IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received January 10, 2019, accepted January 23, 2019, date of publication February 5, 2019, date of current version March 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2897747

Cluster-Based Cooperative Caching With Mobility Prediction in Vehicular Named Data Networking

WANYING HUANG[1], (Student Member, IEEE), TIAN SONG[2], (Member, IEEE), YATING YANG[2], (Student Member, IEEE), AND YU ZHANG[1]

[1]School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China
[2]School of Computer Science, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Tian Song (songtian@bit.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672101, in part by the Aerospace Field Pre-Research Foundation Project under Grant 060501, and in part by the Science and Technology on Communication Networks Laboratory Foundation Project.

ABSTRACT Vehicular named data networking (VNDN) is a networking architecture candidate to support various kinds of content-oriented applications in high dynamic topology environment. Its inherent in-network caching facilitates content delivery and the communication efficiency in the vehicular network. However, during the process of data delivery, fast and varying vehicle mobility changes the position of relay routers, which makes data packet delivered through the reverse path difficult. Furthermore, frequent link disruption and packet retransmission lead to the increase of network load and the decrease of user QoE. To address these problems, we propose a cluster-based cooperative caching approach with mobility prediction (COMP) in VNDN. The main idea of COMP is to establish communication among vehicles with similar mobility pattern to mitigate the impact of vehicle mobility, as the link between nodes with a similar pattern is relatively stable and reliable. Specifically, we design a clustering algorithm to group vehicles with similar mobility pattern via mobility prediction and present a cooperative caching to construct intra-cluster and inter-cluster communication over the vehicle clusters. To increase the cache resource utilization and the diversity of the cached data, we classify the cached data into the most popular data and the less popular data based on request frequency, and furthermore, the corresponding cache placement and transmission schemes are proposed. The evaluation results show that most of the vehicles (>95%) can acquire feasible and efficient data delivery via COMP, and COMP significantly improves network performance and user QoE.

INDEX TERMS Vehicular ad hoc network, vehicular named data networking, cooperative caching, cluster, mobility prediction.

I. INTRODUCTION

In recent years, vehicular ad hoc networks have attracted significant attention of researchers from industries and academics. As the trend of global vehicle industry is shaping from traditional vehicles to autonomous vehicles rapidly, technologies that ensure stable and reliable communication in VANETs become critically important. Various wireless technologies such as WAVE/IEEE 802.11p, LTE, Mobile IP, etc. have been developed to support data delivery in vehicular networks [1]. However, all these wireless technologies may bring about several problems in vehicular networks due to their reliance on IP addresses which are closely linked with the node locations. On one hand, high dynamic topology due to the fast and varying vehicle mobility, intermittent connection, and temporal and spatial network density leads to frequent route recalculations and session reestablishments [2], thus increasing unnecessary network load and transmission latency. On the other hand, the dependence on IP address requires additional infrastructures to support IP allocations. Frequent handovers between moving vehicles and infrastructures increase the computing complexity and the costs in allocation and management of IP addresses.

Named data networking (NDN) [3], a networking paradigm that mainly focuses on data rather than their carriers, is developed to address the above issues arised in IP. NDN decouples the data identification from node location, where all packets are identified by a name instead

The associate editor coordinating the review of this manuscript and approving it for publication was Zhengguo Sheng.

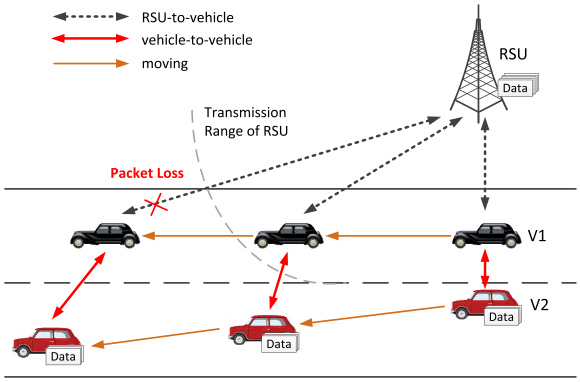

FIGURE 1. **Unstable RSU-to-vehicle communication, and relatively stable communication between V_1 and V_2 with similar mobility pattern.**

of an IP address, since address allocation and management is no longer demanded in communication. This location-independent implementation not only provides a good support for mobility, but its characteristics of multipath data acquisition and in-network caching also improve the communication efficiency in NDN. Therefore, vehicular named data networking (VNDN) has become a potential candidate for VANETs with feasibility and scalability [4], [5].

Although VNDN improves the communication efficiency, the vehicle mobility constantly affects the data delivery. For example, as shown in Fig.1, when vehicle V_1 moves out of coverage area of the road side unit (RSU), it will result in link interruption and packet loss. However, the link between V_1 and V_2 is still stable, and hence V_1 successfully receives packets from V_2. The main reason for the difference is that, unlike the static RSU, V_1 and V_2 have similar mobility patterns (i.e., similar velocities, locations, and moving directions); the relative movements between those two vehicles are more slight so that their link is hardly affected by movements and can maintain for a relatively long time. Consequently, the impact from vehicle mobility on data delivery can be mitigated by establishing communication between vehicles with similar mobility pattern.

Since vehicles and their neighbor vehicles probably travel with similar speed and direction, the links between neighboring vehicles are more likely to be stable and reliable. Allowing vehicles to fetch popular data (i.e., data with high request frequency, like traffic information) from one-hop neighbors may alleviate the mobility issues. However, the in-network caching in VNDN is in a non-cooperative caching manner. Vehicles can only make caching decisions according to local demands rather than global demands, which makes it more difficult for requesting vehicles to fetch desired data from the surrounding vehicles and therefore increases the impact of vehicle mobility on data delivery [6]. Unlike non-cooperative caching, cooperative caching is flexible in managing cache, and it enables RSU to make caching decision for vehicles and cooperatively cache globally popular data with vehicles. Its characteristics like high cache hit ratio and low access delay also help to increase the feasibility and efficiency of acquiring data from one-hop neighbors.

Therefore, we propose a cooperative caching strategy, namely cluster-based COoperative caching with Mobility Prediction (COMP) in VNDN. Considering there is a strong demand to provide users with popular data, such as traffic information, but the high dynamic characteristic of vehicles makes data delivery in sparse network infrastructures difficult. COMP aims to establish stable communication among vehicles to protect the communication quality from vehicle mobility. The contributions of this paper are listed as follows:

- We propose COMP strategy that establishes communication among vehicles with similar mobility pattern to mitigate the mobility issues for VNDN. We design a clustering algorithm to establish and maintain vehicle clusters via mobility prediction.
- We present a cooperative caching strategy to construct intra-cluster and inter-cluster communication over vehicle clusters, which helps to organize communication among vehicles with similar mobility pattern.
- COMP provides relatively stable and reliable data delivery for most of vehicles, and performs better than other commonly used caching strategies.

The rest of this paper is organized as follows: Section II reviews the previous work related to our research and Section III is an overview of COMP. The clustering algorithm and cooperative caching strategy are presented respectively in Section IV and Section V. Evaluation results are shown in Section VI to demonstrate the feasibility of COMP, and conclusions are given in Section VII.

II. RELATED WORK

A. NDN AND ITS CACHING MODEL

NDN uses a pull-based communication model, where the requesting node (named *Consumer*) sends an *Interest* packet, and the node with the requested data (named *Producer*) sends the *Data* packet. The Interest packet carries a name, and the Data packet carries the same name. Each NDN node has three essential data structures: (1) PIT records Interest packet that has not been satisfied; (2) FIB guides the name-based forwarding of Interest packet; (3) CS caches the replicas of Data packets.

Upon receiving an Interest, a node records the Interest in PIT and searches its CS for a hit according to name. If there is a hit, the matched Data is replied to the Consumer. Otherwise, the unsatisfied Interest is forwarded to the next node by looking up matched entries in FIB. The Producer replies Data packet toward the Consumer through the reverse path created by Interest, and when there exists several relay nodes on the path, these nodes will cache the Data replicas in the CS. Such in-network caching increases the chances for nodes to become potential producers, but may bring about some issues like frequent cache updates and over-caching.

To improve cache efficiency and network performance, lots of caching strategies have been studied and proposed. The existing caching strategies in NDN can be classified into non-cooperative caching and cooperative caching depending on whether the caching decisions are made independently at each individual node [6], [7].

In non-cooperative caching approaches, nodes exchange no caching information with others. AlwaysCache, the default caching strategy in NDN, allows relay nodes to cache all the coming Data, even though they are not the consumer. Another commonly used caching strategy in NDN, ProbCache [8], evaluates the caching capability of a path and then caches Data packets with a certain probability constant.

Cooperative caching approaches enable nodes to exchange caching information with each other and aim to reduce access delay and cache redundancy. References [9]–[12] are examples of these approaches. In [9], a cooperative in-network caching scheme is proposed in which each content is divided into multiple chunks and cooperatively cached at several nodes. In WAVE [10], an upstream node decides which chunks can be cached at the next downstream node, and the node closed to the end users tends to cache popular contents. In [11], a cache-aware ICN design is developed with lightweight cooperation between nodes. In [12], an intra-AS cache cooperation scheme is presented to control cache redundancy within the AS, and nodes can cooperatively satisfy requirements of neighbors.

However, the core problem in non-cooperative caching is that the contents cached by nodes outside of request path are unknown by nodes inside the path, which leads to cache redundancy and inefficiency. Unlike non-cooperative caching, cooperative caching enables nodes to share cache information with each other and to make caching decisions for others; and moreover, the flexibility of managing cache can significantly improve the cache efficiency.

B. CACHING IN VNDN

VNDN consists of VANET and NDN [4], and it inherits the in-network caching characteristics from NDN, which can reduce the load of original data producers and access delay. However, as the default caching strategy, AlwaysCache, may increase the caching redundancy, some caching strategies need to be designed for VNDN.

Several non-cooperative caching strategies for VNDN have been proposed [13]–[15]. In LF [13], popular contents are cached at selected vehicles on the request path. In [14], a distributed probabilistic caching strategy is proposed for vehicles to make caching decisions by considering demands and preference of users. In [15], vehicles cache contents based on the request of application types.

Cooperative caching approaches in VNDN have also been studied for several years [16]–[25]. In [16], a set of proxies are deployed to explore user requirements and prefetch contents by leveraging pull-based model in NDN. In [17], a layered cooperative cache management enables vehicles to choose one-hop neighbors as caching nodes according to the layer level caching potential. In CCMP [18], the probability of a vehicle reaching different hot regions is predicted based on its past trajectories, and the vehicle with longer sojourn time is designed to cooperatively cache popular contents with RSUs. In [19], an interval metric is introduced for selective caching with consideration of data types and network density. In UG-Cache [20], caching vehicles are selected based on request frequency and cache distance. In [21], a set of caching nodes are selected based on their mobility similarity and privacy to reduce cache redundancy. In [22]–[25], by predicting the next RSU that vehicles may pass through, desired data of vehicles is pre-cached in the next RSU to enhance seamless mobility support.

