Optimized Content Delivery in Vehicular Networks: A Novel SH-Based Caching Strategy and Replacement Algorithm

Pruthvi C N[1], H S Vimala[1] and Shreyas J[2]

[1]*Department of Computer Science and Engineering, UVCE, Bangalore University, Bangalore, India- 560001*
[2]*Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, Karnataka, India - 576104*

Abstract: Context: Vehicles are essential components of vehicular networks: they function as mobile nodes that exchange and disseminate vital information including emergency warnings, safety alerts, and updates on passenger enjoyment. As a result, the network experiences high data flow and a notable rise in data traffic. ICN architectures have been established in contrast to standard host-centric IP networks, which find it difficult to suit the dynamic needs of vehicular networks. ICN caching makes it possible to store content on the network, decreases content latency, and improves content accessibility.
Objective: This work aims to optimize content delivery in vehicular networks using ICN networks with efficient caching decisions.
Method: An efficient SH-based (Stretch and Hop) caching strategy is proposed to select the cache node with more interfaces in a delivery path. The interest and data packets are modified to identify the bridge node by adding H and stretch fields in the packets. Further, a novel cache replacement policy called PRPI (Popularity-Recency-Probability-Interfaces) eviction policy is proposed to remove outdated data from the cache node when storage is full. PRPI policy considers the content's popularity, recency score, probabilistic factor, and incoming interface score to calculate eviction value. The content with the least eviction value is selected to remove from the cache.
Result: The proposed work is evaluated using simulation, and the results show better efficiency in terms of cache hit ratio, content retrieval delay, and stretch ratio.
Conclusion: Using ICN caching in vehicular networks increases content availability in a network. The proposed work uses a high interface node to cache the content and removes the outdated content by considering multiple factors. This method leads to storing the fresh content at a more connected node, which can satisfy more requests.

Keywords: Caching, ICN, ICN caching, NDN, Vehicular network.

1. INTRODUCTION

Vehicular networks, including Vehicular Ad-hoc Networks (VANETs) and the more recent vehicle-to-everything (V2X) networks, have become a significant focus for researchers, academics, and industry professionals. These networks offer various applications, such as enhancing road safety, improving traffic flow, providing entertainment for drivers and passengers, and offering environmental benefits [1] [2]. Automobiles and transportation networks have long been a focus of research in relation to vehicular communication technologies. Vehicle-to-vehicle (V2V) communication allows for direct interaction between vehicles, removing the necessity for distributed or central servers for data distribution or storage. The communication foundation of cutting-edge technologies like linked autonomous cars, intelligent transportation systems, and smart cities is provided by vehicular ad hoc networks, or VANETs. There has been a significant growth in data traffic within VANETs due to their wide range of uses, as well as the growing number of linked vehicles and the increased use of high-bandwidth apps like social networking and video streaming [3] [4]. In VANETs, vehicles, including both drivers and passengers, can share and receive real-time information with minimal delay. This information includes details on road safety, traffic conditions, accidents, weather updates, parking availability, promotional offers, and targeted advertisements. Most existing architectures for vehicular networks rely on the TCP/IP stack, necessitating a node to locate the address of the content source and establish a route before accessing the content. This process complicates application development for vehicular networks due to the challenge of maintaining routes in high-mobility environments [5]. Assigning IP addresses to vehicles in such dynamic settings is particularly difficult. Consequently, a host-free connection model and content-centric architecture are favored for vehicle-to-vehicle networks.

Named Data Networking (NDN), a novel Information-Centric Networking (ICN) architecture, emphasizes content as the primary network element [6]. NDN nodes utilize application-level content names to exchange and retrieve data packets directly, bypassing the need for IP addresses [7]. This model incorporates in-network caching, making NDN particularly effective for wireless environments char-

E-mail address: pruthvicngowda@gmail.com

acterized by mobility and intermittent connectivity, such as delay-tolerant, opportunistic, and vehicular networks [8]. Unlike traditional systems, NDN nodes possess content stores that hold data. This supports node mobility, as the content store can adapt to users' content demands [9][10].

ICN offers several benefits in vehicular environments. One key advantage is content-based security, where information is tied to the content rather than the communication channel. In-network caching helps address mobility issues by allowing consumer vehicles to retrieve content from the nearest cache points. Routing and forwarding are based on the requested content instead of the host or provider's location. This approach supports high-speed vehicle movement and enhances content security against attacks targeting host addresses, such as DoS attacks. Overall, ICN caching improves efficiency in vehicular networks by ensuring faster and more reliable content access despite the mobility of vehicles [11] [12]. Caching content replicas by network nodes is a built-in feature of ICN, offering several benefits, such as reduced content retrieval delay, shorter path stretches, decreased network traffic, and optimal resource utilization. However, effectively caching multimedia content replicas remains a challenge. This challenge revolves around three main questions: 1) Where should replicas be cached? 2) What content should be cached? 3) How should the cached content be distributed to users? Considering all these benefits and challenges of ICN caching in vehicular networks, the contributions of the proposed work are listed below.

1) Developed an ICN-SH-based content caching strategy for vehicular networks to optimize content delivery by considering the stretch (path length) and hop count, ensuring more efficient routing and reduced latency.
2) The Proposed caching strategy selects nodes with a higher number of interfaces, allowing bridge nodes to cache data and satisfy more requests.
3) Introduced a novel cache replacement algorithm called PRPI to manage cache content when storage is full.
4) To improve the accuracy of cache eviction decisions by considering multiple factors, the PRPI algorithm uses a combination of popularity score, recency score, probabilistic factor, and incoming interface score to decide which content to evict.
5) Proposed work is simulated using the ICARUS simulator, demonstrating improvements in Cache Hit Rate (CHR), Cache Response Delay (CRD), and Stretch Ratio (SR).

A. *Organization of this paper*

The remainder of this paper is structured as follows: Section 2 reviews related work on ICN-caching in vehicular networks. Section 3 outlines the problem statement and the objectives of the proposed study. Section 4 provides an introduction to NDN packet processing. The system design of the proposed work is detailed in Section 5. Section 6 presents the proposed E^2C^2 strategy. The proposed cache eviction policy is discussed in Section 7. Section 8 offers evaluation results of the proposed approach. Finally, Section 9 concludes the paper and suggests directions for future research.

2. Related works

Caching in ICN plays an important role since it's in the early stage of development and has gotten most of the researcher's attention in recent years. Since each node in NDN has limited cache capacity, many researchers concentrated on cache management strategies [13] [14]. Deciding which node should cache which data to achieve higher efficiency becomes a major challenge. Hence, most researchers proposed caching strategies based on the method of forwarding packets with caching mechanisms [15]. In these strategies, nodes handle cache management independently based on the node's parameters, or sometimes, nodes cooperate with other nodes to achieve maximum cache efficiency [16]. Caching all content at all nodes is called CEE [17] and is an in-built cache strategy of ICN. This caching strategy increases the redundant caches in the network and inefficient usage of resources. LCD (Leave Copy Down) caching strategy is developed to reduce redundancy [18]. It reduces redundancy by caching the most popular content at edge nodes; however, when requests increase for particular content, it leads to a cache at all nodes. Probabilistic-based caching (ProbCache) [19] is also a primary caching mechanism based on the probability value p, which enables a high probability of storing content at edge nodes without considering the popularity of the content and also increases the pressure on the edge nodes. Hence, caching strategies will be concentrated more on cache placement, which is where to store the content and cache replacement, that is, which content should be replaced from the cache. [20] proposed an energy-efficient content placement that caches the content at edge nodes by considering the residual energy of the node, content popularity, and cache gain value. This work supports efficient usage of cache resources by checking the remaining energy of the node before selecting it to store the content. However, the authors have not considered the node's importance in the network. [21] a novel ICN-based proactive LRF caching method for VANETs. When material is proactively placed at the right nodes, VANET performance is optimized. Furthermore, the suggested approach guarantees the prompt distribution of messages pertaining to safety. [22] a region-based classification for vehicular network components improves content caching diversity and reduces transmission delay. Vehicles use a strategy to fetch requested content from their current or neighboring regions, considering transmission delay and connection handover. A proposed caching strategy for transient content in VENs enhances content availability, prioritizing items with longer lifetimes and high popularity if they are less distributed in the region. This approach optimizes fast content delivery and efficient caching in vehicular networks.
[23] a Popularity-Incentive Caching Scheme has been introduced. A Stackelberg game is modeled considering rational

utilities to address the conflict of interest between the base station (BS) and vehicles. The proposed solution to this game model is evaluated, including the impact of various weight parameters.

