargento, 2022).

Um esquema de cache consiste em duas partes, uma política de decisão de caching e uma política de substituição de caching. A política de decisão de caching tem como função definir a probabilidade de armazenamento de um determinado conteúdo (Zhang *et al.*, 2020).

Por sua vez, a política de substituição de caching também pode ser dividida em duas partes, a política de substituição e a política de inserção. A substituição tem como responsabilidade designar o conteúdo a ser despejado para que haja espaço para o caching do conteúdo selecionado. Ao haver espaço suficiente, a política de inserção define a posição em que o novo conteúdo será inserido (Qureshi *et al.*, 2007).

Nesta seção serão expostos diversos algoritmos de decisão de caching, proporcionando uma abordagem abrangente e elucidativa sobre as estratégias adotadas para otimizar o armazenamento temporário de conteúdos em nós de rede.

2.3.1 CEE

A política de decisão Caching Everything Everywhere (CEE), também conhecida como Leave Copy Everywhere (LCE), é a política aplicada de forma mais comum em conjunto com os demais algoritmos de substituição devido a sua simplicidade na implementação. Em redes com alta demanda é uma política que pode levar ao *trashing*, armazenando apenas os dados mais recentes, resultando em alta redundância em um contexto local. Por outro lado, é a política com alta efetividade na proliferação de novos conteúdos (Pfender; Valera; Seah, 2018).

2.3.2 Cache Probabilístico

Probabilistic Cache, também referido como Prob(p) ou simplesmente Prob é um algoritmo que adota como princípio de funcionamento uma abordagem randômica, assemelhando-se, em essência, a uma versão aleatória do CEE (Laoutaris; Syntila; Stavrakakis, 2004).

Em toda a extensão da rede, cada nó é considerado elegível para alocar uma cópia do conteúdo em cache. Contudo, a decisão de alocação é tomada de maneira aleatória a cada iteração, seguindo uma distribuição de probabilidade homogênea p que abrange toda a rede. Durante cada iteração do algoritmo, um valor de probabilidade

é gerado de forma aleatória. Se esse valor for inferior a *p*, o conteúdo é alocado no nó, caso contrário, a alocação não ocorre (Laoutaris; Syntila; Stavrakakis, 2004).

Este algoritmo destaca-se em relação ao CEE por sua redução da redundância de dados na rede, além de promover um aumento significativo na diversidade de dados circulando pela rede, fomentando uma distribuição mais equitativa e abrangente de conteúdo entre os nós (Zhang *et al.*, 2020).

2.3.3 pCASTING

O *probabilistic CAching STrategy for the INternet of thinGs* (pCASTING) é uma derivação do Prob(p) que propõe de uma política de decisão que visa considerar as características específicas de redes CCN levando em consideração o wireless como principal meio de transferência de dados. Essa estratégia leva em consideração a atualidade dos dados, o nível de energia e a ocupação do cache do dispositivo para definir de forma dinâmica a probabilidade de caching de cada informação (Hail *et al.*, 2015).

Para o cálculo da atualidade residual normalizada (FR) dos dados é levado em consideração o tempo atual ($currentTime$), o timestamp de produção da informação (t_s) e a atualidade indicada do dado (f), que representa o período de tempo em que o dado pode ser considerado válido. A expressão que representa FR é indicada na Equação 1.

$$FR = 1 - \frac{currentTime - t_s}{f} \tag{1}$$

Já os atributos do dispositivo de Internet das Coisas (IoT) são calculados de forma mais simples, a partir da normalização do respectivo índice, com valores de ocupação do cache (OC) e nível de energia (EN) com valores de ponto flutuante entre 0 e 1.

Para a definição dinâmica da probabilidade de caching, é definido o cálculo da Função de Utilidade do cache (F_u), que leva em consideração simultaneamente FR, OC e EN. Além desses valores é utilizado um índice $n \geq 1$, que representa o impacto dos índices na probabilidade de caching do dado, e a distribuição de pesos para cada índice, representados por w_i. A definição da função está expressa na Equação 2.

$$F_u = w_1 \cdot EN^n + w_2 \cdot (1 - OC)^n + w_3 \cdot FR^n \tag{2}$$

Considerando os valores de F_u sempre contidos no domínio $(-\infty, 1]$, o comportamento do pCASTING pode ser visualizado no algoritmo da Figura 4.

1: **procedure** HANDLE_DATA(Data)
2: $FR \leftarrow$ calculate_freshness(Data)
3: $F_u \leftarrow w_1 \cdot EN^n + w_2 \cdot (1 - OC)^n + w_3 \cdot FR^n$
4: **if** rand() $\leq F_u$ **then**
5: cache(Data)
6: **end if**
7: forward(Data)
8: **end procedure**

Figura 4 – Algoritmo pCASTING

2.4 ALGORITMOS DE SUBSTITUIÇÃO DE CACHE

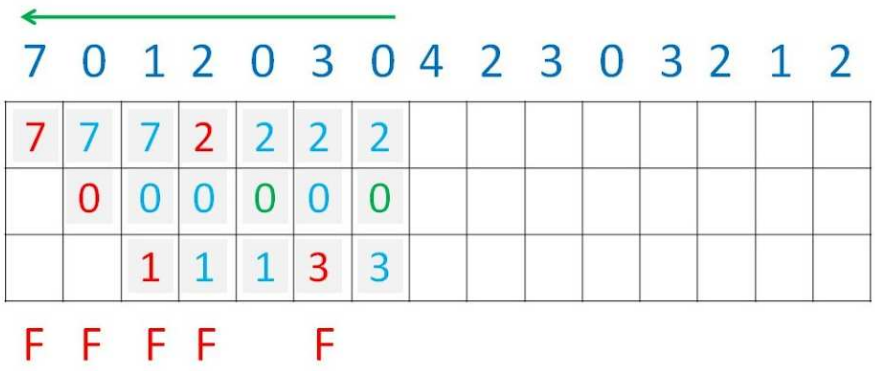
Essa seção apresenta o funcionamento dos principais algoritmos de substituição de cache.

2.4.1 Least Recent Used

O algoritmo Least Recent Used (LRU) é um algoritmo que, ao identificar que uma nova informação deve ser alocada, toma a decisão de qual conteúdo remover de seu armazenamento com base no último acesso de um conteúdo. O algoritmo foi elaborado para funcionar em situações de instruções lógicas, não desempenhando bem no gerenciamento de dados (O'Neil; O'Neil; Weikum, 1993).

Na Figura 5 é demonstrado o comportamento da execução de um cache com implementação do algoritmo LRU. Nas três primeiras solicitações, ao não encontrar a informação armazenada e havendo espaço, a informação é salva. A partir da quarta solicitação, do número dois, não há mais espaço disponível e a informação não está no cache, assim, o algoritmo apaga o número sete, que foi solicitado apenas na interação 1 e armazena a informação no novo espaço vago.

Figura 5 – Exemplo de processamento do algoritmo LRU



Fonte: Tech Academy (2015b).

2.4.2 Random Replacement

O algoritmo Random Replacement (RR) é uma estratégia de gerenciamento de dados que opera de forma aleatória em relação à escolha de quais informações remover em sua estrutura de armazenamento. É o algoritmo com funcionamento mais simples, não mantendo nenhum tipo de informação sobre o conteúdo armazenado e utilizando menor quantidade de recursos do sistema (Tarnoi *et al.*, 2014).

Esse algoritmo é usado de maneira mais frequente em um contexto acadêmico, como parâmetro comparativo na análise de desempenho de outros algoritmos (Pfender; Valera; Seah, 2018).

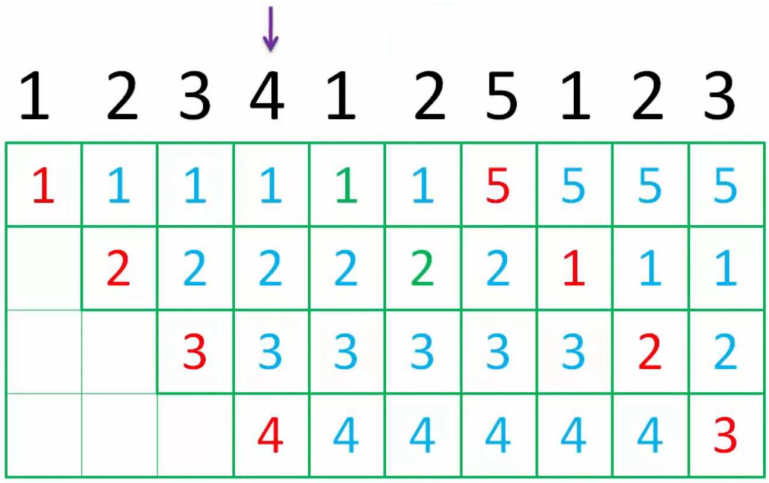
2.4.3 Priority First-In-First-Out

Priority Queue é um conceito solução para situações onde a ordem de execução de tarefas não está necessariamente atrelada à ordem de chegada das informações ou pedidos de execução. Nesses casos, um sistema de prioridade é estabelecido para que as tarefas de maior prioridade tenham menor tempo de espera, aumentando o tempo de espera de tarefas de baixa prioridade em contrapartida (Cobham, 1954).

No contexto de algoritmos de cache, são definidas operações padrão para o gerenciamento dos dados Insert, Delete e DeleteMin, sendo que a função DeleteMin executa a remoção da informação de menor prioridade em uma fila (Larkin; Sen; Tarjan, 2014).

Filas prioritárias são uma adaptação do algoritmo First-In-First-Out (FIFO), que opera com base no princípio de tratar as informações recebidas na ordem em que foram originalmente recebidas (Kurose; Ross, 2013). Um exemplo de como o algoritmo FIFO funciona é ilustrado na Figura 6.

Figura 6 – Exemplo de processamento do algoritmo FIFO



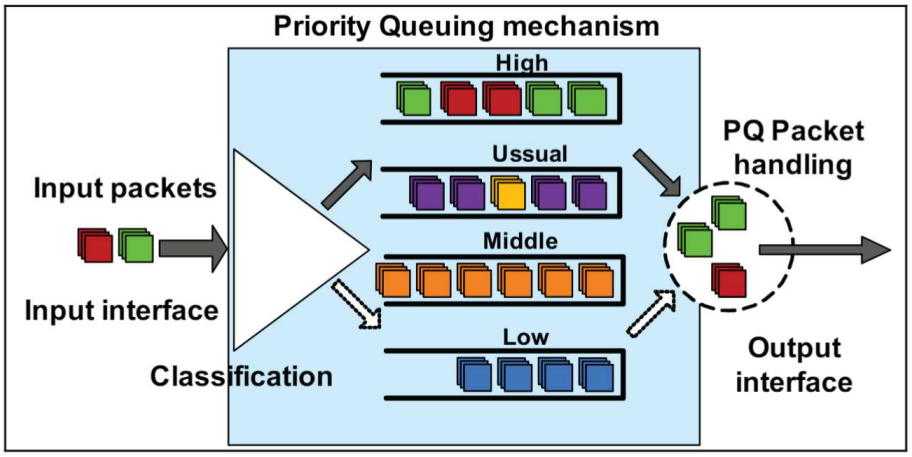
Fonte: Tech Academy (2015a).

O Priority First-In-First-Out (pFIFO) é uma variação do FIFO que atribui

diferentes níveis de prioridade a cada pacote. Essa priorização pode ser realizada incorporando valores diretamente nos pacotes ou por meio de processamento no nó de rede. Os nós de rede, por sua vez, gerenciam várias filas de prioridade, tratando cada uma delas com base no princípio FIFO individualmente (Islam *et al.*, 2012).

No contexto de gerenciamento de cache, a implementação de filas prioritárias permite que informações com maior grau de prioridade sejam mantidas em armazenamento enquanto houverem possíveis despejos de informações com menor prioridade. Na Figura 7 é exemplificado o funcionamento de um algoritmo pFIFO com múltiplas filas.

Figura 7 – Exemplo de processamento do algoritmo pFIFO



Fonte: Klampfer *et al.* (2011, p. 96)

A introdução de novos dados traz consigo informações cruciais, como a atualidade dos dados, o produtor do conteúdo e se o conteúdo foi requisitado e esses são utilizados para seleção da fila de cache do novo conteúdo. No âmbito da substituição de cache, filas de menor prioridade são selecionadas para despejar todo seu conteúdo inicialmente, e filas de maior prioridade são despejadas apenas após o esvaziamento completo de filas de menor prioridade. Entretanto os dados inseridos não necessariamente têm prioridade fixa e podem ser configurados temporizadores para revalidação da prioridade do conteúdo.

2.4.4 MDMR

Considerando os desafios de redes de internet das coisas e redes centralizadas em conteúdo, a estratégia de cache Max Diversity Most Recent (MDMR) foi proposta. Essa estratégia tem como objetivo manter alta disponibilidade de informação ao mesmo tempo que aumenta a economia energética do sistema em que é implementado (Hahm *et al.*, 2017).

O algoritmo assume que seja possível identificar o produtor e o timestamp do

conteúdo por meio do seu nome. A partir disso, primeiramente é realizada a tentativa de despejo de um antigo conteúdo do mesmo produtor, se não for possível, tenta despejar o conteúdo mais antigo do produtor com maior número de informações armazenadas, se nenhuma das alternativas for possível, o conteúdo mais antigo é removido (Hahm *et al.*, 2017). O comportamento do MDMR é definido pelo algoritmo apresentado na Figura 8.

```
1:  procedure ON_INCOMING_DATA(Data)
2:      Queue.attach(Data)
3:      forward(Data)
4:      while Queue.size > Cache.maxSize do
5:          if Queue.producers_list contains Data.producer then
6:              Queue.evict_from_producer(Data.producer)
7:          else if Queue.find_bigger_queue ≠ ERROR then
8:              Queue.evict_from_producer(Queue.find_bigger_queue)
9:          else
10:             Queue.evict_oldest_data()
11:         end if
12:     end while
13: end procedure
```

Figura 8 – Algoritmo MDMR

Quando um novo pacote de dados é recebido e o tamanho máximo do cache é atingido, o MDMR inicia o processo de seleção de conteúdo para despejo. Inicialmente, verifica-se se há algum dado no cache relacionado ao conteúdo a ser inserido. Em caso afirmativo, o conteúdo na primeira posição da fila do produtor é removido. Se não houver correspondência, o produtor com a maior quantidade de dados alocados em cache é selecionado, e um conteúdo dessa fila é despejado. Em último caso, quando não há mais de um produtor com o número máximo identificado de dados em cache, o conteúdo mais antigo é despejado.