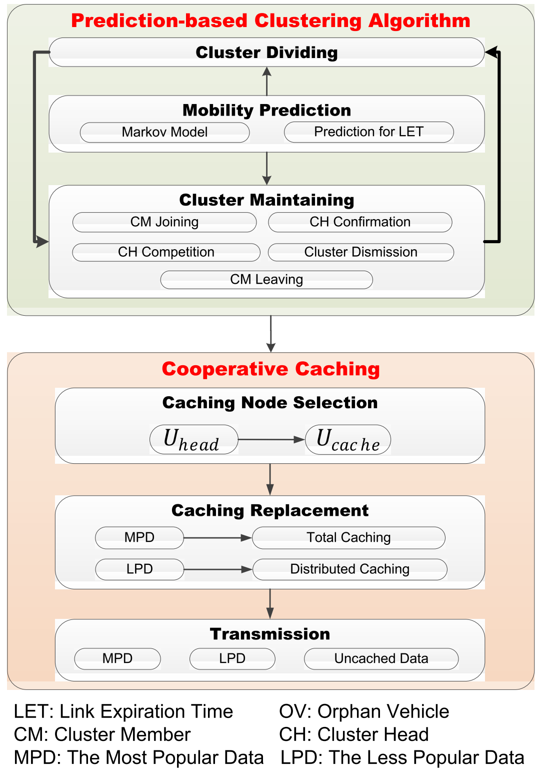

FIGURE 2. System model.

Although there are some work taking advantages of cooperative caching to improve cache efficiency in vehicular network, those work rarely consider the effects of vehicle mobility on communication quality; the unsolved mobility issues limit the expected performance of these caching strategies. To address the gap, our previous work [26] designs a cooperative caching strategy to mitigate the impact of vehicle movements on communication quality. In the present work, we further improve the stability and reliability of vehicle-to-vehicle communication via mobility prediction, and further enhance the cache efficiency by introducing distributed caching.

III. COMP OVERVIEW

In COMP, we mainly consider two types of nodes: RSUs and vehicles; each node maintains three essential data structures: PIT, FIB, and CS. COMP includes two major parts: prediction-based clustering algorithm and cooperative caching, as illustrated in Fig.2. The former groups vehicles with similar mobility pattern via mobility prediction, and the latter establishes intra-cluster and inter-cluster communication over vehicle clusters by caching data at selected cluster heads. Each vehicle is in one of three states: cluster head (CH), cluster member (CM), and orphan vehicle (OV). Every cluster has a vehicle as the cluster head, others are cluster members; vehicles that do not join any cluster are named as orphan vehicles. Vehicles periodically broadcast beacon packets to one-hop neighbors, and the beacon packets carry mobile information (i.e., current location, velocity, and moving direction) and state information (i.e., CH, CM, or OV) of the sender.

The workflow overview of COMP is illustrated as follows. According to the clustering algorithm, vehicles can predict the future mobility patterns via Markov model and prediction model for link expiration time (LET). Vehicle clusters are established depending on the prediction results. To maintain the vehicle clusters, COMP considers several scenarios, which may affect the stability and coverage of clusters, such as CM joining, CH confirmation, CH competition, cluster dismission, and CM leaving. According to cooperative caching, RSUs select some vehicles in the set of cluster heads U_{head} as caching nodes U_{cache} to avoid the interruption of caching process caused by vehicle mobility. Based on request frequency, cached data is categorized into two types: the most popular data (MPD) and the less popular data (LPD). COMP allows each vehicle in U_{cache} to cache total MPD, and different vehicles to cache different partitions of LPD. When a CM generates requirement, it first looks up its CS for a hit. If so, its CS will return the matched Data packets; otherwise, it will send Interest to the local CH. Based on the types of requested data (i.e., MPD, LPD, uncached data), the Interest and matched Data will be forwarded via corresponding transmission schemes.

The frequently used notations are listed in Table 1.

TABLE 1. Key notations.

Notation	Description
N_i	Set of neighbor vehicles of vehicle i
$\mid N_i \mid$	The number of vehicles in N_i
R	Transmission range of vehicle
W_i	Current weight of vehicle i
$W_i^{(t)}$	Weight of vehicle i in the tth period
$\overline{D_i}$	Average distance between i and neighbors
$P(N_i)$	Probability of neighbor maintaining its status
L_{ij}	LTE between vehicle i and j
$\overline{L_i}$	Average LET between vehicle i and neighbors
ΔT	Period of broadcasting beacon packet
s_a	Location a
S	Location sequence
v_i	Speed of vehicle i
θ_i	Moving direction of vehicle i
(x_i, y_i)	Location coordinate of vehicle i
U_{head}	Set of cluster heads
U_{cache}	Set of caching nodes
S_{co}	Total size of cached contents
ρ	Caching Proportion of LPD

IV. PREDICTION-BASED CLUSTERING ALGORITHM

A. CLUSTER DIVIDING

When the network is first established, all vehicles are in an orphan state and then are divided into different clusters via selection for cluster heads. The selection considers various factors including vehicle mobility, average distance between vehicle and its neighbors, and the number of neighbors. The reasons for taking these factors into consideration are as follows: (1) vehicle mobility may lead to link disruption and packet loss, thus affecting the communication quality of vehicular network; (2) due to the feature of random loss in a wireless fading channel, the smaller communication distance represents a higher success rate of data delivery; (3) establishing clusters with extensive coverage can avoid an excess of clusters, as well as the costs of calculation and maintenance. Therefore, a weight with refresh cycle ΔT is used as the suitability of becoming cluster head for each vehicle. The vehicle with the highest weight is regarded as the best to be the cluster head among surrounding vehicles.

An example is given to illustrate the calculation process of weight. Assuming vehicle V_i and its neighbor vehicles $V_j (j = 1, \cdots, \mid N_i \mid)$ can communicate with each other, and each of them periodically broadcasts beacons to other vehicles where the beacons carry the identification, and the mobile and state information of the sender. After receiving and collecting the related information, vehicle i can independently calculate its weight as:

$$W_i^{(t)} = w_1 * \mid N_i \mid + w_2 * \frac{R}{\overline{D_i}} + w_3 * P(N_i) + w_4 * \overline{L_i} \quad (1)$$

where w_1, w_2, w_3 and w_4 are weighted factors, which are set according to the specific application environment and

$$w_1 + w_2 + w_3 + w_4 = 1. \quad (2)$$

Among those factors considered by weight calculation, $\mid N_i \mid$ and $\overline{D_i}$ can be obtained according to the received identification and location information; $P(N_i)$ and $\overline{L_i}$ are related to vehicle mobility, which is computed via mobility prediction (see Section IV-B) After calculating the weight W_i, vehicle V_i records the weight value and adds that value into the next beacon.

In cluster dividing, the vehicle with the highest weight among surrounding vehicles is chosen as the cluster head. The process of cluster dividing is illustrated as Algorithm 1. Assuming the network is first established, all vehicles are in the orphan states, and the cluster dividing is performed. After receiving the first beacon of neighbors $V_j (j = 1, \cdots, |N_i|)$, vehicle V_i calculates the first initial weight W_i^0 by Eq (1) and adds the weight into the second beacon. Upon receiving the weight $W_j^{(0)} (j = 1, \cdots, |N_i|)$ from neighbors, V_i compares them with its own weight. If its weight $W_i^{(0)}$ is the highest, V_i becomes a cluster head, and the state information is updated

Algorithm 1 Cluster Dividing

Input: $W_i^{(0)}, W_j^{(0)} (j = 1, \cdots, \mid N_i \mid)$
Output: S_i

$S_i \leftarrow OV$ ▷ V_i is in the orphan state
while V_i exchanges *beacon* with V_j **do**
  $V_i \leftarrow beacon\_j$ ▷ V_i receives beacon
  $V_i \leftarrow W_i^{(0)}$ ▷ Apply Eq (1)
  $beacon\_i \leftarrow W_i^{(0)}$ ▷ Add $W_i^{(0)}$ into beacon
  while $V_i \leftarrow beacon\_j$ **do**
    if $W_i^{(0)} > W_j^{(0)}$ **then**
      $S_i \leftarrow CH$ ▷ V_i becomes CH
      $beacon\_i \leftarrow CH$ ▷ Add state into beacon
      while $V_j \leftarrow beacon\_i$ **do**
        $S_j \leftarrow CM$ ▷ V_j becomes the CM of V_i
      end while
    else
      if $V_i \leftarrow beacon\_j$ and $beacon\_j == CH$ **then**
        $S_i \leftarrow CM$ ▷ V_i becomes the CM of V_j
      else
        $S_i \leftarrow CH$ ▷ V_i becomes the CH
      end if
    end if
  end while
end while
return S_i

from OV to CH in the next beacon. Then, according to the received state information from V_i, neighbor vehicles will join the cluster and become cluster members of V_i. Otherwise, V_i will wait for instruction from the vehicle which has become a cluster head for a period of $\beta \Delta T (\beta > 1)$. If it still receives no response from any cluster head when the time expires, it will become a cluster head.

B. MOBILITY PREDICTION

In the process of weight calculation, node mobility is an important factor to be considered. Due to the high dynamic of node mobility, clusters constructed based on the current mobility patterns may be unstable. To guarantee the stability of clusters, the vehicle mobility is evaluated by two factors: probability of neighbors maintaining states in the next period, and link expiration time (LET) between the vehicle and neighbors. These two factors are respectively predicted by Markov model and prediction model for LET, which are illustrated in following subsections.

1) PREDICTION BASED ON MARKOV MODEL

Owing to the accurate prediction capability and low calculation costs, COMP adopts second-order Markov model to predict the next location of vehicles according to the current and last locations, and then the probability for two vehicles to keep a connection in the next period can be computed [27].

To facilitate the process of describing moving traces of vehicles, we partition the digital map into multiple cells

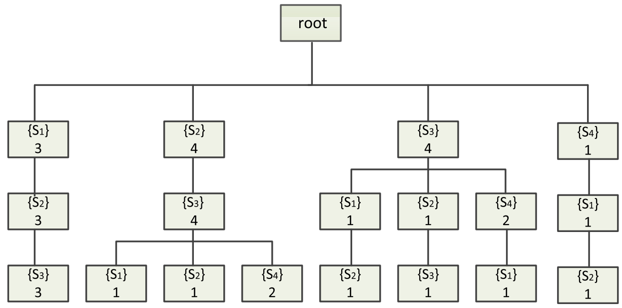

FIGURE 3. An example of Markov parsing tree trained by sequence S.

$s_1, s_2, \cdots, s_n$ with the length of r [28]. According to the location of vehicles, the cells that vehicles locate in can be obtained. In order to decrease the computing complexity and memory storage, COMP assumes the moving trace of a vehicle is a sequence $S = \{s^{(1)}, s^{(2)}, \cdots, s^{(n)}\}$, where $s^{(t)}$ refers to the location in the tth period.

