Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 6539532, 13 pages
https://doi.org/10.1155/2022/6539532

Research Article

Dynamic Naming Scheme and Lookup Method Based on Trie for Vehicular Named Data Network

M. Wasim Abbas Ashraf,[1] **Chuanhe Huang**,[1] **Shehzad Khalid**,[2] **Amir Saeed Rana**,[3] **Mudassar Ahmad**,[4] **and Umar Raza**[5]

[1]*School of Computer Science, Wuhan University, Wuhan 430 072, China*
[2]*Department of Computer Engineering, Bahria University, Islamabad, Pakistan*
[3]*Department of EFS, University of Agriculture, Faisalabad, Pakistan*
[4]*Department of Computer Science, National Textile University, Faisalabad, Pakistan*
[5]*Department of Engineering, Manchester Metropolitan University, Manchester, UK*

Correspondence should be addressed to Chuanhe Huang; huangch@whu.edu.cn

Received 25 January 2022; Accepted 4 May 2022; Published 19 May 2022

Academic Editor: Farhan Ullah

Content naming and lookup are decisive functions of the future architecture named data network (NDN). The core concept of NDN is the content distribution between consumers and content providers. The NDN supports advance vehicular networks that is famous with vehicular-named data network (VNDN) with different naming schemes such as hybrid, flat, attribute-based, and hierarchical names. These schemes are used in a static way for vehicular network, in summary, the hybrid, flat, and attribute-based makes a complex structure, and on the other hand, hierarchical names long in length and name lookup performance are a bottleneck in NDN, which can directly affect the network performance. Therefore, we introduce a dynamic naming scheme and lookup method (DNSL) for VNDN to mitigate these issues. We argue that the dynamic naming scheme is a better approach to VNDN, while the static name is a cost-effective, hefty, integrated fashion, and improper for the vehicular network. This study focuses on (1) a dynamic naming scheme using dynamic-tag and (2) a lookup method based on node partition of trie; the trie approach is very famed in data structure and extensively used for the lookup content, insertion, and deletion processes. Our experimental evaluation shows that the DNSL scheme is highly efficient, scalable, and provably correct for VNDN.

1. Introduction

Coincidentally, NDN is a role model of the information-centric network (ICN) that allows users to request data without knowing the host node. The core focus of NDN is data, identified by names instead of nodes or devices in the networks. In the NDN architecture, all devices keep up information with the NDN table's structure, which commonly uses the content table, pending information, and forward information tables. Typically, VNDN covers these mechanisms like NDN vehicle-to-infrastructure, vehicle-to-vehicle, vehicle-to-everything, and infrastructure-to-vehicle [1].

Inspired by content, the named data networking (NDN) [2] is a new data communication approach that will replace the current IP infrastructure, which can help achieve performance compared to IP-based vehicular networks. Now, much-emerging applications and content distribution networks rely on named-based services for content forwarding and fetching by names. Therefore, a name lookup, insertion, and updates are the core function of the name-based vehicular networks. The vehicular network (VN) is the extended version of mobile ad hoc networks to provide moving vehicles' communication services. Moreover, in [3], VN is very efficient for the large-scale distribution system and many road safety systems. The current trend of VN's communication is entirely based on IP addresses, which may have some issues due to IP limitation. The VN environment requires a vital communication link without session breakage. Hence,

in this regard, VNDN provides a reliable and efficient communication architecture for sharing any information between moving vehicles in the traffic scenario.

The namespace is a crucial component of NDN. The NDN adopts a hierarchical naming structure that is human readable. It seems like a uniform resource locator URL with separate "/." The benefits of the hierarchical structure are the compatibility with the system and minimize the routing aggregation, which helps to improve the search for the routing table. However, the drawback of this scheme has a long length. The hierarchical scheme is not suitable for that network, which already has a long component, and it becomes tough to remember and consumes more memory. On the other hand, a flat naming scheme is introduced and uses in future architecture of information-centric networks (ICN) like MobilityFirst, NetInf, and DONA. The flat naming scheme generates by different encryption-based [4] methods. The flat scheme is globally unique but reduces the aggregation process. Using a flat scheme increases the size of the routing table and may be challenging to read, which requires an additional cost to read it. The suitable naming scheme is still a critical issue for all NDN-based networks and unique in VN.

The vehicular named data network (VNDN) has newly received significant attention in the research community. A name-based lookup method is essential. However, several crucial issues and challenges related to name-based lookup are yet to be addressed to successfully realize a content-oriented network model for the VNDN and the future's internet. The name prefix is considered longer than the IP prefix [5]. The lookup time for the name's unbounded length may take more time and increase the forwarding table's size compared to the IP prefix in a vehicular network. A name-based lookup scheme faces significant problems. As per the NDN rule, the packet's reputation depends on the application naming method for every request identified by a unique name.

Lookup is a core function of NDN to scan hundreds and millions of characters to find the long prefix matching (LPM) in the forwarded information base (FIB). The name length is correlated to the lookup time, which has still a big challenge to achieve the lookup speed [6]. Many naming schemes (flat, hierarchical, and hybrid) are presented to improve the lookup time, but FIB forwarding tables could not give desired results, and the static naming scheme not suitable for VNDN. Therefore, the existing baseline lookup and naming methods based on trie are less efficient as per the above discussion in achieving unique prefix naming and lookup method for the dynamic scenario in VNDN. Furthermore, the trie can play a crucial role in forwarding tables [7] to get the desired content. This data structure supports both the longest prefix match (LPM) and the exact match (EM). Trie is also beneficial to update and removal function, which is the reverse of the insertion [8]. The challenges of existing studies and the benefits of trie motivated us to design a unique naming and fast lookup method to improve the performance of the whole network in VNDN.

In this paper, we address the content naming scheme and name lookup issues for VNDN. We propose a dynamic naming scheme and lookup method, efficient for the VNDN environment, and offers a content lookup guarantee. The dynamic naming scheme uses shared features of hierarchical and hashed naming methods. At the same time, the lookup method is based on node partition of Patricia trie (PT). Moreover, our proposed method does not require any additional computational expenses to manage the trie structure. Our proposed work is specially designed to meet the following objectives:

(i) We design the novel trie based on node partition using compact trie

(ii) The dynamic tag is introduced in this study used for VNDN, which is unique to identify the producer node and the content itself

(iii) We improved the content lookup process and memory usage performance via node partitions that merge the same prefix bytes on a single node. Moreover, which helps to improve the overall network performance

(iv) We also provide a quantitative comparison of our DNSL scheme with the existing approaches

The existing naming approaches and lookup methods based on the trie for vehicular are discussed in Section 2. Section 3 presents the system model and interest process for VNDN. Section 4 introduces the proposed scheme and the evaluation of proposed work with other states of the art. Finally, we conclude our work in Section 5.

