A Cooperative Caching Scheme for VCCN with Mobility Prediction and Consistent Hashing

Lin Yao*‡, Xiaoying Xu†, Jing Deng§, Guowei Wu†, and Zhaoyang Li†
*DUT-RU International School of Information Science & Engineering, Dalian University of Technology, China
†School of Software, Dalian University of Technology, China
‡Network Research Centre, Peng Cheng Laboratory, Guangdong Shenzhen 518057
§Department of Computer Science, UNC Greensboro, U.S.A.

***Abstract*—In order to mitigate the performance degradation of intermittent vehicular network caused by traffic mobility and sporadic connectivity issues, Vehicular Content Centric Network (VCCN) has been proposed to apply many technologies in Content Centric Network (CCN) into vehicular ad hoc networks. The open in-network caching stragegy of CCN enables sharing and coordination of the cached data among multiple nodes as an efficient data access without relying on remote fetching. Nonetheless, few studies have considered the effective use of overall cache capacity with these cooperative nodes, especially the issues of cache duplication. Furthermore, most content replacement polices have ignored the needs of cooperative contents when making cache decisions. In this paper, we design a novel Cooperative Caching scheme by using Mobility Prediction and Consistent Hash for VCCN (called CCMPCH). Specifically, based on the observation that vehicles with the same trajectory are more likely to maintain stable communication links, we adopt Prediction by Partial Matching (PPM) to forecast each vehicle's path and cluster the vehicles with similar future path, moving direction and moving speed into one group. In each cluster, the consistent hash algorithm is used to allocate contents among cooperative nodes, thereby reducing unnecessary cache duplication while maintaining strong content availability. A popularity-based cache replacement policy is also developed to prioritize cooperative contents. We evaluate CCMPCH via extensive simulations, which demonstrate its higher cache hit ratio, shorter content access delay, and lower hop count compared to other state-of-the-art schemes.**

I. Introduction

VEHICULAR Ad-hoc Network (VANET) is an important component of Intelligent Transportation System (ITS), consisting of vehicles and Road Side Units (RSUs). Besides improving road safety, VANET is expected to offer commerce, information, and entertainment services to drivers and passengers [1]. Natural movement of vehicles causes rapidly changing topology, and short lived links between vehicles. Therefore, traditional TCP/IP protocol based on the static topology is unsuitable for VANET without the ability of dealing with its frequent link disruptions. Besides, vehicles usually maintain communication over the Dedicated Short-Range Communications (DSRC) based on IEEE 802.11p standard, which provides a limited communication bandwidth.

Interestingly, recent studies have shown a surprising correlation among interests from different users, leading to the concept of content caching in local regions to support users' requests and reduce the average bandwidth consumption, giving rise to the so-called Vehicular Content Centric Network (VCCN) by emerging CCN into VANET [1][2][3], In CCN or VCCN, there are usually two types of message exchanges: content request packet (called "Interest") and content return packet (called "Data"). Users send Interests requesting for specific contents without knowing target addresses. Such Interests are then forwarded within a limited scope either in time or in number of hops (time-to-live) with all intermediate nodes helping to forward them. Data packets will eventually be sent back to the requesting users. Each node in the forwarding path of the Data packet can decide whether to cache it based on its cache policy. Consequently, an efficient in-network caching strategy will be needed in order to make this approach successful.

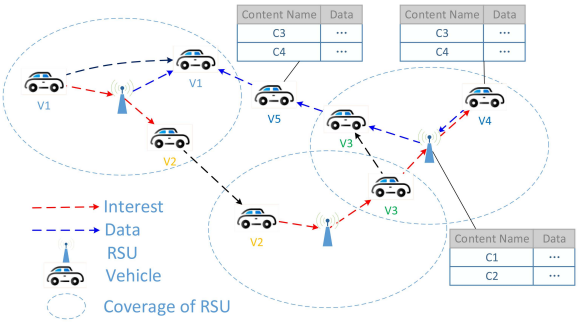


Figure. 1 VCCN Network Model

Caching schemes in CCN can be classified into non-cooperative caching and cooperative caching ones [4]. In non-cooperative caching, each CCN node decides its cache replacement policy independently, which may lead to problems of frequent cache updates, cache redundancy, etc. For example, the same content may be cached among multiple nodes in close proximity, causing low cache utilization. Cooperative caching, conversely, allows the content sharing and buffer resource coordination among multiple nodes, mitigating the deficiencies of non-cooperative caching. Cooperative nodes can further support off-path caching under the control of a centralized manager [5][6] or on-path caching by caching Data packets along the forwarding path of the corresponding Interest [7][8][9].

Due to vehicle mobility, off-path caching is not suitable

where a centralized manager must be determined to take charge of maintaining the cache state of the network. Still, efficiency of current cooperative caching schemes remains very low, mainly due to the difficulty of identifying the best nodes to cache popular contents [10][11]. Though [11][12][13][14] made use of the past trajectory information of vehicles to predict their future moving paths in order to select the appropriate caching nodes, they have not considered how to allocate the cooperative contents between these nodes. The main issue is that nodes mostly consider popular contents independently and there lacks coordination among themselves. To improve the cache hit ratio and reduce content retrieve delay, the total cache space in the network must be utilized efficiently and in a collective fashion. In order to address these challenges, we incorporate mobility prediction and consistent hashing into our design of a new content caching scheme with mitigating cache duplicates. Our design is called Cooperative Caching scheme based on Mobility Prediction and Consistent Hash (CCMPCH). The main contributions of this work include:

(1) We propose a novel cooperative caching strategy with the integrated consideration of bridging centrality as well as the vehicular moving pattern. Vehicles are clustered based on their path predictions, current moving directions, and speeds. Nodes with higher bridging centrality tend to be elected as the cooperative caching node.
(2) We are the first to perform consistent hashing on cooperative caching nodes to allocate different contents to improve the storage utility within the same cluster and avoid the cache duplication among the adjacent nodes.
(3) We design a popularity-based caching replacement policy by combining the need for cooperative contents and the local distribution of Interests. A preference constant is introduced in popularity calculation on cooperative contents to raise their priority in the cache decision. We then optimize the preference constant to achieve the lowest average hop count for successful retrievals.
(4) CCMPCH is compared with other state-of-the-art cooperative caching techniques through extensive simulations. The results show that CCMPCH outperforms others in terms of cache hit ratio, access delay, and hop count.