[24] proposes a proactive in-network caching scheme to improve data sharing. The scheme begins by dividing each onboard service into multiple content units cached at ICVs and small cell base stations to reduce content retrieval delays. The system is modeled as an INLP problem to optimize QoE by strategically placing content units at suitable caching locations.

[25] A new system for delivering content in vehicular networks has been proposed, which considers social interactions and traffic conditions. The system includes three main parts. First, a special search algorithm is used to find the shortest and most relevant paths for content delivery by looking at social connections between vehicles. Second, a recommendation scheme is used to suggest content based on the social context of the vehicle, using a technique called "vehicle2vec" to store information about previously consumed content. Finally, DRL is used to efficiently distribute content provider vehicles throughout the network, ensuring that content is delivered quickly and efficiently to where it is needed most.

[26] a new edge-computing-enabled hierarchical cooperative caching framework has been implemented. Firstly, the spatio-temporal correlation between historical vehicle trajectories and user requests has been deeply analyzed. A system model has been constructed to predict vehicle trajectories and content popularity, forming the basis for mobility-aware content caching and dispatching. Privacy protection strategies have also been explored to create a privacy-preserved prediction model.

[27] an efficient cluster-based caching in NDN NDN-VN has been proposed. This ensures that consumers receive content proactively after handover during their mobility. Additionally, the A-CB-PC-DMM method in the NDN-VN has been developed to improve the packet delivery ratio and to reduce handover delay and cluster overhead.

[28] a cooperative content caching approach is proposed for VENs. This approach utilizes K-means clustering to manage multi-tier caching servers, including base stations and roadside units, to collaboratively store identical content across various servers. Communication models for V2V and V2I interactions are developed, along with content caching and delivery models. The cuckoo search algorithm is then applied to determine vehicle locations and optimize content delivery according to these models.

[29] a TOCP scheme is proposed for CCVNs based on tolerable delay time. First, a numerical model is provided to determine the necessity of precaching by calculating the possibility of content provision. Then, a Delay-tolerant Content Management Module (DMM) is created to manage updated information. New packet formats are designed to reduce the movement of large-size content, achieving the optimization goals of TOCP.

[30] a collaborative caching method is introduced, organizing vehicles into clusters with a designated head to manage caches across the cluster. This strategy enables vehicles to access content cached on nearby vehicles even if they are not directly in the communication path.

[31] an EDC technique and a PACM have been proposed. This method takes into account the location and mobility speed of cars in a VANET and groups them into clusters according to their placements. The purpose of this architecture is to ensure consistent and dependable communication between member vehicles (MVs) and cluster heads.

Each caching framework for vehicular networks has its own benefits and limitations. The rapid increase in connected vehicles generates significant traffic, creating numerous challenges. Efficient resource utilization in these networks is a complex issue. Energy consumption and computational demands affect the performance and lifespan of network components in vehicles. Solutions must balance these factors carefully to avoid optimizing one at the expense of others. Therefore, further research and the creation of strong caching systems and strategies are essential to fully use vehicular networks' potential in ICN.

3. Problem description and objectives

In vehicular networks, efficient content caching is critical for ensuring timely data delivery and optimal network performance. Traditional caching strategies often fall short in dynamic and resource-constrained environments, such as those encountered in vehicular networks. To address these challenges, ICN caching has emerged as a promising approach. Current caching strategies do not adequately consider the stretch (path length) and hop count, leading to inefficient routing and higher latency in content delivery. The problem statement of the proposed work is to develop an efficient content caching strategy and novel replacement algorithm for optimized content delivery in vehicular network by considering stretch ratio and number of interfaces. The objectives of the proposed work are:

1) To optimize content delivery in vehicular network and improve cache eviction accuracy.
2) To enhance node selection for caching in vehicular network.
3) To increase the cache hit ratio of vehicular network.
4) To decrease the content retrieval delay in vehicular network.
5) To decrease the number of hops travelled to deliver the content.

4. NDN: An Overview

In this section, we briefly introduce the NDN working process along with packet structures. ICN is a future internet architecture that resolves almost all IP-based network issues, such as routing and content-sharing issues based on addresses. ICN focuses on content rather than end-to-end connections between the source and destination for content retrieval. The data or information vehicles produced in vehicular networks can be considered content. Vehicles or consumers can request content in the network by using the name of the content without worrying about the location of

the content producers. Each node in an ICN network works as a replica node using cache stores. Content stored at cache nodes served for future requests instead of reaching till original content producers. Enabling caching at intermediate nodes is called in-network caching in ICN. This is the most used feature of ICN. This in-network caching enhances data availability and data retrieval and reduces the latency in the network. Among all ICN projects, NDN is the most used project by researchers and is well suitable for vehicular networks. NDN is considered a promising architecture for the future of computer networks, providing a means to efficiently communicate data and content in the upcoming Internet era. NDN transfers the current IP-based network to a content-based network by addressing the content instead of the traditional host addressing method by allowing hosts to name the appropriate content [32].

The NDN architecture employs hierarchically structured names that are easily readable by humans. These names identify specific data, enabling content discovery and delivery to request consumers [33]. Communication in NDN happens through two types of packets: 1) Interest packet: Generated by the requester or consumer to request the particular content when needed. 2) Data packet: Generated by the producer as a response to consumers, which contains the data requested by the consumer and produced by the producer. The intermediate nodes in NDN are responsible for caching the content and acting as relay nodes. They forward the interest packet to producers, and the data packet is sent downward to consumers. These are called replica nodes because they store a copy of the content in their buffers while receiving the data packet. This copy is stored to fulfill future requests. The content name is added to both the interest and data packet, and the interest packet is forwarded hop-by-hop using the content name until it finds the corresponding content in the replica node or the original content producer. The same content name is added to the data packet while it travels back and follows the same path that the interest packet used. NDN is a receiver-driven architecture where each node in a network maintains three data structures. 1) Content Store (CS): Each node caches the content in CS and serves the future request without forwarding the request to the primary content provider. This increases the content-sharing probability, saves bandwidth, and decreases the content retrieval delay [34].
2) Pending Interest Table (PIT): If the requested content is available at the received node, it sends a data packet to the user. The request count will be added to PIT if the data is unavailable. If the same content request is already available, then the count is increased else, the new request is added. It helps to cache the content while encountering the data packet [35].
3) Forward Interest Base (FIB): FIB functions similarly to a routing table of the current architecture. It maintains records of all incoming and outgoing interfaces. Used to forward interest to the most suitable hops.
The communication process within the NDN network is depicted in Figures 1 and 2. The node checks its CS for