2. Related Work

2.1. Naming Schemes. We first address the proposed naming schemes presented in [9], a proposed naming system for vehicular network traffic details, as this paper, focuses on the naming scheme and lookup process. The following is the naming scheme: "/traffic/geolocation/timestamp/datatype/nonce." The author uses the traffic variable as the identifier of the application, current location as road-ID and segment value, timestamp as the time date, data type as the type of data, and nonce as the nonce to prevent interest duplication. To simplify NDN names, the author invented the hash encoding scheme [10] for vehicular networks. The procedure is divided into two parts: (a) compress the name and (b) lookup procedure used the Wu-Manber method, which is expensive due to its complexity.

Communication in ICN uses content information instead of the geolocation and address of the interest source [11]. Inspired by the content's characteristics, the smart house is introduced in [12], which gives an idea about a smart house idea using a hierarchical naming structure based on the NDN architecture. As per a naming scheme structure, a component has specified the activity action and sensing. The subcomponent identifies which exact action will be executed to turn off the light, get its temperature, and get the mac address of the terminal devices inside the indicates by location components. The names in this system are just too long.

The NINQ (name-integrated query) systems introduced a hybrid scheme [13] for the NDN architecture. This scheme is divided into three parts: (1) hierarchical namespace, (2) flat portion based on hashing, and (3) query, which used command and interest as a satisfaction rate and typical latency, use of energy, and volume of data in a network; this scheme seems promising; the namespace and request procedure may be too much longer.

A similar scheme is suggested in [14] in which hierarchical names for reliable models are used to ensure autonomous vehicles' validity in producing data. The scheme follow the format as/applicationprefix/typeof data/datalocation/namemarker/vehiclename/timestamp. The prefix of application is a component that specifies the application name, which generates a data type element that signifies the produced data type and the current position of the component that indicates the content's location. For keeping the historical detail of the vehicle movement, the label is used to identify the starting and ending points. In this strategy, the location may have numerous name components that make redundancy and trouble in access.

The global name scheme is proposed using services-based architecture (SEVen) [15]. The global name is the keystone of this architecture [16]. This naming scheme consists of uppermost groups that indicate three main classes: (1) services for safety, (2) transfer information, and (3) infotainment. Prioritization of content is based on the class level. Applications have identified the type and group of the service into class levels. In conclusion, the name identifier and metadata restrict discrepancies that result in demanding data from the various neighboring producers.

To enhance the names lookup, the author designed the adaptive prefix Bloom filter (NLAPB) [17] and uses a Bloom filter to match the first part of a naming scheme, and a simple trie processes the other part of the scheme.

The similar work presented with a hybrid scheme [18] for vehicular networks (VN's) in ICN takes advantage of flat attributes based on hash method and hierarchical naming structure. The naming format is distributed with three chunks: (1) firstly, the name is used as an identifier of VN; (2) the next part, which has the detail of consumers based on hierarchy scheme, this portion supports simplifying the routing information and name aggregation; and (3) the third or last portion is a flat portion based on the Base64 format to contain owner signature or item information using the hash method. The flat portion promises content integrity. This scheme is theoretical and needs a detailed feasibility analysis. Furthermore, the proposed scheme has some drawbacks, containing long variables and the absence of a name length restriction, resulting in a bulky prefix table and a consumer takes much time to search the contents. Meanwhile, in [19], we improve this study using a managing scheme based on compact trie (CT) for VN. Moreover, the authors examined VN imitation outcomes and determined that the proposed scheme improves space consumption and lookup. The hybrid naming scheme has many benefits rather than hierarchical or flat. However, a critical combination may make a more complex scheme and increases lookup time because of rehashing. Thus, a lithe and suitable naming scheme is required yet for vehicular networks based on NDN.

2.2. Name Lookup Method. In PlusBitmap Caching (PBC) [20], the first lookup request/com/yahoolindexll, since the cache is already empty, the corresponding prefix is admitted to the cache after the main table examine. If no false positives occur when checking the bitmap, the second lookup request/com/yahoolnews returns a cache hit. PBC accepts the third lookup request/com/alibabalindex with a leaf flag since it misses the cache but fits a leaf prefix in the main table. If there are no false positives when checking the bitmap, the fourth lookup request/com/yahoolindexll also hits the cached entry (/come/yahool∗, Rl). As per our knowledge, PCB achieves the cache performance and increases the trie height due to bitmap structure.

The NCIS [21] method adopts dual DHT rings to classify the contents elements. The first portion of this method is based on names, while the other ring is based on contents. Moreover, a pair of keys and values are used for a table. The key represents a content's name, and a value is considered a pointer to the contents. The pointer acts like the lookup service to the entry point, takes NDN-based names as input, and then pushes it back to the contents. It might be possible that if the NDN name fails to be routed, then the pointer will carry the prefix information for routing. A name-based node serves the push-it-back contents and handles the keys/value pair. The ring, based on contents, provides the functions in a similar way to conventional DHTs. The distinction is that the name/other names pair is often pushed back to the name-based ring by the content-based one. This technique joins names with the content of hashes. NCIS is still in its infancy phase and needs to implement on the NDN structure for content lookup and sharing.

The authors suggested BF-PDT [22] name search strategy, which integrates the Bloom filter, popularity graph, and the tire properties. The flowering filter in BF-PDT will help to find out the variables in the application. Meanwhile, the popularity graph is dependent on the content-centric network characteristics and trie properties to improve the trie structure. They also conducted tests for BF-PDT, demonstrating that BF-PDT improves the searching speed at reduced memory cost.

To provide NDN-based location data retrieval, the author suggested a hierarchical name data structure in [23]. The safety info is broadcast in vehicular networks via the publisher and subscriber; the author effort in [24] uses NDN and the hierarchical naming scheme.

To improve efficiency, the authors created a list of localized hops. Subsequently, a piece of information is likely to share with a prefix with earlier data, and the skip list (SL) checks start nearly the node of that prefix that was earlier node. This proposed method eliminates the significant node redundancy for lookup time [25]. As a result, it significantly decreases the searching time. The assessment results indicate that this design achieves a performance compared to the initial design, but the proposed design is complex to implement in reality.

The lookup speed can be achieved by port information in the trie. The study introduced P-Trie [26] based on is based on the trie. Each node has three portions: (i) acceptance (store last bit information of prefix), (ii) advanced node (the node that holds the prefixes of its leaf node that have a similar outgoing port but no one can further be

divided like a child node, and (iii) the intermediary node (holding others nodes). When an identical node arrives at the advanced node, the port's component is given back instantly. Meanwhile, if an identical node occurs on the intermediary node or acceptance node, the process will continue, and the succeeding bits will be compared. On the other hand, if a mismatching happens anywhere, the port push it back to all elements, and the lookup process starts. The proposed trie increase computational cost and not good for VN.