The remainder of this paper is organized as follows. In Section II, we discuss the related work. The detailed design of the CCMPCH scheme is presented in Section III. In Section IV, we derive an analysis of the hop count based on a so-called preference constant β. We evaluate the performance of CCMPCH in Section V and conclude the work in Section VI.

II. Related Work

Our review of related work consists cooperative content caching and consistent hashing.

Cooperative Content Caching: We further divide cooperative caching schemes into ones without a central-controlled node and those with a central-controlled node.

Cooperative caching schemes *without a central-controlled node* can collaborate to complete content distribution within all the mobile nodes. A caching scheme for Mobile Ad Hoc Network (MANET) was first proposed in [15], where the network was divided into several regions and clusters of mobile devices residing in each region. The cluster head was responsible for distributing contents within the cluster members. In [16], K-nearest method was used to cluster neighbors by adjusting content placement and cluster size to balance content diversity and spectrum efficiency. A set of network central locations were chosen to cache contents based on a probabilistic selection metric, which could be easily accessed by other nodes [17][18]. In [19], the demands for documents were not restricted to specific application messages but they were codified into the messages when the path to the server was being searched. Furthermore, these messages could provide useful information about potential localizations of the documents. In DPC [10], each vehicle independently decided its caching strategy by taking into account three factors, the received Interests, its centrality in the network, and its moving direction. Game theory was adopted to determine optimal cooperators and the size of cached data in [20]. In [21], an edge caching scheme was proposed to make full use of the limited storage space for edge nodes.

Since vehicles are expected to move, their mobility pattern should be used in cache consideration [12][13][14]. In [11], a clustering algorithm was proposed to establish and maintain vehicle clusters based on the prediction of future moving path and link expiration time, with each cluster head responsible for content distribution. Social attributes were exploited to design content caching strategy and determine cooperative nodes in mobile ad hoc networks [22][23].

A *central-controlled* caching scheme was proposed for MANET [24], where a pre-fetch manager was designed to evaluate the popularity index of the data. If the popularity index was lower than a threshold, its space would be released to cache popular contents. Suno et al. [25] proposed a cooperative caching invalidation scheme along with its enhancement for VANET, where both a server and location management agent coordinate for cache invalidation. A server asynchronously sends an individual report to a home agent rather than blindly broadcasts to vehicles. The agent could distribute the report to appropriate gateways, which can answer the validity of the queried data item to reduce the unavoidable query delay. Similarly, the gateways in different regions also cooperated to maintain the invalidate contents so as to reduce the query delay for urban vehicles in [26]. A central-controlled caching scheme was proposed for VANET [27], where each RSU maintained the cache list of all vehicles in its domain and helped to forward requests. A centralized approach was presented in which the base station computed an estimate of the popularity profile based on the requests observed during the time interval and this estimate was used to optimize the caching probability for the terminals in its region.

None of the above related works can achieve a balance for content duplication and cache availability. Any caching node mobility may disrupt the delicate caching structure of the entire system. Instead, we propose the use of consistent hashing in order to ensure a strong balance between content

duplication and cache availability, as well as lower traffic overhead even when caching nodes move around.

Consistent Hashing is a distributed hashing scheme that operates independently of the number of servers or objects in a distributed hash table. It was first proposed in [29] for web caching to map items to servers. Because it can achieve load balance and fault tolerance, most solutions based on consistent hashing focus on file allocation in distributed storage systems [30][31][32][33]. To reduce time delay of processing data and improve the efficiency of cloud computing, a clustering algorithm based on the principle of minimum distance was proposed to store user-based and item-based data [34]. In [35], routers are grouped into multiple autonomous systems and then consistent hashing is applied in each system.

III. Cooperative Caching based on Mobility Prediction and Consistent Hashing (CCMPCH)

In this section, an overview of our algorithm is provided first. Then, our CCMPCH scheme will be introduced in detail including cluster division in all the vehicles, content allocation among the cooperative nodes, and design of cache replacement strategy.

A. Overview

Our VCCN model in Figure. 1 consists of connected vehicles and RSUs. Each vehicle can communicate with its neighbors via vehicle-to-vehicle communication or connect to an RSU via vehicle-to-infrastructure communication within its wireless communication range. RSUs are connected to each other via wired communication and are preloaded with different contents. A pull model is adopted to disseminate contents in VCCN. A vehicle first sends an Interest containing the required content name and broadcasts it to its neighbors. The Interest can be stored, carried and forwarded by vehicles or RSUs for at most time-to-live (TTL) seconds until the interested content is reached. Each RSU also acts as a gateway, responsible of forwarding Interests to other RSUs if the corresponding contents are unavailable in the local region.

Each vehicle is assumed to maintain three data structures described in Table I, Content Store (CS) to cache popular contents, Pending Interest Table (PIT) to keep track of forwarded Interests, and Vehicle Information Table (VIT) to record its neighbor information. Each RSU only maintains CS and PIT. When a node receives an Interest, it will first search the requested content name in its CS and reply with the Data packet if there is a hit. Otherwise, the vehicle will record the requested content name in PIT and update the corresponding counter for popularity calculation. When receiving a Data chunk, each node will decide whether to cache it based on its own replacement policy.

The CCMPCH scheme consists of three modules: cluster generation, content allocation, and cache decision, as illustrated in Figure. 2. In the cluster generation, RSU is responsible for grouping vehicles with the most common attributes including moving direction, moving speed and future trajectory into the same cluster and choosing core nodes to perform cooperative caching. In the content allocation, we apply consistent hashing to distribute contents within the core nodes. In the cache decision, we design the popularity-based cache replacement policy to make cache decision.

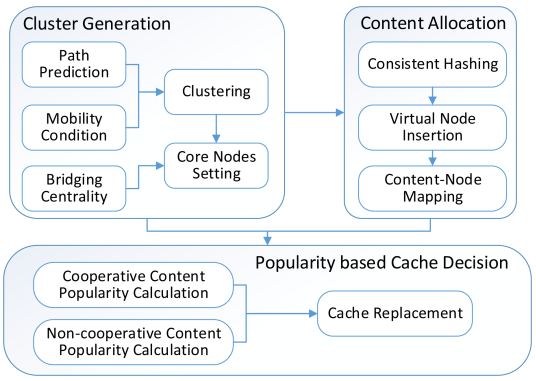


Figure. 2 Architecture of our CCMPCH Scheme

B. Cluster Management

Working with mobile nodes, our clustering algorithm must be able to deal with cluster dynamics caused by joining/leaving vehicles, while maintaining link stability of cooperative caching nodes. First, we list the frequently used notations in Table II.