The Markov prediction model has a training stage to construct a parsing tree by processing a single or multiple known location sequences where the tree starts with a root node and the training location sequence is processed one location at a time. Once a location s_a and its previous k locations L_k appear in the training sequence at the first time, a new path $L_k s_a$ is added into the parsing tree and the number of times that the path appears in the training sequence is recorded as $N(L_k s_a)$. The k value depends on the order of adopted Markov model.

Fig.3 is an example for this. A training sequence $S = \{s_1, s_2, s_3, s_1, s_2, s_3, s_4, s_1, s_2, s_3, s_2, s_3, s_4\}$ is processed to generate a parsing tree for second-order Markov prediction model. The first location s_1 and its subsequent two (i.e., $k = 2$) locations s_2, s_3 form the first path of the tree, and then the second path consisting of locations s_2, s_3, s_1 is added into the tree. Other paths are also shown in Fig.3 as well.

After establishing the Markov parsing tree, the probability of $s_a (s_a \in S)$ is the next location that can be computed by a given location sequence L. Based on the received beacons from neighbors, vehicles collect and record the current and last locations of these neighbors, and then predict their next locations via the parsing tree. The prediction uses escape mechanism [29] and is presented as

$$P(s_a|L) = \begin{cases} P_k(s_a|L) & s_a \in U_{L_k} \\ P_k(s_\xi|L) \cdot P(s_a|L_k') & s_\xi \in U_{L_k}, s_a \in U_{L_k'} \end{cases} \tag{3}$$

where

L_k denotes the last k locations of a given sequence;
L_k' denotes the first i ($i = k - 1$) locations of L_k;
U_{L_k} denotes the set of locations appearing after L_k, which is obtained from the training sequences.

When location s_a appears after a known k-size sequence L_k, i.e., path $L_k s_a$ has been added into the parsing tree in the training stage, the probability $P(s_a|L)$ equals $P_k(s_a|L)$ as

$$P_k(s_a|L) = \frac{N(L_k s_a)}{\sum_{s_b \in U_{L_k}} N(L_k s_b) + |U_{L_k}|}. \tag{4}$$

When the path $L_k s_a$ has not been previously added into the parsing tree, the probability for s_a to become the future location can be computed by using subsequence L'_k:

$$P_k(s_\xi|L) = \frac{|U_{L_k}|}{\sum\limits_{s_b \in U_{L_k}} N(L_k s_b) + |U_{L_k}|} \tag{5}$$

and

$$P(s_a|L'_k) = \frac{N(L'_k s_a)}{\sum\limits_{s_b \in U_{L'_k}} N(L'_k s_b) + |U_{L'_k}|}. \tag{6}$$

For example, according to the parsing tree in Fig.3 and a given location sequence where the last two locations are $s_3 s_2$, the probability of s_4 is the next location that can be predicted by Eq (3). Because there is no path $s_3 s_2 s_4$ in the parsing tree, the probability $P(s_4|s_3 s_2)$ equals $P(s_\xi|s_3 s_2) \cdot P(s_4|s_3)$, where

$$\begin{aligned} P(s_4|s_3 s_2) &= P(s_\xi|s_3 s_2) \cdot P(s_4|s_3) \\ &= \frac{|U_{s_3 s_2}|}{\sum\limits_{s_b \in U_{s_3 s_2}} N(s_3 s_2 s_b) + |U_{s_3 s_2}|} \\ &\quad \cdot \frac{N(s_3 s_4)}{\sum\limits_{s_b \in U_{s_3}} N(s_3 s_b) + |U_{s_3}|} = \frac{2}{11}. \end{aligned}$$

The prediction result in this example shows the probability $P(s_4|s_3 s_2)$ equals $\frac{2}{11}$.

Assuming a parsing tree involving n locations has been established in the training stage, and then vehicle V_i records location of neighbors $V_j(j = 1, \cdots, | N_i |$ based on the received beacon. According to Eq (3), vehicle V_i calculates the probability of the next location for each neighbor vehicle as well as itself. The prediction results are presented as a vector $P_i = [\pi_i, \pi_1, \cdots, \pi_j, \cdots, \pi_{|N_i|}]$, where π_i and π_j denote the prediction results of V_i and $V_j(1 \leq j \leq| N_i |$), respectively. The elements of P_i also consist of vectors, for example, $\pi_i = [\pi_{i1}, \pi_{i2}, \cdots, \pi_{in}]$, $\pi_{ik}(k = 1, \cdots, n)$ represents the probability that the next location of V_i is s_k. The probability for V_i to keep connecting with neighbor V_j can be calculated by V_i as

$$P(ij) = \sum_{k,l} \pi_{ik} \pi_{jl} \tag{7}$$

where the distance between s_k and s_j should be smaller than the transmission range R. To consider other neighbors, the probability that all neighbors of V_j maintain the states in the next period can be computed as

$$P(N_i) = P(i1)P(i2) \cdots P(in) \tag{8}$$

2) PREDICTION MODEL FOR LINK EXPIRATION TIME

Link expiration time (LET) is defined as the length of time that two vehicles can keep connecting. Therefore, the prediction model for LET can be used to evaluate the stability of link between two vehicles.

Based on the mobile information carried by periodically exchanged beacons, a vehicle can independently predict the LET between its neighbor and itself through trigonometry. Fig.4 shows an example of predicting LET, a vehicle V_i intents to predict the LET of link between neighbor V_j and itself. Upon receiving beacons broadcasted by neighbor V_j, V_i extracts and records the mobile information of V_j including location (x_j, y_j), velocity v_j, and moving direction $\theta_j (0 < \theta_j < 180^o)$. The transmission range R can be computed by

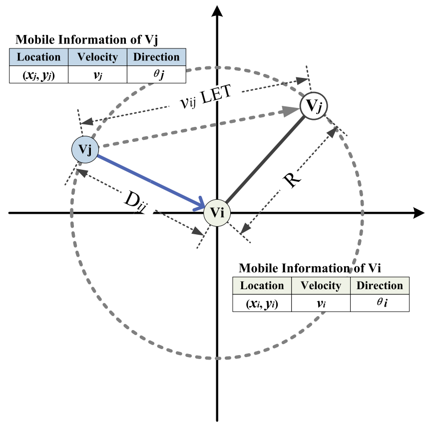

FIGURE 4. An example of predicting link expiration time.

$$R^2 = (v_{ij} L_{ij})^2 + D_{ij}^2 - 2 v_{ij} D_{ij} L_{ij} \cos \Delta\theta \tag{9}$$

where

$$\begin{aligned} v_{ij} &= ((v_j \cos\theta_j - v_i \cos\theta_i)^2 + (v_j \sin\theta_j - v_i \sin\theta_i)^2)^{\frac{1}{2}} \\ D_{ij} &= ((x_j - x_i)^2 + (y_j - y_i)^2)^{\frac{1}{2}} \\ \Delta\theta &= \theta_j - \theta_i \end{aligned}$$

Finally, the LET between vehicle V_i and V_j can be calculated by

$$L_{ij} = \frac{D_{ij} \cos\Delta\theta + (R^2 - D_{ij}^2 \sin^2 \Delta\theta)^{\frac{1}{2}}}{v_{ij}} \tag{10}$$

Both the Markov prediction model and the prediction model for LET enable vehicles to predict the mobility pattern independently, and the prediction results can be used to evaluate the suitability of a vehicle to become the cluster head. Therefore, the results are regarded as important factors to be considered in the establishment and maintenance of vehicle clusters.

C. CLUSTER MAINTAINING

The maintenance of vehicle clusters is necessary because vehicle movements can affect the stability and coverage of the clusters. Assuming all vehicles desire to join the clusters, as the links between vehicles with similar mobility pattern are relatively stable, i.e., the links based on the clusters can provide good support for stable data delivery. Hence the stability and coverage of the clusters is closely related to link quality. To maintain the clusters, we propose corresponding solutions to address scenarios including CM joining, CH confirmation,

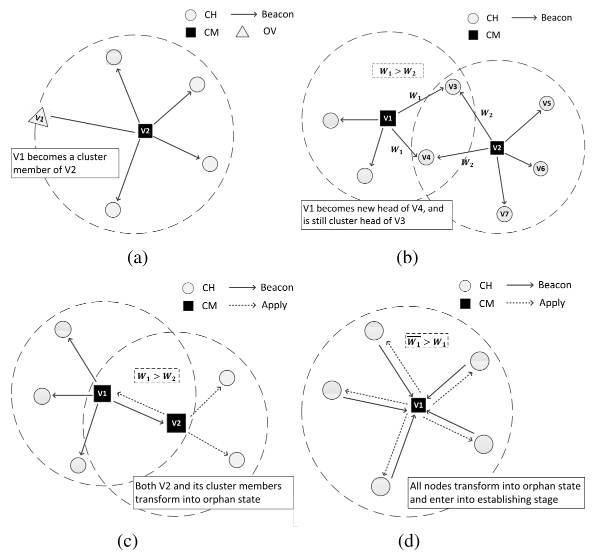

FIGURE 5. Scenarios considered in the maintenance of clusters. (a) CM joining. (b) CH confirmation. (c) CH competition. (d) Cluster dismission.

CH competition, cluster dismission, and CM leaving. The details are listed as follows.

Scenario 1(CM Joining): Upon cluster dividing, some vehicles may still keep an orphan state and wait for joining clusters. These orphan vehicles can join clusters when they move into the coverage area of clusters. As shown in Fig.5(a), an orphan vehicle V_1 moves into the coverage area of a cluster. After receiving beacon from cluster head V_2, V_1 transforms its state from OV to CM.

Scenario 2(CH Confirmation): Due to the varying mobility of vehicles, a cluster member occasionally moves into the coverage area of other clusters, and then receives beacons from both the local cluster head and other heads. It is reasonable to enable the cluster member to select a better head, which can offer more stable and reliable data delivery. Based on the weight carried in beacons, a new head is selected by comparing the weight of multiple heads; the cluster member chooses the cluster head with the highest weight as its new head. As shown in Fig.5(b), vehicle V_3 and V_4 are cluster members of cluster heads V_1 and V_2, respectively. They move in the coverage of both clusters and receive the beacons from both heads. By comparing the weight of V_1 and V_2, V_3 and V_4 find that W_1 is larger than W_2, which indicates that V_1 is more suitable than V_2 to be the cluster head and therefore they choose V_1 as their new head.

Scenario 3(CH Competition): As vehicles are moving around, some cluster heads may enter into the coverage area of each other. The solution is to enable one of them to be the head continually, while other heads recover to the orphan state. When the distance between two cluster heads is smaller than the half of transmission range ($R/2$), this solution is started. As shown in Fig.5(c), cluster head V_1 enters the transmission range of V_2, and the distance between them is smaller than $R/2$. Upon receiving the beacons from each other, vehicle V_2 finds that its weight W_2 is smaller than that of V_1, which indicates V_1 is more suitable than V_2 to be a head. Then V_2 and its cluster members are turned into the orphan state after V_2 broadcasts apply packets to the members.