Another trie-based method is proposed to resolve the longest prefix matching (LPM) issues in NDN [27]. The leaf node is merged with their rooted node to achieve memory and processing time performance. The author impresses by the following reasons: (i) binary trie (it can be easy to compress), (ii) use bit string (can be processed quickly in any naming system), and (iii) namespace (name is used for direct lookup and forward it without parsing). The author has used the longest prefix classification (LPC) based on dual binary trie instead of longest prefix matching (LPM). It works like LPM but stores only first-level components rather than whole components' names. A secured method is introduced to extract the hybrid feature in [28], and the deep learning model is designed for different application storages in IoT devices [29].

The complex name strategy takes more processing time in the FIB table [30, 31] than IP-based networks because of the unbounded length of names. Name processing is a core issue in the forwarding engine and needs to reduce the NDN structure's table size. The global services demand a faster lookup method in the forwarding table of NDN.

3. Proposed DNSL For VNDN

This section discusses the system model and presents the proposed naming scheme and name lookup method using dynamic tag. We aim to improve the overall performance of the lookup process, insertion, and deletion entries in the FIB table and reduce the memory using the DNSL scheme.

3.1. System Model. We consider a vehicular network by a graph $G = \{N, C\}$ set of vehicle nodes N, and C is the connections between the two $(n_a, n_b) \in N$ vehicles, where (n_a, n_b) indicates the number of vehicles the moving in a different location. Each node exchanges the packets (interest and data), and each packet carries the content information. A desired content can get from the producer node D_n or content server C_s.

3.1.1. Packet Process. When a request arrives at N, it first checks the local cache; if matched then forward to the requested node. If the content is not matched in local cache, then it will check in PIT; if content is found then add the interface information and discard it. Otherwise, the content can be taken from C_s via FIB.

3.2. Proposed Naming Scheme. Dynamic naming offers a flexible approach for VNDN to maintain communication and record of moving vehicles based on ICN. Our proposed DNSL scheme reduces the memory cost and achieves the desired lookup performance in the routing table that is shown in Figure 1. Firstly, we introduce the basic naming format for a vehicular network and then presented the proposed DNSL scheme's structure and names lookup method for VNDN.

3.2.1. Naming Scheme Format. The naming scheme is an essential part, which helps to implementation in the NDN. This section gives an overview of the NDN naming scheme based on a hierarchical structure. Each application has its proprietary namespaces, and then, the publisher generates the desired data using a namespace giving to prearranged policies. A general naming format of NDN is shown in Figure 2, which is divided into three parts: (a) the first consist of the vehicle information; (b) the second part keeps the content information, which is stored in the content table to serve the consumer request; and (c) used for authentication purpose.

Both the consumer and publisher first register their namespaces to join the NDN network. Each publisher is declaring a list of contents, which they can provide to consumers after registration. The consumers are generating request with their namespace, such as "V117/Wasi/video/aa.mp4/(L1, RSU1)/...," where V117 indicates the vehicle identity, "Wasi" represents the user information, video belongs to a type of content, and aa.mp4 shows the subtype of content, and (L1, RSU1) signifies the current status with location and roadside unit identity of the consumer and publisher. The novelty of the DNSL scheme provides many advantages for VNDN based on ICN.

(1) Static part: in VNDN, the static prefix remains constant once the consumer generates it. The static part must be unique in the VNDN networks. Content naming is played the role of uniqueness in VNDN

(2) Dynamic tag: we proposed a dynamic tag, which is change by the movement of vehicles and gives the current status of vehicles

(3) Dynamic tag helps aggregation, reduces the routing table entries, and improves lookup speed compared to CT scheme [19] and others (seen in Section 4). This portion of our proposed scheme also allows the facility to keep track of vehicle movements with the NDN architecture. Figure 1 shows the structure of the FIB table using a dynamic naming scheme

Figure 1 first stores unique static prefix using the NDN storage structure and a second portion consist of dynamic tag and captures the current position of moving vehicle, and the roadside unit (RSU) acts as a next hop, when a new request arrives, by default is stored in the stack. Then, it performs more queries operations; the most top record is returned by default from the stack of FIB. When RSU collects the payback data, the removal process begins after getting the data from the consumer. Our proposed works reduce the routing table size in the same interest routing; the record lookup and storage based on PT are discussed in Section 3. Our proposed DNSL scheme uses a hierarchical namespace and converts it into twofold chunks of the static and dynamic parts shown in Figure 1.

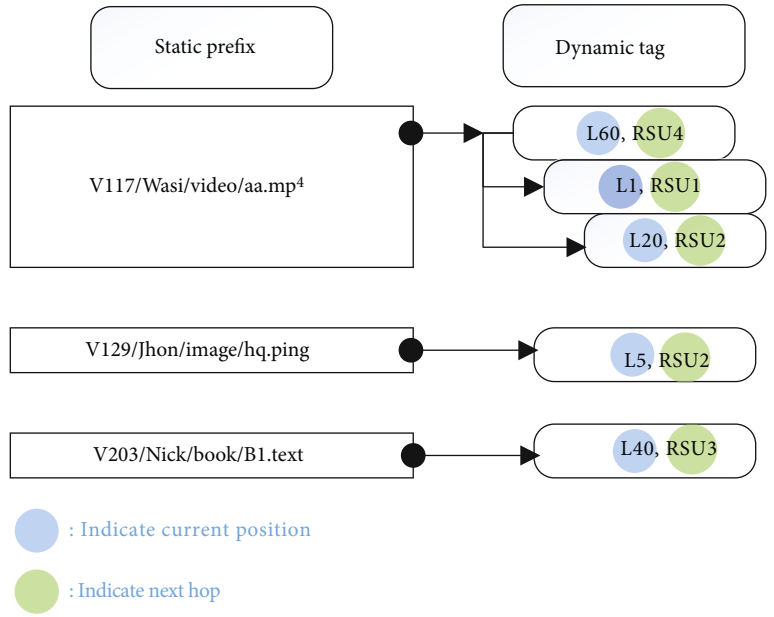


Figure 1: Dynamic naming process in FIB.

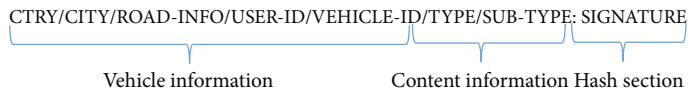


Figure 2: NDN naming format.

Figure 3 shows the comparison of the static naming scheme and dynamic naming scheme in which our proposed dynamic tag is associated with the dynamic scheme. Moreover, 1, 2, and 3 indicate that a single dynamic name can be used for multiple dynamic tag, improving lookup efficiency and reducing memory usage rather than a static naming scheme. Furthermore, Figure 3(b) shows the dynamic and static naming schemes using the same contents in the FIB table, and we found redundancy in the static scheme, whereas the dynamic restricts the redundancy using dynamic tag. Moreover, in Figure 3(a), our proposed scenario illustrates with dynamic naming for VNDN, in which each vehicle first registered and generated its routing information to its nearby RSU. Figure 3(b) shows the process of both dynamic and static naming schemes using the same contents in the FIB table, and we found redundancy in the static scheme, whereas dynamic restricts the redundancy using dynamic-tag. Moreover, the performance is discussed in Section 4.