TABLE I: Three Data Structures

(a) Content Store

Notation	Description
c_i	The i-th cached content
N_i	Content name of c_i
ξ_i	Popularity value of c_i

(b) Pending Interest Table

Notation	Description
I_i	Counter of requests for c_i in the current time slot

(c) Vehicle Information Table

Notation	Description
$List_n$	A list recording its neighbors
$List_{clu}$	A list recording its cluster members
$List_{core}$	A list recording core nodes of its cluster

TABLE II: Notations

Notation	**Description**
v_k	The k-th vehicle
l_m	The m-th grid in location
L	Trip sequence of a vehicle.
g_n	The number of core nodes in a cluster
R_v	Vehicle communication range
R_R	RSU communication range
μ	Duration of each time slice
r	Rank of content popularity

1) Cluster Formation: Vehicles are divided into clusters based on their future trajectory, moving speed, and moving direction. In order to model vehicular mobility patterns and obtain accurate predictions, we adopt discrete geographical regions instead of infinite continuous locations [36][37]. As shown in Figure. 3, a road network can be divided into multiple grids. Coordinates of each vehicle can be discretized and converted into the corresponding grids. Each grid may cover several roads or even no road depending on the actual city map. A trip sequence is formed with a set of grids to record the trajectory of a vehicle, $L = \{l_1, l_2, ..., l_n, ...\}$, where l_n refers to the n-th grid.

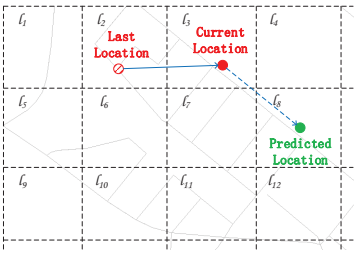


Figure. 3 Grid Division and Mobility Prediction

First, we use Prediction based on Partial Matching (PPM) [38] to make the mobility prediction.

Step 1: Based on a vehicle's trip sequence, a trie is constructed in Figure. 4, where each node except the root includes two symbols, the n-th grid l_n and a counter representing the number of times that l_n appears in all subsequences. Each element l_n and its context form a trip path. If the path has not appeared before, it will be added into the trie. Otherwise, all the counters along the path will be incremented. For the given sequence $\{l_5, l_2, l_3, l_4, l_2, l_3, l_8, l_2, l_3, l_8\}$, l_3 occurs two times as the leading element with $\{l_3, l_4, l_2\}$, and $\{l_3, l_8, l_2\}$ which will be added into the trie. Obviously, there are two sub-branches on the third branch of the tree. Similarly, there is only one branch of $\{l_2, l_3, l_8\}$ in this tree, though $\{l_2, l_3, l_8\}$ occurs two times in the trip sequence.

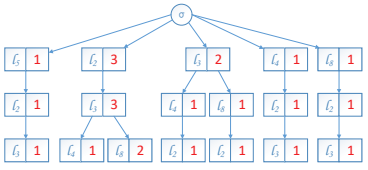


Figure. 4 Trie Constructed by Trip Sequence

Step 2: Given the grid l_{m-1} in the last slice and l_m in the current slice, we predict l_{m+1} where the vehicle is the most likely to go in the next slice if $\{l_{m-1}, l_m\}$ can be parsed in the trie. For instance, given $\{l_{m-1}, l_m\} = \{l_2, l_3\}$, the probabilities of l_4 and l_8 being l_{m+1} are 1/3 and 2/3 respectively according to the second branch in Figure. 4. Therefore, l_8 is the predicted location, which is shown as an example in Figure. 3.

Step 3: The grid with the highest probability is selected as the predicted grid whose center coordinate is set to be the predicted coordinate $\hat{\boldsymbol{v}}_k$ for the vehicle v_k.

Then, we determine the cluster head as follows:

Step 1: Within its communication range, each RSU chooses the vehicle with the highest bridging centrality as the initial cluster head. *Bridging Centrality* is defined as a measure of the extent to which a node controls over the information flowing among neighboring nodes. For a vehicle v_k,

$$b_k = d_k \cdot \frac{|\mathcal{S}_k|}{|\mathcal{S}|}, \tag{1}$$

where d_k is the degree of v_k, defined as the number of neighbors that are connected to itself, $\mathcal{S}$ is the set of all possible paths among all neighbors, and $\mathcal{S}_k$ is the set of paths in $\mathcal{S}$ that go through v_k (therefore, $\mathcal{S}_k \subseteq \mathcal{S}$). The $\frac{|\mathcal{S}_k|}{|\mathcal{S}|}$ term in Eq. (1) is the so-called *Betweenness Centrality* for each node. To reduce the complexity of centrality calculation, we only take one-hop neighbors into consideration.

Step 2: If multiple nodes possess the same bridging centrality, RSU will calculate the average physical distance from each node to all its neighbors. The smaller the average distance, the more compact the node with its neighbors. The node with the smallest average distance will be chosen as the cluster head.

Each RSU assigns each vehicle in its coverage into the nearest cluster head while ensuring that the physical distance between each node and its cluster head is within the communication range before and after a time slot. If a vehicle lies in an overlapping area of multiple RSUs, it will choose the nearest RSU to be its unique RSU.

To maintain stability between cached contents and caching nodes, each RSU will choose half of all nodes in each cluster to serve as cooperative caching nodes or core nodes in the descending order of bridging centrality, because a node with higher bridging centrality can keep relatively stable relationship with other nodes.

2) Cluster Adjustment: Due to the vehicle mobility, it often occurs that a vehicle leaves its old cluster and joins a new cluster. When a node finds itself disconnected from all nodes in its original cluster, it will declare itself an orphan node and request the nearest RSU to assign itself to a new cluster. As soon as receiving such request, the RSU will assign it into the nearest cluster whose cluster head satisfies the mobility similarity requirement. When a cluster head departs, the corresponding RSU will select a core node with the highest bridging centrality to act as the new head. The leaving of core nodes will be processed in a similar fashion. As more vehicles join a cluster, the original number of core nodes may be less than 1/4 of the total member, which means the number of nodes is about twice as the original number. In this case, RSU will split the members into two clusters.