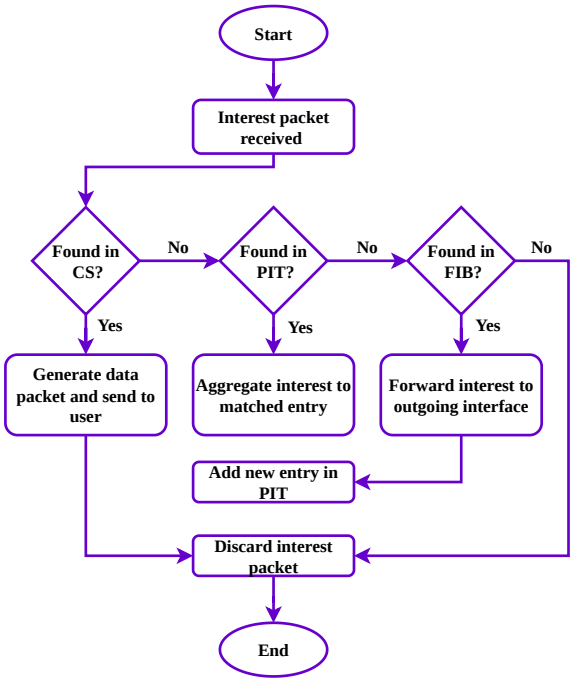

Figure 1. Interest packet processing in NDN

the desired data when it obtains the interest packet. If the data is available, it creates and transmits the packet to the requester. Based on the FIB history, the request information is added to PIT and the interest is passed to the subsequent node. The request is aggregated if the PIT record for the same content already exists; if not, a new entry is made. A node verifies its PIT after receiving data packets from other nodes. The node forwards the packet to every other node that has previously requested the same content if it discovers an entry that matches the content request. If not, the node considers the packet to be unimportant. Based on their respective caching policies, nodes along the data pipeline decide concurrently whether to cache the data or not.

A. Caching in ICN

One of ICN's most prominent and well-researched aspects is the in-network content caching at intermediate nodes. This feature has garnered significant attention from researchers, as noted in references [36]-[37]. Intermediate nodes can keep content retrieved from servers through in-network caching. Many advantages come with this approach, including decreased network traffic, more effective content distribution, fewer content retrieval latency, and more network capacity by delivering requested content closer to the user. However, there are a number of issues with in-network caching, including security risks, cache overflow, excessive duplication of cached content, and restrictions on where and how often data can be cached. All of these problems have an impact on the network's overall performance. To efficiently manage ICN caches, a number of caching techniques, content placement strategies, and

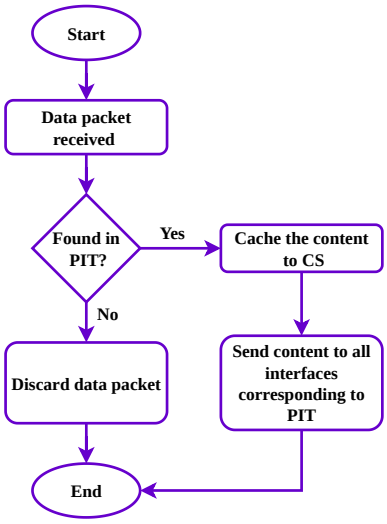

Figure 2. Data packet processing in NDN

replacement algorithms have been presented.

5. System Design

This section discusses the main elements and structure of our suggested approach.

A. Major components

The design of our suggested vehicular network is represented as a graph $G = (V, E)$, in which V stands for vertices, or vehicles, and E for edges, or connections. $V = v1, v2, v3, ..., vn$ represents the vertices, while $E = E1, E2, E3, ..., Em$ represents the edges. The contents that are now available at the server are $F = f1, f2, ..., fn$. In this network, requests are created by the end-user and are responded to by the server. Data packets that are going via an ICN network vehicle (vertex) can be cached. Every data chunk on the network is provided by a single server, V_s. The FIB's forwarding plan determines how requests for these pieces are sent. Every node v_i has a variable power $P(v_i)$ and a finite cache capacity $C(v_i)$, which are updated following the transmission of each request or response packet via v_i.

B. Assumptions

To improve understanding, let's assume each content chunk is uniform and represents the smallest unit for caching within the network. The general assumptions of this work are listed below:

- $\lambda(f_j, v_i)$= Speed of a request traveling from one node to another.
- $P(f_j)$= Local popularity of a content chunk traveling from one node to another.
- $T(t)$= Time of the day. Because request frequencies might vary at different times, with peaks during certain hours.

The request frequency of content chunk f_j at vehicle v_i is represented as:

$$Req(f_j, v_i) = \lambda(f_j, v_i) * P(f_j) * T(t) \tag{1}$$

A random number generator will be used to assign power to each vehicle at random.

C. Optimization model

Our suggested approach seeks to reduce retrieval latency and data redundancy in the network by optimizing content placement on cache-capable vehicles. To improve caching advantage and minimize path length, we address the problem of caching individual chunks on vehicles with different power levels and storage capacities. The notations used in this paper are shown in Table **??**.

Notation	Description
V	Set of all nodes in the network
F	Set of all content chunks
v_i	A node in the network
f_j	A content chunk
Req_{f_j,v_i}	Number of requests for content chunk f_j by node v_i
$H(v_i)$	Number of interfaces of node v_i
$B(f_j, v_i)$	Binary variable indicating if content chunk f_j is cached at node v_i
$\text{Stretch}_{f_j,v_i}$	Number of hops traveled by content chunk f_j to reach node v_i
$\text{size}(f_j)$	Size of content chunk f_j
$C(v_i)$	Storage capacity of node v_i
H_d	Minimum required number of interfaces for a node to cache content
P_i	Popularity score of content item i
R_i	Recency score of content item i
Q_i	Probabilistic factor of content item i
I_i	Incoming interfaces score of content item i
E_i	Eviction score of content item i

Table I. List of Notations

For effective content caching in vehicle networks, we formulate the following optimization function as a maximization problem:

$$\max \left(\sum_{v_i \in V} \sum_{f_j \in F} \left(\text{Req}_{f_j,v_i} \times H(v_i) \times B(f_j, v_i) \right) - \sum_{v_i \in V} \sum_{f_j \in F} \left(\text{Stretch}_{f_j,v_i} \times B(f_j, v_i) \right) \right) \tag{2}$$

Subject to the constraints:

1. Binary Caching Decision:

$$B(f_j, v_i) \in \{0, 1\} \tag{3}$$

Each content chunk f_j at node v_i is either cached ($B(f_j, v_i) = 1$) or not ($B(f_j, v_i) = 0$). Ensures that content is either cached or not at a node.
2. Cache Capacity:

$$B(f_j, v_i) \times \text{size}(f_j) \leq C(v_i) \quad \forall v_i \in V, f_j \in F \tag{4}$$

The total size of cached content at node v_i should not exceed its capacity $C(v_i)$. Ensures that the cached content does not exceed the node's capacity.
3. Single Instance of Content:

$$\sum_{v_i \in V} B(f_j, v_i) = 1 \quad \forall f_j \in F \tag{5}$$

Each content chunk f_j must be cached at exactly one node in the network.

Ensures each content chunk is cached at only one node in the network.
4. Interface Requirement:

$$H(v_i) \geq H_d \quad \forall v_i \in V \tag{6}$$

The interface count of node v_i should be at least H_d. Ensures nodes selected for caching have a sufficient number of interfaces.
5. Eviction Strategy:

$$E_i = \frac{1}{P_i \times R_i \times Q_i \times I_i} \tag{7}$$

The eviction score E_i for each content item i in the cache is calculated based on its popularity score P_i, recency score R_i, probabilistic factor Q_i, and incoming interfaces score I_i. Items with the lowest E_i score are selected for eviction. Uses the calculated eviction score E_i to decide which content to evict, prioritizing items with lower scores for eviction.

By integrating these constraints and the objective function, the proposed algorithm optimizes content caching to enhance network performance, minimize latency, and ensure efficient use of resources.

6. Proposed SH-based caching Strategy

To achieve efficient caching, content chunks should be placed at nodes with more interfaces and enough energy to serve more requests. This helps to reduce content redundancy, path stretch, and bandwidth usage. The proposed strategy considers the number of interfaces of a node in the delivery path to select the optimal cache node. In the proposed work, the packet is forwarded in two directions: upstream and downstream. Upstream direction is when the user sends the request through the interest packet and forwards it upstream to the content producer. Downstream is where packets route from producer to user. We have modified the interest and data packets to identify each node's interface in the delivery path, as shown in Fig. 3.