Scenario 4(Cluster Dismission): Because of the ever-changing location and speed, a cluster head may not always be an appropriate head. For example, either acceleration or deceleration may make a cluster head unstable to keep the CH state. A less qualified vehicle in the CH state would reduce the stability and reliability of clusters and therefore a cluster head should be self-detective and able to adaptively transform the state. The comparison of current and previous weights can reflect the change in the qualification of cluster head, and hence cluster heads are designed to record their weight in past n times and get an average value as

$$\overline{W_i} = (W_i^{(1)} + W_i^{(2)} + \cdots + W_i^{(n)})/n. \tag{11}$$

If the current weight $W_i^{(n)}$ is smaller than $\overline{W_i}$, it implies that V_i becomes less eligible to be the cluster head, and the cluster needs to be dismissed and reconstructed. An example is shown in Fig.5(d), cluster head V_1 finds that its current weight $\overline{W_1}$ is smaller than $W_1^{(n)}$, which means that it is no longer suitable to manage the cluster. After vehicle V_1 sends an apply packet to its cluster members, all vehicles within the cluster will recover from the orphan state and then enter into the phase of cluster dividing. A new cluster head and a new cluster will be generated soon.

Scenario 5(CM Leaving): When a cluster member leaves the located cluster, i.e., moves out of the coverage area of the cluster head, it can not receive any beacon from the head. Upon finding the time interval from receiving the last beacon is longer than $\gamma \Delta T (\gamma > 1)$, the cluster member will recover to the orphan state and then join a new cluster according to the Scenario 1.

Besides the above five scenarios that should be considered for the maintenance of cluster and corresponding solutions are given to extend the cluster lifetime, there are several extreme cases that also should be taken into account:

- If all vehicles are in the orphan state, they will enter into cluster dividing stage (Section II-A), and then new vehicle clusters will be established;
- If all vehicles are in the CH state, i.e., there is no cluster member in any cluster, these vehicles will discard the CH state, and then enter into the cluster dividing stage;

Through the prediction-based clustering algorithm, vehicle clusters can be established with high stability and coverage, and the link between CH and CM should be more stable than that between any two nodes with uncertain mobility pattern. This cluster-based communication is able to be established via cooperative caching, which can improve the communication quality of vehicular network.

V. COOPERATIVE CACHING STRATEGY

After establishing vehicle clusters, COMP uses cooperative caching strategy to construct cluster-based communication by

Algorithm 2 Caching Node Selection

Input: U_{head}
Output: U_{cache}
while RSU $i \leftarrow beacon_{U_{head}}$ **do**
  $U_{cache} \leftarrow \phi$, $(x, y) \leftarrow (0, 0)$
  $V, \theta, LET \leftarrow 0$ ▷ Parameter initialization
  for all j in U_{head} **do**
    $V \leftarrow v_j, \theta \leftarrow \theta_j$ ▷ Record mobile information
    $(x, y) \leftarrow (x_j, y_j)$
    $LET \leftarrow L_{ij}$ ▷ Apply Eq (10)
    if $LET \geq \eta T_{co}$ **then**
      $U_{cache} \leftarrow j$ ▷ j is selected as caching node
    end if
  end for
end while
return U_{cache}

Algorithm 3 Cache Placement

Input: U_{cache}
Output: CS_j
$U_{MPD}, U_{LPD} \leftarrow \phi$ ▷ Set of MPD and LPD
while RSU i selects U_{cache} as caching nodes **do**
  $k \leftarrow\mid U_{cache} \mid, n \leftarrow 0$
  $U_{MPD} \leftarrow (1-\rho)S_{co}$ contents
  $U_{LPD} \leftarrow \rho S_{co}$ contents
  for all j in U_{cache} **do**
    $CS\_j \leftarrow U_{MPD}$ ▷ Cache the whole MPD at j
    for all C_r in U_{LPD} **do**
      for all $Chunk_t$ in C_r **do**
        if t mod $n == 0$ **then**
          $CS\_j \leftarrow Chunk_t$ ▷ Cache $Chunk_t$ at j
        end if
      end for
    end for
    $n \leftarrow n + 1$
  end for
end while
return CS_j

enabling RSUs to cache popular data to cluster heads, and thus these heads can offer stable and reliable data services for requesting vehicles.

A. CACHING NODE SELECTION

When the vehicle clusters are formed, the cluster heads broadcast beacons to the neighbors including their nearest RSUs, and then wait to receive data from the RSUs. To prevent vehicle mobility from the cooperative caching, RSUs are designed to select appropriate cluster heads as caching nodes according to their mobility patterns.

Assuming RSUs can cooperatively cache S_{co} size data with each caching node and the transmission rate of RSU is R_{rsu}, thus the transmission delay in cooperative caching is expressed as

$$T_{co} = \frac{S_{co}}{R_{rsu}}. \tag{12}$$

Considering the possible situations in data delivery, such as link interruption and packet loss, $\eta T_{co}(\eta > 1)$ is presented as the benchmark of caching node selection, where the LET of links between caching nodes and RSUs should be longer than ηT_{co}.

The procedure of caching node selection is described as Algorithm 2. Upon receiving beacons from a set of cluster heads U_{head}, it empties the set of caching nodes U_{cache} for selecting and adding new caching nodes. For a cluster head j in U_{head}, RSU i records its mobile information including the current velocity, moving direction, and location, and then computes the LET L_{ij} of the link between j and itself by applying Eq (10). Through the comparison between L_{ij} and ηT_{co}, RSU i assesses the suitability of being a caching node for j. If L_{ij} is longer than ηT_{co}, RSU i adds vehicle j into U_{cache} and then prepares to deliver cached data to j; otherwise, vehicle j will take no action until it encounters an appropriate RSU.

B. CACHE PLACEMENT

RSUs cooperatively cache data with caching nodes according to cache placement scheme, but caching the same data at every caching node may cause cache redundancy and the waste of cache resource, which limits the diversity of cached data and the performance of COMP.

To address the above issues and fully utilize the finite cache space, the cache placement classifies the cached data into two types based on the request frequency: the most popular data (MPD) and the less popular data (LPD). In each node, $(1-\rho)S_{co}$ and ρS_{co} size space in CS are used to cache MPD and LPD, respectively. COMP enables MPD to be totally cached at each caching node, and enables LPD to be distributedly cached among caching nodes.

Algorithm 3 describes the procedure of cache placement, where CS_j represents the CS of node j ($j \in U_{cache}$). All nodes in U_{cache} are numbered from 0 to $k-1$ ($k = \mid U_{cache} \mid$). MPD is totally cached at caching nodes; each caching node receives the same MPD from RSU i. As for LPD, assuming $Chunk_t$ denotes the chunk t of content C_r, RSU i makes caching decision for a caching node numbered $n (0 \leq n < k)$ according to the modulus operation between t and n.

Fig.6 shows an example of cache placement, content C_1 is MPD, and content C_2 and C_3 are LPD with 9 chunks. Assuming RSU selects cluster heads CH_1 and CH_2 as caching nodes after receiving beacons, MPD C_1 is totally cached at both cluster heads. Content C_2 and C_3 are cached by taking the number of chunk modulo 2 (2 caching vehicles), for example, 4 mod 2 is 0, therefore Chunk 4 is stored at the vehicle CH_1. By that analogy, caching vehicle CH_1 and CH_2 cache chunks (0, 2, 4, 6, 8) and (1, 3, 5, 7), respectively. It is noteworthy that extra popular content C_3 is able to be cached

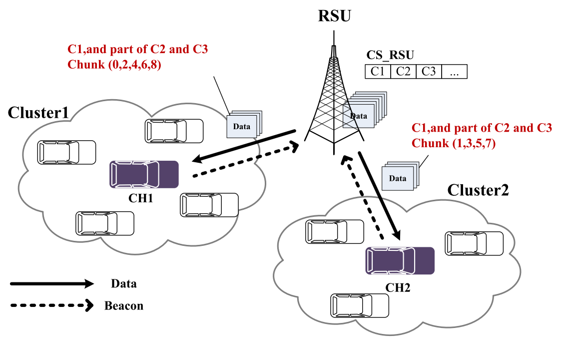

FIGURE 6. An example of cooperative caching.

TABLE 2. Neighbor head table.

Notation	Description
ID	vehicle identification
content	content name
number	number of cached chunks

at caching nodes, which indicates that distributed caching can improve the diversity of cached data.

Meanwhile, RSUs record the caching information of each caching node into Neighbor Head Table (see Table 2), where the table contains the vehicle ID, the name of cached contents, and the number of cached chunks. Each caching node holds the table accepted from the RSUs, and hence the nodes can fetch desired data according to the table.

In the designed cache placement, the MPD is totally cached at all nodes, and the LPD are distributedly cached at different nodes. It encourages caching nodes to cache more popular contents, thus increasing the chances of requests satisfied through the communication over vehicle clusters.

C. TRANSMISSION

When new data requests are generated, the requested data should belong to one of these three types: MPD, LPD, and uncached data. COMP designs three transmission schemes based on these data types to enable requesting vehicles to fetch data from intra-cluster and inter-cluster links, as these links are relatively stable and reliable.

Scenario 1(Request MPD): As MPD is cached at all caching nodes, the cluster member can directly fetch MPD from local cluster head. As shown in Fig.7(a), V_2 and V_3 are cluster heads of vehicle V_1 and V_4, respectively. MPD C_1 is cached at both V_2 and V_3 according to the proposed cache placement scheme. When V_1 and V_4 request for MPD C_1, they sent Interest packets toward their local heads V_2 and V_3, respectively. After receiving the Interests, V_2 and V_3 find that their CSs have cached the C_1, and then send the packet to the requesting vehicles.

Scenario 2(Request LPD): Fig.7(b) shows an example. Cluster members V_1 and V_4 request for LPD C_2 and C_3, respectively. The partitions of C_2 and C_3 are cached at caching nodes V_2 and V_3 according to cache placement scheme, chunk (0, 2, 4, 6, 8) and the chunk (1, 3, 5, 7) of C_2 and C_3 have been cached at cluster head V_2 and V_3, respectively. When V_1 requests for C_2, V_1 firstly sends the Interest packet to the local head V_2. After receiving the Interest, V_2 delivers the chunk (0, 2, 4, 6, 8) of C_2 to V_1, and then forwards the Interest toward neighbor head V_3 based on the caching information recorded in Neighbor Head Table. Then V_3 delivers the rest chunk (1, 3, 5, 7) of C_2 to V_1 along the request path and finally V_1 can receive total chunks of C_2. Member V_4 can obtain the C_3 from V_2 and V_3 in the same way.

Scenario 3(Request Uncached Data): When a cluster member requests for data that has not been cached in advance at caching nodes, its local head will return a NACK packet after receiving Interest packets from the requesting member. An example is shown in Fig.7(c), where both cluster members V_1 and V_4 request for uncached data C_4. Upon receiving the Interest and finding that there is no matched Data in CSs, their local cluster heads V_2 and V_3 send Nack packets to V_1 and V_4, respectively. In the end, V_1 and V_4 request for and fetch C_4 from the nearest RSU.