3.3. Name Lookup for VNDN. The name searching process is the primary function of the forwarding engine in NDN. The proposed scheme keeps elements with familiar characters or symbols in the content set Z and can be signified by a recursive function.

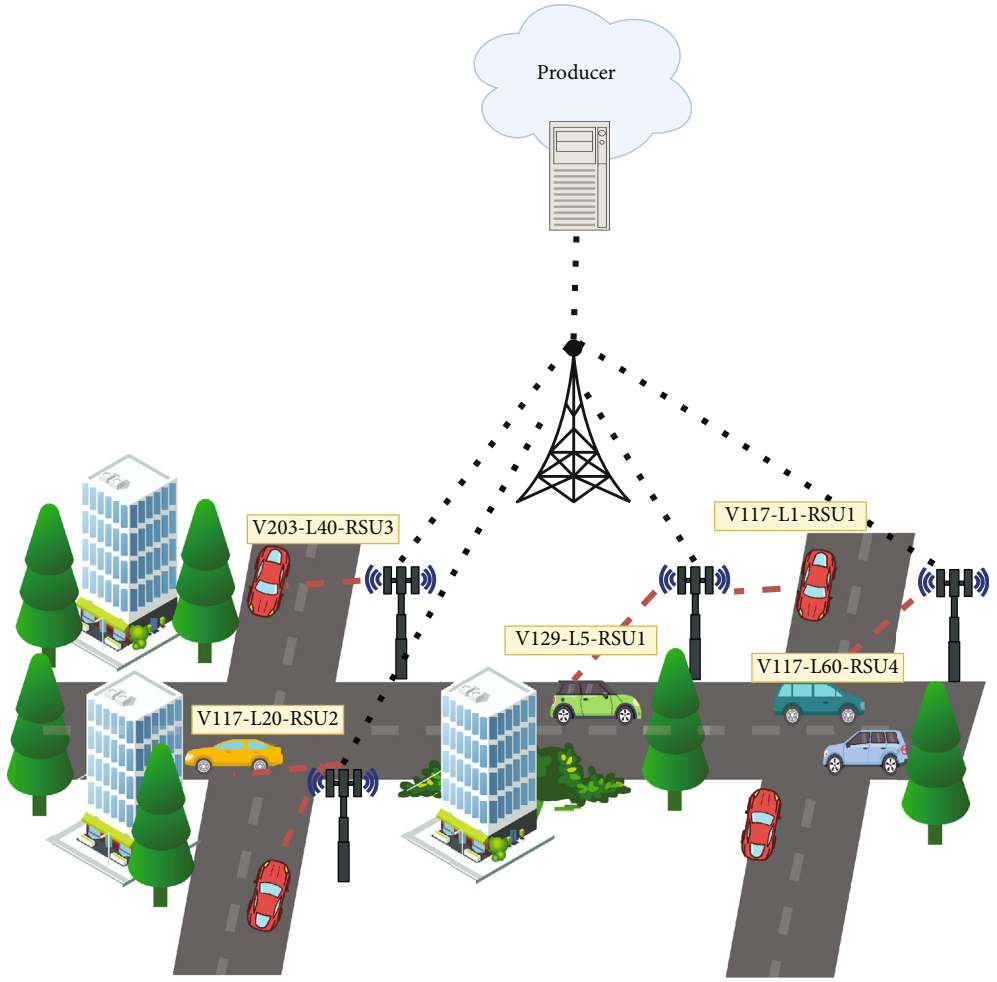
$$\mathrm{DNSL}(W) = \left\langle r, \mathrm{DNSL}\left(\xi_{[p1]}W\right), \mathrm{DNSL}\left(\xi_{[p2]}W\right), .., \mathrm{DNSL}\left(\xi_{\left[a_{p_j}\right]}W\right)\right\rangle, \tag{1}$$

where r represents the root trie, ξ_{p_i} confines the prefix starting set with p_i number of prefix characters at the internal leaf node, and j represents the number of leaf nodes in $DS(W)$ and $j \leq \max_{1\leq i\leq |W|} W[i]$. However, the existing approaches simple trie has overhead in height and width, NLAPB requires many pointers, consumes processing time and more memory to fetch the desired result, and CT has overhead in width that affects lookup time and takes more memory. Therefore, we proposed DNSL, in which the lookup method is based on node partition using PT; which keeps all records in one node with the same characteristics, symbol, and figures. The prefix matching for the given string or name t with the length of y; in our proposed scheme, DNSL $P\mathrm{match}_{\mathrm{DNSL3pt}}(ty)$ required a small number of pointers to fetch the node and branching the nodes, such as

$$P\mathrm{match}_{\mathrm{DNSL3pt}}(ty) = \sum_{m=1}^{y} \begin{cases} \mathrm{match}(t[m]), \\ \delta. \end{cases} \tag{2}$$

Suppose the symbols and characters from the ASCII code are represented with C, where $t[m] \in C$. Besides, $\leq$ use as a pointer to fetch the node's record. As per Equation (2), the lookup processing time is reduced rather than simple trie, NLAPB, and CT. Moreover, CT required rehashing to mitigate the collision concerning lookup and insertion.

We design the trie based on node partition. The partition policy has two characteristics: first, we keep the record of the



(a) VNDN scenario

Static Naming Scheme

Content	Next-Hop
V117/Wasi/video/aa.mp4/L60/	RSU4
V129/Jhon/image/hq.png/L5/	RSU5
V117/Wasi/video/aa.mp4/L1/	RSU1
V203/Nick/book/b1.txt/L40/	RSU3
V117/Wasi/video/aa.mp4/L20/	RSU2

Dynamic Naming Scheme

Content		Dynamic-Tag
V117/Wasi/video/aa.mp4/	1	L60,RSU4
	2	L1,RSU1
	3	L20,RSU2
V129/Jhon/image/hq.png/		L5,RSU1
V203/Nick/book/b1.txt/		L40,RSU3

(b) Naming scheme

Figure 3: VNDN scenario using dynamic tag.

previous node; second, a new node is created instead of the previous. A new node is further split into classes: (a) remaining bytes of the previous node and (b) the rest of the input name. Furthermore, we discussed a node partition model in Section 3.4, and Figure 4(a) shows the node partition when we insert two prefixes, "/V117/wasim/Movie/Mission1" and "/V117/wasim/Movie/Mission2."