2.5 SIMULADORES

Nesta seção do capítulo de fundamentação teórica, este trabalho direciona o enfoque para a apresentação dos simuladores que desempenham papel central na avaliação e validação dos algoritmos de gerenciamento de cache propostos. Serão apresentados e detalhados os simuladores selecionados para a condução das avaliações, destacando suas características essenciais e a relevância de sua aplicação no âmbito deste estudo.

2.5.1 ndnSIM

O NDN surgiu como a proposta de ser um substituto viável para o TCP/IP e, como consequência, diferentes propostas de design para o protocolo precisam ser validadas em larga escala, entretanto, é inviável realizar experimentos com uma infraestrutura física completa em todos os casos. O objetivo da ferramenta ndnSIM é oferecer uma ferramenta comum para simulações de estruturas NDN de forma simples, amigável ao usuário e código aberto (Mastorakis *et al.*, 2016).

O simulador implementa as principais funcionalidades e estruturas do NDN, como CS, PIT e FIB, além de regras de encaminhamento, processamento de dados e gestão de interfaces. A implementação das features se dá com base no simulador de redes open source NS-3, a biblioteca C++ ndn-cxx e o NDN Forwarding Daemon (NFD), sendo esses implementações do time oficial do NDN e aplicados ao contexto do simulador (Mastorakis *et al.*, 2016).

O NFD foi integrado no ndnSIM desde a atualização 2.0, sendo responsável por toda a implementação de encaminhamento de pacotes. Esse foi um passo importante no desenvolvimento do ndnSIM, já que, por ser a implementação oficial de encaminhamento do protocolo NDN, implementações usadas no simulador e em aplicações práticas são diretamente compatíveis, sem necessidades de adaptações.

O Quadro 1 apresenta os principais componentes utilizadas no ndnSIM e suas respectivas funcionalidades.

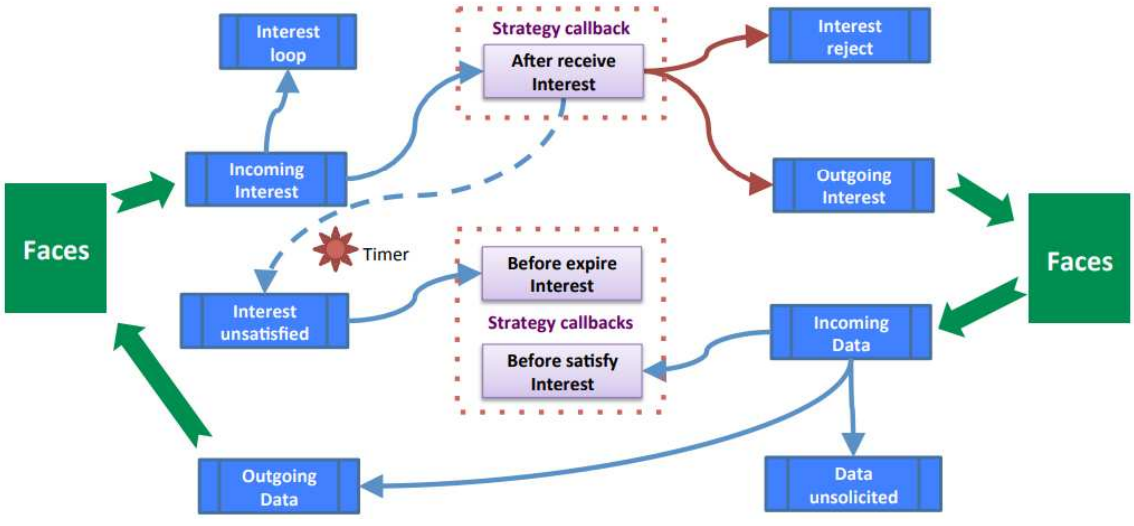
Quadro 1 – Principais componentes do ndnSIM

Componente	Funcionalidade
ndn::L3Protocol	Abstração da pilha NDN para o NS-3
nfd::Forwarder	Principal classe do NFD, armazena todas as faces e tabelas do nó NDN e implementa as pipelines de encaminhamento
nfd::Face	Abstração da face NFD que implementa as comunicações primitivas necessárias para enviar e receber pacotes de Interesse e de Dados
nfd::face::LinkService	Faz a tradução entre pacotes da camada de rede (Interesses, Dados e Nacks) e quadros da camada de enlace
nfd::face::Transport	Fornece serviço de entregas de pacotes best-effort para o LinkService de uma face
nfd::Cs	O cache de pacotes de Dados utilizados pelo NFD
nfd::Pit	Abstração da PIT do NFD
nfd::Fib	Abstração da FIB do NFD
nfd::fw:Strategy	A estratégia de encaminhamento decide se, quando e onde os pacotes de Interesse serão encaminhados. É uma classe abstrata que precisa ser implementada por estratégias customizadas ou pré-instaladas

Fonte: Elaborado pelo Autor

O Forwarder abstrai a tomada de decisões a respeito do encaminhamento de dados a partir da implementação de pipelines, e é implementado de forma que possa se adaptar a diferentes contextos sem necessidade da reconstrução de todo o pipeline para cada nova política. Uma visão geral do funcionamento do pipeline de encaminhamento pode ser visualizado na Figura 9.

Figura 9 – Visão geral do pipeline de encaminhamento do ndnSIM / NFD



Fonte: Mastorakis *et al.* (2016, p. 5)

Como pode ser visualizado na Figura 9, ao receber um novo Interesse, o callback ”After receive Interest” é acionado. Nesse callback são analisados a validade do pacote, se o pacote foi solicitado (a partir da PIT) e então invoca o método afterInsert para que a política de evicting realize o despejo de alguma informação caso seja necessário (Afanasyev *et al.*, 2021).

Após ser invocado, a política de decisão decide se o novo dado deve ser aceito ou não. Em caso positivo, o iterador portando a nova entrada deve ser inserido na lista, caso o contrário, a Content Store é acionada a partir do sinal beforeEvict, para que a entrada seja limpa da estrutura principal da CS (Afanasyev *et al.*, 2021).

Caso o dado seja aceito, a política invoca o método de inserção a partir do método insertToQueue, que adiciona o novo dado à estrutura virtual de controle de cache da política, e então aciona o método evictEntries, para que, caso seja necessário, a política selecione e despeje os dados para cumprir o requisito de alocação máxima de memória da Content Store (Afanasyev *et al.*, 2021).

O NFD tem implementado de forma nativa os algoritmos de evicting LRU e pFIFO, os quais foram explicados na Seção 2.3. Além dessas políticas, é possível implementar novas políticas com o uso de heranças da classe base Policy. Maiores informações sobre a implementação de novas políticas serão dispostas no Capítulo 3.

Entretanto, como o ndnSIM é baseado no NS-3, com foco em estruturas de rede padrão, com nós fixos, tem limitações em relação à simulação direta de VANET's. Para tal, é necessário de um simulador de sistemas veiculares que ofereça a dinâmica de uma estrutura malha viária.

2.5.2 SUMO

O Eclipse Simulation of Urban MObility (SUMO) é um simulador open source de tráfego contínuo em escala microscópica, o que significa que cada veículo e suas dinâmicas são modelados de forma individual em seu contexto de simulação. É um dos simuladores mais populares, sendo baixado mais de trinta e cinco mil vezes a cada ano (Lopez *et al.*, 2018).

Uma simulação de tráfego na ferramenta toma como base algus elementos, sendo os mais importantes para a construção de uma simulação os dados de malha como estradas, infraestrutura de tráfego como semáforos e a demanda de tráfego (Lopez *et al.*, 2018).

Como geralmente os cenários são usados para estudo de comportamento estocástico, é extremamente vantajoso analisar de forma visual o comportamento dos diferentes objetos de simulação. Para esse fim, o SUMO oferece o SUMO-GUI, aplicação que permite a visualização da simulação de forma gráfica.

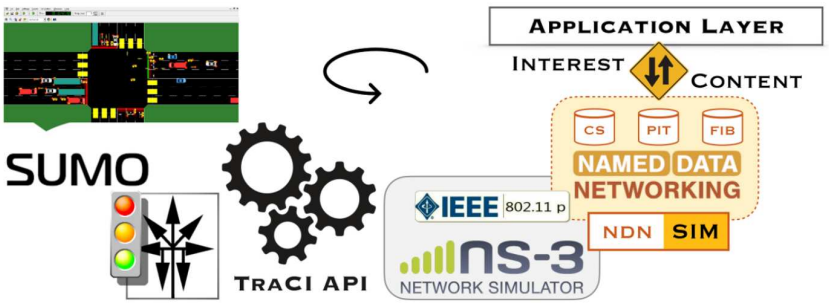
Utilizando o SUMO, é possível examinar com precisão o comportamento dos veículos em diferentes cenários, incluindo situações complexas como semáforos,

intersecções, acidentes que bloqueiam a via e congestionamento de rotas. Contudo, é importante destacar que o funcionamento do SUMO ocorre de forma independente do simulador de rede. Portanto, é indispensável o uso de uma ferramenta de integração capaz de estabelecer uma ligação eficaz entre essas duas plataformas distintas.

2.5.3 NDN4IVC

Para realizar a integração entre as ferramentas de simulação de redes e simulação de tráfego, a ferramenta NDN for Inter-Vehicle Communication (NDN4IVC) foi proposta. A partir da utilização da API TraCI, é possível realizar a simulação de cenários VANET a partir de um único contexto de simulação, como apresentado na Figura 10 (Araujo; Peixoto; Sampaio, 2023).

Figura 10 – Estrutura de conexão do NDN4IVC



Fonte: Araujo; Peixoto; Sampaio (2023, p. 4)

A integração entre os simuladores apresenta um desafio devido às diferenças na manipulação de objetos entre NS-3 e SUMO. Enquanto o NS-3 requer a existência prévia de todos os nós, o SUMO permite uma lógica dinâmica de exclusão de objetos. Para resolver isso, o NDN4IVC utiliza o conceito de *node pool*, permitindo a conexão e desconexão dinâmica dos nós de rede no NS-3, sincronizando com os eventos correspondentes no SUMO (Araujo; Peixoto; Sampaio, 2023).

A organização de diretórios do framework é esquematizada no Quadro 2. Contudo, na versão disponibilizada por meio da máquina virtual, a pasta `results` não se encontra presente (Araujo; Peixoto; Sampaio, 2023). O NDN4IVC é posicionado na pasta `contrib` dentro da estrutura do NS-3, reservada para códigos que ainda não foram incorporados à árvore principal do simulador de rede (UNIVERSITY OF WASHINGTON NS-3 CONSORTIUM,).

Quadro 2 – Estrutura de diretórios do NDN4IVC

Diretório	Descrição
`doc/`	Documentação adicional
`examples/`	Cenários de simulação do NS-3
`helper/`	Códigos de auxílio à configuração
`model/`	Aplicações e códigos de usuário
`traces/`	Dados de mobilidade e mapas do SUMO
`results/`	Arquivos de log, gráficos e dados processados

Fonte: Araujo, Peixoto e Sampaio (2023)

2.6 CONSIDERAÇÕES DO CAPÍTULO

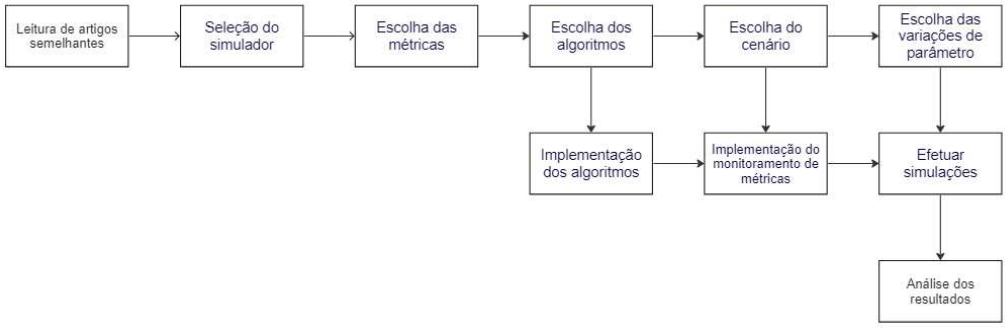
Este capítulo abordou os fundamentos teóricos essenciais para o desenvolvimento e compreensão do presente estudo. Inicialmente, foram apresentados os conceitos de VANETs e em seguida introduziu-se a arquitetura NDN e sua implementação específica em VANETs. Os algoritmos de decisão e substituição de cache utilizados para otimizar a gestão de conteúdo foram apresentados, além de detalhes sobre o simulador SUMO e o módulo ndnSIM, bem como a estrutura da ferramenta NDN4IVC.

Com este arcabouço teórico sólido, o próximo capítulo adentrará à metodologia, delineando os passos e abordagens adotados para investigar e comparar os algoritmos de gerenciamento de cache em cenários de VANETs baseadas em NDN.

3 MATERIAIS E MÉTODOS

Neste capítulo são descritos os métodos de avaliação empregados para analisar o desempenho dos algoritmos de cache mencionados no Capítulo 2, a partir da metodologia mostrada na Figura 11. Essa abordagem pode ser caracterizada como uma pesquisa experimental, submetendo o objeto de estudo a distintos tratamentos e avaliando estatisticamente a relevância das variações nas respostas (Fonseca, 2002).

Figura 11 – Sumário da metodologia de analise comparativa de algoritmos de cache



Fonte: Elaborado pelo autor

Os tópicos a seguir explicarão o desenvolvimento de cada uma das etapas da Figura 11.

3.1 TRABALHOS RELACIONADOS

Com o objetivo de embasar o desenvolvimento do trabalho, o início do projeto se deu a partir da procura de trabalhos com objetivos similares ao deste. A seleção de base de dados para busca foi realizada a partir da análise do índice Qualis de periódicos, sistema usado para classificar a produção científica dos programas de pós-graduação com base nas publicações em periódicos científicos (UFRN; CAPES, 2022).

O primeiro trabalho selecionado como referência foi o desenvolvido por Pfender, Valera e Seah (2018), *Performance Comparison of Caching Strategies for Information-Centric IoT*, com a análise de algoritmos de gerenciamento de cache no conceito de Redes Centradas em Informação (ICN). O foco do trabalho foram algoritmos de decisão e de evicting, considerando uma estrutura de fila para a inserção de novos dados.