As for the requests generated from cluster head, the cluster head looks up the local CS for a hit first and then obtains data according to the corresponding scenarios described above. As for the requests generated from orphan vehicle, the orphan vehicle directly requests and fetches from the nearest RSU.

Based on the transmission schemes, vehicles can acquire desired data through inter-cluster and intra-cluster first, which promotes vehicles to access to stable and reliable data service via COMP.

VI. EVALUATION AND DISCUSSION

We design an experiment to evaluate the feasibility and efficiency of the proposed COMP strategy. Experimental results are showed and discussed in this section.

A. EXPERIMENTAL SETUP

Both SUMO [30] and ndnSIM [31] are employed to build our experimental platform, where they are respectively used to simulate an urban vehicular scenario and implement the NDN communication model.

We use SUMO to simulate a Manhattan model and generate location sequence of vehicles. The Manhattan model consists of three horizontal and three vertical bidirectional streets with an area of $2 \times 2\ km^2$. Twelve RSUs are uniformly deployed along the streets with a 500 m interval, and varying network sizes ranging from 100 to 700 vehicles in a street are simulated. Part of location sequences are used to train the Markov model, and the rest sequences are used to simulate the vehicular network via ndnSIM. We mainly focus on the evaluation of cooperative caching, and thus we consider the cache space of vehicles is only used for cooperative caching and the RSUs have cached all simulated contents. The rest simulation parameters are summarized in Table 3.

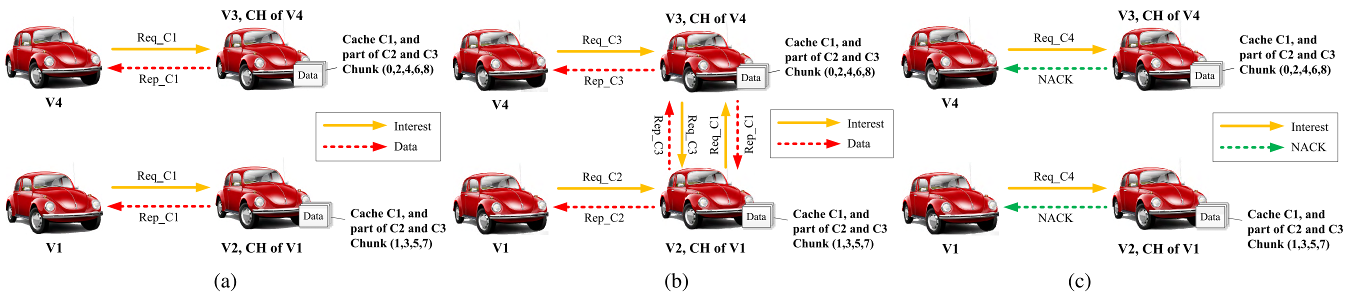

FIGURE 7. Transmission schemes for different request scenarios. (a) Request MPD. (b) Request LPD. (c) Request uncached data.

TABLE 3. Simulation parameters.

Parameter	Value
Total Caching Size	40%
Network Size	100 vehicles
No. of Contents	1000
Content Size	1 M
Simulation Time	120 s
Maximum Vehicle Speed	7 m/s
RSU Transmission Range	500 m
Vehicle Transmission Range (R)	100 m
Length of cluster (l)	50 m
Period of broadcasting beacon (ΔT)	1 s
Parameter (β, η, γ)	1.2
LPD caching ratio (ρ)	0.0-0.7

At the beginning we analyze the cost of the proposed COMP strategy and later evaluate the performance of prediction-based clustering algorithm by observing the stability and coverage of vehicle clusters. The performance of cooperative caching is assessed by comparing COMP, AlwaysCache, and ProbCache, according to several metrics linked with network performance (i.e., average access delay, server request ratio, cache hit ratio and average hit distance). Both AlwaysCache and ProbCache are the most commonly used caching strategies in VNDN, which have been illustrated in Section II. To evaluate the feasibility and efficiency of distributedly caching LPD, we also compare COMP with $\rho = 0$ (vehicle only caches MPD) and $\rho = 0.5$ (half the cache space is used to cache MPD, and the rest is used to cache LPD). Finally we study the effects of caching portion of LPD by varying ρ from 0.1 to 0.7 to learn which strategy performs the best.

The metrics related to the network performance are illustrated as follows:

- **Average Access Delay**, the average time interval between requesting vehicle sending Interst and retrieving Data.
- **Server Request Ratio**, the ratio of number of requests satisfied from RSUs to the total number of requests.
- **Cache Hit Ratio**, the ratio of number of Interests satisfied from the cache to the total number of Interests arrived at cache.
- **Average Hit Distance**, the average number of hops that the received Data traveling from the data provider.

B. COST ANALYSIS

In COMP, the computation complexity of clustering algorithm and cooperative caching is $O(MN^2)$ and $O(CL)$, where M, N, C denote the number of vehicles being predicted, locations predicted in Markov model, and cluster heads, respectively, and L denotes the size of LPD. Particularly, the analysis results show that the number of predicted locations N and the size of LPD L are closely linked with the cost and efficiency of COMP.

The number of predicted locations N relies on the length of location r, which not only affects the cost of COMP but also the prediction accuracy. A smaller r can lead to a more complex model and possibly produce over-fitting problems. A larger r corresponds to a simpler model but may represent the possible future location inaccurately. Through testing for different r ranging from 5 m to 25 m, we find that the setting of $r = 10$ m can get a representative Markov model while reducing the computation complexity of the algorithm.

The size of LPD L depends on the cache size and LPD caching ratio ρ. The larger cache size and ρ are corresponding to a larger LPD size, which means more content chunks need to be processed in cache placement and then higher computation complexity will be produced. The smaller cache size and ρ will lead to a smaller LPD size, which is corresponding to lower computation complexity but lower diversity of cached data. The lower diversity of cached data may affect the performance of COMP. The effect of cache size and ρ will be further discussed in later subsections.

C. PERFORMANCE ANALYSIS OF CLUSTERING ALGORITHM

In the experiment, vehicle clusters are established and maintained according to the proposed clustering algorithm. This approach aims to group vehicles with similar mobility pattern and alleviate the impact from vehicle movements on the stability and coverage of vehicle clusters. To validate the

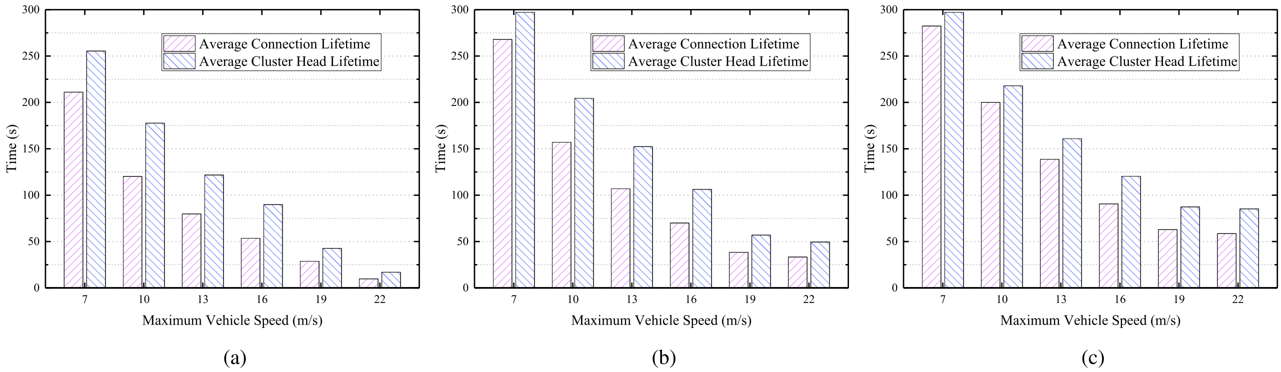

FIGURE 8. Comparison results of average connection lifetime and average cluster head lifetime at different vehicle speeds. (a) $R = 100$ m. (b) $R = 200$ m. (c) $R = 300$ m.

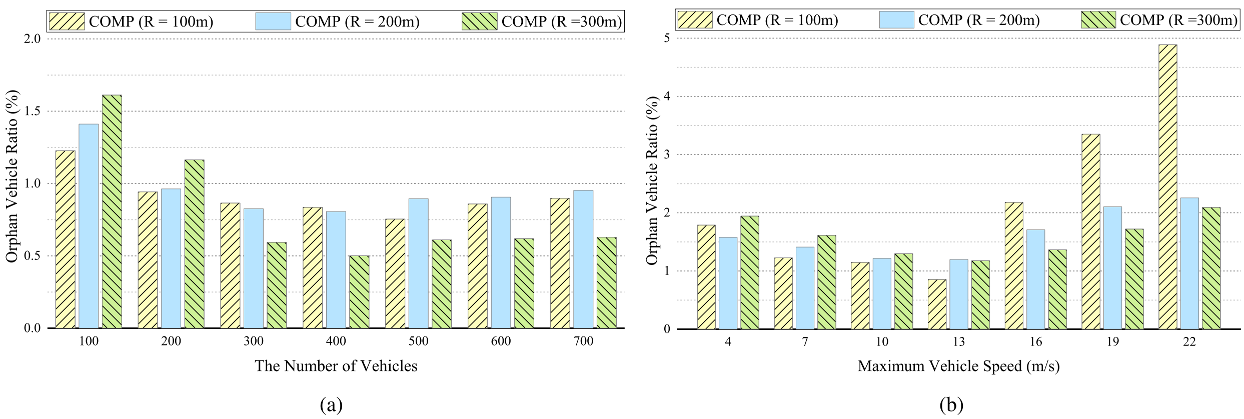

FIGURE 9. Orphan vehicle ratio with different network sizes and maximum vehicle speeds. (a) Orphan vehicle VS. The number of vehicles. (b) Orphan vehicle VS. Maximum vehicle speeds.

performance of clustering algorithm, we assess the stability and coverage of vehicle clusters constructed via COMP.

The stability of vehicle clusters can be accessed by comparing average connection lifetime and average cluster head lifetime, where the former is the average connection duration of any neighboring vehicles and the latter is defined as the time interval between cluster head entering and discarding CH state (i.e., cluster lifetime). Fig.8 shows the comparison at different vehicle speeds and different transmission ranges.

With the maximum vehicle speed grows from 7 m/s to 22 m/s, both the average connection lifetime and the average cluster head lifetime decrease because high speed enhances the dynamic of network topology as well as the uncertainty and unpredictability of vehicle mobility pattern. It is evident that the average cluster head lifetime is always longer than the average connection lifetime (up to 73.5%), which indicates that the intra-cluster communication constructed by COMP can provide stable and reliable data service for VNDN. One reason is that, the prediction-based clustering algorithm can accurately predict the future mobility pattern of vehicles, thereby selecting the most suitable vehicles as cluster heads and establishing stable vehicle clusters. Another reason is that, the cluster maintenance in clustering algorithm mitigates the impact of vehicle mobility on the stability of vehicle clusters so that the lifetime of clusters can be extended.