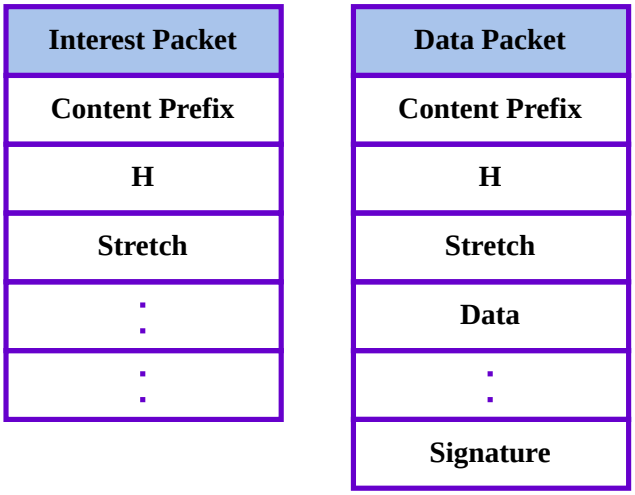

Figure 3. Modified NDN packets

The H field in the interest and data packets represents the number of hops connected to each node, which is the number of interfaces. The Stretch field specifies the number of hops traveled by each packet. The flowchart of the proposed packet forwarding strategy is represented in Fig. 4. The main idea of our proposed strategy is in Algorithm 1 also listed below as different cases:

1) The main objective of the proposed work is to determine a bridge node with a greater number of interfaces in the delivery path.
2) The node with the highest number of interfaces in the delivery path is selected to cache the content.
3) Each time an interest packet visits another node, it follows the proposed upstream rules.

Upstream rules:

- When the interest packet reaches the next node, i.e. next to the requester, the number of interfaces of that node will be added to the H field, and the Stretch field value will be increased by one. Initially, H and stretch values are zero.

$$Stretch_{new} = Stretch_{current} + 1 \tag{8}$$

- The Existing value of the H field will be replaced only if the number of interfaces of the next visited nodes in a path is greater than the existing H value in the interest packet.

$$H_{new} = \max(H_{current}, \text{interfaces of current node}) \tag{9}$$

- : When interest visits the content producer, the H filed will have the highest interface value.
- Increasing the stretch value to one when interest visits a new node helps to identify how many nodes that interest packet is passed to reach the content producer.

4) The data packet follows the proposed downstream rules while traveling from the producer to the user.
Downstream rules:
   - When a data packet is produced at the content producer, the Stretch and H values of the interest packet will be copied to the data packet.
   - When the data packet visits the next nodes in a path, then the value of stretch will be decreased by one. This means that in the same path of interest packet, the data packet is covered by one node, and the remaining value indicates how many nodes remain in the path.

$$Stretch_{new} = Stretch_{current} - 1 \quad (10)$$

   - Also, the H value of the data packet is compared with the current node's number of interfaces.
   - If both the H value and the current node's interface value match, then that node will be selected to cache the content.
   - Since the dynamic nature of the vehicular network, if the number of interfaces doesn't match any node value, then the node that exists next to a user is selected to cache the data. It can be identified when the stretch value becomes zero.

5) When the selected cache node's storage is full, the unwanted data is removed using the proposed cache replacement policy, and new data will be added.

7. Popularity-Recency-Probability-Interfaces (PRPI) Eviction Strategy

We proposed a new cache replacement strategy to evict the unwanted content from the cache when the cache is full. We considered the popularity score, recency score, probability factor, and incoming interface score as a key points to measure the unwanted data from the cache. Including the number of incoming interfaces can add another dimension to the eviction decision, reflecting the network traffic load on each content item. Algorithm 2 explains the proposed PRPI strategy. The key components of our proposed cache placement strategy are listed below:

1) **Popularity Score** (P_i): The popularity score quantifies how frequently a content chunk has been accessed. This metric helps to identify the most requested content within the network. Prioritizing content with higher popularity scores for caching ensures that the most in-demand content is readily available. This reduces retrieval latency as popular content is more likely to be cached near where it is requested, improving overall user satisfaction. By maintaining frequently accessed content within the cache, the system can efficiently handle common requests without repeatedly fetching the same data from distant servers, thus optimizing network performance and resource utilization.

Algorithm 1 Content Caching Based on Node Interfaces

Require: Network of nodes, Interest packets, Data packets
Ensure: Efficient content caching strategy
function ContentCaching(Network, InterestPackets, DataPackets)
  1. Initialization:
  Define H field in interest and data packets to represent the number of interfaces.
  Define $Stretch$ field to specify the number of hops traveled by each packet.
  2. Upstream Process:
  Objective: Place content at a bridge node with the highest number of interfaces in the delivery path.
  Interest Packet Handling:
  for each interest packet visiting a node **do**
    Add the node's interface count to the H field.
    if it is the first node after the requester **then**
      Set the $Stretch$ field value to 1.
    end if
    Replace the existing H value only if the current node's interface count is greater than the existing H value in the interest packet:
    $H_{\text{new}} = \max(H_{\text{current}}, \text{interfaces of current node})$
    Increment the $Stretch$ field by 1 for each new node visited:
    $Stretch_{\text{new}} = Stretch_{\text{current}} + 1$
    At the content producer, the H field will hold the highest interface value, and the $Stretch$ field will indicate the total number of nodes traversed.
  end for
  3. Downstream Process:
  Data Packet Handling:
  for each data packet returning from the content producer **do**
    Copy the $Stretch$ and H values from the interest packet to the data packet.
    Decrease the $Stretch$ field by 1 at each node visited:
    $Stretch_{\text{new}} = Stretch_{\text{current}} - 1$
    Compare the H value in the data packet with the current node's interface count.
    if the H value matches the current node's interface count **then**
      Select this node to cache the content.
    else if no node matches the H value **then**
      Select the node next to the user (identified when the $Stretch$ value becomes zero) to cache the content.
    end if
  end for
  4. Cache Management:
  if the selected cache node's storage is full **then**
    Remove unwanted data as per the proposed cache replacement policy.
    Add the new data to the cache.
  end if
end function=0