Nos algoritmos de decisão, além do estudo do comportamento do CEE e do pCasting, os autores Pfender, Valera e Seah (2018) também analisaram o algoritmo probabilístico de probabilidade estática *Prob*(0.5), que tem um comportamento com características similares ao pCasting, porém, sem a existência de variáveis. A partir de uma probabilidade estática definida para a aplicação, um fator aleatório é gerado a

cada entrada e comparado com o limiar, que para a aplicação desse trabalho foi de 50%.

Para os algoritmos de despejo, nomeados no trabalho como algoritmos de substituição, Pfender, Valera e Seah (2018) escolheram três algoritmos. O LRU, por considerar ser o algoritmo mais utilizado em aplicações, o algoritmo Random, como parâmetro comparativo para o desempenho, e o MDMR, por ser um algoritmo desenvolvido especialmente para o trabalho com redes ICN.

As métricas selecionadas para avaliação de desempenho dos algoritmos foram carga do servidor, taxa de cache hit, atraso no recebimento dos dados, taxa de retransmissão de interesses, total de despejos de cache, métrica de diversidade e taxa de retenção de cache (Pfender; Valera; Seah, 2018).

A métrica de carga do servidor é calculada como a porcentagem de interesses que chegam até algum produtor de conteúdo ao invés de serem satisfeitos pelo cache de algum nó da rede, e pode ser expressa pela Equação 3. Um índice menor significa que o cache é mais eficiente em satisfazer os interesses da Rede.

$$\text{Carga no servidor} = \frac{\text{Interesses recebidos pelo servidor}}{\text{Quantidade total de Interesses}} \tag{3}$$

A métrica de taxa de cache hit representa a proporção de interesses que podem ser satisfeitos por uma cópia alocada em cache de algum dos nós da Rede e é representado pela Equação 4. A combinação de carga do servidor e taxa de cache hit são indicadores de quão bem distribuídos estão os conteúdos mais populares na rede local (Pfender; Valera; Seah, 2018).

$$\text{Taxa de Cache Hit} = \frac{\text{Cache hits}}{\text{Cache hits} + \text{Cache misses}} \tag{4}$$

O conjunto de métricas de carga no servidor e taxa de cache hit pode inicialmente parecer redundante, no entanto, a avaliação conjunta dessas métricas é essencial para compreender o impacto de outros parâmetros da rede, como a política de encaminhamento e o protocolo de comunicação adotados, no desempenho das políticas de cache.

O atraso no recebimento dos dados representa o tempo médio que um interesse leva para ser satisfeito, incluindo possíveis retransmissões (Hail *et al.*, 2015). O atraso pode ser afetado por vários fatores independentes ao cache, como densidade e congestionamento de rede, mas também é relacionado à diversidade de dados da rede

(Pfender; Valera; Seah, 2018). O cálculo do atraso no recebimento dos dados é descrito pela Equação 6.

$$\text{Soma dos Atrasos} = \sum_{i=1}^{\text{Interesses}} (\text{Timestamp do recebimento}(i) - \text{Timestamp da criação}(i)) \tag{5}$$

$$\text{Atraso no recebimento} = \frac{\text{Soma dos Atrasos}}{\text{Quantidade total de Interesses}} \tag{6}$$

A taxa de retransmissão de interesses mede a porcentagem de interesses que são retransmitidos por terem atingido o limite de tempo estabelecido. É uma medida diretamente ligada ao tempo de atraso no recebimento de dados, também sendo afetada pela densidade e congestionamento da rede, além da eficiência da estratégia de cache. A taxa de retransmissão pode ser expressa pela Equação 7.

$$\text{Taxa de Retransmissão} = \frac{\text{Quantidade de retransmissões}}{\text{Quantidade total de Interesses}} \tag{7}$$

É importante ressaltar que em redes ICN, uma retransmissão não necessariamente implica que o dado precisa percorrer todo o caminho até o produtor do conteúdo. Caso a informação tenha sido transmitida até parte do caminho antes de ser perdida, ela pode ser obtida do cache de algum dos nós. Dessa forma, o impacto de uma retransmissão na carga da rede pode variar, e retransmissões subsequentes têm maiores chances de impactarem menos (Pfender; Valera; Seah, 2018).

O total de despejos de cache serve como indicador de quão bem uma estratégia de cache é capaz de adaptar os dados armazenados em cache em relação à popularidade e propagação de conteúdo. Em casos onde a estratégia não é capaz de lidar bem com a diversificação de conteúdos populares, pode ocorrer o efeito de thrashing em situações onde o número de conteúdos populares é maior que o tamanho de cache disponível (Pfender; Valera; Seah, 2018).

A métrica de diversidade (DM) indica a diversidade de conteúdos armazenados ao longo de toda a rede. Levando em consideração o número de produtores (S) e a quantidade de produtores diferentes que possuem algum conteúdo armazenado ao longo da rede (C_{disj}), a métrica de diversidade demonstra a porcentagem de produtores de conteúdos representados em cache em um determinado momento. A Equação 8 demonstra o cálculo da métrica.

$$DM = \frac{C_{disj}}{S} \tag{8}$$

A métrica de diversidade mede apenas a diversidade em termos de produtores e não em termos de conteúdo real, e, em uma rede real, é possível que alguns produtores de conteúdo sejam muito mais populares do que outros. Nesse caso, a DM

não seria capaz de capturar o quão bem o conteúdo desse produtor está distribuído pela rede em comparação com o restante do conteúdo.

Para esse caso, a taxa de retenção de cache (CRR) é utilizada como métrica. De forma semelhante à métrica de diversidade, ela representa a proporção de conteúdos únicos (D_q), em relação a todos os conteúdos já gerados pela rede local (D_p), que estão armazenados em cache ao longo da rede para um determinado instante de tempo. A taxa de retenção de cache é calculada a partir da Equação 9.

$$CRR = \frac{D_q}{D_p} \tag{9}$$

As métricas acima foram utilizadas para analisar a desempenho dos algoritmos em um cenário implementado na Future Internet Testing Facility Internet of Thing Laboratory (FIT IoT-LAB), um ambiente com mais de 2700 dispositivos que permite teste e implementação de aplicações IoT em 6 lugares diferentes (Adjih *et al.*, 2015).

Os hardwares utilizados para implementação foram um micro-controlador STM32 com 512 kB de memória de leitura (ROM) e 64 kB de memória de acesso aleatório (RAM) e um microchip transceiver Atmel AT86RF231 com 2.4 GHz operando com o padrão IEEE 802.15.4, atuando como o dispositivo de IoT (Pfender; Valera; Seah, 2018).

Foram implementados sessenta micro-controladores STM32 com tamanho de cache de vinte objetos cada e foram distribuidos de forma que a rede utilize multi-hop com média de dois a três saltos por caminho de transmissão. Cada nó gerava um conteúdo aleatório em intervalos entre um e cinco segundos e, através da Interface de Programação de Aplicação (API) do FIT IoT-LAB, o controlador instruía cada nó a requisitar uma informação aleatória a cada segundo (Pfender; Valera; Seah, 2018).

Além disso, no modelo de seleção aleatória de conteúdo requisitado, foram utilizados dois cenários. No primeiro, os conteúdos têm chances uniformes de serem selecionados, enquanto no segundo modelo foi utilizado o padrão de distribuição Zipf, que classifica as chances com base em popularidade probabilística do conteúdo (Pfender; Valera; Seah, 2018).

A conclusão de Pfender, Valera e Seah (2018) é de que, com base nos resultados obtidos, o esforço na implementação de algoritmos extremamente complexos é pouco vantajosa, sendo que em alguns cenários os algoritmos mais simples e populares podem ter desempenho semelhante ou até mesmo melhor que os algoritmos mais elaborados (Pfender; Valera; Seah, 2018).

Outra pesquisa utilizada como referência para o desenvolvimento deste trabalho foi o artigo *NDN Content Store and Caching Policies: Performance Evaluation*, desenvolvido por Silva, Macedo e Costa (2022). O foco desse trabalho foi entender e comparar o comportamento de quatro políticas de cache em um ambiente NDN. As políticas selecionadas para estudo foram Random, LRU, FIFO e Least Frequently Used

(LRU).

As métricas selecionadas para análise no trabalho de Silva, Macedo e Costa (2022) foram taxa de cache hit, carga do servidor, atraso no recebimento dos dados, taxa de retransmissão de erros, as quais foram utilizadas também no trabalho de Pfender, Valera e Seah (2018), além da métrica de taxa de número de saltos upstream.

A taxa de número de saltos upstream representa quantos saltos em média um interesse precisa realizar para que seja satisfeito, seja por meio de cache hit ou por alcançar o produtor. A Equação 11 apresenta o cálculo da métrica.

$$\text{Saltos totais} = \sum_{i=1}^{\text{Number of Interests}} \text{Saltos upstream} \tag{10}$$

$$\text{Taxa de Saltos Upstream} = \frac{\text{Saltos totais}}{\text{Quantidade total de Interesses}} \tag{11}$$

No estudo de Silva, Macedo e Costa (2022) o simulador ndnSIM foi selecionado como ambiente de testes, com a aplicação de duas topologias de estudo. A primeira topologia consiste em 42 roteadores genéricos onde cada roteador está ligado a apenas um consumidor, e há apenas um produtor entre os 42 nós. A segunda topologia foi baseada na rede Abilene, com 11 roteadores no total, e entre 4 e 5 consumidores por roteador, sendo que, assim como na primeira topologia, há apenas um nó produtor.

O principal diferencial deste trabalho em relação aos trabalhos de Pfender, Valera e Seah (2018) e Silva, Macedo e Costa (2022) é que em ambos a topologia da rede é estática, enquanto neste trabalho, devido à seleção de uma rede veicular para implementação, a topologia é dinâmica, influenciando diretamente na maneira que a FIB se comporta.

3.2 SELEÇÃO DAS MÉTRICAS

Considerando os trabalhos apresentados na Seção 3.1, as métricas selecionadas para a análise de desempenho dos algoritmos foram carga do servidor, taxa de cache hit, atraso no recebimento de dados, taxa de retransmissão de interesses, total de despejos de cache e taxa de saltos upstream.

O conjunto das métricas de carga do servidor e taxa de cache hit foram considerados para a análise da eficiência na decisão de caching e substituição de conteúdo despejado. O atraso no recebimento de dados, a taxa de retransmissão de interesses, o total de despejos de cache e a taxa de saltos upstream foram escolhidas para indicar a eficiência do algoritmo em lidar com o fluxo de dados, a comunicação entre os nós e a interação com o ambiente de rede.

As métricas de diversidade (DM e CRR) são diretamente relacionadas à proporção da diversidade de conteúdo contida na rede. Entretanto, devido à baixa

diversidade do conteúdo produzido no cenário de simulação selecionado, conforme definido na Seção 3.6, as métricas podem não capturar adequadamente o conteúdo mais representativo para análise, potencialmente levando a conclusões tendenciosas. Por esse motivo essas métricas não foram utilizadas.

Após a definição das métricas selecionadas, foi necessário selecionar o conjunto das políticas a serem analisadas, pois a forma de implementação de cada política poderia influenciar de forma direta na maneira de captar os dados utilizados para a análise das métricas.

3.3 SELEÇÃO DAS POLÍTICAS DE CACHE

Para este trabalho, assim como no trabalho de Pfender, Valera e Seah (2018), foram selecionados apenas algoritmos de decisão de caching e algoritmos de substituição. Algoritmos de inserção de cache não foram considerados nesta análise pois têm uma relevância maior no âmbito de caching de sistemas de tempo real. Além disso, a combinação com algoritmos que utilizam filas (como pFIFO e MDMR) poderiam levar à uma alta complexidade de código, podendo resultar em um alto atraso de processamento na rede.

Dentre os algoritmos de decisão selecionados, o CEE e o pCASTING foram incluídos na análise. Além desses, outros algoritmos, como o Prob(p), também foram considerados. No entanto, devido à sua semelhança com o pCASTING, combinada à consideração de menos variáveis (o pCASTING leva em conta a energia do carro, a ocupação do cache e a atualidade do conteúdo), é provável que o desempenho do Prob(p) seja inferior ao do pCASTING. Os algoritmos ProbCache, EgoBetw, PopNetCod e PT-Sharing também foram considerados, porém, exigem estabilidade na topologia da Rede, que não se aplica a VANETs.

Para a substituição foram adotados os algoritmos LRU, pFIFO, Random e MDMR. Embora os algoritmos FIFO e LFU tenham sido considerados, acabaram não sendo escolhidos devido à sua semelhança com as demais opções de algoritmos. O algoritmo PT-Sharing não foi selecionado devido à necessidade de constante troca de informações entre os nós da rede, o que poderia levar à poluição dos dados (Chen *et al.*, 2014).

3.4 IMPLEMENTAÇÃO DAS POLÍTICAS DE CACHE

Como mencionado na Seção 2.5.1, o ndnSIM é uma plataforma com base no NFD implementado de forma real na pilha NDN. Dessa forma, é possível desenvolver algoritmos que são equivalentes a uma implementação real no cenário de simulação selecionado.

Entretanto, a implementação de novas políticas de cache no NFD é um processo que enfrenta barreiras como a defasagem da documentação oficial do ndnSIM, que em muitos momentos ainda utiliza o não mais utilizado ndn::Forwarder como base de sua documentação, além da escassa divulgação de métodos de fácil entendimento, geralmente concentrados no Fórum NDN (UNIVERSITY OF CALIFORNIA, 2012).

A documentação base para a implementação considerada para essa etapa foi o Guia Oficial do NFD, que foi utilizado como base para compreender o funcionamento das políticas pré-instaladas (Afanasyev *et al.*, 2021).

A partir da extrapolação das informações e do estudo dos códigos de erro do compilador do ndnSIM, foi possível estabelecer um procedimento padrão para a implementação de novas políticas de cache. A estrutura dos novos algoritmos implementados foi idealizada com base nos exemplos implementados nativamente no ndnSIM (LRU e pFIFO).

O procedimento sintetizado de como implementar novas políticas de cache no ndnSIM pode ser encontrado no Apêndice A.