The coverage of vehicle clusters can be evaluated by the orphan vehicle ratio and average cluster spreading degree, where the former is the percentage of orphan vehicles in all moving vehicles and the latter is the average number of cluster members in a cluster. Fig.9 and Fig.10 respectively present the orphan vehicle ratio and average spreading degree at different number of vehicles and maximum vehicle speeds. In Fig.9(a), with the number of vehicles raise from 100 to 300, orphan vehicle ratio at $R = 100$ m, 200 m, 300 m decreases from 1.23% to 0.86%, 1.41% to 0.83%, 1.61% to 0.59%, respectively. In Fig.10(a), average cluster spreading degree with different transmission range increases from 4.4 to 7.85, 6.83 to 15.62, 12.24 to 23.40, respectively. When the number of vehicles exceeds 300, both orphan vehicle ratio and average cluster spreading degree generally level off. On one hand, the difficulty of establishing vehicle clusters is increasing in the sparse environment, as the distance between vehicles is always longer than the length of cluster l. On the other hand, with the growing number of vehicles, the number of cluster heads also increases and hence the cluster spreading degree can not grow endlessly.

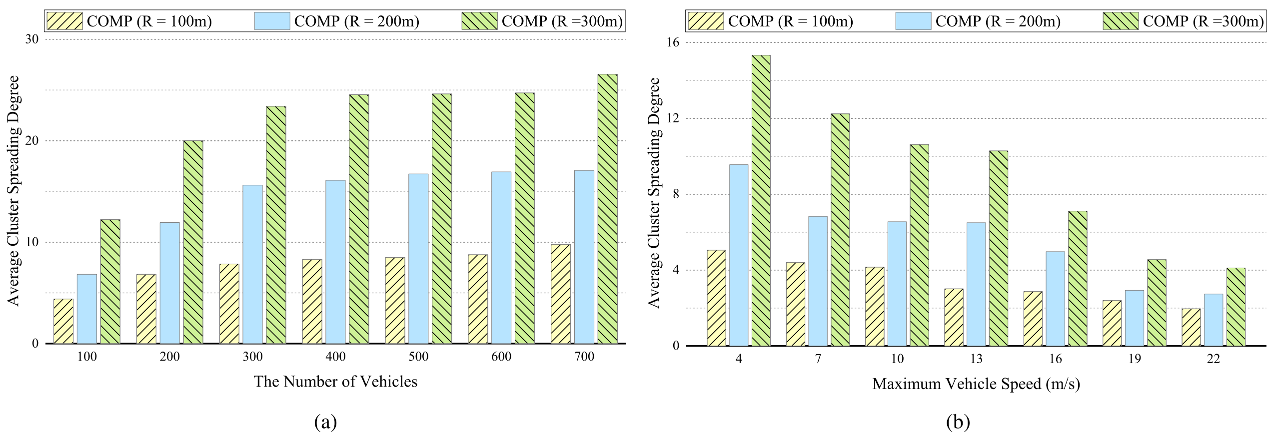

FIGURE 10. **Spreading degree with different network sizes and maximum vehicle speeds. (a) Average cluster spreading degree VS. The number of vehicles. (b) Average cluster spreading degree VS. maximum vehicle speeds.**

Fig.9(b) depicts that when the maximum vehicle speeds go up from 4 m/s to 13 m/s, the orphan vehicle ratio declines and soon begins rising up to no more than 4.89%. Fig.10(b) shows that, as the speed increases, the average cluster spreading degree decreases all the time. It can be illustrated that the higher vehicle speed causes the drop in average spreading degree while facilitating orphan vehicles to join clusters; however, overspeed ($>$13 m/s) makes that joining difficult and increases the number of orphan vehicles.

From Fig.9 and Fig.10, it can be seen that the vehicle clusters established via COMP can cover most of vehicles ($>$95%) and keep a certain cluster spreading degree ($2-26$) in different cases. It is proven that the prediction-based clustering algorithm is feasible and efficient to establish and maintain stable vehicle clusters, hence this approach provides a foundation to build stable and reliable intra-cluster and inter-cluster communication via cooperative caching.

D. EFFECT OF TOTAL CACHE SIZE

To study the effect of total cache size on system performance, the metrics related to network performance are measured at total cache sizes ranging from 10% to 70%, where the total cache size is the ratio of the size at total cache space to total simulated contents. COMP with different LPD caching ratio ($\rho = 0, 0.5$) are compared with two most widely used caching strategies, AlwaysCache, and ProbCache.

Fig.11(a), 11(b) and 11(d) reveal that as the cache gets larger, the average access delay, server request ratio and average hit distance will all decline. This is because the larger cache size causes vehicles to cache more data, and then consumers can obtain desired data from surrounding nodes more easily. It is noteworthy that COMP strategy performs better than other two contrast strategies. For example, when the total cache size is 25%, the average access delay of AlwaysCache and ProbCache are 0.17 s and 0.27 s, respectively, the delay of COMP ($\rho = 0.5$) is 0.12s, achieving decrease of 29.58% and 56.22%. In addition, COMP ($\rho = 0.5$) has lower server request ratio than both contrast proposals with 26.71-66.85% and 54.60-69.66%, respectively, and always keeps a lower level of average hit distance ($<$2.1). The main reason is that, COMP enables vehicles to acquire data through stable intra-cluster and inter-cluster communication instead of long-distance RSUs, thus improving the network performance. Notably, when the total cache size is larger than 55%, the average hit distance of AlwaysCache becomes shorter than COMP ($\rho = 0.5$). Because larger cache size means that more data can be cached among vehicles, requesters have more chances to fetch data from surrounding vehicles. However, the design of distributed caching of COMP ($\rho = 0.5$) allow requesters to obtain LPD from several caching nodes, which causes COMP ($\rho = 0.5$) to have a larger hit distance than other contrast strategies.

Fig.11(c) shows that the average hit ratio improves with increasing cache size for all schemes. We observe that COMP ($\rho = 0.5$) produces at least 47.98% and 86.41% higher hit ratio than AlwaysCache and ProbCache, respectively. It can be inferred that the larger cache size allows more popular contents to be cached in vehicles and later creates more chances for requesters to obtain data from neighbor nodes, thereby effectively raising the cache hit ratio.

It is noteworthy that although the server load and hit ratio of COMP ($\rho = 0.5$) perform better, its delay and hit distance are higher than $\rho = 0$. The main reason is that, requesting vehicles need to fetch LPD from multiple nodes as COMP ($\rho = 0.5$) distributedly caches LPD at multiple nodes. We still reckon that COMP ($\rho = 0.5$) overall performs better, as this approach enhances the diversity of cached data and significantly improves the cache hit ratio and server load for vehicular network.

E. EFFECT OF NETWORK SIZE

To study the effect of network size on system performance, the network performance is evaluated by increasing the number of vehicles from 100 to 700.

Fig.12(a) and 12(b) indicate that average access delay and server request ratio of four schemes decline with the

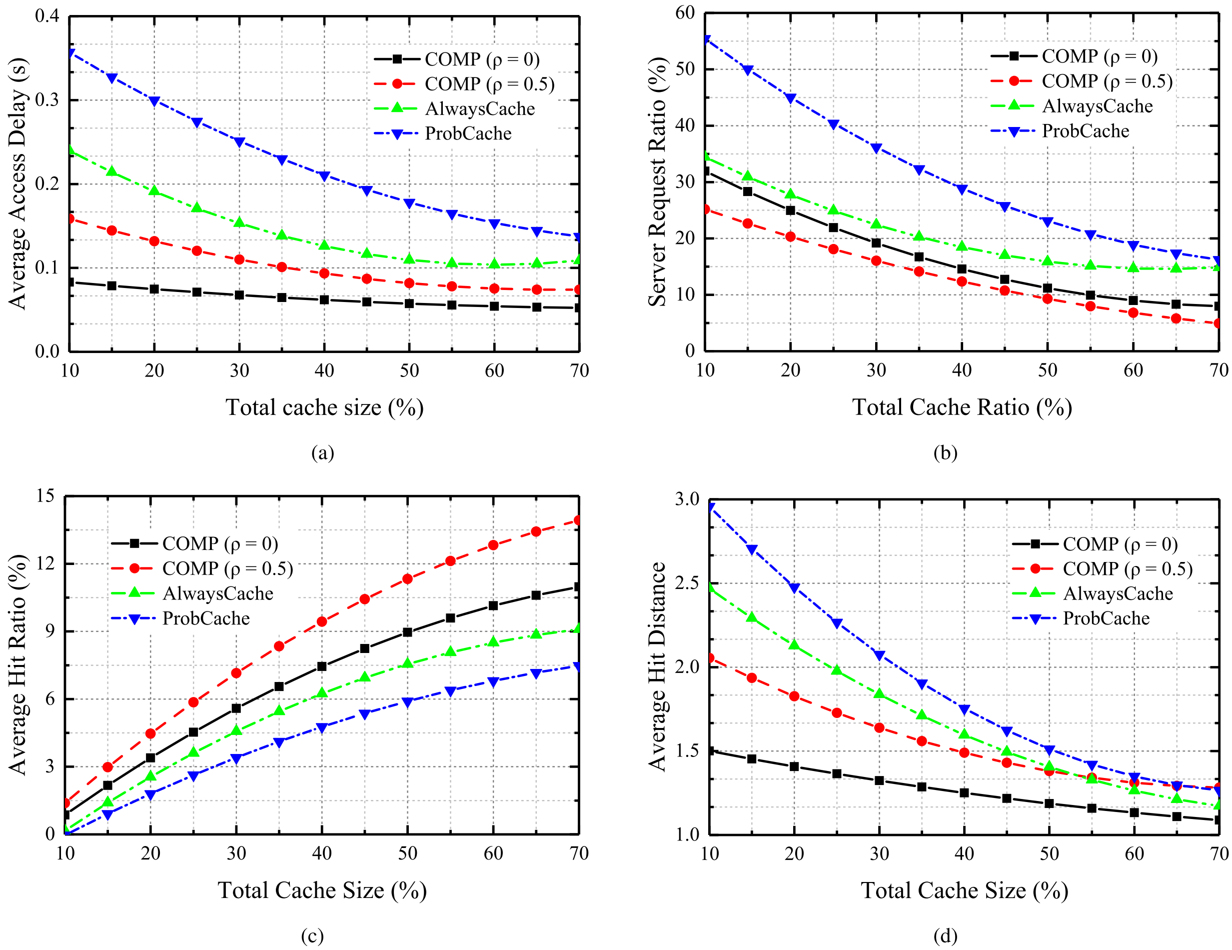

FIGURE 11. The effect of total cache size on system performance. (a) Average access delay. (b) Server request ratio. (c) Average cache hit ratio. (d) Average hit distance.

growth of number of vehicles, but the average hit ratio and hit distance are on the contrary. The larger network size represents more contents are cached among vehicles, and hence requesting vehicles have more chances to fetch data from surrounding vehicles.