3.4. Node Partition Model

Definition 1. (node partition model): gven table T and split node position N_p, each requested interest is divided into two portions at position N_p, namely, P_n and C_n $(P_n + C_n)$. The name lookup function for desired interest is determined by

$$\text{Interest} = \text{LPM}(X_1 \cup X_2) = \begin{cases} \text{LPM}(X_1), \\ y \cup \text{LPM}(X_2), \end{cases} \tag{3}$$

where $P_n \in X_1$, $C_n \in X_2$, and y = remaining bytes.

$$X_2 = \{y | \in T \text{ and } y \geq C_n\} \tag{4}$$

As per giving Definition 1, the node partition is based on content bytes. If the contents partially match the given

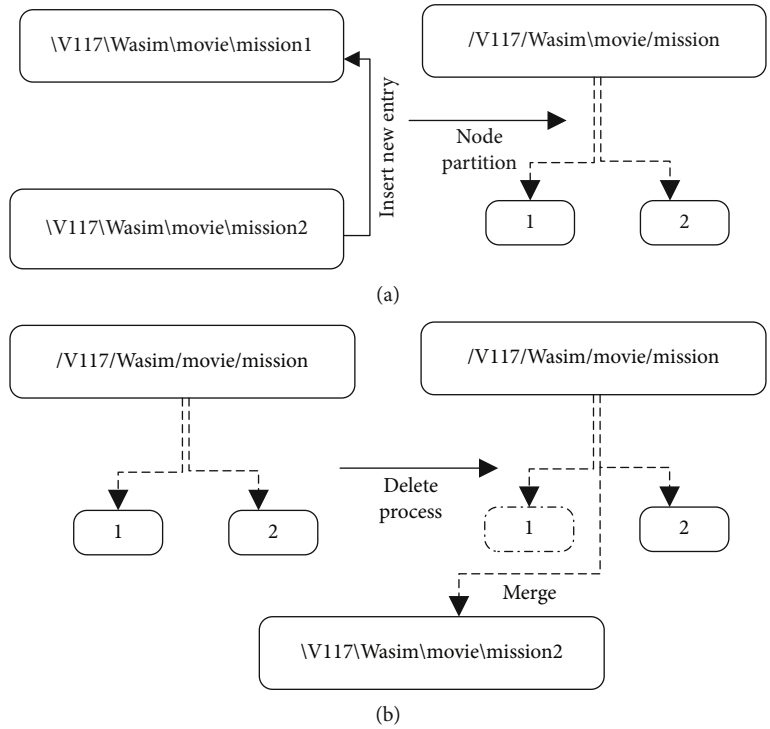


Figure 4: Insertion and deletion process with node partition.

Input: Interest [Vehicle_information, Content_information]
Output: Get desired data
1 Node=R_n;
2.**if** $C_n[b_{id}]$!= "/" **then**
3. Return 0; //contents must start with "/"
4.**while**($b_{id} < C_{len}$)**do**
5. **if** !(C_n)= H_t**then**
6. return 0; //not matched with hash table
7. Node = N_{node}; //insert new record
8. **if** Exact match **then**
9. V_n [V_{id}] = node;
10. V_{id} + +;
11 **while**($N - B_{id}$!$Node_{len}$&&B_{id}!C_{len})**do**
12 **if** (node[N_{id}]= $C_n[B_{id}]$**then**
13 B_{id} + +;
14 N-B_{id} + +;
15 continue;
16 **if**(!$N - B_{id}$)**then**
17 return 0; //first byte not matched
18 break;
19 **if** ($b_{id} = C_{len}$) **then**
20 **if**(($N - B_{id} = Node_{len}$)**then**
21 return node; //content found
22 **else**
23 return 0; //lookup failed due to partially match
24 **if**($N - B_{id} = Node_{len}$)**then**
25 $N - B_{id}$=0; continue;
26 **else**
27 return 0; //invoke Algorithm-2(partition)

Algorithm 1: Lookup and interest contents.

Input: Enter prefix
Output: Set the node with parent and child relationship
1 Initialize hash table;
2 **if**P_n matched & C_n not matched then
3 Keep record P_n; //at position N_p
4 Create N_{node} // with remaining bytes;
5 ontinue: until node is not set;
6 else
7 Node partition failed;

Algorithm 2: Node partition.

interest, then the splitter node such as C_n will keep a record of its corresponding node P_n and then creates a new node.

The deletion process is the inverse of insertion. When the content is matched, we delete the leaf entry node, and the content will merge when the root entity is left with a single node or child content. We consider the invalid removal function if the content mismatching. The invalid has occurred because the lookup function fails to match the contents inside the trie. Figure 4(b) illustrates the reverse of insertion with remove â€œ/V117/wasim/Movie/Mission2."

3.5. Algorithm for Interest and Data Processing. Assume all contents starting with slash "/." When a request arrives for content by the consumer, it invokes the following algorithms. Algorithms 1 and 2 are a core function of our DNSL scheme for VNDN. When a request arrives in the consumer's form of interest, it invokes Algorithm 1. Algorithm 1 first matched the consumer's request using lookup function.). If the content is a partial match for insertion, it invokes the partitioning Algorithm 2, in which it keeps the P_n previous information and merge the prefix with bytes and creates the new node for mismatched bytes. The process will continue until set the prefix bytes, and the process is shown in Figure 4. Moreover, if the requested content is not matched, and it looks in the leaf node, and for insertion entry, and then nothing changes in content tables if an exact match occurred. This paper designed an Algorithm 1 for lookup and insertion contents requested by the consumer to get the desired information. Lookup algorithm can be implemented in NDN routing architecture without changing the core design of the NDN for the vehicular networks. Our proposed algorithms are suitable as compared to existing approaches regarding fast lookup, quick insert contents, and less time to remove contents from the content table. Algorithm 3 is designed for content deletion process, which is the reverse of the insertion Algorithm 1 and the detail process shown in Figure 4(b). Table 1 describes some important notations, which used in this work.

4. Performance Evaluation

This section has conducted an immense analysis to evaluate the proposed dynamic naming scheme's performance and the lookup method in the FIB table.

Input: Desired interest for delete
Output: Delete interest from content table
1 **if**!$(C_n) = H_t$ then
2 return 0; // not matched with hash table
3 Nothing change;
4 **if** Exact match **then**
5 $V_n[v_{id}]$ = Node;
6 V_{id}− −Node Merge;
7 **else**
8 return 0; failed to remove contents

Algorithm 3: Remove content.

Table 1: Table of notations.

Notation	Description
C_{n}	Content name
R_{n}	Root node
V_{n}	Visited node
V_{id}	Visited node index
B_{id}	Name of index
$N - B_{\text{id}}$	Current node index
C_{len}	Content length
Node_{len}	Working node length
P_{n}	Parent node
C_{n}	Child node
V_{n}	Visited node

4.1. Simulation Parameters and Metrics. Our proposed work is aimed at achieving the performance of prefix lookup, insertion, and deletion times in milliseconds (ms) and reduce memory consumption. Besides, we choose NLAPBT [17], simple tire [18], and CT [19] as comparison. The proposed trie is designed using C++ language, proposed naming scheme is implemented on Figure 3(a) using NDN simulator (NDNsim-2.7) [32], and performance measure in a virtual machine on PC, and with Linux Ubuntu 20.04.1 (64-bit) dual processor of inter-core i7-4600U CPU 2.7 GHZ and RAM 8 GB. Moreover, the empirical distribution function shows the distribution process time in milliseconds (ms) of