A implementação das políticas de cache consiste basicamente na aplicação de classes derivadas da superclasse Policy, a qual implementa métodos virtuais para sobrescrita. Os principais métodos empregados na criação de novas políticas são:

- **doAfterInsert**: Responsável por tratar o dado assim que é alocado no cache, ativado a partir de um sinal da classe CS. Este método desempenha a função da política de decisão.
- **evictEntries**: Verifica se o cache está cheio, e em caso positivo, seleciona o dado a ser despejado.
- **insertToQueue**: Encarregado de realizar a inserção do dado alocado no cache, seguindo a estrutura do algoritmo definido.

Inicialmente, a intenção era implementar o algoritmo de decisão CS antes do envio do sinal de alocação na estrutura física do Content Store. Contudo, devido a algumas limitações na sobrescrita dos métodos da classe, optou-se por realizar a implementação da política de decisão no método **doAfterInsert**. Nesse cenário, caso a alocação do conteúdo fosse negada, seria acionado um sinal de despejo novamente para a estrutura do Content Store. Essa escolha resultou, na prática, na existência de oito arquivos de implementação de algoritmo de gerenciamento de cache, consistindo na combinação par a par dos dois algoritmos de decisão, CEE e pCASTING, e dos quatro algoritmos de substituição, LRU, pFIFO, Random e MDMR.

Na implementação do pCASTING, adicionou-se à superclasse Policy um novo método, **toStore**, responsável pelo cálculo da probabilidade de alocação conforme definido na Seção 2.3.3. Os parâmetros utilizados para a implementação do algoritmo pCASTING são descritas na Tabela 1.

Tabela 1 – Parâmetros utilizados para o pCASTING

Parâmetro	Valor
w_1	0
w_2	0.5
w_3	0.5
n	1

Fonte: Elaborado pelo autor

O parâmetro w_1 foi fixado em 0, uma vez que está relacionado à energia do veículo no momento do cálculo da probabilidade de caching. No entanto, devido a configurações específicas do cenário de simulação, os veículos mantêm uma energia constante em 100%. Nesse contexto, considerando que o impacto desse índice seria mínimo, optou-se por aumentar o peso dos demais fatores para melhor avaliação da eficiência do pCASTING.

O método de inserção na fila foi simplificado, adotando uma fila única para os algoritmos LRU e Random. A inserção ocorre na última posição da fila, e no caso do LRU, há reinserção em caso de o conteúdo já estar presente.

Para o pFIFO, foram adotadas três filas auxiliares para o gerenciamento dos dados armazenados: `QUEUE_UNSOLICITED`, `QUEUE_STALE` e `QUEUE_FIFO`. O dado é direcionado para a primeira fila caso o interesse não tenha sido gerado pelo veículo que está armazenando a informação. Ele é alocado na segunda fila se o conteúdo tiver sido recebido devido a um interesse gerado pelo veículo, mas não for considerado novo. Já na última fila, o conteúdo é armazenado se tiver sido recebido devido a um interesse gerado pelo veículo e for considerado novo. Além disso, ao serem armazenados na última fila, é definido um temporizador para mover o dado para a fila `QUEUE_STALE` caso o período de validade seja ultrapassado.

Para o MDMR, considerando a estrutura de simulação selecionada, foram empregadas 11 filas no total, sendo 10 filas correspondentes a cada via da malha e uma fila adicional denominada "trash", utilizada para armazenar dados não relevantes gerados pela atualização das FIBs. A cada novo dado recebido e aceito, ele é salvo na fila correspondente à sua via na malha.

O método de despejo implementado para os algoritmos seguiu os princípios já apresentados na Fundamentação Teórica, conforme descrito na Seção 2.4.1 para o LRU, na Seção 2.4.2 para o Random e na Seção 2.4.2. No caso do pFIFO, o despejo ocorre primeiro na fila `QUEUE_UNSOLICITED`, caso haja dados, e posteriormente nas filas `QUEUE_STALE` e `QUEUE_FIFO`, removendo sempre o item que encabeça cada fila.

Após a criação dos arquivos de implementação e cabeçalho, procede-se à inclusão das novas políticas no StackHelper, encarregado da configuração dos nós no simulador. Durante a inicialização da classe, é necessário empregar o método **insert** para registrar o construtor de cada uma das classes. Essa prática possibilita que o

helper instale a política especificada no cenário em todos os nós pertinentes.

Ao final desse procedimento, as políticas criadas estão disponíveis para utilização em cenários de simulação no ndnSIM.

3.5 REGISTRO DAS MÉTRICAS

Após a implementação dos algoritmos de gerenciamento de cache, tornou-se essencial realizar a incorporação dos registros de simulação, os quais fornecerão as informações necessárias para a análise de desempenho dos algoritmos.

O armazenamento das métricas foi distribuído em quatro arquivos, listados no Quadro 3. O acesso aos arquivos de texto é realizado por meio de um ponteiro compartilhado, possibilitando que o NS-3 gerencie os recursos de forma eficiente em ambientes multi-thread. A adoção desse método foi necessária devido à presença de diversos logs gerados pelo ndnSIM com informações de pouco interesse, o que poderia dificultar a análise subsequente dos dados obtidos. A escolha desse método de arquivar as informações implicou em erros na escrita de dados que precisaram ser tratados, como apresentado na Seção 3.8.

Quadro 3 – Arquivos de logs das simulações

Arquivo	**Estrutura**	**Descrição**
evictions	Tipo de ação (ADDITION e EVICTION), nome do Interesse e timestamp	Arquivo que controla a inserção e remoção de arquivos na Content Store
hitmiss	Tipo de ação (HIT e MISS), nome do Interesse e timestamp	Arquivo para controle de hits e misses na Content Store
interests	Tipo de ação (SENT, RECEIVED e RESEND), nome do Interesse, solicitante e timestamp	Arquivo para controle de emissão e recebimento de interesses, além do índice ded retransmissões
server_load	Nome do Interesse e timestamp	Arquivo para controle de Interesses que chegam até o servidor

Fonte: Elaborado pelo autor

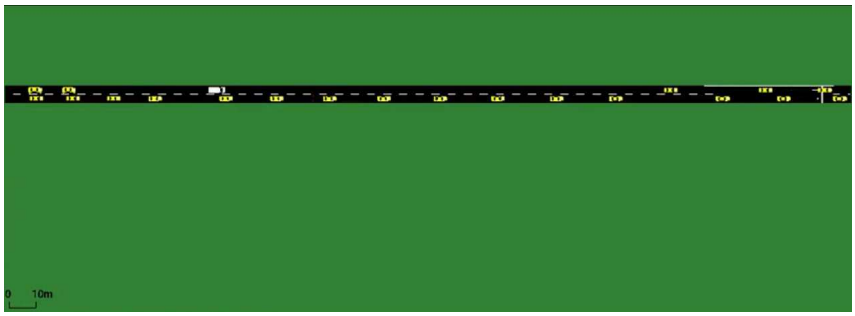
As saídas do arquivo **evictions** foram atreladas aos métodos *evictOne* e *attachQueue* de cada algoritmo de gerenciamento de cache. O arquivo **hitmiss** foi atrelado aos métodos *onContentStoreHit* e *onContentStoreMiss* da classe Forwarder. Os arquivos **interests** e **server_load** foram atrelados, respectivamente, aos programas dos nós consumidores (veículos) e dos nós produtores (RSUs).

3.6 DEFINIÇÃO DO CENÁRIO DE SIMULAÇÃO

A definição do cenário de simulação foi uma etapa crucial no desenvolvimento deste projeto, uma vez que influencia diretamente o desenho e a execução da simulação. Alguns aspectos considerados na seleção do cenário envolvem o comportamento dos veículos ao longo da simulação, a estrutura da malha viária e o tipo de dado consumido pela rede.

O ndn4IVC dispõe de dois cenários pré estabelecidos para uso em testes e simulações. No cenário *beacon*, há uma única rodovia, com duas faixas em sentido único, no qual o tipo de dado enviado é a presença de um veículo de emergência, que ativa o comportamento de permissão de passagem (todos os veículos se direcionam para a faixa da direita) por um curto período de tempo em cada veículo. Um frame deste cenário pode ser visto na Figura 12.

Figura 12 – Cenário de simulação *beacon*



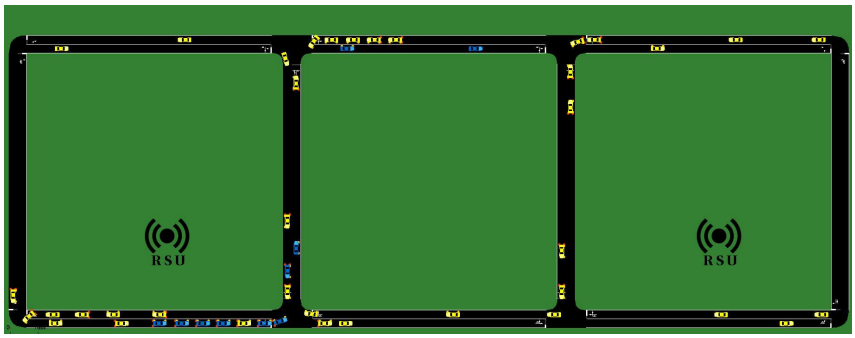
Fonte: Araujo; Peixoto; Sampaio (2013)

O segundo cenário disponibilizado, *tms-consumer*, apresenta um cenário com três quadras de circulação em mão dupla, dispondo também de semáforos nas intersecções das vias. O tipo de dado gerado é uma estrutura com a velocidade média dos veículos da via e a densidade veicular, representada por uma porcentagem da capacidade máxima de veículos, de cada sentido de cada via, permitindo que os algoritmos internos de cada veículo escolham rotas alternativas em caso de engarrafamentos ou acidentes de trânsito com bloqueio de vias. Um frame deste cenário pode ser visto na Figura 13.

Considerando que o propósito deste trabalho é analisar o comportamento de algoritmos de cache em um contexto dinâmico e com topologia de rede pouco previsível, o cenário *tms-consumer* foi escolhido, uma vez que o conteúdo exerce maior influência no funcionamento da rede, facilitando a visualização dos impactos dos diversos algoritmos de cache.

A estrutura dos nós da rede compreende duas RSUs, posicionadas nos pontos indicados na Figura 13, atuando como produtores de conteúdo, e sessenta veículos, que são introduzidos dinamicamente ao longo do período de simulação. Todos os

Figura 13 – Cenário de simulação *tms-consumer*



Fonte: Araujo; Peixoto; Sampaio (2013)

veículos possuem um ponto de partida, um destino final e uma rota inicial definida por meio de arquivos de configuração xml do SUMO. Adicionalmente, foi estabelecida a ocorrência de um acidente no início da simulação com duração de 150 segundos.

Ao longo da simulação, os veículos solicitam atualizações de status da próxima via da sua rota atual, e, caso a velocidade média de alguma esteja abaixo de 20% da velocidade máxima permitida da via ou estiver mais de 80% ocupada, verifica qual rota alternativa tomar, atualizando o roteiro até o destino final.

Os veículos estão configurados com uma variável *windowSize*, que determina o período de validade de uma informação. Nesse cenário, a janela foi estabelecida em 30 segundos. Dessa forma, um veículo que solicitar informações sobre uma determinada via aos 29 segundos de simulação pode receber tanto uma informação gerada com 1 segundo de simulação quanto uma gerada no mesmo intante do recebimento do Interesse pelo servidor, ambas consideradas válidas. Além disso, o timeout, que representa o tempo de ida e volta (Round Trip Time - RTT), definido para os interesses enviados pelos nós consumidores, é de 1 segundo. Caso o Interesse não tenha sido satisfeito ao final desse período, ocorre uma retransmissão.

O Interesse criado pelos veículos é configurado com nomes estruturados de forma que o sentido da via é explícito, por exemplo, o Interesse “/service/traffic/C1C0/0” solicita o status da via entre as esquinas C1 e C0 no sentido da primeira para a segunda, dentro da primeira janela de validade, entre 0 e 30 segundos. Dessa forma, os pacotes de dados são menores e evitam a transição de informações indesejadas.

As RSUs, desempenhando o papel de produtores nesse cenário, processam o Interesse recebido se estiver dentro da janela de tempo especificada no nome, como "0", por exemplo, no caso do Interesse “/service/traffic/C1C0/0”. Elas geram o conteúdo correspondente se estiverem na mesma janela de tempo da simulação atual. Os pacotes de dados gerados são estruturados por meio de um arquivo json e contém as informações de ID da via, velocidade média da via e taxa de ocupação da via.

Considerando que a janela de validade dos dados está incorporada na própria nomeação dos dados, em vez de o dado ter o tempo de validade $t_c + 30$, onde t_c é o

tempo de criação do dado, sua validade se estende até a próxima mudança de janela de validade. Por exemplo, um dado gerado aos 45 segundos de simulação terá sua validade estabelecida até 60 segundos de simulação, que corresponde ao término da segunda janela (entre 30 e 60 segundos).

Os comportamentos descritos acima para produtores (RSUs) e consumidores (veículos) são instalados em cada nó gerado na definição do cenário de simulação. Além disso, são instalados os adaptadores de rede, que para este caso foi implementado o protocolo IEE 802.11p, específico para funcionamento em redes veiculares, com potência de sinal de transmissão de 21 dBm, equivalendo a um alcance próximo de 70m.

Após a configuração do módulo de rede, é instalado o protocolo NDN nos nós. A estratégia de encaminhamento selecionada foi a multicast-vanet, que consiste em encaminhar para todas as interfaces registradas na FIB que não estão bloqueadas de alguma forma. Caso não exista interface disponível, o dado é recusado e não é salvo na PIT.

Além da estratégia de encaminhamento, são configuradas questões como o algoritmo de gerenciamento de cache, tamanho da Content Store e frequência de envio de interesses, que são as variáveis escolhidas para desenvolvimento das análises de comparação de resultados.

3.7 VARIÁVEIS DE SIMULAÇÃO

Além dos diferentes algoritmos de gerenciamento de cache, foi necessário realizar a seleção de alguns parâmetros para variação no ambiente de simulação com o propósito de entender como esses algoritmos se comportam em diferentes situações.

As variáveis selecionadas foram **período entre o envio de interesses** e **tamanho da Content Store**, e os possíveis valores assumidos estão dispostos na Tabela 2. Os valores de período de envio foram selecionados tomando como base o valor proposto no cenário inicial do NDN4IVC, validados pelos valores utilizados nos estudos de Pfender, Valera e Seah (2018) e Silva, Macedo e Costa (2022). Para a definição do tamanho da Content Store foram analisados os trabalhos de Cai *et al.* (2006), Kaplan e Winder (1973) e Meade (1970). Embora esses estudos sugiram um padrão de tamanho de cache de 128 unidades de informação, considerando a baixa diversidade de dados do ambiente proposto (limitada a 20 conteúdos por janela de validade) e alguns testes preliminares, o valor foi reduzido para promover maior interação dos algoritmos de gerenciamento de cache.