It is evident that COMP performs the best. For instance, when the number of vehicles is smaller than 550, COMP ($\rho = 0.5$) results in at least 25.26% and 46.11% lower access delay, 41.23% and 62.72% fewer server request, 9.5% and 12.59% higher hit ratio, and 13.68% and 14.79% lower hit distance than AlwaysCache and ProbCache, respectively. The high dynamic of vehicle movements deteriorate the performance of AlwaysCache and ProbCache, as these proposals rarely consider the impact of vehicle movements. Unlike the contrast proposals, COMP establishes stable cluster-based communication through cooperative caching, which efficiently alleviate the mobility issues in VNDN. When the number of vehicles is larger than 550, the delay of AlwaysCache is smaller than that of COMP ($\rho = 0.5$), and almost equals that of COMP ($\rho = 0$). The increase of network size means more vehicles to become producers, hence more data demands can be met by one-hop neighboring vehicles in AlwaysCache. Because vehicles need to request and fetch LPD from multiple caching nodes in COMP ($\rho = 0.5$), its delay is longer than that of AlwaysCache.

It should be noted that the performances of COMP ($\rho = 0.5$) and COMP ($\rho = 0$) are similar except average hit ratio. The main reason about the difference is that the diversity of cached data in COMP ($\rho = 0.5$) is higher than that in COMP ($\rho = 0$), thus increasing the success ratio of searching hits in the CS of neighbor vehicles.

F. EFFECT OF VEHICLE SPEED

To study the effect of vehicle speed on system performance, the network performance is assessed by varying maximum vehicle speeds from 4 m/s to 22 m/s.

In Fig.13, with the vehicle speed grows, the average access delay, server request ratio and average hit distance of all strategies increase, and the average hit ratio decreases. For example, Fig.13(a) shows that the delay of AlwaysCache and ProbCache rise from 0.10 s and 0.20 s to 0.61 s and 0.60 s, respectively, as the higher vehicle speed represents the more severe impact of vehicle mobility. Frequent link disruption and packet loss and retransmission due to high dynamic of

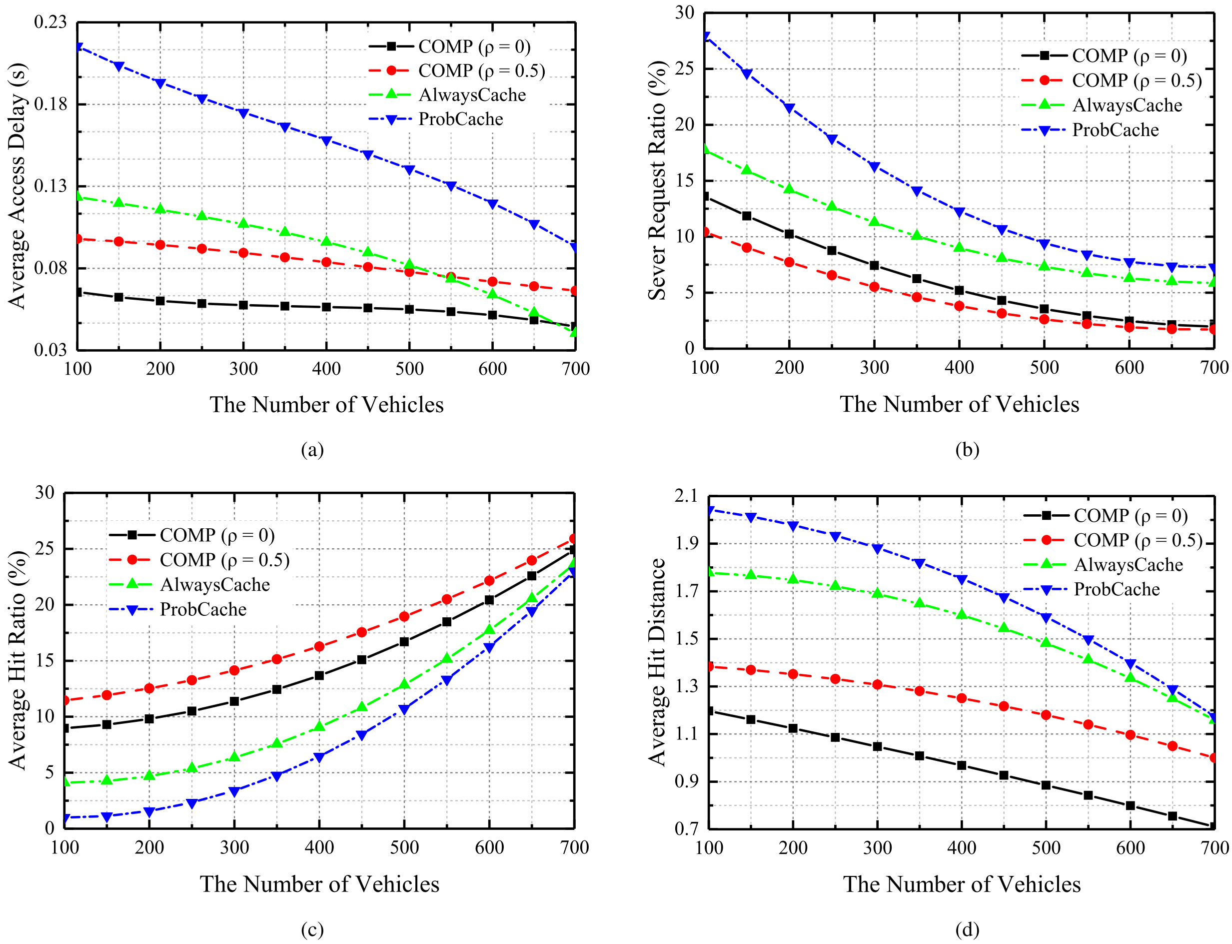

FIGURE 12. The effect of network size on system performance. (a) Average access delay. (b) Server request ratio. (c) Average cache hit ratio. (d) Average hit distance.

network topology lead to the degradation of network performance.

It is observed that, COMP performs the best and is less affected by vehicle mobility than other strategies in most cases. For example, as shown as Fig.13(b), in the cases of different speeds, COMP ($\rho = 0.5$) has lower server request ratio than AlwayCache and ProbCache with 17.46-33.97% and 31.38-48.15%. The main reason is that, via cooperative caching, the intra-cluster and inter-cluster communication provide good support for stable data retrieval, which significantly mitigates the impact of vehicle mobility on network performance. But when the maximum vehicle speed is lower than 7 m/s, AlwaysCache performs better than COMP ($\rho = 0.5$) in terms of average hit distance. This is because that AlwaysCache is little affected by mobility issues in the cases of small vehicle speed, and COMP ($\rho = 0.5$) needs vehicles to get LPD through multi-hop links. In addition, COMP ($\rho = 0.5$) has lower server request ratio and higher average hit ratio but higher access delay and hit distance than $\rho = 0$. Compared with COMP ($\rho = 0$), COMP ($\rho = 0.5$) is more affected as it needs requesters to fetch LPD from multiple nodes; however, this approach increases the diversity of cached contents and hence facilitates to reduce server load and increase hit ratio.

G. EFFECT OF DISTRIBUTEDLY CACHING LPD

To validate the feasibility and efficiency of distributedly caching LPD, its effect is also studied with different LPD caching ratio ρ, ranging from 0.1 to 0.7.

To compare with the strategy, which totally caches MPD at all caching nodes (i.e., COMP ($\rho = 0$)), in a more convenient way, we normalize the values relative to the baseline of COMP ($\rho = 0$)); e.g., for COMP ($\rho = 0.5$) the normalized average access delay will be $NormDelay_{\rho=0.5} = \frac{Delay_{\rho=0.5}}{Delay_{\rho=0}}$. Note that for average access delay, server request ratio and average hit distance, higher values of the normalized metric indicate worse performance and for average hit ratio, lower values indicate worse performance.

Fig.14 shows the comparison results of normalized values. When ρ is increasing from 0.1 to 0.7, both average hit ratio and server load of COMP are decreasing. Because higher ρ implies that more data can be cached among vehicles via COMP, it can help more requests to be satisfied by caching vehicles. Additionally, COMP ($\rho = 0.1 - 0.7$) has higher

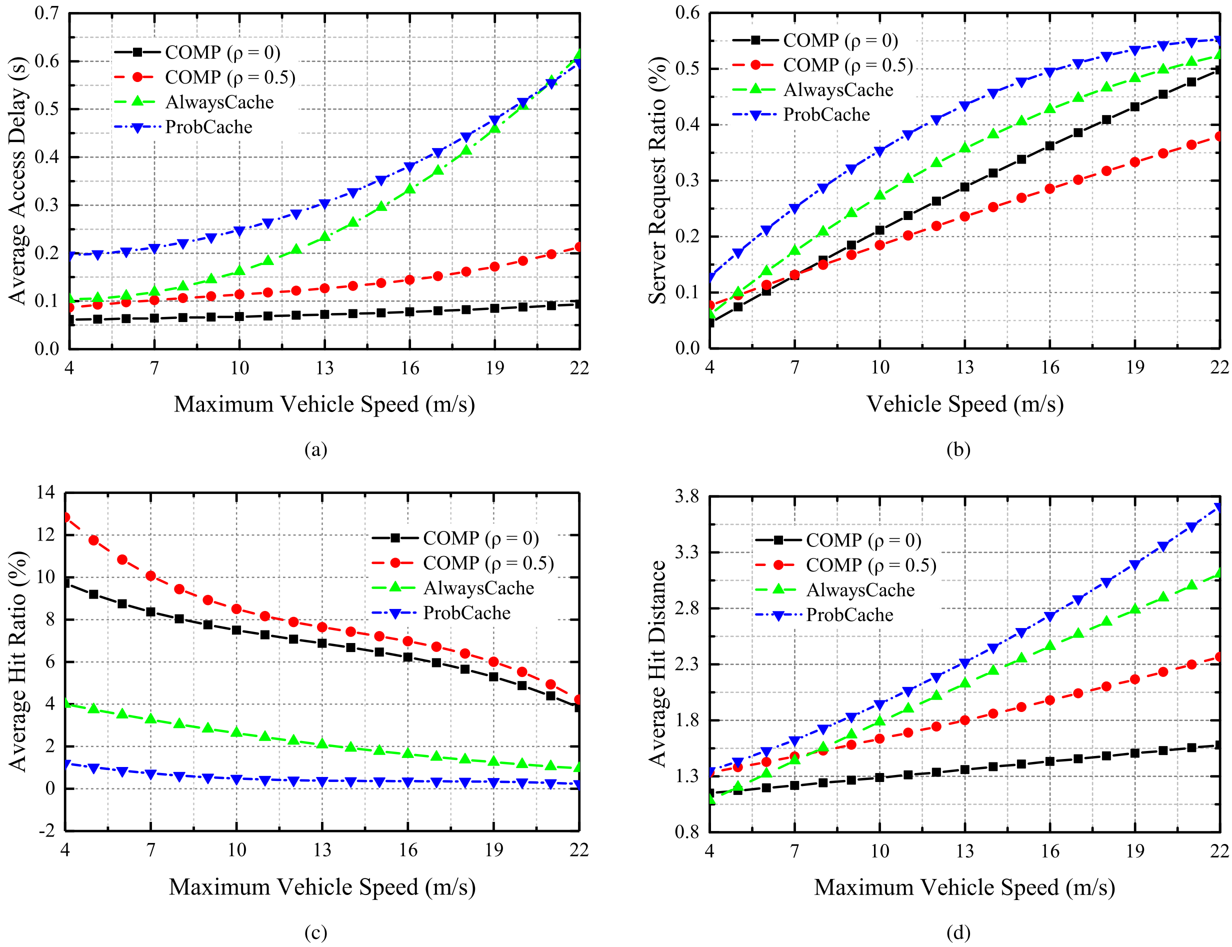

FIGURE 13. The effect of vehicle speed on system performance. (a) Average access delay. (b) Server requesting ratio. (c) Average cache hit ratio. (d) Average hit distance.