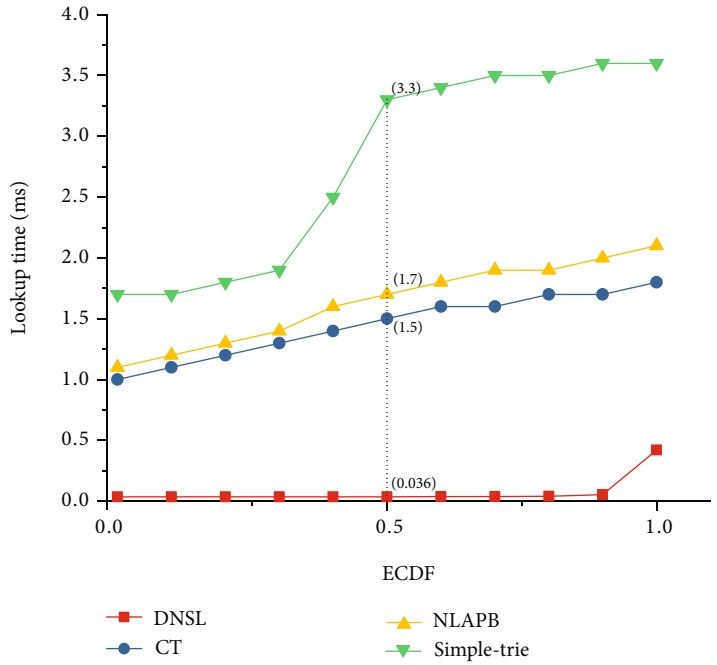


Figure 5: Lookup time per content from content table.

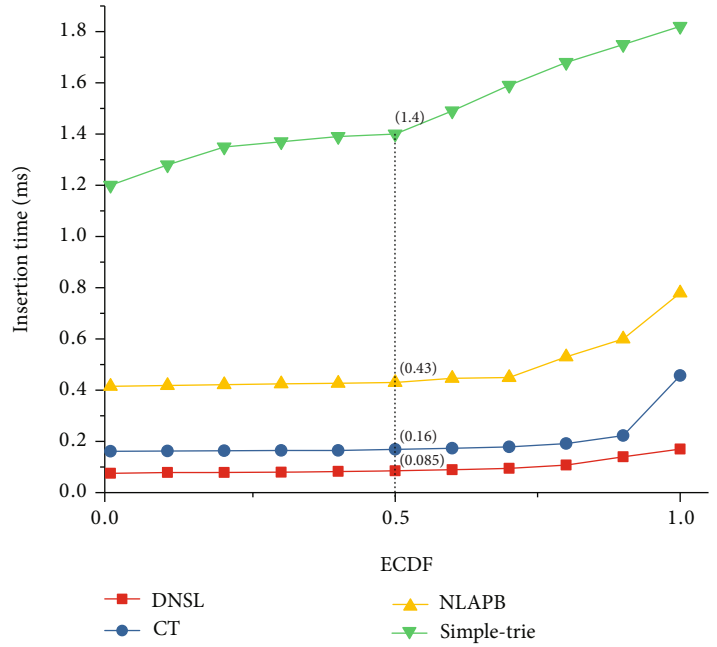


Figure 6: Insertion-Time to add contents in content table.

each prefix lookup, insert, and delete operation. The URL dataset is collected from Shallalist [33], Alexa [34], DGA [35], and DMOZ [36]. The range of prefix names are 10^10 = 10,000 to 50,000,000 with 16 characters to 250 characters. The program has run 1000 times to compute the result with the 95% confidence interval.

4.2. Content Lookup Time. The lookup function is the primary step of insert content in the table and removes the content from the table. We randomly choose 100 contents for the lookup operation that has shown in Figure 5. The size of the content table is 1 M to analyze the lookup performance. Average lookup processing time 0.036(ms), which is less than CT, NLAPB, and simple trie that is 1.5 (ms), 1.7 (ms), and 3.3 (ms), respectively. The DNSL scheme achieves the lookup performance 97.6% CT, 97.8% NLAPB, and 98.90% from simple trie.

4.3. Content Add Time. In Figure 6, we show the contents insertion time for 100 contents inserted in the content table

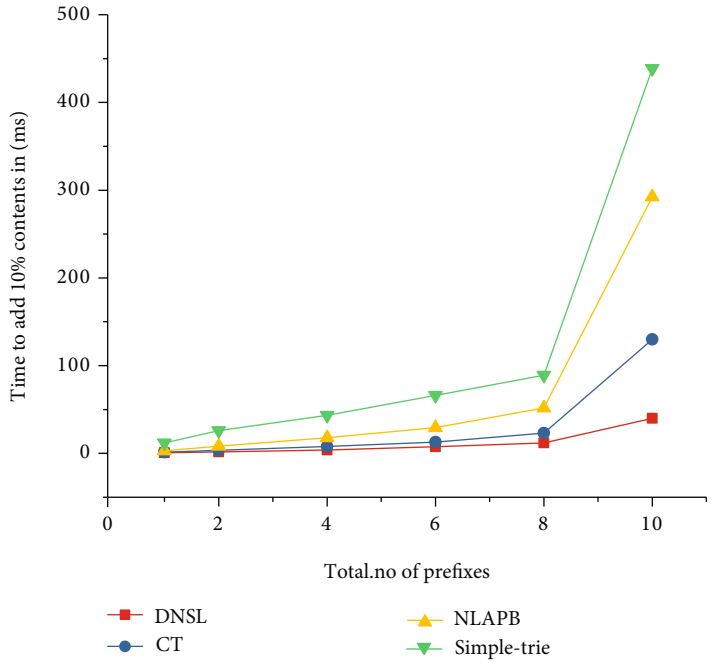


Figure 7: Processing time to add 10% of total contents.

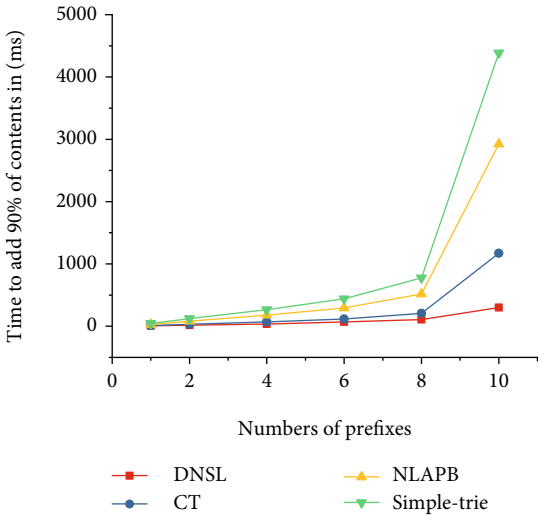


Figure 8: Processing time to add 90% of total contents.

for a total of 1 million. We noted that our proposed DNSL scheme takes less time with 0.085 ms as with CT takes 0.16 ms, and NLAPB takes 0.43 ms, and the simple trie takes 1.4 ms. Figure 6 indicates that the proposed scheme DNSL achieves insertion performance 47% from CT, and 80% from NLAPB, and 94% from simple trie.