A escolha das variáveis mencionadas deve-se à alta influência desses fatores no gerenciamento de cache. Aumentar a frequência de envio de Interesses proporciona

Tabela 2 – Variáveis de simulação e seus possíveis valores

Parâmetro	Valor mínimo	Valor máximo
Período	1	5
CS Size	32	64

Fonte: Elaborado pelo autor

mais ciclos de decisão de caching, ao passo que a redução do tamanho da Content Store diminui o número de ciclos necessários para as primeiras interações de seleção de conteúdo de despejo.

Outras variáveis, como a janela de validade dos conteúdos, o alcance da rede WiFi e a estrutura de nomes dos Interesses e Pacotes de dados, foram consideradas e analisadas. Contudo, a correlação entre esses parâmetros e as métricas selecionadas é baixa, o que poderia resultar em análises inconclusivas ao final do trabalho. Além disso, o ambiente de simulação escolhido atenua o impacto da alteração desses parâmetros devido à baixa diversidade de conteúdo, que minimiza o impacto da variação dos parâmetros de janela de validade e estrutura de nomes. A baixa extensão geográfica do ambiente também reduz a influência da variação no alcance do módulo WiFi.

3.8 SIMULAÇÕES

Após a definição de todas as variáveis para a realização das simulações, foi desenvolvido um script em Python para a execução automática dos experimentos. A cada iteração, os parâmetros da simulação eram ajustados, assim como os algoritmos de gerenciamento de cache. Com 2 algoritmos de decisão, 4 algoritmos de despejo e 2 parâmetros de simulação, cada com 2 valores possíveis, o número total de cenários foi de 32. As informações sobre as características gerais da simulação são apresentadas no Quadro 4.

O script Python automatizava a modificação das variáveis de simulação, assim como o conjunto de algoritmos de gerenciamento de cache no arquivo de configuração da simulação. Em seguida, realizava a compilação e a execução do cenário. Após cada simulação, as informações coletadas eram armazenadas em uma pasta compartilhada entre a máquina virtual e a máquina hospedeira e iniciava novamente o ciclo. O período médio estimado para a conclusão de todos os cenários foi de aproximadamente 5 horas.

Como mencionado anteriormente na Seção 3.5, a escolha de realizar a escrita em tempo real causou alguns problemas na gestão do arquivo TXT, levando a falhas ocasionais nos registros de entrada, especialmente nas informações relacionadas a armazenamentos e despejos em cache, que representam um volume significativo de dados transitados.

Para lidar da melhor maneira possível com esse problema, uma análise

Quadro 4 – Parâmetros de simulação

Parâmetro	Valores assumidos
Período	1 ou 5 segundos
CS Size	32 ou 64
Política de decisão	CEE ou pCASTING
Política de despejo	LRU, pFIFO, Random ou MDMR
Protocolo WiFi	IEEE 802.11p
Alcance do WiFi	70 metros
Estratégia de encaminhamento	Multicast VANET
Produtores	2
Consumidores	60
Conteúdos	20
Métricas de análise	Despejos, cache hit, carga no servidor, atraso, retransmissão e saltos upstream
Simulações por cenário	10
Duração da simulação	210 segundos
Tamanho de cada via	50 metros

Fonte: Elaborado pelo autor

das falhas de escrita no arquivo ‘evictions.txt‘ foi realizada, identificando dois erros principais: linhas de registro de atividades juntas, sem quebra de linha, e a falta de registro do tempo de simulação. Para esses casos, foi criado um algoritmo que insere automaticamente a quebra de linha no primeiro caso e o tempo 0, considerando que as análises realizadas no Capítulo 4 não utilizam a dimensão temporal. A implementação dessa correção permitiu reduzir a quantidade de erros em 50%.

Além do tratamento mencionado no parágrafo anterior, não foram aplicados outros ajustes nos dados. O período de *warmup* foi incluído na análise dos dados, ou seja, não foi realizada a exclusão dos dados do início de cada simulação, que representaria um intervalo destinado ao preenchimento inicial das Content Stores dos veículos, inicialmente vazias. Isso se deve ao fato de que trabalhos relacionados empregam uma análise temporal que leva em consideração o início da simulação. Assim, a fim de manter consistência com as referências selecionadas, optou-se por não incluir o período de *warmup*.

A consolidação dos dados e sua representação gráfica foram executadas mediante a utilização de um script Python. Os valores apresentados no gráfico final correspondem à média simples derivada das 10 simulações, enquanto a barra de erro indica o desvio padrão entre as médias obtidas em cada execução do cenário.

3.9 CONSIDERAÇÕES DO CAPÍTULO

No presente capítulo, delineou-se o processo de escolha das políticas de cache para análise, estabelecendo as métricas necessárias para a condução da avaliação.

Detalhou-se a criação do ambiente de simulação para uma rede VANET utilizando o protocolo NDN, incluindo a implementação de arquivos de registro para a compilação das informações de cada simulação. Ademais, discutiu-se o processo de seleção das variáveis do ambiente, culminando na definição de quatro cenários distintos de simulação. No próximo capítulo, serão apresentados e discutidos os resultados obtidos nesses cenários, proporcionando uma análise crítica dos dados coletados.

4 RESULTADOS E DISCUSSÕES

Neste capítulo serão apresentados os resultados das simulações realizadas, agrupando-os em quatro seções de acordo com o conjunto de variáveis de cenário (intervalo de interesses e cache size). Seguidamente, as discussões referentes ao desempenho dos algoritmos em cada cenário serão realizadas. Por fim, serão apresentadas hipóteses sobre o caso de uso de cada conjunto de algoritmos.

Os cenários serão apresentados a partir do menor estresse da rede (maior período entre interesses e maior tamanho de cache) até o maior estresse da rede (maior frequência de envio de interesses e menor tamanho de cache) para melhor compreensão, conforme dispostos na Tabela 3.

Tabela 3 – Parâmetros utilizados em cada cenário de simulação

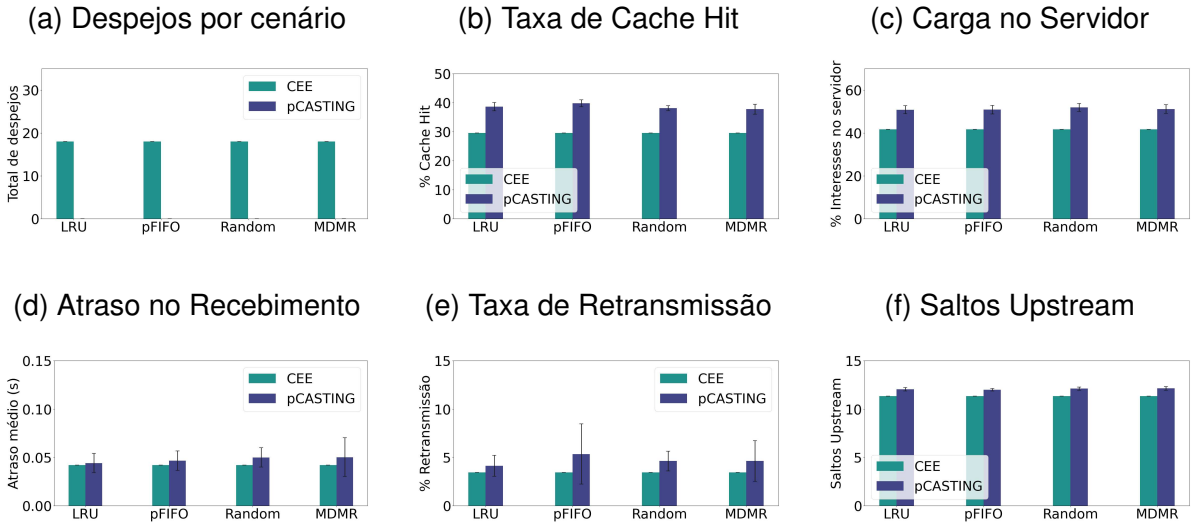
Cenário	Período (s)	Tamanho da CS
A	5	64
B	1	64
C	5	32
D	1	32

Fonte: Elaborado pelo autor

4.1 CENÁRIO A

Nessa seção são explorados os resultados obtidos sob as condições estabelecidas pelo Cenário A, no qual o intervalo entre requisições de informações na rede foi configurado para 5 segundos e o tamanho da cache foi fixado em 64 objetos. É o cenário com maior intervalo e maior tamanho de cache, avaliando o desempenho dos algoritmos em situações de menor estresse. Os resultados desse cenário são apresentados na Figura 14.

Figura 14 – Resultados do cenário A



Fonte: Elaborado pelo autor

No Cenário A, observa-se, de maneira abrangente, que os algoritmos de substituição apresentam desempenho estatisticamente comparável quando empregam o mesmo algoritmo de decisão de caching. No entanto, ao direcionarmos nossa atenção para a comparação de desempenho entre os diferentes algoritmos de decisão, torna-se evidente uma distinção significativa, o que requer uma discussão mais profunda.

A discrepância mais notável acontece na métrica de Despejos por cenário, conforme ilustrado na Figura 14a. A observação mais evidente é a não existência de despejos na implementação do pCASTING, uma vez que este algoritmo tem natureza probabilística para a decisão de caching. Em outras palavras, como não é todo dado que chega até o nó que será alocado em cache, e a carga de rede deste cenário é baixa, não ocorrem despejos.

A taxa de cache hit demonstra-se notavelmente superior ao utilizar o algoritmo pCASTING, contrariando a expectativa inicial de que seria menor devido à sua prática de salvar menos informações ao longo da rede. Todavia, essa aparente contradição encontra sua explicação na característica específica do cenário de simulação selecionado, a qual exerce um impacto direto nos índices do pCASTING. A brevidade das rotas nesse contexto implica que cada veículo tem uma permanência relativamente curta na simulação, resultando em poucos casos nos quais o cache atinge sua capacidade máxima no Cenário A. Como consequência, é provável que os veículos recém-inseridos na simulação, cujos caches estão inicialmente vazios, apresentem maior probabilidade de armazenar informações mais recentes, preservando-as ao longo do tempo.

Como discutido na Seção 2.3.3, um dos índices que o pCASTING utiliza para decidir sobre o armazenamento do dado é a atualidade, ou seja, informações próximas

ao vencimento terão uma probabilidade menor de serem guardadas. Por consequência, a cache tende a guardar mais dados que serão válidos. Por outro lado, o CEE guardar todos os dados, inclusive os vencidos, o que tende a diminuir a taxa de acerto.

Em relação aos resultados na Figura 14c, nota-se uma conformidade esperada, visto que o uso de um algoritmo probabilístico diminui a probabilidade de encontrar uma informação no cache, resultando em trajetos mais extensos na rede, como refletido na Figura 14f, que apresenta uma média de saltos superior com a aplicação do algoritmo pCASTING.

A necessidade de realizar mais saltos até obter a informação requerida também se reflete no atraso do recebimento dos dados, como evidenciado na Figura 14d. Essa correlação influencia a taxa de retransmissão, que tende a aumentar devido à ativação associada a um limite superior de atraso, demonstrado na Figura 14e. Enquanto no CEE a natureza determinística nos apresenta uma variação nula entre as diferentes simulações, a natureza probabilística do pCASTING traz uma variação que torna os desempenhos estatisticamente iguais nas métricas de atraso e retransmissão.

Contudo, é crucial observar que, ao analisar o desempenho médio, o CEE demonstra um desempenho superior, como era esperado devido à maior probabilidade de um conteúdo específico estar armazenado no cache de um nó de rede próximo em comparação com o pCASTING.

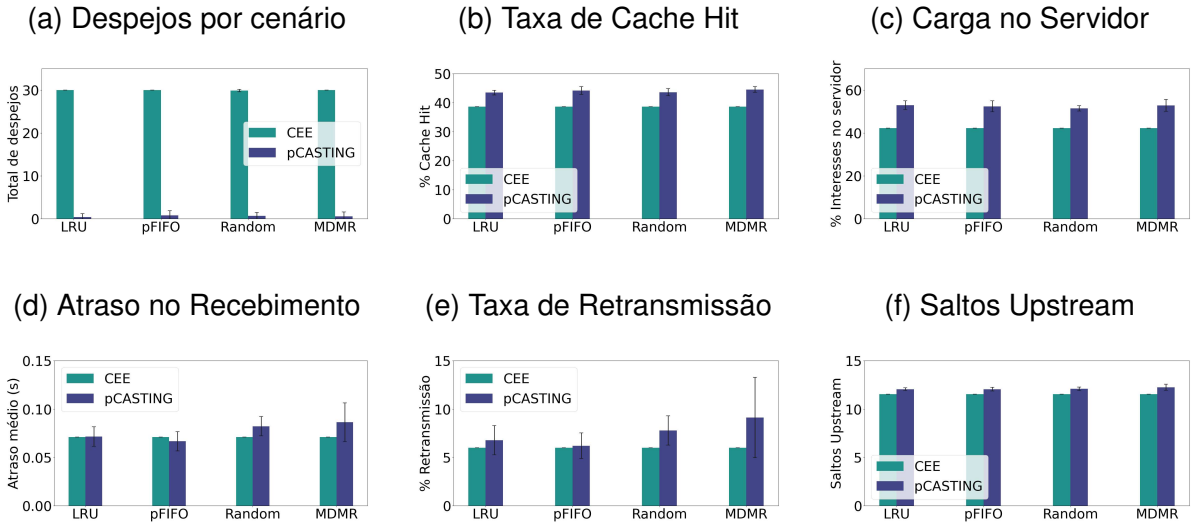
Após a simulação do cenário com menor carga de rede, optou-se por modificar a variável de frequência de envio de interesses, uma escolha fundamentada na suposição de que o impacto seria menos significativo do que a alteração na variável de tamanho do cache. A motivação subjacente a essa decisão está alinhada com o objetivo do projeto, que busca compreender a evolução gradual do cenário à medida que o estresse da rede se intensifica.