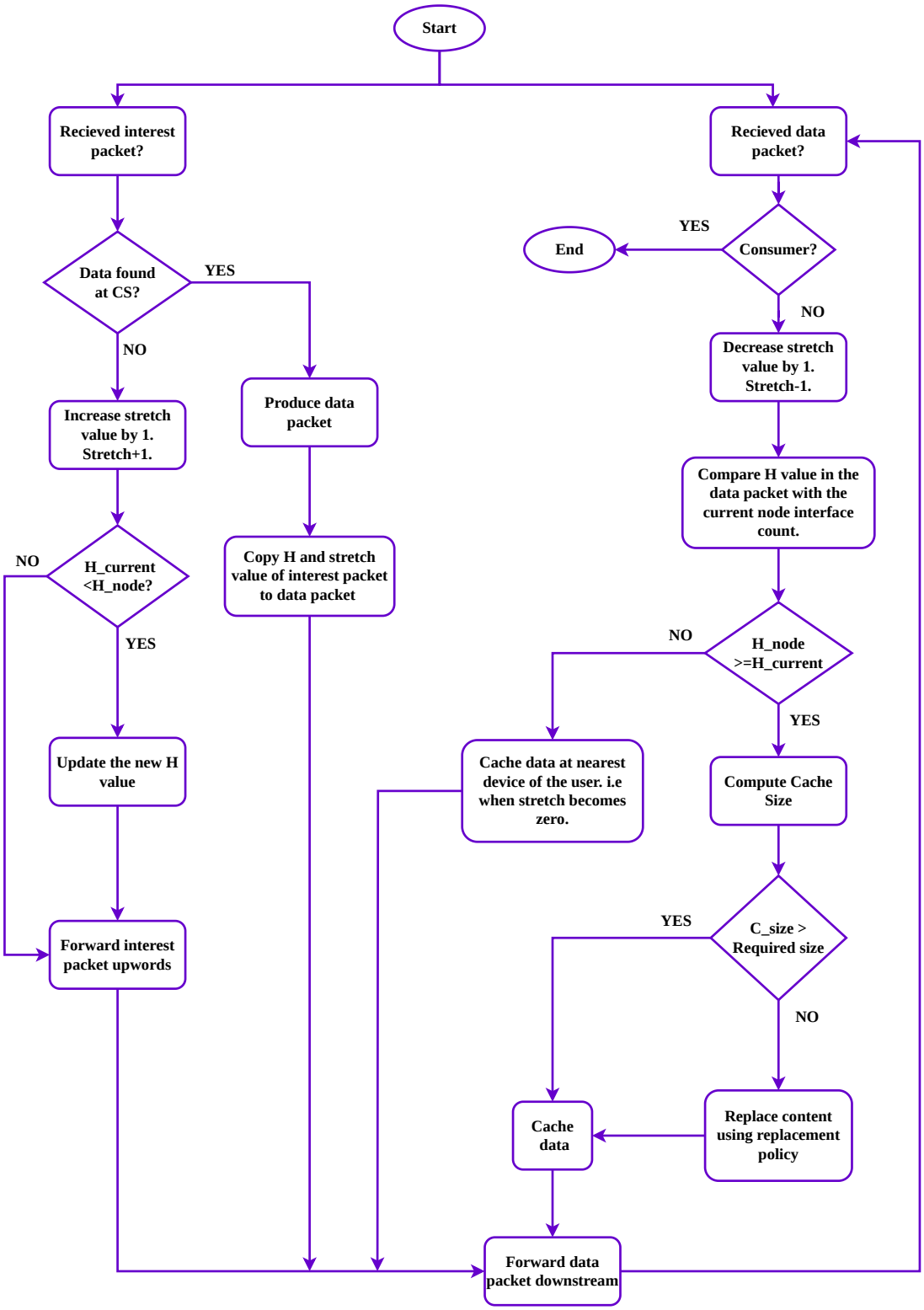

Figure 4. Flow chart of the proposed interest and data packet processing

$$P_i = \frac{\text{Number of accesses of content } i \text{ in the time window}}{\text{Total number of accesses in the time window}}$$

$$P_i = \frac{a_i}{\sum_{j=1}^{N} a_j} \tag{11}$$

where a_i is the number of accesses of content i and N is the total number of content items in the cache.

2) **Recency Score (R_i)**: The recency score indicates how recently a piece of content has been accessed, with higher scores for more recent accesses. Caching content based on its recency helps the system adapt to evolving user preferences and trends. By prioritizing recently accessed content, the cache stays relevant and current, providing users with quick access to the latest information. This method minimizes the storage of outdated content, enhancing the overall effectiveness of the cache.

$$R_i = \frac{1}{t_i} \cdot \frac{1}{\max\left(\frac{1}{t_1}, \frac{1}{t_2}, \ldots, \frac{1}{t_n}\right)} \tag{12}$$

where t_i is the time since the last access of content i.

3) **Probabilistic Factor (Q_i)**: The probabilistic factor adds an element of randomness to the caching decision process, which can change according to a set probability distribution. Incorporating a probabilistic factor into the caching strategy adds an element of diversity, reducing the predictability of eviction decisions. This randomness prevents the cache from becoming too homogenous and ensures a broader range of content is available. It is particularly useful in scenarios where content access patterns are highly variable or unpredictable.

$$Q_i \sim \mathcal{U}(0, 1) \tag{13}$$

Where, Q_i: Represents the random component for content i. $\mathcal{U}$: Denotes a uniform distribution. 0: Lower bound of the uniform distribution. 1: Upper bound of the uniform distribution.

4) **Incoming Interfaces Score (I_i)**: The incoming interfaces score measures the number of different interfaces or sources requesting a particular content chunk. This score reflects the demand for content from multiple points within the network. The caching strategy prioritizes content in demand from multiple sources by considering the number of incoming interfaces. This ensures that cache space is used efficiently by storing content that benefits a larger network portion. By serving content requested from various nodes, the strategy reduces overall bandwidth usage and minimizes path stretch, leading to more efficient content delivery.

$$I_i = \frac{n_i}{N} \tag{14}$$

where I_i represents the score for content i, n_i is the number of different interfaces that have requested content i, and N is the total number of interfaces.

The overall eviction value of each content can be calculated as The overall eviction score (E_i) is calculated as:

$$E_i = \alpha P_i + \beta R_i + \gamma Q_i + \delta I_i \tag{15}$$

where:

- E_i represents the eviction score for content i.
- α, β, γ, and δ are weights determining the importance of each factor.
- P_i, R_i, Q_i, and I_i are the Popularity Score, Recency Score, Probabilistic Factor, and Incoming Interfaces Score for content i, respectively.

Algorithm 2 PRPI Eviction Strategy

```
Input: Cache with content items
Output: Content item to evict
function EvictItem(Cache)
    for each content item i in the cache do
        Calculate Popularity Score (P_i) using Equation 11
        Calculate Recency Score (R_i) using Equation 12
        Generate Probabilistic Factor (Q_i) using Equation 13
        Calculate Incoming Interfaces Score (I_i) using Equation 14
        Calculate Eviction Score (E_i) using Equation 15
    end for
    Select item with the lowest E_i score for eviction
end function=0
```

A. *Example scenario for selecting the content with the least eviction score*

Suppose you have the following content items with their access patterns:

- Content A: accessed 10 times, last accessed 2 minutes ago, requested from 3 interfaces.
- Content B: accessed 5 times, last accessed 1 minute ago, requested from 1 interface.
- Content C: accessed 8 times, last accessed 5 minutes ago, requested from 2 interfaces.

Popularity Scores:

$$P_A = 10$$
$$P_B = 5$$
$$P_C = 8$$

Recency Scores (assuming normalized inverse time):

$$R_A = \frac{1}{2}$$
$$R_B = 1$$
$$R_C = \frac{1}{5}$$

Probabilistic Factors (random values between 0 and 1):

$$Q_A = 0.7$$
$$Q_B = 0.2$$
$$Q_C = 0.9$$

Incoming Interfaces Scores (normalized to 0-1):

$$I_A = 1$$
$$I_B = \frac{1}{3} \approx 0.33$$
$$I_C = \frac{2}{3} \approx 0.67$$

Assuming equal weights for simplicity ($\alpha = \beta = \gamma = \delta = \frac{1}{4}$):

$$\begin{aligned} E_A &= \frac{1}{4}(10) + \frac{1}{4}(0.5) + \frac{1}{4}(0.7) + \frac{1}{4}(1) \\ &= 3.05 \\ E_B &= \frac{1}{4}(5) + \frac{1}{4}(1) + \frac{1}{4}(0.2) + \frac{1}{4}(0.33) \\ &= 1.63 \\ E_C &= \frac{1}{4}(8) + \frac{1}{4}(0.2) + \frac{1}{4}(0.9) + \frac{1}{4}(0.67) \\ &= 2.44 \end{aligned}$$

Content B would be evicted in this example as it has the lowest eviction score.

Benefits:

- Balanced Approach: Combines multiple factors to make a more informed eviction decision.
- Network Demand Consideration: Includes the number of incoming interfaces, which can reflect broader network-wide demand.
- Reduced Predictability: The probabilistic factor introduces randomness to prevent the eviction strategy from being too predictable.

By incorporating these factors, the PRPI eviction strategy provides an efficient approach to cache management, potentially improving cache hit rates and overall network efficiency.