4.4. Processing Time to Add 10% and 90% Contents. The overall processing time is shown in Figure 7 for the total prefixes 10^n where n = {2, 4, 6, 8, 10} and 2 million contents for Figure 8 in the tables. The 10% contents are insert of the total contents. Similarly, Figure 8 shows the whole time to insert 90% contents of the total contents. The results show that CT, NLAPB, and simple trie are linearly high when the numbers of contents increase compared to the DNSL scheme. The reason is behind that existing approaches have complex methods to add the content in the content table.

4.5. Content Delete Time. Figure 9 shows the average deletion time of 100 contents from the content table. The result clarifies that the DNSL scheme is efficient for deletion function. As per graph figures, CT consumes 1.8 ms, NLAPB 2.3 ms, and simple trie takes 3.3, respectively, while DNSL takes 0.73 ms. DNSL achieve performance 54.4%, 76.8%, and 77.8% than CT, NLAPB, and simple, respectively.

4.6. Memory Performance. The DNSL scheme has a unique structure and efficient lookup method for VNDN. According to our proposed scheme, the routing information with the same prefix or common contents store in a FIB stack only once time and a dynamic tag is used for further routing rather than storing complete content information. This naming scheme helps to reduce the FIB table size and restrict the redundant information. Let suppose, static name occupies the memory using the prefix static length *Sl*, dynamic tag *Dl*, and *Hl* is the next-hop length. Thus the length of each record using a static scheme in the table for M_s:

$$M_s = \sum_{i=1}^{n} S\ell_i + \sum_{j=1}^{j} D\ell_j + \sum_{k=1}^{k} H\ell_k, \tag{5}$$

and the length of each record in DNSL occupies the memory with M_d is

$$M_d = \sum_{i=1}^{1} S\ell_i + \sum_{j=1}^{j} D\ell_j + \sum_{k=1}^{k} H\ell_k. \tag{6}$$

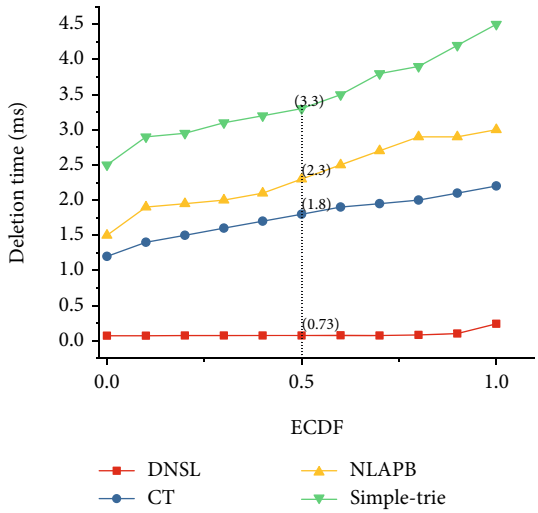


Figure 9: Deletion time to delete contents from content table.

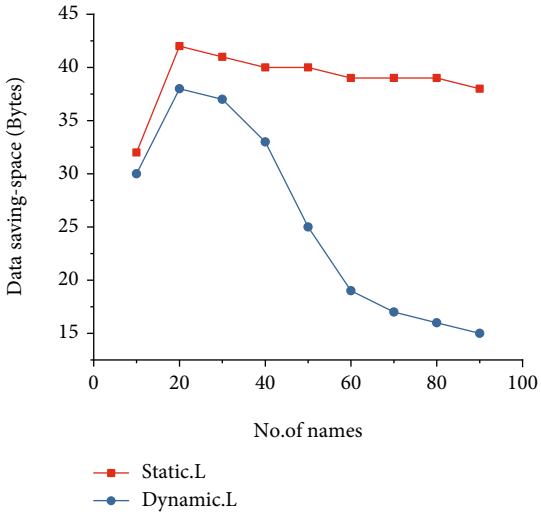


Figure 10: Memory performance between the static and dynamic naming schemes.

The calculation of the compression ratio C_r to save the total space in the FIB table T_s between the static and dynamic naming schemes is

$$C_r = \frac{M_s}{M_d}, \quad C_r = 1 - \frac{M_s}{M_d}. \tag{7}$$

Figure 10 compares both the dynamic and static naming schemes. We achieve 33% memory performance. Our proposed scheme DNSL is better than static naming for VNDN. Moreover, it illustrates that when a vehicle joined the structure of DNSL, and it takes a little bit more memory but also less than a static approach. After registration, the dynamic tag is enough to store the vehicle's information, reduce the content length stores in the FIB table, consume less memory, and provide a dynamic environment for VNDN.

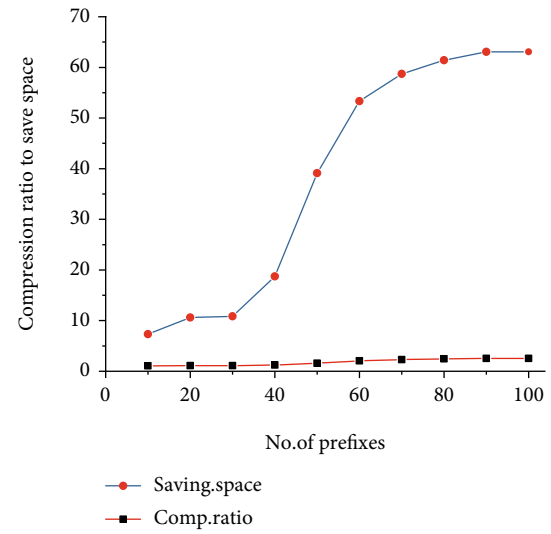


Figure 11: Compression ratio to save memory.

We can see in Figure 11 that the compression rate of static content and achieve the space performance from 5% to 60% and averagely 33% using the above mathematical model.

5. Conclusion

The choice of the naming scheme for lookup in vehicular named data networks can significantly affect the network performance. The static and dynamic naming approaches have not yet been studied well. This article makes three key contributions: (1) our proposed dynamic naming scheme to solve the lengthy namespace problem and restrict the redundancy contents in the naming scheme. Moreover, our proposed work reduces the FIB table size and achieves memory performance. (2) Our proposed work is unique to identify the consumer and producer requests using the common characteristic of the vehicle and the dynamic-tag, which improves the vehicular network's overall performance. (3) Lookup plays a decisive role in improving network performance. Therefore, we designed a lookup method based on node partition based to improve the lookup, insertion, and deletion time. However, the results show (see Section 4) that our proposed DNSL scheme is best for VNDN and achieves the merit of structure in terms of fast lookup, unique approach, and less memory usage. In conclusion, our proposed work was found great when the short namespace, fast lookup, and low memory requirements are the genuine concern.

Data Availability

The simulation data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research study is funded by the National Natural Science Foundation of China under Grant 61772385. The authors acknowledge this financial support.