C. Content Allocation by Consistent Hashing

In order to avoid cache duplication among the adjacent nodes, we adopt consistent hashing on core nodes in each

cluster to allocate different contents[1]. Though the basic hash function has been widely used for this purpose, it is not consistent and creates unbalanced loads [35]. In basic hash, nodes in a group are used as buckets and modulo of the hash value is taken with respect to the total number of nodes. Each node stores the received Data packets based on its given label from the output value. One member's leaving or joining the group can abruptly change these modulus values. In this case, modulo hashing becomes inconsistent and most of nodes must be re-assigned new labels and cache updated. Consequently, this leads to high traffic overhead as new contents are transferred between nodes. Compared with the basic hash function, consistent hashing can achieve load balancing and avoid the modulo operation for all members in case of a member joining/leaving the cluster [35].

Suppose integers from 0 to $2^{32}-1$ form a hash ring. In consistent hash, each core node is mapped into a unique integer which can identify the node's position on the ring. Similarly, the global content names are mapped to integers so that each content has a unique position on the hash ring. In fact, the contents are allocated to the closest core node following clockwise direction along the hash ring, forming a many-to-many mapping between contents and nodes. Further, a certain number of virtual nodes will be inserted on the hash ring to achieve cache balance [39].

We take Figure. 5 as a simple example to illustrate the content allocate process. There are 6 nodes including three physical nodes v_1, v_2 and v_3 and their corresponding virtual nodes v_1', v_2' and v_3'. 6 contents from c_1 to c_6 are also mapped on the hash ring. As introduced before, c_1 should be allocated to the closest node v_3' in clockwise direction. Because v_3' is a virtual node, c_1 should be allocated to the corresponding real node v_3. When v_3 leaves the cluster, RSU will determine an actual node along the ring until c_1 is assigned. When the Data packet of c_1 arrives at the designated node, this node will decide whether to cache it based on its cache replacement policy. However, the cached contents in the other members still remain the same.

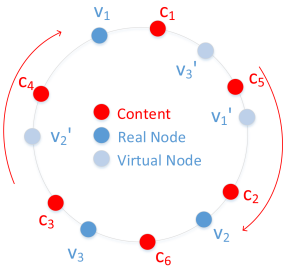


Figure. 5 Consistent Hashing with Virtual Nodes

By following the above steps, each core node as a cooperative member will be pre-allocated some contents. Generally, we divide the cached contents into two categories, cooperative contents and non-cooperative contents. The contents mapped to a node from the computation of consistent hashing are called cooperative ones, while the other contents are non-cooperative. For example, c_4 and c_5 are cooperative contents for v_1. Similar to CCN, VCCN also suits pull-based applications where a given content is transmitted on demand in reply to an Interest, even though the content might be available from multiple providers [1]. To improve the storage utility, we should try to cache more cooperative contents under the same measurement metric such as the total number of requests in one time slot. In the next section, we will introduce a preference constant to give the cooperative contents higher priority in order to make them remain longer in the buffer.

D. Popularity-based Cache Replacement

We use content popularity as an effective measure for making caching decision. In this section, we present our popularity-based cache replacement algorithm in detail.

We define an interest Counter I_i for recording the requests on c_i. As a new Interest on c_i is received, both I_i and a total interest counter A will be incremented. Each node calculates the content popularity ξ_i of c_i as follows,

$$\xi_i \Leftarrow (1+\beta\cdot\mathbf{1}(c_i))\left[\lambda\frac{I_i}{A^{(t)}}+(1-\lambda)\xi_i\right], \qquad (2)$$

where $\mathbf{1}(\mathbf{c_i})$ returns 1 if c_i is a cooperative content and 0 otherwise, so that preference constant $\beta > 0$ could be added into multiplier to increase the weight of cooperative contents. $A^{(t)}$ is the total number of interests arrived in time slot t, t is the current time slot. And we further use the rate between the request number of c_i and the total number of all requests to suggest the frequency of requests on c_i. λ is time average constant to suggest weight decrease of former popularity. Eq. (2) is used to compute the comprehensive content popularity whenever it is necessary to perform cache replacement. The content popularity values are sorted in descending order and whether to cache a content is dependent on its popularity value. When the buffer is full, those contents with lower popularity values will be replaced.

When receiving a request, if the desired content is already cached, the content will be directly returned. Else if the request is over TTL, it will be discarded. Else if the request is within its TTL, it will be forwarded to the neighboring vehicles. For occasion of receiving a content, if the node buffer is not full, the content will be directly cached. Otherwise the node should first find the mapping node of the content according to consistent hashing. If the content is a cooperative content for the node, we add preference constant to improve content popularity. Then the cached content with the lowest popularity will be replaced, or the new coming content will not be cached when it possesses the lowest popularity. Following the above steps, each node tends to cache more cooperative contents so as to reduce cache redundancy between nodes and avoid the overhead of transferring contents between nodes.

[1]When a node receives a content, the content will only be cached on one node in the cluster, not on several different nodes. Only the number of packets delivered is increased in the process of caching a content, and it does not occupy more cache space.

IV. Analysis

We optimize the preference constant β in this section. Our optimization is to achieve the lowest average hop count for successful retrievals.

Generally, there are three ways to retrieve data for any node: local buffer (L), core nodes in its cluster (C), or non-core nodes (N). The hop count for retrieving from local buffer is zero. The hop count for retrieving from its cluster is one, while retrieving from non-core nodes can take multiple hops. The average hop $E[h]$ to get any content is calculated as follows,

$$E[h] = \Pr(C) + \Pr(N)h(N), \tag{3}$$

where $h(N)$ is the expected number of hops when data is retrieved from non-core nodes, and $\Pr(C)$ and $\Pr(N)$ represent the probability that the content is retrieved through C and N, respectively.

In order to make our analysis more tractable, we assume that content requests follow the Zipf's Law distribution [40],

$$f(r;\alpha,\theta_c) = \frac{1/r^{\alpha}}{\sum_{n=1}^{\theta_c}(1/n^{\alpha})}\ , \tag{4}$$

where θ_c is the number of contents, $r \in \{1, 2, \cdots, \theta_c\}$ is the rank of a content, and α is the exponent characterizing the distribution. We define two ranking thresholds r_1 and r_2 in the following way. Each caching node only caches cooperative contents ranking higher than r_1 and non-cooperative contents ranking higher than r_2. These two thresholds are defined such that the comprehensive popularity of the least popular cooperative content and non-cooperative content should be the same,

$$\bar{\xi} = \xi_2 = (1+\beta)\xi_1, \tag{5}$$

where ξ_1 is the popularity value of the least popular cooperative content and ξ_2 is that of the least popular non-cooperative content. After some simple rearrangement of Eq. (5), we have the cooperative to non-cooperative content ratio r_1/r_2 in any core node,

$$\frac{r_1}{r_2} = (1+\beta)^{1/\alpha}. \tag{6}$$

If ω is denoted as each caching node's capacity (i.e. the number of contents each caching node can cache), there are roughly $\frac{r_1}{r_1+r_2}\omega$ cooperative contents and $\frac{r_2}{r_1+r_2}\omega$ non-cooperative contents in each core node. We assume that cooperative contents have no duplicates, while non-cooperative contents may duplicate among different nodes. For a cluster including g_n core nodes, the total number of distinctive contents in each cluster can be estimated as

$$\omega g_n \left(\frac{r_1}{r_1+r_2} + \frac{r_2}{g_n(r_1+r_2)}\right) = \omega\left(\frac{g_n(1+\beta)^{1/\alpha}+1}{(1+\beta)^{1/\alpha}+1}\right). \tag{7}$$