4.2 CENÁRIO B

Na presente seção, exploramos os resultados obtidos sob as condições estabelecidas pelo Cenário B, onde o intervalo entre requisições de informações na rede foi reduzido para 1 segundo, e o tamanho da cache foi mantido em 64 objetos. Esta configuração específica busca avaliar o desempenho dos algoritmos de gerenciamento de cache em um contexto mais dinâmico, onde as solicitações de dados são mais frequentes, e a capacidade de armazenamento da cache permanece constante. Os resultados desse cenário são demonstrados na Figura 15.

O padrão observado anteriormente no Cenário A apresenta notável semelhança com o comportamento observado no Cenário B. A análise da Figura 15a revela que, embora haja um aumento no número de despejos nos conjuntos com CEE e

Figura 15 – Resultados do cenário B



Fonte: Elaborado pelo autor

a ocorrência de alguns despejos nos conjuntos com pCASTING, a similaridade entre os conjuntos com o mesmo algoritmo de decisão permanece significativamente elevada.

Ao examinar a métrica de saltos upstream, como ilustrado na Figura 15f, é possível observar a inexistência de alterações notáveis. Essa constância sugere a viabilidade de considerar os cenários A e B estatisticamente iguais em relação a essa métrica de avaliação.

Um comportamento interessante de ser observado é o apresentado na Figura 15c, que demonstra alteração praticamente nula na métrica de carga do servidor, mesmo com o aumento da frequência de envio de interesses. Esse fenômeno pode ser explicado ao se observar a Figura 15b, que demonstra que a carga desses novos Interesses trafegando na rede foi suprida pelo cache dos próprios veículos, indicado pelo aumento significativo na taxa de cache hit.

Um fenômeno inesperado se revela no aumento da taxa de retransmissão (Figura 15e), contrariando a expectativa teórica devido ao incremento na probabilidade de um dado necessário estar armazenado no cache de um nó próximo, fortalecida pelo aumento correspondente na taxa de cache hit. No entanto, uma análise detalhada dos registros revela que a maioria das retransmissões ocorreu nos estágios finais da validade das informações em circulação na rede. Nesse cenário, as informações em cache naquele momento não atendiam mais aos interesses enviados, explicando assim o aparente paradoxo entre o aumento na probabilidade de cache hit e o correspondente acréscimo na taxa de retransmissão.

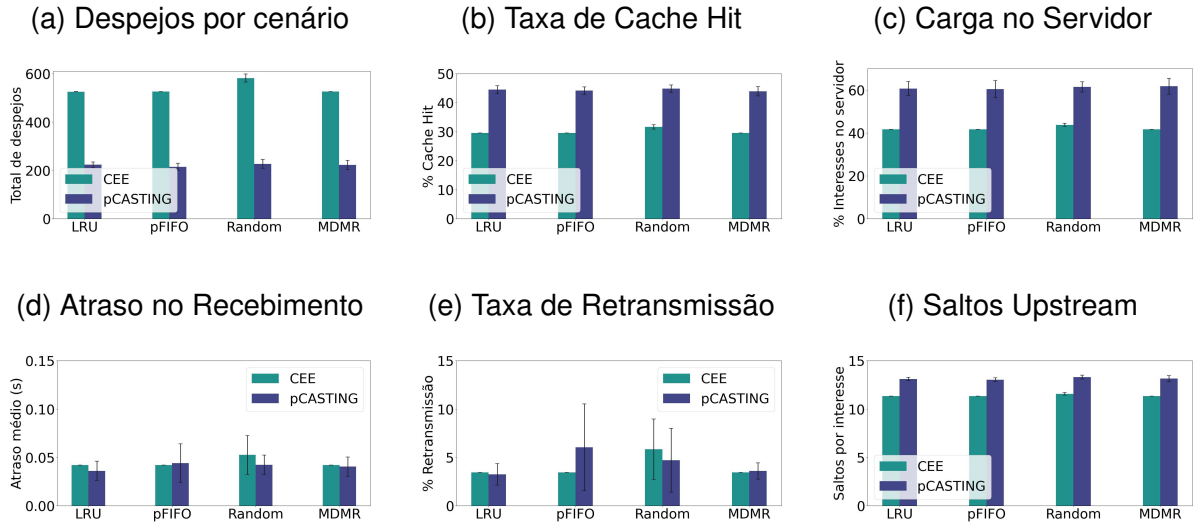
Na Figura 15d é possível observar um aumento significativo do atraso em todos os cenários. Com o aumento do número de interesses processado por cada nó, era esperado o aumento dessa métrica devido ao aumento do atraso por fila, simulado no

NS-3 por meio da estrutura de fila dos nós da rede.

4.3 CENÁRIO C

Na terceira etapa da análise, retorna-se ao intervalo de tempo entre o envio de interesses para 5 segundos, entretanto, reduzindo pela metade o tamanho da cache para 32 objetos. Esta modificação tem como objetivo investigar o comportamento da rede diante da restrição de recursos, sem comprometer a estabilidade devido à sobrecarga resultante da circulação de interesses. Os resultados são apresentados na Figura 16.

Figura 16 – Resultados do cenário C



Fonte: Elaborado pelo autor

Na análise da métrica de despejos (Figura 16a), foi necessário ajustar a escala do eixo Y para otimizar a visualização dos dados, dada a expressiva elevação no número de despejos, aproximadamente 20 vezes superior. Em relação aos conjuntos que empregam CEE, destaca-se a significativa disparidade de desempenho do algoritmo Random em comparação com os demais. Já no contexto do uso de pCASTING, os algoritmos mantêm desempenho estatisticamente equivalente, evidenciando uma redução de mais de duas vezes nos despejos em comparação aos conjuntos que utilizam CEE.

Ao analisar os conjuntos que utilizam pCASTING, destaca-se um aumento significativo na taxa de retransmissão (Figura 16e), refletindo diretamente no aumento expressivo na carga do servidor (Figura 16c). Similar ao Cenário B, esse aumento é explicado pelos momentos em que a janela de validade dos dados é alterada. Contribui para esse aumento o fato de os conteúdos estarem mais amplamente distribuídos pela rede, pois nem todo conteúdo recebido é armazenado, o que substancialmente eleva a

probabilidade de um interesse alcançar o servidor, conforme evidenciado no aumento da métrica de saltos upstream na Figura 16f. A métrica de atraso também apresentou o aumento esperado, conforme demonstrado na Figura 16d.

A métrica de taxa de cache hit, apresentada na Figura 16b, também teve um aumento significativo ao observar o comportamento dos conjuntos que utilizam pCASTING. Esse incremento pode ser atribuído à abordagem do pCASTING, que considera a ocupação da cache como um fator determinante na decisão de caching, e a redução do tamanho da Content Store aumenta a rapidez com que ela é ocupada. Consequentemente, valores que provavelmente não seriam referenciados no curto prazo não são armazenados, favorecendo a retenção de valores mais frequentemente acessados na cache. No entanto, é importante ressaltar que essa justificativa requer uma análise mais aprofundada da carga de trabalho associada aos acessos de interesse, uma análise a ser conduzida em trabalhos futuros.

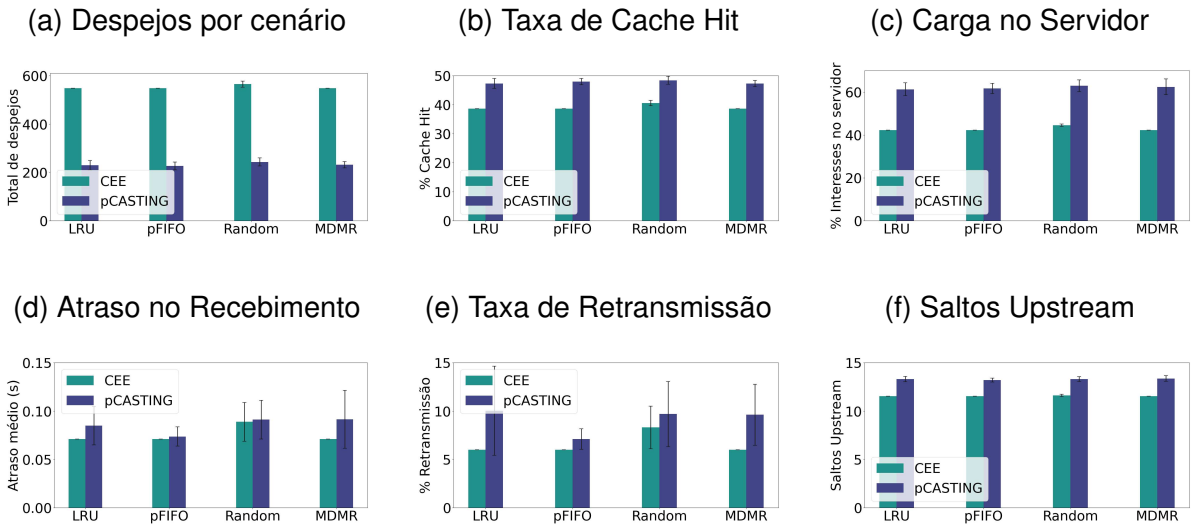
Destaca-se que, embora o algoritmo MDMR siga a tendência dos demais algoritmos, seu comportamento diferencia-se em comparação a estes. Ao contrário do observado nos outros casos, o MDMR demonstra uma diminuição na taxa de retransmissão, assemelhando-se ao LRU, quando comparado ao Cenário A. No que diz respeito à métrica de cache hit, mesmo apresentando um aumento e sendo considerado estatisticamente comparável, o MDMR exibe uma porcentagem inferior em relação aos demais algoritmos. Essa discrepância pode ser atribuída possivelmente à maneira como as fileiras de produtores foram definidas, uma vez que cada rua é considerada um produtor. No entanto, é importante ressaltar que, simultaneamente, dois conjuntos de dados da mesma rua são válidos (sentido de ida e sentido de volta), levando o algoritmo MDMR a escolher a exclusão do outro arquivo proveniente da mesma rua em alguns momentos, resultando em uma forma de alternância entre os dados válidos da mesma via.

Ao observar o desempenho do CEE, destaca-se uma estabilização em comparação com o Cenário A, tanto na taxa de cache hit quanto no atraso, carga do servidor, taxa de retransmissão e média de saltos upstream. Essa estabilidade é evidenciada na maioria dos algoritmos, com a exceção do algoritmo Random. O comportamento deste algoritmo pode ser explicado pela possibilidade de despejo de conteúdos com alta frequência de consulta e longa duração da validade, o que demanda uma busca mais extensa na rede por nós capazes de suprir a demanda do interesse. Em relação aos demais algoritmos, a estabilidade é um comportamento esperado. Ao reduzirmos o cache mantendo a frequência de envio de interesses baixa, e considerando que a limitação de recursos force o despejo de mais conteúdos ao longo da rede, o conteúdo continua a existir em algum nó dentro do alcance do nó atual. Esse cenário resulta principalmente no comportamento de um alto número de despejos.

4.4 CENÁRIO D

Na última etapa de análise direta dos cenários, introduzimos o Cenário D, concebido para impor o maior nível de estresse à rede. Neste cenário final, a frequência de envio de interesses foi reduzida para 1 segundo, enquanto o tamanho da cache foi mantido em 32 objetos. Essa configuração específica visa explorar os limites da rede, provocando uma sobrecarga mais substancial em comparação aos cenários anteriores. O Cenário D, portanto, desafia a capacidade dos algoritmos de gerenciamento de cache em ambientes de alta demanda, com a expectativa de identificar comportamentos distintos e avaliar a robustez dos algoritmos sob condições extremas.

Figura 17 – Resultados do cenário D



Fonte: Elaborado pelo autor

Em comparação ao Cenário C, nota-se grande estabilidade nos indicadores analisadas. As métricas de despejos (Figura 17a), carga no servidor (Figura 17c) e saltos upstream (Figura 17f) são estatisticamente comparáveis entre os dois cenários.

A taxa de cache hit apresentou um aumento, conforme previsto, uma vez que parte da nova demanda de interesses foi atendida pelo conteúdo armazenado em cache. Este aumento é esperado devido à maior probabilidade de o conteúdo estar presente no cache de vários nós da rede após múltiplas rodadas de interação.

Ao analisar as métricas de atraso no recebimento (Figura 17d) e taxa de retransmissão (Figura 17e), observa-se um aumento significativo, especialmente nos conjuntos que empregam pCASTING. A explicação para esse comportamento pode ser encontrada ao considerarmos as alterações em relação ao Cenário A. A redução do tamanho da Content Store, combinada com a manutenção de uma frequência elevada, resulta em um aumento no número de saltos upstream (Figura 17f). Como mencionado anteriormente, um efeito colateral do aumento na frequência de envio de interesses é o aumento do atraso na fila. Ao combinar essas duas condições, a cada salto adicional

que o interesse realiza, o acumulado do atraso aumenta. Como resultado, alguns interesses excedem o tempo de espera e são retransmitidos.

Após a análise minuciosa de todos os cenários e a compreensão das modificações no comportamento dos algoritmos selecionados em resposta às alterações nas variáveis de ambiente, a próxima seção se dedicará a examinar o desempenho de cada algoritmo no contexto geral. Nesse contexto mais amplo, serão apresentadas considerações e propostas acerca de situações específicas em que cada algoritmo pode ser mais eficaz, fornecendo assim orientações para a escolha apropriada de algoritmos em diferentes cenários e condições de rede.

4.5 CONSIDERAÇÕES PARCIAIS

Na presente seção, dedicada à análise do desempenho dos conjuntos de algoritmos, empreenderemos uma comparação entre os conjuntos de algoritmos, visando identificar cenários específicos nos quais cada conjunto demonstra vantagens. Salienta-se que a compreensão plena do desempenho é intrinsecamente vinculada ao contexto da simulação adotado. No cenário simulado, a geração dinâmica de dados com uma janela de validade reduzida, desconsiderando a que a simulação em si tem um período curto, e a atribuição de alta criticidade ao valor dos dados fornecem os parâmetros essenciais para a avaliação do desempenho dos algoritmos em questão.

A avaliação das métricas, como despejos, taxa de cache hit, taxa de retransmissão e saltos upstream, desempenha um papel crucial na comparação e compreensão do funcionamento dos diferentes conjuntos de algoritmos. Contudo, é importante destacar que as métricas de Carga no Servidor e Atraso no Recebimento têm relevância singular, uma vez que exercem impacto direto no desempenho percebido da rede. Nesse contexto, a ênfase das análises desta seção estará direcionada particularmente para essas duas métricas.