8. Simulation results

To evaluate the proposed work, we used the ICN caching simulator ICARUS [38]. Table II contains a list of all the parameters utilized and the configuration of the system. The Salama method creates a random network structure utilizing an ICN network of 100 nodes. We consider the uniform capacity allocation strategy, which means all cache nodes have the same storage capacity. The content popularity distribution follows Zipf's law, $1/|K|^{\alpha}$, which is widely used in literature. The parameter α varies between 0.6 and 1.00 in the experiments. Higher α values indicate more concentrated user preferences.

Table II. Simulation parameters

Parameters	Default values
Total number of nodes	100
Number of servers	1
Number of cache nodes	30
User nodes	69
Request rate	15 req/sec
Cache size	10-100 MB
α value	0.6 to 1.0

A. *performance metrics*

To analyze the performance of the proposed work, we have chosen the most commonly used performance metrics in the existing works on ICN caching. ICN caching aims to minimize content retrieval delay by efficiently managing the available resources in the network. Therefore, most researchers in existing works are concentrated on the literature on CHR, CRD, and Stretch ratio. Considering the related works, the proposed work considered the following performance metrics.

1) **Cache Hit Ratio (CHR):** CHR is a metric used to evaluate the performance of a cache system by comparing the number of interest packets successfully satisfied by the cache to the total number of interest packets transmitted across the network. This metric effectively measures the efficiency of routers. Equation 16 provides the formula for calculating CHR.

$$\text{CHR} = \frac{\text{cache hits}}{\text{cache hits} + \text{cache misses}} \tag{16}$$

2) **Content Retrieval Delay (CRD):** It refers to the time required to deliver content to a user. This metric, known as delivery time, can be calculated using the formula provided in Equation 17.

$$\text{CRD} = \sum_{i=1}^{C_{\text{total}}} \text{IPD} + \sum_{i=1}^{C_{\text{total}}} \text{DPD} \tag{17}$$

Where $C_t otal$ is the total content chunks requested, IPD is interest packet duration and DPD is data packet duration.

3) **Stretch Ratio (SR):** The number of hops traveled by the interest packet until the corresponding data is found.

$$\text{SR} = \frac{\text{Number of hops traveled}}{\text{Total number of hops from user to data producer}} \tag{18}$$

B. Evaluation results

The result of the proposed work is evaluated against various existing strategies, namely, ProbCache, CL4M, and LCD, which are explained in Section II. We divided the evaluation into three subsections, using three major parameters: content popularity, cache size, and request rate. In this context, the performance of the described algorithms is evaluated by adjusting various parameters that influence performance metrics. These measures include the average delay for content retrieval, the hit rate, and the stretch ratio—the decrease in the number of hops needed to get content.

1) **For varying content popularity** We assessed the effectiveness of several algorithms by manipulating the Zipf-Mandelbrot distribution's α skewness index, which signifies that requests focus on a more limited set of content, thereby making that content more popular. Fig. 5 shows the evaluation result of the proposed work in terms of CHR for varying popularity index. The graph indicates that when we increase the popularity factor, then the CHR of all the caching strategies increases. This shows that more requests are focused on fewer items, making those items more popular. Compared to existing strategies, the proposed work indicates better performance.
The efficiency of the proposed work is calculated by determining the average relative improvement. The average relative improvement is found by taking the mean of the relative improvements across all data points for each comparison. The average relative improvements of the proposed method (CHR ratio) over the other methods are as follows: proposed method over ProbCache: (27.76%), proposed method over CL4M: (28.00%), proposed method over LCD: (33.78%). Fig. 6 shows the content retrieval delay for different popularity values. When the popularity parameter increases, the delay decreases. This shows that when the popular content becomes less, it easily stores those popular contents in the cache and can satisfy many requests. The popularity parameter becomes a major factor in the proposed policy as we calculate the eviction score to replace the content by using popularity as one of the major factors. The suggested policy inspects the number of interfaces and stretch ratio for select-

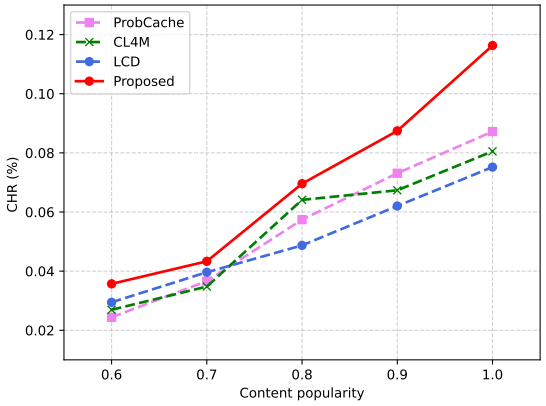

Figure 5. CHR for different α values

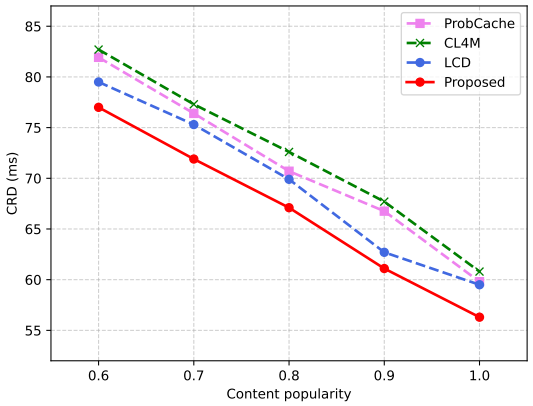

Figure 6. CRD for varying α values

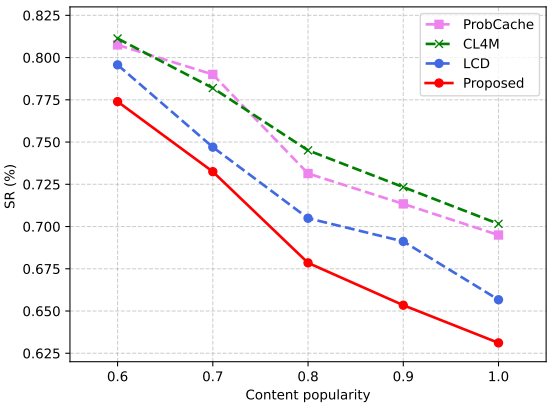

Figure 7. SR for varying α values

ing the best node for caching the content, which enables the selection of the efficient node in the network so that it can satisfy many requests. Whereas the ProbCache only considers the probability value to cache the content, the LCD leaves the content downstream without considering the node's position, and CL4M focuses on graph-based centrality. Hence, the proposed strategy yields better results than the existing methods. The average relative improvements of the suggested method over the other methods in terms of CRD are as follows: Proposed method over Probcache: 6.27% Proposed method over CL4M: 7.72% Proposed method over LCD: 3.92%. These values represent the average percentage reduction in the Content Retrieval Delay of the recommended method compared to the other methods across all the data points. This indicates that the proposed method achieves a lower CRD, thus demonstrating improved efficiency in terms of content retrieval time.

Fig. 7 depicts the stretch ratio for the increasing popularity parameter. The proposed work selects the efficient cache using the PRPI eviction policy and stores content at the bridge node. It decreases the number of hops required to travel in the delivery path.

The average relative improvements of the suggested method over the other methods regarding SR are as follows: Proposed method over ProbCache: 7.25% Proposed method over CL4M: 7.92% Proposed method over LCD: 3.55% These values represent the average percentage reduction in the Stretch Ratio of the proposed method compared to the other methods across the evaluated data points. This indicates that the proposed method achieves a lower Stretch Ratio, thus demonstrating improved efficiency in the number of hops traveled.