Besides, a node can get the content from its reachable RSUs within one hop. We further assume there are N_R RSUs in a rectangle region of size $X * Y$ and these RSUs are randomly placed. Actually, they should be placed strategically, but we will use this approximation to make our analysis more straightforward and further evaluate our analysis in Section V. Each RSU covers a region of πR_R^2, where R_R is the transmission range of each RSU and the total coverage area of these RSUs is $\pi R_R^2 \cdot N_R$. Then, the expected number of RSUs each vehicle can reach from any random location is about $\frac{\pi R_R^2 \cdot N_R}{XY}$.

As a result, we can get the total capacity of caching nodes (core nodes and RSUs) that are within one-hop of a vehicle,

$$\varsigma = \left\lfloor \left(\frac{g_n(1+\beta)^{1/\alpha}+1}{(1+\beta)^{1/\alpha}+1}\right)\omega + \Omega\frac{\pi R_R^2 \cdot N_R}{XY} \right\rfloor, \tag{8}$$

where $\lfloor . \rfloor$ is simply the floor function and Ω is the caching capacity of each RSU. ς represents the total number of unique contents that have been cached and can be retrievable.

$\Pr(C)$ is estimated as

$$\Pr(C) = \sum_{r=1}^{\varsigma} f(r;\alpha,\theta_c). \tag{9}$$

Note that

$$\Pr(N) = 1 - \Pr(C) - \Pr(L), \tag{10}$$

where $\Pr(L)$ is the probability that the content is retrieved directly from the local cache. Since the local cache has a capacity of ω, the composition of cache is,

$$r_1 \cdot \frac{1}{g_n} + r_2 \cdot \frac{g_n - 1}{g_n} = \omega. \tag{11}$$

Combining with Eq. (6), we can deduce

$$r_1 = \frac{(1+\beta)^{1/\alpha} g_n \omega}{g_n - 1 + (1+\beta)^{1/\alpha}} \tag{12}$$

and

$$r_2 = \frac{r_1}{(1+\beta)^{1/\alpha}} = \frac{g_n\omega}{g_n - 1 + (1+\beta)^{1/\alpha}}. \tag{13}$$

$\Pr(L)$ can be calculated as

$$\Pr(L) = \sum_{r=1}^{r_2} f(r;\alpha,\theta_c) + \frac{1}{g_n}\sum_{r=r_2}^{r_1} f(r;\alpha,\theta_c). \tag{14}$$

To get $E[h]$ in Eq. (3), we need an estimation of $h(N)$. Assuming that N_v vehicles are randomly distributed in the area of $X*Y$. On an average, each RSU is in range of $\frac{\pi R_R^2 \cdot N_v}{XY}$ vehicles within its communication range. As discussed before, the proportion of core nodes is 1/2 in each cluster. We then estimate the total capacity within the communication range of each RSU as

$$\frac{\pi R_R^2 \cdot N_v}{2XY} \cdot \left(\frac{(1+\beta)^{1/\alpha}+1/g_n}{(1+\beta)^{1/\alpha}+1}\right)\omega + \Omega. \tag{15}$$

Assuming a perfect zero-duplication in content caching among RSUs, a request may need to be transmitted through the following number of RSUs in order to find the corresponding content:

$$\frac{\theta_c}{\frac{\pi R_R^2 \cdot N_v}{2XY} \cdot \left(\frac{(1+\beta)^{1/\alpha}+1/g_n}{(1+\beta)^{1/\alpha}+1}\right)\omega + \Omega}. \tag{16}$$

Under normal operational settings, the number above is usually rather small. Actually, most requests do not need to

go through the full number of hops to find the corresponding content, so we can calculate the mathematical expectation as the expected number of hops. And we can get the average hop $h(N)$:

$$h(N) = \frac{1}{2} \cdot \frac{\theta_c}{\frac{\pi R_R^2 \cdot N_v}{2XY} \cdot \left(\frac{(1+\beta)^{1/\alpha}+1/g_n}{(1+\beta)^{1/\alpha}+1} \right) \omega + \Omega}. \quad (17)$$

We further use numerical calculations to identify optimum β value in Section V-B.

V. Performance Evaluation

In this section, we evaluate the performance of CCMPCH via the Opportunistic Network Environment (ONE) simulator [41]. ONE supports several mobility models, from simple Random Waypoint to more realistic Map-Based Movement. We adopt movement models in ONE from the real map of Helsinki area [42]. In the designed scenario, there are 30 RSUs and 270 users (including 100 private cars, 100 buses, and 70 taxis) in Figure. 6. Private cars move following the Working-Day-Movement Model including staying at home, working in the office, and going shopping after work. Buses run through fixed routes and taxis follow the Random Waypoint Model in ONE. Because these models have been constructed by actual traffic database, so we ignored the road settings such as the number of lanes and signal setting at intersections. Furthermore, we adopt 801.11p for PHY and MAC protocols and spray and wait protocol [43] for routing, which is widely used in intermittently connected mobile networks. In spray and wait, real timer instead of hop count is used to measure how long packets have been traveling in the network and when they should be discarded. This eliminates the performance issue inherent in utility-based schemes caused by long wait for the next hop.

All nodes are assumed to move with a speed randomly chosen from a fixed range, the same wireless transmission/reception range, and transmission data rate. During warm-up (training), we collect trip sequences of vehicles in working days for mobility prediction training. In the evaluation stage, every user generates content requests following the Zipf's law distribution [40] as described in Section IV. Each vehicle updates its physical coordinates every 100 seconds and then uploads them to the nearest RSU for second order prediction in its beacon frame. Performance statistics are collected after 8,000 seconds of operation. Important simulation parameters are listed in Table III, unless specified otherwise.