Como discutido nas seções anteriores, a análise dos diversos cenários indicam que a métrica de atraso no recebimento está mais associada à variação da frequência de envio de interesses, enquanto a carga do servidor é mais sensível à redução no tamanho do Content Store. Com base nessa observação, torna-se possível inferir que a primeira métrica assume um caráter crítico em situações de maior carga na rede, ao passo que a segunda desempenha um papel crucial em contextos nos quais há limitação nos recursos disponíveis na rede.

Em diferentes contextos de rede, a ênfase em uma métrica específica pode ser justificada. No cenário da simulação selecionada, onde o enfoque recaía sobre a disseminação de informações críticas acerca das condições viárias, a análise da métrica de atraso no recebimento surge como mais pertinente. Isso se deve à natureza dessas informações, cujo atraso potencialmente comprometeria as ações de

veículos de emergência e a eficácia do reajuste de rotas na rede, resultando em maior congestionamento.

Em contextos de baixa demanda de rede, todos os conjuntos de algoritmos podem ser considerados estatisticamente equivalentes. Entretanto, é observável que os conjuntos que utilizam o algoritmo de decisão pCASTING apresentam um desempenho notavelmente inferior em comparação com os demais conjuntos. Em situações de alta demanda, a disparidade de desempenho entre os algoritmos de decisão é reduzida, destacando-se a consistência do desempenho nos conjuntos pFIFO e pCASTING. Este fenômeno sugere que o pCASTING é possivelmente um algoritmo que sofre menos impacto em seu desempenho conforme a carga na rede aumenta. Um comportamento adicional a ser destacado é que o algoritmo pCASTING, devido ao seu método de tomada de decisão baseado em cálculos de probabilidade, manifesta uma maior instabilidade em seu comportamento dentro de um mesmo cenário.

Ao incorporar a análise da variação do tamanho do cache, nota-se que no Cenário C, onde a frequência de envio de interesses é baixa, mas há uma limitação no tamanho do cache, o pCASTING supera o CEE em quase todos os conjuntos. Contudo, no Cenário D, os conjuntos de pCASTING apresentam uma significativa degradação de desempenho, sendo superados pelo CEE em todos os casos.

Em contextos semelhantes ao do cenário de simulação escolhido, os quais as informações transmitidas entre os nós da rede podem ser consideradas de alta criticidade, ou mesmo o tempo de resposta dos atuadores da rede é essencial, como redes com alto tráfego de veículos de emergência, ou com alto índice de acidentes, incidentes ou mesmo desastres naturais, constata-se que a escolha do algoritmo de substituição possui pouca relevância no resultado final. Embora os algoritmos de substituição tenham pouco impacto nos resultados da métrica de atraso, é importante ressaltar que os algoritmos LRU e MDMR podem ser mais interessantes devido à estabilidade de desempenho, possuindo as menores latências na maioria dos cenários quando utilizados em conjunto com o algoritmo CEE.

Em relação aos algoritmos de decisão, enquanto o pCASTING revela um melhor desempenho em situações específicas, como no Cenário B, caracterizado por uma ampla disponibilidade de tamanho de cache, ou no Cenário C, com limitação de cache, mas uma demanda reduzida na rede, o referido algoritmo demonstra-se instável. Tal instabilidade se evidencia tanto na análise do desempenho médio, à medida que há variação nas variáveis da simulação, quanto na análise da variação intrínseca a um mesmo cenário, com considerável erro percentual. Dessa forma, em cenários onde o interesse reside na estabilidade do desempenho na métrica de atrasos, é recomendável a utilização do CEE, decisão reforçada na análise da métrica de carga do servidor, a qual demonstra consistentemente valores inferiores ao utilizar o CEE em todos os cenários apresentados.

No entanto, o pCASTING exibe uma propensão à estabilização em contextos de maior estresse na rede, mostrando-se menos afetado em situações críticas em comparação com o CEE. Observa-se uma tendência de melhor desempenho do pCASTING em cenários que apresentam maior diversidade de conteúdo na rede e uma extensão geográfica mais ampla. Apesar disso, é necessário validar essa hipótese por meio de estudos futuros em ambientes com maior variedade de conteúdo como ambientes VANET com aplicações de streaming de áudio e vídeo.

4.5.1 Limitações

Nesta seção serão discutidas as restrições e considerações importantes que permeiam a análise realizada nos resultados apresentados. Reconhecendo a complexidade do ambiente de simulação e as escolhas feitas na configuração dos parâmetros, esta seção busca fornecer uma visão crítica sobre os limites do estudo, promovendo uma compreensão das implicações e interpretações dos resultados alcançados.

É crucial destacar que o contexto selecionado para a simulação apresenta dados altamente homogêneos, com uma ampla diversidade de produtores decorrente da decisão de considerar cada via como um produtor. Entretanto, essa diversidade contrasta com a baixa variação de conteúdo, uma vez que cada produtor gera apenas dois conteúdos, correspondentes aos sentidos opostos da via. Além disso, a janela de validade dos conteúdos é proporcionalmente extensa em relação ao tempo total da simulação.

Além disso, é relevante observar que o cenário adotado possui dimensões consideravelmente reduzidas no contexto de análise de malhas viárias urbanas. Essa característica específica pode ter um impacto significativo na métrica de carga no servidor, uma vez que a distância média entre os servidores abrange aproximadamente a metade da malha. Essa limitação geográfica associada ao fato de que a estrutura de forwarding do NDN consiste em um broadcast para todos os nós da FIB pode ter ocultado a métrica real de saltos upstream, além de aumentar consideravelmente a carga no servidor de maneira artificial.

Ao comparar os resultados deste trabalho com os trabalhos de referência de Silva, Macedo e Costa (2022) e Pfender, Valera e Seah (2018), observa-se uma disparidade nos resultados. As métricas de cache hit, server load, saltos upstream apresentadas pelos autores são menores quando comparadas a este trabalho, enquanto as métricas de atraso no recebimento do dado e taxa de retransmissão são maiores. No entanto, nos trabalhos citados, são utilizadas topologias estáticas e geograficamente dispersas, ao passo que este trabalho adota uma topologia altamente dinâmica e com pouca extensão em tamanho. Adicionalmente, índices como a diversidade de conteúdo e o tamanho do cache size são de ordens de grandeza

distintas entre este trabalho e os referenciados. O primeiro trabalho citado utiliza 1000 como tamanho padrão do Content Store, enquanto este trabalho adota 64, e gera 10^6 conteúdos diferentes, em contraste com os 20 conteúdos diferentes gerados neste estudo. Ambos os trabalhos referenciados também empregam uma distribuição *MZipf* para simular a popularidade de conteúdos, o que contribui para um impacto mais significativo na escolha dos algoritmos.

Por fim, é crucial considerar que, em situações práticas, as redes são populadas por conteúdos que são dinamicamente gerados (variando conforme o nome do interesse recebido), conteúdos estaticamente gerados (mantendo-se inalterados independentemente do interesse recebido), ou uma combinação de ambos. Nos cenários estudados neste trabalho, focamos em conteúdos que não variam pelo nome do interesse, mas sim pelo tempo. As análises aqui apresentadas são válidas para esse contexto específico, e avaliações mais aprofundadas são necessárias para uma compreensão mais completa do comportamento desses algoritmos em situações reais, considerando a diversidade de padrões de tráfego e características dinâmicas das redes veiculares.

5 CONCLUSÕES

Redes orientadas a conteúdo surgiram como uma abordagem de arquitetura inovadora, transformando a busca de informações em um paradigma central nas redes. Essa característica, combinada com a capacidade de se ajustar a topologias altamente dinâmicas, conferiu às redes orientadas a conteúdo uma vantagem considerável no contexto do desenvolvimento de redes veiculares. Otimizar o desempenho das redes veiculares é essencial, e o cache emerge como uma das ferramentas fundamentais para reduzir o atraso no acesso e entrega de conteúdo, conforme destacado por Khelifi *et al.* (2020). Nesse contexto, este trabalho teve como propósito a análise comparativa de algoritmos para gerenciamento de cache em redes veiculares que adotam o NDN como protocolo de rede.

Na primeira fase do desenvolvimento do projeto, o objetivo de *investigar mecanismos de gerenciamento de cache* foi integralmente alcançado. Introduziu-se o conceito de estratificação do gerenciamento de cache em três níveis (decisão, substituição e inserção), e optou-se por implementar os níveis de decisão e substituição. Além dos algoritmos de decisão pCASTING e CEE, e dos algoritmos de substituição LRU, pFIFO, Random e MDMR, outros algoritmos de decisão, como Prob(p), ProbCache, EgoBtw, PopNetCod e PT-Sharing, e de substituição, como FIFO e LFU, foram inicialmente considerados para o escopo do estudo. No entanto, questões como a complexidade na implementação, a alta semelhança com os algoritmos já selecionados e a necessidade de topologia estática para obter alto desempenho e levaram à exclusão desses algoritmos do conjunto final.

Após a escolha dos algoritmos a serem estudados, a etapa subsequente deste trabalho focou no objetivo de *definir métricas de comparação de algoritmos*. Nesse contexto, as pesquisas de Pfender, Valera e Seah (2018) e Silva, Macedo e Costa (2022) embasaram a condução do projeto. Além das métricas de despejos totais, taxa de cache hit, carga no servidor, atraso no recebimento, taxa de retransmissão e saltos upstream, que foram escolhidas para a avaliação dos algoritmos, o trabalho de Pfender, Valera e Seah (2018) também incorporou métricas de diversidade, DM e CRR. No entanto, dada a baixa diversidade intrínseca ao cenário selecionado para a simulação, concluiu-se que as informações obtidas por essas métricas teriam pouca relevância para o contexto apresentado.

Um dos objetivos específicos foi *estabelecimento de um ambiente de simulação* que integrasse eficazmente uma rede com o protocolo NDN, aliada a uma topologia dinâmica típica de redes veiculares. Nesse sentido, empregou-se o integrador de simuladores NDN4IVC, possibilitando a integração completa entre o simulador de redes NS-3 e o simulador de malhas viárias SUMO. Um ambiente de simulação foi proposto,

contemplando duas variáveis com dois valores possíveis em cada, o que resultou em quatro cenários distintos.

Durante o desenvolvimento do objetivo de *avaliar comparativamente os algoritmos* nos diferentes cenários de simulação, observou-se uma maior correlação entre a variável de frequência de envio de interesses e a métrica de atraso no recebimento de dados, ao passo que a métrica de carga do servidor demonstrou maior relação com a variável de tamanho da Content Store. Foi possível inferir que, em situações de baixa diversidade e alta criticidade do conteúdo da rede, os algoritmos de substituição têm baixa influência no desempenho da rede. No entanto, os algoritmos LRU e MDMR demonstraram maior estabilidade quando utilizados em conjunto com o algoritmo de decisão CEE. Ao analisar os algoritmos de decisão, em ambas as métricas destacadas, o algoritmo CEE apresentou melhor desempenho médio, evidenciando também maior estabilidade ao longo dos diversos cenários apresentados.

Além dos objetivos propostos inicialmente, durante o desenvolvimento desse ambiente, foi possível identificar desafios em encontrar referências para a implementação de novas políticas de cache devido à desatualização dos manuais de referência em certa medida. Assim, é elevante destacar que um dos resultados obtidos neste projeto foi a padronização de um processo para a implementação de novas políticas de cache, disponibilizado no Apêndice A.

O objetivo específico de *propor melhorias nos mecanismos de gerenciamento de cache* não foi alcançado pois ao longo do desenvolvimento do trabalho, constatou-se que os dados obtidos nos cenários de simulação propostos são bastante específicos. Nesse sentido, reconheceu-se a necessidade de um estudo mais abrangente, visando compreender se o comportamento observado nas situações apresentadas permanece consistente em outros ambientes e contextos.

Dessa forma, sugere-se que em trabalhos futuros sejam considerados cenários com maior diversidade de conteúdo, como streamings de áudio e vídeo, malhas viárias mais extensas e conteúdos de popularidade dinâmica para uma compreensão mais abrangente do comportamento da rede, enriquecendo a análise e propiciando propostas de melhorias nos algoritmos.

REFERÊNCIAS

ADJIH, C. *et al.* FIT IoT-LAB: A large scale open experimental IoT testbed. *in*: **Proceedings** of the 2015 IEEE 2ND WORLD FORUM ON INTERNET OF THINGS (WF-IOT). Milan, Italy, p. 459–464, 2015. Acesso em: 04 nov. 2023.

AFANASYEV, A. *et al.* **NFD developer's guide - named Data Networking (NDN)**. 2021. Disponível em: https://named-data.net/wp-content/uploads/2021/07/ndn-0021-11-nfd-guide.pdf. Acesso em: 25 jun. 2023.

AL-SULTAN, S. *et al.* A comprehensive survey on vehicular ad hoc network. **Journal of Network and Computer Applications**, v. 37, p. 380–392, 2014.

ALAYA, B. *et al.* Study on QoS management for video streaming in vehicular ad hoc network (VANET). **Wireless Personal Communications**, v. 118, n. 4, p. 2175–2207, 2021.

ARAUJO, G.; PEIXOTO, M.; SAMPAIO, L. A comprehensive and configurable simulation environment for supporting vehicular named-data networking applications. **Computer Networks**, v. 235, p. 109949, 2023.

BIAN, C. *et al.* Boosting named data networking for data dissemination in urban VANET scenarios. **Vehicular Communications**, v. 2, n. 4, p. 195–207, 2015.

CAI, Y. *et al.* Cache size selection for performance, energy and reliability of time-constrained systems. *in*: **Proceedings** of the ASIA AND SOUTH PACIFIC CONFERENCE ON DESIGN AUTOMATION, 2006. Yokohama, Japan, p. 6 pp.–, 2006. Acesso em: 04 nov. 2023.

CHEN, M. *et al.* Vendnet: Vehicular named data network. **Vehicular Communications**, v. 1, n. 4, p. 208–213, 2014.

COBHAM, A. Priority assignment in waiting line problems. **Operations Research**, v. 2, n. 1, p. 70–76, 1954.

DUA, A.; KUMAR, N.; BAWA, S. A systematic review on routing protocols for vehicular ad hoc networks. **Vehicular Communications**, v. 1, n. 1, p. 33–52, 2014.

FONSECA, J. J. S. d. **Apostila de Metodologia da Pesquisa Científica**. 2002. Disponível em: https://books.google.com/books/about/Apostila_de_metodologia_da_pesquisa_cien.html?id=oB5x2SChpSEC. Acesso em: 22 jul. 2023.