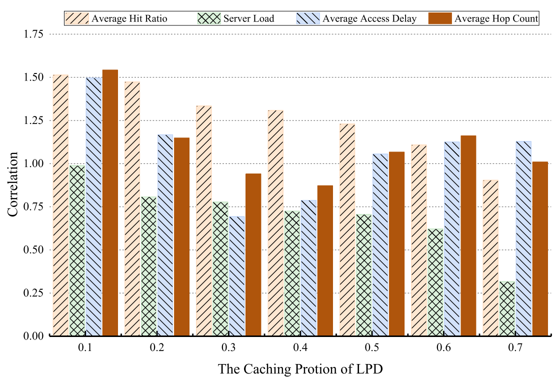

FIGURE 14. Performance comparison among COMP with different ρ.

average hit ratio and lower server request ratio than COMP ($\rho = 0$). For example, when the LPD caching ratio $\rho = 0.4$, the normalized values $NormHit_{\rho=0.4}$ and $NormSevr_{\rho=0.4}$ are 1.31 and 0.73, respectively. This is because, compared with COMP ($\rho = 0$), distributed caching can increase the diversity of cached data, which improves the average hit ratio and the server load up to 33% and 78%, respectively.

With the ρ increasing from 0.1 to 0.3, average access delay and average hop count are decreasing, and then both metrics are increasing when the ρ increases from 0.4 to 0.7. The main reason is that, higher ρ increases the chances of requesters to obtain data from neighboring caching vehicles, and then mitigates the impact of vehicle mobility. It is obvious that when ρ ranges from 0.3 to 0.4, both average access delay and average hop count perform better than $\rho = 0$. In other cases, distributedly caching LPD is not so effective. This is because when a requester demand cannot be met by local and neighbor cluster heads and receives a Nack response, the requester requests and fetches data from long-distance RSU. A smaller ρ corresponds to lower diversity of cached data and causes more requests to be satisfied by RSU-to-vehicle communication. A higher ρ represents higher data diversity, so that more requests need to be satisfied by multiple nodes.

To sum up, the performance of COMP is closely linked with the LPD caching ratio ρ, and distributed caching performs better than centralized caching when ρ value is set to be 0.2-0.6. Notably, COMP can obtain a best performance with $\rho = 0.3$

VII. CONCLUSION

In this paper, we propose a cooperative caching strategy to mitigate the impact of vehicle mobility on data delivery by constructing stable communication via prediction-based clustering algorithm and cooperative caching. To evaluate the feasibility and efficiency of proposed COMP approach, we construct a realistic urban vehicular network through the SUMO and ndnSIM emulator. The evaluation results show that, the proposed COMP strategy can provide stable and reliable data delivery for most of vehicles (>95%), and COMP can improve system performance up to 86.4% by comparing with commonly used caching strategies. The data type based design of COMP is able to increase the cache efficiency and network performance, where the COMP with 0.3 caching ratio of less popular data can obtain the best performance.

REFERENCES

[1] A. Bazzi, B. M. Masini, and G. Pasolini, "Real time acquisition of traffic information through V2V, V2R, and V2I communications," in *Roadside Networks for Vehicular Communications: Architectures, Applications, and Test Fields*. Pennsylvania, PA, USA: IGI Global, 2013, pp. 72–92.

[2] A. Pineda-Briseño, "A probabilistic approach to location estimation in manets," *Adhoc Sensor Wireless Netw.*, vol. 28, no. 1, pp. 97–114, 2013.

[3] L. Zhang *et al.*, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[4] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang, "VANET via named data networking," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr./May 2014, pp. 410–415.

[5] Z. Yan, S. Zeadally, and Y.-J. Park, "A novel vehicular information network architecture based on named data networking (NDN)," *IEEE Internet Things J.*, vol. 1, no. 6, pp. 525–532, Dec. 2014.

[6] W. Gao, G. Cao, A. Iyengar, and M. Srivatsa, "Cooperative caching for efficient data access in disruption tolerant networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 611–625, Mar. 2014.

[7] I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali, and A. Habbal, "Caching in information-centric networking: Strategies, challenges, and future research directions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1443–1474, 2nd Quart. 2018.

[8] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. ICN*. New York, NY, USA: ACM, 2012, pp. 55–60.

[9] Z. Li and G. Simon, "Time-shifted TV in content centric networks: The case for cooperative in-network caching," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–6.

[10] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *Proc. IEEE INFOCOM Workshops*, Mar. 2012, pp. 316–321.

[11] S. Saha, A. Lukyanenko, and A. Ylä-Jääski, "Cooperative caching through routing control in information-centric networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 100–104.

[12] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-AS cooperative caching for content-centric networks," in *Proc. 3rd ACM SIGCOMM Workshop Inf.-Centric Netw.* New York, NY, USA: ACM, 2013, pp. 61–66.

[13] L. Zhang, J. Zhao, and Z. Shi, "LF: A caching strategy for named data mobile ad hoc networks," in *Proc. 4th Int. Conf. Comput. Eng. Netw.* Springer, 2015, pp. 279–290.

[14] G. Deng, L. Wang, F. Li, and R. Li, "Distributed probabilistic caching strategy in vanets through named data networking," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 314–319.

[15] D. D. Van, Q. Ai, Q. Liu, and D.-T. Huynh, "Efficient caching strategy in content-centric networking for vehicular ad-hoc network applications," *IET Intell. Transport Syst.*, vol. 12, no. 7, pp. 703–711, 2018.

[16] X. Vasilakos, V. A. Siris, G. C. Polyzos, and M. Pomonis, "Proactive selective neighbor caching for enhancing mobility support in information-centric networks," in *Proc. 2nd Ed. ICN Workshop Inf.-Centric Netw.* New York, NY, USA: ACM, 2012, pp. 61–66.

[17] Y. Wei, C. Xu, M. Wang, and J. Guan, "Cache management for adaptive scalable video streaming in vehicular content-centric network," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Jul. 2016, pp. 410–414.

[18] L. Yao, A. Chen, J. Deng, J. Wang, and G. Wu, "A cooperative caching scheme based on mobility prediction in vehicular content centric networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5435–5444, Jun. 2018.

[19] Y. Li, J. Wang, A. Lindgren, X. Su, and J. Riekki, "Implicit cooperative caching based on information popularity for vehicular networks," in *Proc. 12th Workshop Challenged Netw.* New York, NY, USA: ACM, 2017, pp. 51–56.

[20] F. de Moraes Modesto and A. Boukerche, "Utility-gradient implicit cache coordination policy for information-centric ad-hoc vehicular networks," in *Proc. IEEE 42nd Conf. Local Comput. Netw. (LCN)*, Oct. 2017, pp. 10–17.

[21] W. Zhao, Y. Qin, D. Gao, C. H. Foh, and H.-C. Chao, "An efficient cache strategy in information centric networking vehicle-to-vehicle scenario," *IEEE Access*, vol. 5, pp. 12657–12667, 2017.

[22] Y. Rao, H. Zhou, D. Gao, H. Luo, and Y. Liu, "Proactive caching for enhancing user-side mobility support in named data networking," in *Proc. 7th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput. (IMIS)*, Jul. 2013, pp. 37–42.

[23] D. Grewe, M. Wagner, and H. Frey, "PeRCeIVE: Proactive caching in ICN-based VANETs," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2016, pp. 1–8.

[24] H. Farahat and H. Hassanein, "Optimal caching for producer mobility support in named data networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[25] N. Abani, T. Braun, and M. Gerla, "Proactive caching with mobility prediction under uncertainty in information-centric networks," in *Proc. 4th ACM Conf. Inf.-Centric Netw.* New York, NY, USA: ACM, 2017, pp. 88–97.

[26] W. Huang, T. Song, Y. Yang, and Y. Zhang, "Cluster-based selective cooperative caching strategy in vehicular named data networking," in *Proc. 1st IEEE Int. Conf. Hot Inf.-Centric Netw. (HotICN)*, Aug. 2018, pp. 7–12.

[27] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating next-cell predictors with extensive Wi-Fi mobility data," *IEEE Trans. Mobile Comput.*, vol. 5, no. 12, pp. 1633–1649, Dec. 2006.

[28] S. Qiao, D. Shen, X. Wang, N. Han, and W. Zhu, "A self-adaptive parameter selection trajectory prediction approach via hidden Markov models," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 284–296, Feb. 2015.

[29] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order Markov models," *J. Artif. Intell. Res.*, vol. 22, no. 1, pp. 385–421, 2004.

[30] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, *Sumo-Simulation of Urban Mobility: An Overview*. Simul, 2011, pp. 63–68.

[31] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," Tech. Rep., 2012.

WANYING HUANG (S'18) received the B.Eng. degree from the Guangdong University of Technology, Guangzhou, China, in 2017. She is currently pursuing the Ph.D. degree with the School of Information and Electronics, Beijing Institute of Technology, China. Her current research interests include named-data networks, vehicular ad hoc networks, and wireless networks.

TIAN SONG (S'05–M'08) received the M.Eng. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 2005 and 2008, respectively. He is currently an Associate Professor with the School of Computer Science, Beijing Institute of Technology, China. His current research interests include green networking, information-centric networking, high-speed packet processing, and network security.

YATING YANG (S'17) received the B.Eng. degree in computer science from the Beijing Institute of Technology, China, in 2016, where she is currently pursuing the Ph.D. degree with the School of Computer Science. Her research interests include future Internet, information-centric networks, and high-speed network processing.

YU ZHANG received the Ph.D. degree in electrical engineering from Peking University, China, in 2001. He is currently an Assistant Professor with the School of Information and Electronics, Beijing Institute of Technology, Beijing, China. He is currently interested in the delay and tolerant networks and named-data networks. His research interests include the modeling and simulation of the communication systems and networks and the performance analysis of the wireless networks.