A. Performance Metrics

We compare our CCMPCH scheme with DPC [10], COMP [11], CCMP [13], and CCSAMP [14], all of which are cooperative schemes. In DPC, each vehicle independently makes its caching decision taking three factors into account: received Interests, its centrality in the network, and its moving direction. In COMP, mobility prediction is used to cluster vehicles using a Markov model and a cluster head is responsible for content distribution. In CCMP, hot regions and trajectory of vehicles

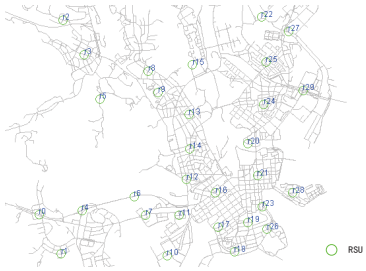


Figure. 6 Simulation Map and Location of RSUs

TABLE III: Simulation Parameters

Parameter Description	Value
Buffer Size of Each Node	50 MB
Each Content Size	1 MB
Request Interval	[450 s, 550 s]
Message TTL	1 minute
Network Area [X*Y]	4 km×3 km
Simulation Time	12 hours
RSU Number N_R	30
RSU Transmission Range R_R	500 m
Vehicle Number N_v	270
Vehicle Speed	[15m/s, 20 m/s]
Vehicle Transmission Range R_v	100 m
Transmission Rate	10 Mbps
Zipf's Distribution Exponent a	0.8
Number of Contents	$2,000$

are used to select cache nodes. In CCSAMP, social attributes and vehicle's future trajectory are used to pick caching nodes.

The following performance metrics are used to compare these schemes:

- ***Cache Hit Ratio***: The probability of obtaining a cache hit from a caching node. This is defined as the ratio of the number of cache hits to the total number of Interests received.
- ***Average Access Delay*:** The average delay of obtaining the requested contents among all successful queries.
- ***Average Hop Count*:** The average hop count between each pair of content requester and provider in successful queries.
- ***Average Delivery Ratio*:** The average ratio of packets delivered successfully before they are removed from the network due to an expired TTL. This is necessary to gauge how efficient a routing scheme is.
- ***Communcation Overhead*:** The ratio between the number of Interest packets plus the number of the generated packets for saving the cooperative contents and the number of the corresponding Data packets.
- ***Content Diversity*:** The ratio between the number of different contents cached and the number of total contents cached within a cluster.
- ***Prediction Accuracy*:** The probability of successful predictions, which is defined as the ratio between the number of predictions that turn out correct and the total number of predictions.

B. Effect of β

The preference constant β is an adjustable parameter for popularity calculation of cooperative contents. We compare expected hop count under different β values in this subsection. Numerical calculations are based on Eq. (3). When β takes a small value, caching nodes suffer from heavy cache redundancy and Interest packets must go through more hops to find a hit, resulting in a high hop count, as shown in Figure. 7. On the other hand, large β values would replace non-cooperative contents of high popularity values with less popular cooperative contents (even though cache redundancy is lower) and this also ends up with higher hop counts. Both numerical and simulation results show similar trends as β changes.

Our numerical analysis shows that the hop count reaches the minimum as $\beta = 2.483$, matching well with the trend in our simulation results. The discrepancy of about 0.15 in hop counts between numerical results and simulation results should have been caused by the several simplifications that we made in Section IV. Consequently, we adopt the optimal $\beta = 2.483$ in further simulations.

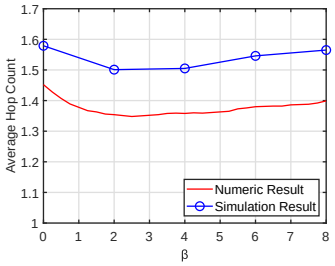


Figure. 7 Effect of β

C. Effect of α

Content requests follow the Zipf's Law distribution, where α is the exponent characterizing the distribution. Fig. 8 shows the cache hit ratio increases with α, because the content distribution becomes dense with α.

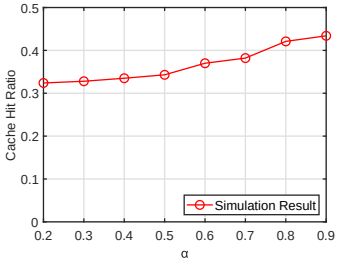


Figure. 8 Effect of α

D. Effect of Content Size

We change the content size from 1MB to 5MB and evaluate its effect on cache hit ratio. As the content size increases, each node caches fewer contents and this leads to lower cache hit ratio in Figure. 9. Our CCMPCH scheme shows the best performance, improving the cache hit ratio by 4% compared to CCSAMP, 5% over CCMP, 12% over COMP and 14% over DPC, because CCMPCH can adaptively distribute contents more effectively among different nodes, in turn making full use of cache space. In CCSAMP and CCMP, only selected nodes are allowed to cache contents, constraining its performance. In COMP, contents are simply classified into most popular and less popular type with fixed caching strategies. DPC only considers the cooperation between providers and receivers, but lacks of cooperation within neighboring nodes.

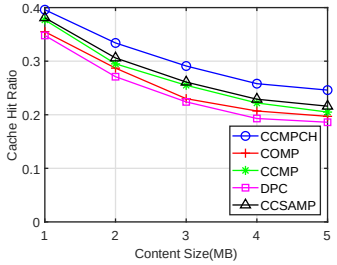


Figure. 9 Effect of Content Size

E. Effect of Cache Buffer Size

In Figure. 10, the cache buffer size varies from 50MB to 250MB. The cache hit ratio increases with the cache buffer size, because bigger buffer can cache more contents. As can be observed in this figure, our CCMPCH scheme consistently outperforms other schemes.

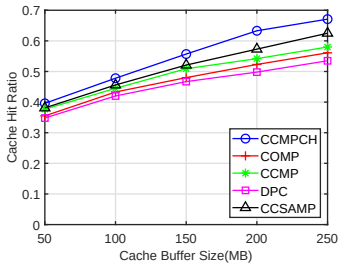


Figure. 10 Effect of Cache Buffer Size

F. Effect of Maximum Vehicle Speed

To evaluate the effect of vehicle mobility, we set the maximum vehicle speed as $10m/s$, $20m/s$ and $30m/s$, respectively. Figure. 11 demonstrates that the cache hit ratio declines with the speed in all five schemes, as higher speed leads to shorter link duration and lower prediction accuracy. Nonetheless, CCMPCH still outperforms the other four references since it takes the moving trajectory, speed, and direction into consideration. When maximum vehicle speed is high ($30m/s$), CCMPCH enjoys a cache hit ratio that is about 10% better over CCSAMP, 14% over CCMP, 16% over COMP, and 44% over DPC.