FOROUZAN, B. A.; GRIESI, A.; FEGAN, S. C. **Comunicação de dados e redes de computadores.** Porto Alegre: McGraw Hill, 2010.

HAHM, O. *et al.* Low-power internet of things with NDN & cooperative caching. *in*: **Proceedings** of the 4TH ACM CONFERENCE ON INFORMATION-CENTRIC NETWORKING. ACM, New York, set. 2017. Disponível em: https://doi.org/10.1145/3125719.3125732. Acesso em: 01 out. 2023.

HAIL, M. A. *et al.* Caching in named data networking for the wireless internet of things. *in*: **Proceedings** of the 2015 INTERNATIONAL CONFERENCE ON RECENT ADVANCES IN INTERNET OF THINGS (RIOT). Singapore, p. 1–6, 2015. Acesso em: 06 jun. 2023.

ISLAM, M. Z. *et al.* A comparative analysis of different real time applications over various queuing techniques. *in*: **Proceedings** of the 2012 INTERNATIONAL CONFERENCE ON INFORMATICS, ELECTRONICS VISION (ICIEV). Dhaka, Bangladesh, p. 1118–1123, 2012.

JACOBSON, V. *et al.* Networking named content. *in*: **Proceedings** of the 2009 ACM CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES. Palo Alto Research Center, p. 1–12, 2009. Disponível em: https://dl.acm.org/doi/10.1145/1658939.1658941. Acesso em: 06 jun. 2023.

KAPLAN, K.; WINDER, R. Cache-based computer systems. **Computer**, v. 6, n. 3, p. 30–36, 1973.

KHELIFI, H. *et al.* Named data networking in vehicular ad hoc networks: State-of-the-art and challenges. **IRRR Communications Surveys and Tutorials**, v. 22, n. 1, p. 320–351, 2020.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a internet**: uma abordagem top down. São Paulo: Pearson, 2013.

LAOUTARIS, N.; SYNTILA, S.; STAVRAKAKIS, I. Meta algorithms for hierarchical web caches. *in*: **Proceedings** of the IEEE INTERNATIONAL CONFERENCE ON PERFORMANCE, COMPUTING, AND COMMUNICATIONS, 2004. Phoenix, AZ, USA, p. 445–452, 2004. Acesso em: 04 nov. 2023.

LARKIN, D. H.; SEN, S.; TARJAN, R. E. A back-to-basics empirical study of priority queues. In: ____. **Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)**. [s.n.], 2014. p. 61–72. Disponível em: https://epubs.siam.org/doi/abs/10.1137/1.9781611973198.7.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY. F. Thomson Leighton e Daniel M. Lewin. **Global hosting system**. 2000. Patent US6108703A, 22 ago. 2000.

LEIRA, L.; LUíS, M.; SARGENTO, S. Context-based caching in mobile information-centric networks. **Computer Communications**, v. 193, p. 214–223, 2022. ISSN 0140-3664.

LOPEZ, P. A. *et al.* Microscopic traffic simulation using SUMO. *in*: **Proceedings** of the 2018 21ST INTERNATIONAL CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEMS (ITSC). Maui, HI, USA, p. 2575–2582, 2018. Acesso em: 25 jun. 2023.

MASTORAKIS, S. *et al.* **ndnSIM 2: An updated NDN simulator for NS-3**. [S.l.], 2016.

MEADE, R. M. On memory system design. *in*: **Proceedings** of the NOVEMBER 17-19, 1970, FALL JOINT COMPUTER CONFERENCE. In: . New York, NY, USA: Association for Computing Machinery, 1970. (AFIPS '70 (Fall)), p. 33–43. ISBN 9781450379045. Disponível em: https://doi.org/10.1145/1478462.1478468. Acesso em: 04 nov. 2023.

NAMED DATA NETWORK. Disponível em: https://ndnsim.net/current/. Acesso em: 20 jul. 2023.

NAMED DATA NETWORKING. **NDN frequently asked questions**. 2023. Disponível em: https://named-data.net/project/faq/. Acesso em: 26 abr. 2023.

O'NEIL, E. J.; O'NEIL, P. E.; WEIKUM, G. The LRU-K page replacement algorithm for database disk buffering. **SIGMOD Rec.**, Association for Computing Machinery, New York, NY, USA, v. 22, n. 2, p. 297–306, jun 1993.

PAUL, A. *et al.* Introduction: intelligent vehicular communications *in:* PAUL, A. et al. **intelligent vehicular networks and communications**. Elsevier, Amsterdam, p. 1–20, 2017.

PFENDER, J.; VALERA, A.; SEAH, W. K. G. Performance comparison of caching strategies for information-centric IoT. Association for Computing Machinery, New York, NY, USA, p. 43–53, 2018. Disponível em: https://doi.org/10.1145/3267955.3267966.

POPESCU-ZELETIN, R.; RADUSCH, I.; RIGANI, M. A. **Vehicular-2-X communication.** Berlin: Springer, 2010.

PRESS, L. The net: Progress and opportunity. **Communications of the ACM**, v. 35, n. 12, p. 21–25, 1992.

QURESHI, M. K. *et al.* Adaptive insertion policies for high performance caching. Association for Computing Machinery, New York, v. 35, n. 2, p. 381–391, jun 2007.

SALTARIN, J. *et al.* Popnetcod: A popularity-based caching policy for network coding enabled named data networking. 2019. Disponível em: https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsarx&AN=edsarx.1901.01187&lang=pt-br&site=eds-live&scope=site&authtype=ip,uid&group=main&profile=eds. Acesso em: 6 jun. 2023.

SILVA, E. T. d.; MACEDO, J. M. H. d.; COSTA, A. L. D. Ndn content store and caching policies: Performance evaluation. **Computers**, v. 11, n. 3, 2022.

TARNOI, S. *et al.* Performance of probabilistic caching and cache replacement policies for content-centric networks. *in*: **Proceedings** of the 39TH ANNUAL IEEE CONFERENCE ON LOCAL COMPUTER NETWORKS. Edmonton, AB, Canada, p. 99–106, 2014. Acesso em: 06 jun. 2023.

TECH ACADEMY. **FIFO/FCFS Page Replacement Algo Explanation**. 2015. Disponível em: https://www.youtube.com/watch?v=UTXkbcJUY74. Acesso em: 25 set. 2023.

TECH ACADEMY. **Least Recently Used (LRU) Explanation**. 2015. Disponível em: https://www.youtube.com/watch?v=4wVp97-uqr0. Acesso em: 24 jun. 2023.

UFRN; CAPES. **Plataforma Sucupira**. 2022. Disponível em: https://sucupira.capes.gov.br/. Acesso em: 04 out. 2023.

UNIVERSITY OF CALIFORNIA. 2012. Disponível em: https://www.lists.cs.ucla.edu/mailman/listinfo/ndnsim. Acesso em: 04 nov. 2023.

UNIVERSITY OF WASHINGTON NS-3 CONSORTIUM. **NS-3: Contrib**. Disponível em: https://www.nsnam.org/docs/release/3.8/doxygen/group__contrib.html. Acesso em: 04 nov. 2023.

WANG, L. *et al.* Rapid traffic information dissemination using named data. *In*: **Proceedings** of the 1ST ACM WORKSHOP ON EMERGING NAME-ORIENTED MOBILE NETWORKING DESIGN - ARCHITECTURE, ALGORITHMS, AND APPLICATIONS. Association for Computing Machinery, New York, p. 7–12, 2012. Disponível em: https://doi.org/10.1145/2248361.2248365. Acesso em: 6 jun. 2023.

YAO, L. *et al.* V2X routing in a VANET based on the hidden Markov model. **IEEE Transactions on Intelligent Transportation Systems**, v. 19, n. 3, p. 889–899, 2018.

ZHANG, Z. *et al.* An SDN-based caching decision policy for video caching in information-centric networking. **IEEE Transactions on Multimedia**, v. 22, n. 4, p. 1069–1083, 2020.

APÊNDICE A - GUIA DE CRIAÇÃO DE POLÍTICAS DE CACHE

Durante o desenvolvimento do trabalho de conclusão de curso, ao utilizar o ndnSIM, foi constatado que a documentação necessária para a implementação de novas políticas de cache está desatualizada na documentação oficial do simulador. Além disso, as informações atualizadas são dispersas, exigindo a busca em diversas fontes de informação.

Dessa forma, foi idealizada a criação deste guia, com o objetivo de fornecer instruções do processo de implementação de novas políticas em um único documento. As principais fontes de informação para as orientações apresentadas neste guia incluem a documentação oficial do ndnSIM (NAMED DATA NETWORK,), o guia de utilização do NFD (Afanasyev *et al.*, 2021), e o fórum de discussões do NDN (UNIVERSITY OF CALIFORNIA, 2012).

É importante ressaltar que este guia não contempla o desenvolvimento de uma política de gerenciamento de cache. Seu objetivo é puramente esclarecer o processo de implementação de algoritmos dentro da arquitetura do simulador.

A pasta na qual as políticas implementadas nativamente no ndnSIM se localizam é a `ndnSIM/ns-3/src/ndnSIM/NFD/daemon/table`. Para cada política é possível encontrar um arquivo de implementação e um arquivo de header.

A implementação das políticas de cache consiste basicamente na aplicação de classes derivadas da superclasse Policy, implementada no arquivo *cs-policy*. Os principais métodos empregados na criação de novas políticas são:

- **doAfterInsert**: Responsável por tratar o dado assim que é alocado no cache, ativado a partir de um sinal da classe CS. Este método desempenha a função da política de decisão.
- **evictEntries**: Verifica se o cache está cheio, e em caso positivo, seleciona o dado a ser despejado.
- **insertToQueue**: Encarregado de realizar a inserção do dado alocado no cache, seguindo a estrutura do algoritmo definido.

Os métodos acima podem utilizar outros métodos e funções auxiliares para o processamento dos dados recebidos, entretanto é necessário compreender que as chamadas internas do simulador disparam a execução dos métodos listados acima.

Após a implementação dos algoritmos desejados na política e a definição do namespace da classe, é necessário declarar a nova política no arquivo StackHelper, localizado no caminho `ndnSIM/ns-3/src/ndnSIM/helper/ndn-stack-helper.cpp`, responsável pela instalação das configurações dos nós no ambiente de simulação.

No construtor do StackHelper é necessário inserir uma nova linha utilizando o método *insert* na pilha de políticas registradas. Os argumentos recebidos pelo método

são o namespace da política e o construtor da classe da política. Na Figura 18 é possível ver um exemplo com as políticas utilizadas neste trabalho.

Figura 18 – Exemplo de declaração das políticas de cache

```
StackHelper::StackHelper()
  : m_isForwarderStatusManagerDisabled(false)
  , m_isStrategyChoiceManagerDisabled(false)
  , m_needSetDefaultRoutes(false)
{
  setCustomNdnCxxClocks();

  m_csPolicies.insert({"nfd::cs::lru", [] { return make_unique<nfd::cs::LruPolicy>(); }});
  m_csPolicies.insert({"nfd::cs::priority_fifo", [] () { return make_unique<nfd::cs::PriorityFifoPolicy>(); }});

  m_csPolicies.insert({"nfd::cs::random", [] () { return make_unique<nfd::cs::random::RandomPolicy>(); }});
  m_csPolicies.insert({"nfd::cs::mdmr", [] () { return make_unique<nfd::cs::mdmr::MDMRPolicy>(); }});

  m_csPolicies.insert({"nfd::cs::lrupcasting", [] () { return make_unique<nfd::cs::lrupcasting::LruPolicyPcasting>(); }});
  m_csPolicies.insert({"nfd::cs::priority_fifo_pcasting", [] () { return make_unique<nfd::cs::PriorityFifoPcastingPolicy>();

  m_csPolicies.insert({"nfd::cs::randompcasting", [] () { return make_unique<nfd::cs::randompcasting::RandomPcastingPolicy>(
  m_csPolicies.insert({"nfd::cs::mdmrpcasting", [] () { return make_unique<nfd::cs::mdmrpcasting::MDMRpCastingPolicy>(); }})

  m_csPolicyCreationFunc = m_csPolicies["nfd::cs::lru"];
```

Fonte: Elaborado pelo autor

Após a declaração de todas as políticas, é necessário configurar uma política padrão, que será utilizada caso o argumento de namespace da política não seja fornecido na configuração do nó. No exemplo acima foi utilizado o LRU como política padrão.

Após essa configuração, é necessário apenas aplicar o comando de build do simulador, para que as mudanças sejam efetivamente implementadas. No exemplo da Figura 19, além do comando `.waf`, comando padrão utilizado para build do NS-3, foram utilizados outros argumentos para inicializar uma simulação padrão pré-configurada no NDN4IVC.

Figura 19 – Exemplo de comando de build

```
ndn4ivc@NDN4IVC-VM: ~/ndnSIM/ns-3
File  Edit  View  Search  Terminal  Help
ndn4ivc@NDN4IVC-VM:~/ndnSIM/ns-3$ ./waf --run "vndn-example-tms --s=210 --sumo-gui"
```

Fonte: Elaborado pelo autor

Para utilização das políticas instaladas é necessário usar o método *setPolicy* da classe StackHelper, fornecendo como argumento da função o namespace configurado. No exemplo da Figura 20 um nó é configurado com a política LRU implementada e uma Content Store com tamanho de 128 unidades disponíveis.

O procedimento acima, apresentado de maneira simples e objetiva, fornece as diretrizes necessárias para a implementação bem-sucedida de novas políticas de cache no ambiente de simulação ndnSIM. Ao seguir essas etapas, os pesquisadores e desenvolvedores podem integrar algoritmos de gerenciamento de cache personalizados em seus experimentos, contribuindo para avanços contínuos na pesquisa em redes centradas em conteúdo.

Figura 20 – Configuração de um nó no ndnSIM

```
// install Ndn stack
std::cout << "Installing Ndn stack in " << nVehicles + nRSUs << " nodes ... " << std::endl;
ndn::StackHelper ndnHelper;
ndnHelper.AddFaceCreateCallback (WifiNetDevice::GetTypeId (), MakeCallback (FixLinkTypeAdhocCb));
ndnHelper.setPolicy("nfd::cs::lru");
ndnHelper.setCsSize (128);
ndnHelper.SetDefaultRoutes (true);
ndnHelper.InstallAll ();
```

Fonte: Elaborado pelo autor