References

[1] X. Wang, Z. Wang, and S. Cai, "Data delivery in vehicular named data networking," *IEEE Networking Letters*, vol. 2, no. 3, pp. 120–123, 2020.

[2] H. Khelifi, S. Luo, B. Nour et al., "Named data networking in vehicular ad hoc networks: state-of-the-art and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 320–351, 2020.

[3] L. B. Rondon, J. B. da Costa, G. P. Rocha Filho, D. Rosário, and L. A. Villas, "Degree centrality-based caching discovery protocol for vehicular named-data networks," in *2020 IEEE 91st vehicular technology conference (VTC2020-spring)*, pp. 1–5, Antwerp, Belgium, May 2020.

[4] B. Li, D. Huang, Z. Wang, and Y. Zhu, "Attribute-based access control for ICN naming scheme," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 2, pp. 194–206, 2018.

[5] X. Guo, Y. Chen, L. Cao, D. Zhang, J. Jiang, and L. Villas, "A smart forwarding scheme for the interest packet in VNDN," in *2019 2nd International Conference on Hot Information-Centric Networking (HotICN)*, vol. 2019, pp. 7–12, Chongqing, China, December 2019.

[6] W. Xu, X. Ji, C. Zhang, B. Liu, and J. Jiang, "NIHR: name/ID hybrid routing in information-centric VANET," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, Seoul, Korea (South), May 2020.

[7] B. Indira, K. Valarmathi, D. Deverag, and D. Zhang, "A trie based IP lookup approach for high performance router/switch," in *2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, pp. 1–6, Tamilnadu, India, April 2019.

[8] D. Saxena and V. Raychoudhury, "Scalable, memory-efficient pending interest table of named data networking," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 533–540, Delhi, India, December 2020.

[9] B. Sun and V. Raychoudhury, "An improved PLC-trie based routing table design for variable length IP address lookup," in *2019 In Proceedings of the 14th International Conference on Future Internet Technologies*, pp. 1–8, New York, USA, August 2019.

[10] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang, "Data naming in vehicle-to-vehicle communications," in *2012 in IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 328–333, Orlando, FL, USA, March 2012.

[11] H. Khelifi, S. Lou, B. Nour, and H. Moungla, "A name-to-hash encoding scheme for vehicular named data networks," in *2019 International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 603–608, Tangier, Morocco, June 2019.

[12] I. Din, B. S. Kim, S. Hassan, M. Guizani, M. Atiquzzaman, and J. J. Rodrigues, "Information-centric network-based vehicular communications: overview and research opportunities," *Sensors*, vol. 18, no. 11, article 3957, 2018.

[13] M. Rehman, R. Ullah, and B. S. Kim, "NINQ: name-integrated query framework for named-data networking of things," *Sensors*, vol. 19, no. 13, article 2906, 2019.

[14] M. Chowdhury, A. Gawande, and L. Wang, "Secure information sharing among autonomous vehicles in NDN," in *2017 IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 15–26, Pittsburgh, PA, USA, April 2017.

[15] F. M. Modesto and A. Boukerche, "SEVeN: a novel service-based architecture for information-centric vehicular network," *Computer Communications*, vol. 117, pp. 133–146, 2018.

[16] W. Quan, C. Xu, J. Guan, H. Zhang, and L. A. Grieco, "Social cooperation for information-centric multimedia streaming in highway VANETs," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pp. 1–6, Sydney, Australia, June 2014.

[17] W. Quan, C. Xu, J. Guan, H. Zhang, and L. A. Grieco, "Scalable name lookup with adaptive prefix Bloom filter for named data networking," *IEE Communications Letters*, vol. 18, pp. 102–105, 2014.

[18] S. H. Bouk, S. H. Ahmed, and D. Kim, "Hierarchical and hash-based naming scheme for vehicular information centric networks," in *2014 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 765-766, Vienna, Austria, November 2014.

[19] S. H. Bouk, S. H. Ahmed, and D. Kim, "Hierarchical and hash based naming with compact trie name management scheme for vehicular content centric networks," *Computer Communications*, vol. 71, pp. 73–83, 2015.

[20] C. Zhang, Y. Feng, H. Song et al., "PBC: effective prefix caching for fast name lookups," in *2020 IFIP Networking Conference (Networking)*, pp. 1–9, Paris, France, June 2020.

[21] S. H. Park, Y. Y. Shin, and N. Ko, "A scalable name lookup service for NDN," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1025–1027, Jeju, Korea (South), October 2020.

[22] H. Hao, C. Xu, S. Yang, J. Guan, Y. Lu, and L. Zhong, "BF-PDT: a new name lookup mechanism in content-centric networking," in *2017 Networking and Network Applications (NaNA)*, pp. 69–74, Kathmandu City, Nepal, October 2017.

[23] Y. Kurihara, Y. Koizumi, and T. Hasegawa, "Compact data structures for location-based forwarding in NDN networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[24] J. Chen, M. Jahanian, and K. Ramakrishnan, "Black ice! Using information centric networks for timely vehicular safety information dissemination," in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pp. 1–6, Osaka, Japan, June 2017.

[25] T. Pan, T. Huang, J. Liu et al., "Fast content store lookup using locality-aware skip list in content-centric networks," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 187–192, San Francisco, CA, USA, April 2016.

[26] D. Li, J. Li, Z. Du, and J. Zhang, "An improved trie-based name lookup scheme for named data networking," *2016 IEEE*

Symposium on Computers and Communication (ISCC), pp. 1294–1296, Messina, Italy, April 2016.

[27] G. Ghasemi, H. Yousefi, K. G. Shin, and B. J. Zhang, "On the granularity of trie-based data structures for name lookups and updates," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 777–789, 2019.

[28] H. Naeem, B. Guo, and M. R. Naeem, "A light-weight malware static visual analysis for IoT infrastructure," in *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 240–244, Chengdu, China, May 2018.

[29] F. Ullah, M. R. Naeem, A. S. Bajahzar, and F. Al-Turjman, "IoT-based cloud service for secured android markets using PDG-based deep learning classification," *ACM Transactions on Internet Technology (TOIT)*, vol. 22, no. 2, pp. 1–17, 2021.

[30] S. Arshad, B. Shahzaad, M. A. Azam, J. Loo, S. H. Ahmed, and S. Aslam, "Hierarchical and flat-based hybrid naming scheme in content-centric networks of things," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1070–1080, 2018.

[31] J. Huang, P. Wang, Z. Wang, and Y. Zhu, "TCAM-based IP address lookup using longest suffix split," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 976–989, 2018.

[32] NDNSim, 2019, https://ndnsim.net/2.7/getting-started.html.

[33] Shalla's blacklists, 2015, http://www.shallalist.de.

[34] Alexa the Web Information Company, 2017, http://www.alexa.com/.

[35] Dga families, 2015, http://data.netlab.360.com/dga/.

[36] ODPnn- Open Directory Project, 2015, http://www.dmoz.org/.