2) **For varying cache size** Fig. 8,9, and 10 represent the process of adjusting performance metrics for different schemes by varying the cache size in network nodes. The proposed workplaces content at nodes with the highest number of interfaces in the delivery path, and the replacement algorithm ensures that frequently accessed content is cached at nodes with higher connectivity. This increases the likelihood that future requests can be served from these high interface nodes, reducing the average content retrieval delay. LCD strategy copies data down to the node closest to the requester, potentially leading to redundant caching and inefficient use of cache space. Probcache and CL4M place content probabilistically and based on the graph centrality along the path, leading to random placement of content and non-optimal cache hits. Hence, the proposed policy gives better CHR, CRD, and SR results when we increase the cache size.

The average relative efficiency of the suggested technique when we increase the cache size over the other methods in terms of CHR is 17.83% more efficient than ProbCache, 36.54% more efficient than CL4M, and 9.94% better than LCD. The average

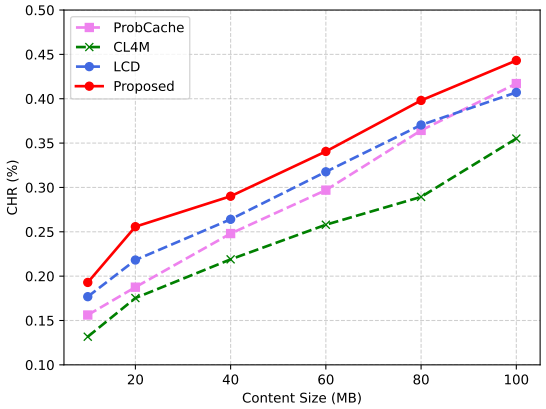

Figure 8. CHR for different cache size

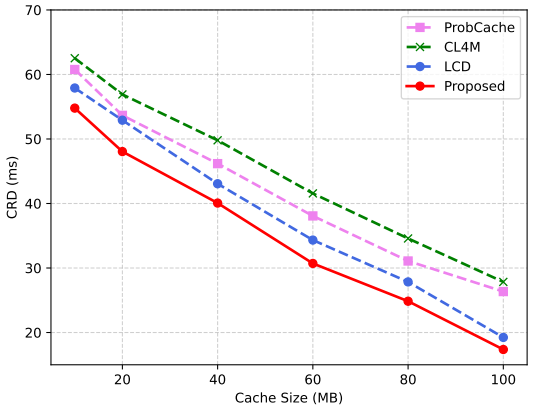

Figure 9. CRD for different cache size

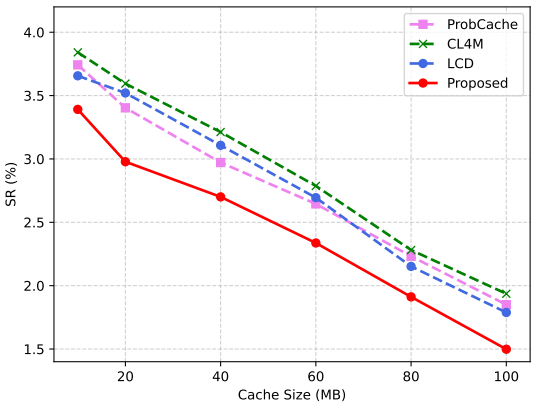

Figure 10. SR for different cache size

relative efficiency of the proposed technique over the other methods in terms of CRD when we vary the cache size is 17.83% more efficient than Probcache, 23.22% more efficient than CL4M, and 8.75% efficient than LCD. The average relative efficiency of the suggested method over the other methods in terms of SR for varying cache size is 12.67% efficient than ProbCache, 16.63% efficient than CL4M, and 12.73% efficient than LCD.

3) **For varying number of requests** When we increase the number of requests, the packets generated in the network also increase. In the proposed work we consider the interface count and strategically placing content at high-connectivity nodes, the retrieval delay is minimized as these nodes are likely to serve more future requests efficiently. The proposed work's stretch and H field mechanism allows dynamic adjustments based on current network conditions, further reducing the delay.

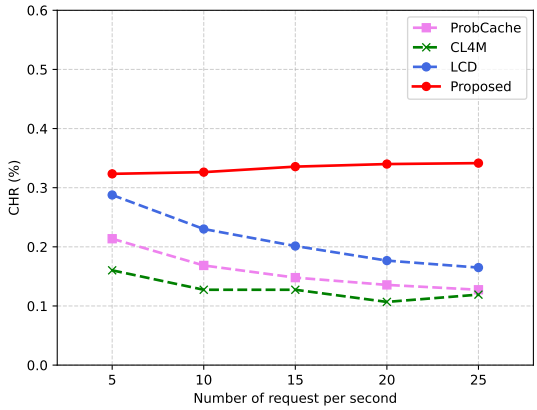

Figure 11. CHR for different number of requests

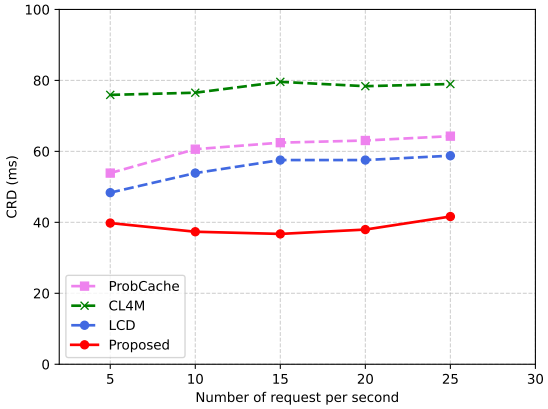

Figure 12. CRD for a varying number of requests

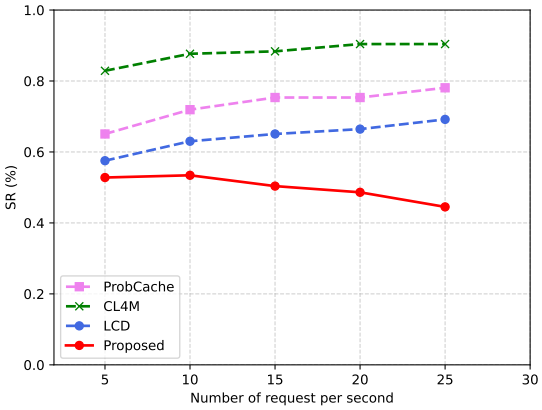

Figure 13. SR for different number of requests

The proposed cache replacement policy combines popularity, recency, probabilistic factors, and incoming interfaces. This eviction strategy ensures that the most relevant content stays in the cache. This reduces the need to fetch content from the source and delays in retrieval. This strategy adapts better to varying traffic patterns in vehicular networks, maintaining a more efficient cache. The existing LCD, ProbCache, and CL4M strategies do not optimize for node connectivity and use simpler eviction strategies, which can lead to higher retrieval delays, lower hit ratio, and lower stretch ratio. Compared to other methods, the proposed work satisfies all requests and reduces the number of hops and time taken to travel. Fig. 11, 12 and 13 indicate the better results of the proposed work compared to existing methods.

9. Conclusion

The rapid increase of data traffic in vehicular networks increases the content retrieval delay when consumer requests travel to the producer every time. ICN's in-network caching feature enables the storage of a copy of the content in the network, which decreases the content retrieval delay and increases content availability in the network. In this paper, we proposed an SH-based caching strategy that selects the cache node using each node's interface count in a delivery path. This strategy places the content at a more connected node in the path that can satisfy several requests. Further, the PRPI eviction policy is developed to evict outdated content when the cache is full. PRPI cache replacement considers the content's popularity, recency, probability, and interfaces to detect unwanted content from the cache. The proposed work is evaluated through simulation against ProbCache, CL4M, and LCD strategies regarding CHR, CRD, and SR. The simulation results indicate that the proposed work achieves better results than the existing methods. The future direction of the proposed work includes enhanced Cache Selection Mechanisms. While the proposed SH-based caching strategy utilizes interface

count to select cache nodes, future research could explore more sophisticated metrics for cache node selection. Energy efficiency is critical in vehicular networks, especially for battery-powered devices. Future work could also explore energy-efficient caching strategies that minimize the power consumption of cache nodes.