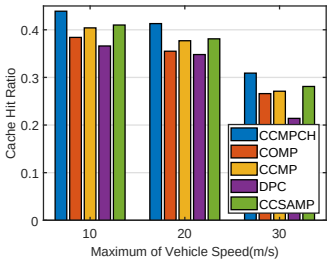


Figure. 11 Effect of Vehicle Speed

G. *Effect of Vehicle Transmission Range*

In Figure. 12, we compare the effect of vehicle transmission range on cache hit ratio. Cache hit ratios for all schemes increase with the transmission range. A larger transmission range covers more neighbors, making it easier to retrieve the cooperative cache contents. As can be seen from this graph, our CCMPCH scheme consistently outperforms other schemes.

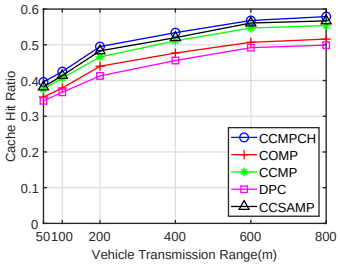


Figure. 12 Effect of Transmission Range

H. *Effect of RSU Number*

In Figure. 13, we compare the effect of RSU number on cache hit ratio. More RSUs possess more contents, increasing the cache hit ratio in all schemes. Our CCMPCH still maintains the highest cache hit ratio as more contents can be accessed inside the cluster.

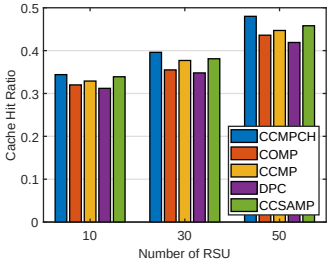


Figure. 13 Effect of RSU Number

I. *Effect of Grid Size*

The cache hit ratio and prediction accuracy of CCMPCH are evaluated as the grid size varies from 50m to $1,000$m in Figure. 14 (right label). The total number of grids in the whole map decreases as the size increases, naturally resulting in higher prediction accuracy.

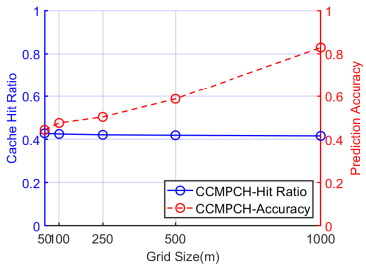


Figure. 14 Effect of Grid Size

J. *Effect of Path Prediction*

In this set of simulations, we measure the impact of path prediction on the cache hit ratio from the grid size and vehicle speed. In Figure. 14 (left label), cache hit ratio remains the same as grid size changes. This is mostly because of the constant wireless communication range.

In Figure. 15, cache hit ratio decreases as speed increases. The rapid movement of vehicles causes the instability among clusters, leading to a lower hit ratio.

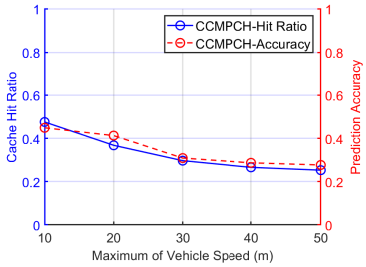


Figure. 15 Effect of Moving Speed

K. *Comparison of Cache Redundancy*

In order to evaluate cache redundancy, we collect the number of distinct contents. First we made the collection at simulation time $8,000s$, which is the end of the warmup phase. This number represents the total number of distinct contents among all contents that have been cached by all nodes. Obviously, a large number indicates lower cache redundancy. On Table. IV, our CCMPCH scheme's number of distinct content copies is 15% higher than COMP, 28% higher than CCMP, 20% higher than DPC, and 23% higher than CCSAMP. This is because CCMPCH takes advantage of cooperative caching inside the cluster and makes use of the whole cache space with high efficiency.

L. *Other Performance Comparisons*

We evaluate the performance of routing protocols adopted by these five schemes on protocols' average access delay, average hop count, average delivery ratio, communication overhead, and content diversity. To evaluate each metric, we

TABLE IV: Number of Distinct Contents

Scheme	Number
CCMPCH	743
COMP	648
CCMP	589
DPC	617
CCSAMP	623

vary the content size from 1MB to 5MB, set the maximum vehicle speed to be 10 m/s, 20 m/s, and 30 m/s respectively, and set the RSU number to be 10, 30, and 50.

Average Access Delay. As the content size increases, the transmission of each individual content takes longer with constant bandwidth. Therefore, the access delay shows an upward trend in Figure. 16(a). As vehicle speed increases, the link duration between vehicles decreases, which increases the difficulty of delivering packets. Therefore, the access delay increases as vehicles move faster in Figure. 16(b). More RSUs can share more data helping to reduce access delay in Figure. 16(c). Our CCMPCH has the best performance because has taken into account the moving trajectory, speed, and direction to determine the cooperative caching nodes.

Average Hop Count. As the content size increases, each node caches fewer contents, causing higher hop count in Figure. 17(a). As vehicles move faster, link instability also causes higher hop count in Figure. 17(b). More RSUs can cache more contents and vehicles can obtain more contents from the nearby RSUs, thus reducing the average hop count in Figure. 17(c). Our CCMPCH still has the best performance.

Average Delivery Ratio. Compared to average access delay and average hop count, the average delivery ratio tends to change more slowly with the content size in Figure. 18(a), which shows the vehicle can still successfully get contents. In Figure. 18(b), the delivery ratio decreases with vehicle speed, because the increasing speed makes it more difficult to deliver packets successfully. In Figure. 18(c), more RSUs cache more contents, increasing the average delivery ratio.

Communication Overhead. As the content size increases, each node caches fewer contents. Therefore, it becomes more difficult to obtain the requested contents, increasing the communication overhead shown in Figure. 19(a). In Figure. 19(b), as vehicles move faster, the links between vehicles are more likely to disrupt, increasing the communication overhead too. In Figure. 19(c), as more RSUs cache more contents, the communication overhead becomes lower.

Content Diversity. Figure. 20(a) shows the content diversity decreases with the content size less than 3MB. Though the number of total contents cached becomes smaller as the size increases more than 3MB, the content diversity becomes larger. As a vehicle moves faster, it can meet more neighbors and make the forwarded Interest and Data packets possess more types of contents, making the content diversity increase in Figure. 20(b). Similarly, more RSUs can share more diverse contents, leading to larger content diversity in Figure. 20(c).