References

[1] H. Al-Omaisi, E. A. Sundararajan, R. Alsaqour, N. F. Abdullah, and M. Abdelhaq, "A survey of data dissemination schemes in vehicular named data networking," *Vehicular Communications*, vol. 30, p. 100353, 2021.

[2] P. K. Singh, S. K. Nandi, and S. Nandi, "A tutorial survey on vehicular communication state of the art, and future research directions," *Vehicular Communications*, vol. 18, p. 100164, 2019.

[3] J. Yangjie, Z. Zewei, Y. Ziru, Y. Huang, Y. Zhang, W. Zhang, L. Xiong, and Y. Zhuoping, "Towards autonomous vehicles: a survey on cooperative vehicle-infrastructure system," *Iscience*, 2024.

[4] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1243–1274, 2018.

[5] A. A. Prates, I. V. Bastos, and I. M. Moraes, "Geozone: An interest-packet forwarding mechanism based on dissemination zone for content-centric vehicular networks," *Computers & Electrical Engineering*, vol. 73, pp. 155–166, 2019.

[6] C. Pruthvi, K. Yashitha, R. Rekha, H. Vimala, J. Shreyas *et al.*, "Information-centric caching solutions for vehicular networks: A survey," in *2023 4th International Conference on Computation, Automation and Knowledge Management (ICCAKM)*. IEEE, 2023, pp. 1–6.

[7] C. Pruthvi, H. Vimala, J. Shreyas, and S. Devayya, "Icn based cooperative edge caching policy for transient iot data," 2023.

[8] M. Amadeo, "A literature review on caching transient contents in vehicular named data networking," in *Telecom*, vol. 2, no. 1. MDPI, 2021, pp. 75–92.

[9] L. V. Yovita and N. R. Syambas, "Caching on named data network: a survey and future research." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 8, no. 6, 2018.

[10] Z. Lin, M. Kuai, and X. Hong, "Reliable forwarding strategy in vehicular networks using ndn," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2016, pp. 1–5.

[11] H. Khelifi, S. Luo, B. Nour, and S. C. Shah, "Security and privacy issues in vehicular named data networks: An overview," *Mobile Information Systems*, vol. 2018, no. 1, p. 5672154, 2018.

[12] H. Khelifi, S. Luo, B. Nour, and H. Moungla, "In-network caching in icn-based vehicular networks: Effectiveness & performance evaluation," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[13] M. S. Zahedinia, M. R. Khayyambashi, and A. Bohlooli, "Fog-based caching mechanism for iot data in information centric network using prioritization," *Computer Networks*, vol. 213, p. 109082, 2022.

[14] D. Man, Q. Lu, H. Wang, J. Guo, W. Yang, and J. Lv, "On-path caching based on content relevance in information-centric networking," *Computer Communications*, vol. 176, pp. 272–281, 2021.

[15] F. Khandaker, S. Oteafy, H. S. Hassanein, and H. Farahat, "A functional taxonomy of caching schemes: Towards guided designs in information-centric networks," *Computer Networks*, vol. 165, p. 106937, 2019.

[16] A. Reshadinezhad, M. R. Khayyambashi, and N. Movahedinia, "An efficient adaptive cache management scheme for named data networks," *Future Generation Computer Systems*, vol. 148, pp. 79–92, 2023.

[17] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1473–1499, 2015.

[18] N. Laoutaris, H. Che, and I. Stavrakakis, "The lcd interconnection of lru caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.

[19] N. Lal, S. Kumar, G. Kadian, and V. K. Chaurasiya, "Caching methodologies in content centric networking (ccn): A survey," *Computer Science Review*, vol. 31, pp. 39–50, 2019.

[20] D. Gupta, S. Rani, A. Singh, and J. J. Rodrigues, "Icn based efficient content caching scheme for vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[21] I. U. Din, B. Ahmad, A. Almogren, H. Almajed, I. Mohiuddin, and J. J. Rodrigues, "Left-right-front caching strategy for vehicular networks in icn-based internet of things," *IEEE Access*, vol. 9, pp. 595–605, 2020.

[22] D. Gupta, S. Rani, B. Tiwari, and T. R. Gadekallu, "An edge communication based probabilistic caching for transient content distribution in vehicular networks," *Scientific Reports*, vol. 13, no. 1, p. 3614, 2023.

[23] C. Wang, C. Chen, Q. Pei, N. Lv, and H. Song, "Popularity incentive caching for vehicular named data networking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3640–3653, 2020.

[24] C. Wang, C. Chen, Q. Pei, Z. Jiang, and S. Xu, "An information-centric in-network caching scheme for 5g-enabled internet of connected vehicles," *IEEE Transactions on Mobile Computing*, vol. 22, no. 6, pp. 3137–3150, 2021.

[25] N. Aung, S. Dhelim, L. Chen, A. Lakas, W. Zhang, H. Ning, S. Chaib, and M. T. Kechadi, "Vesonet: Traffic-aware content caching for vehicular social networks using deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[26] B. Feng, C. Feng, D. Feng, Y. Wu, and X.-G. Xia, "Proactive content caching scheme in urban vehicular networks," *IEEE Transactions on Communications*, 2023.

[27] M. Y. Khan, M. Adnan, J. Iqbal, B.-H. Roh, J. Ali *et al.*, "A hierarchal clustered based proactive caching in ndn-based vehicular network." *Computer Systems Science & Engineering*, vol. 47, no. 1, 2023.

[28] S. K. u. Zaman, S. Mustafa, H. Abbasi, T. Maqsood, F. Rehman, M. A. Khan, M. Ahmed, A. D. Algarni, and H. Elmannai, "Cooperative content caching framework using cuckoo search optimization

in vehicular edge networks," *Applied Sciences*, vol. 13, no. 2, p. 780, 2023.

[29] Y. Nam, H. Choi, Y. Shin, and E. Lee, "Traffic optimized content precaching scheme based on tolerable delay time in content-centric vehicular networks," *IEEE Access*, 2023.

[30] H. Tokunaga and S. Tang, "Efficient v2v communications by clustering-based collaborative caching," *Electronics*, vol. 13, no. 5, p. 883, 2024.

[31] V. Sampath, S. Karthik, and R. Sabitha, "Position-based adaptive clustering model (pacm) for efficient data caching in vehicular named data networks (vndn)," *Wireless Personal Communications*, vol. 117, no. 4, pp. 2955–2971, 2021.

[32] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, "Named data networking: a survey," *Computer Science Review*, vol. 19, pp. 15–55, 2016.

[33] B. Nour, K. Sharif, F. Li, H. Khelifi, and H. Moungla, "Nncp: A named data network control protocol for iot applications," in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2018, pp. 1–6.

[34] B. Nour, K. Sharif, F. Li, H. Moungla, A. E. Kamal, and H. Afifi, "Ncp: A near icn cache placement scheme for iot-based traffic class," in *2018 IEEE global communications conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[35] C. Pruthvi, H. Vimala, and J. Shreyas, "A systematic survey on content caching in icn and icn-iot: Challenges, approaches and strategies," *Computer Networks*, vol. 233, p. 109896, 2023.

[36] Y. Hua, L. Guan, and K. G. Kyriakopoulos, "A fog caching scheme enabled by icn for iot environments," *Future Generation Computer Systems*, vol. 111, pp. 82–95, 2020.

[37] K. Geethakumari, H. Vimala, and C. Pruthvi, "Ecins: Efficient caching for icn-iot based on node score," in *2023 IEEE 3rd Mysore Sub Section International Conference (MysuruCon)*. IEEE, 2023, pp. 1–6.

[38] L. Saino, I. Psaras, and G. Pavlou, "Icarus: a caching simulator for information centric networking (icn)," in *SimuTools*, vol. 7. ICST, 2014, pp. 66–75.