M. Algorithm Complexity Evaluation

In this section, we first analyze the algorithm complexity of these five schemes and evaluate them through experiments.

DPC determines the probability of a node caching contents by considering the demands of users, the importance of nodes, and the relative movement of vehicles. The requested file names are divided into different categories first, which incurs the complexity of $O(n^2)$. Node importance is measured as centrality, which also has the complexity of $O(n^2)$. DPC calculates the distance between the provider and receiver by combining the vehicle moving direction. The complexity of this process is $O(1)$. So the overall complexity of DPC is $O(n^2)$.

COMP first predicts the future mobility pattern and link expiration time with neighbors for each node. This process incurs the complexity of $O(n)$. Then, vehicles are clustered based on the prediction results and cluster heads are chosen as caching nodes, causing the complexity of $O(n^3)$. The overall complexity of COMP is $O(n^3)$.

CCMP selects regions with a high density of mobile trajectories as hot regions. The complexity of this process is $O(n^2)$. By predicting each node's probability of reaching hot regions, vehicles with longer sojourn time in the hot regions are chosen as caching nodes. This process incurs the complexity of $O(n^3)$. So the overall complexity of CCMP is $O(n^3)$.

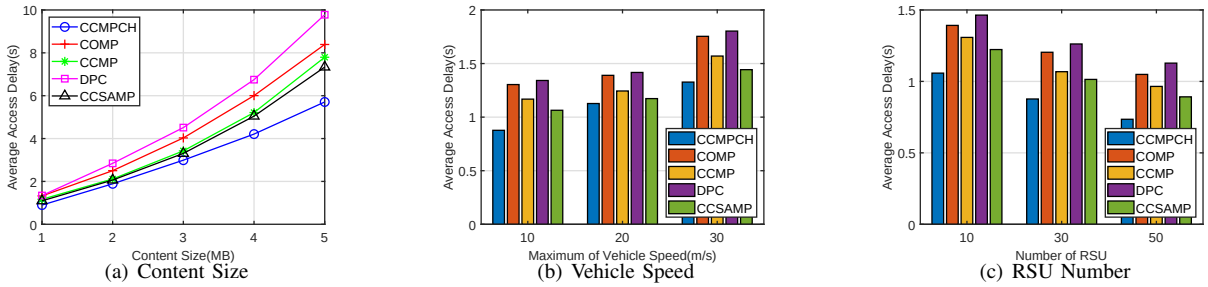
CCSAMP selects the appropriate caching nodes based on social similarity, bridging centrality, and future trajectory. The calculation of social similarity and bridging centrality incurs complexities of $O(n)$ and $O(n^2)$ respectively. CCSAMP predicts the trajectory based on its past trajectory with the hidden markov model, which incurs the complexity of $O(n^2)$. So the overall complexity of CCSAMP is $O(n^2)$.

In our CCMPCH, RSU is responsible for grouping vehicles with the most common attributes into the same clusters. The complexity of this process is $O(n^2)$. Consistent hashing is adopted to allocate different contents in each cluster and the complexity is $O(n)$. Moreover, CCMPCH calculates the content popularity as its cache replacement, which brings the complexity of $O(1)$. In total, the complexity of our CCMPCH is $O(n^2)$.

These are shown numerically below (processing time is measured by the time consumption of handling 10,000 packets.) These results were obtained in a system with a single CPU (Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz) and 8 GB RAM. As shown on Table. V, our CCMPCH scheme has the least processing time, 0.243 seconds. Our CCMPCH scheme also has the lowest CPU usage rate and memory occupancy, 23.2% and 369 MB memory, respectively, while the COMP scheme has the worst performance in Table. V.

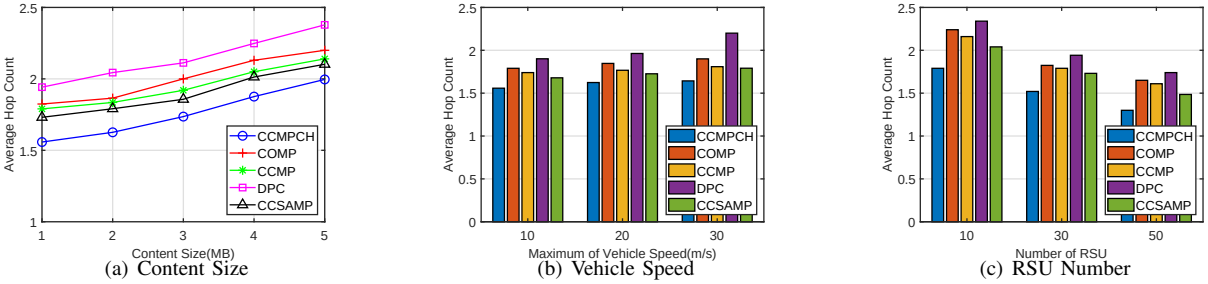
VI. Conclusion

In this paper, we have proposed a novel cooperative caching scheme based on trajectory prediction and consistent hashing. Vehicles are first clustered based on their similar future trajectory and nodes with higher bridging centrality are chosen



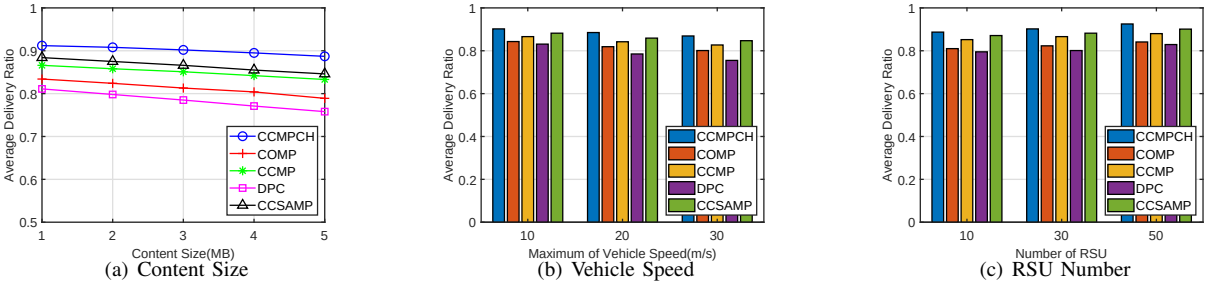
(a) Content Size (b) Vehicle Speed (c) RSU Number

Figure. 16 Comparison of Average Access Delay



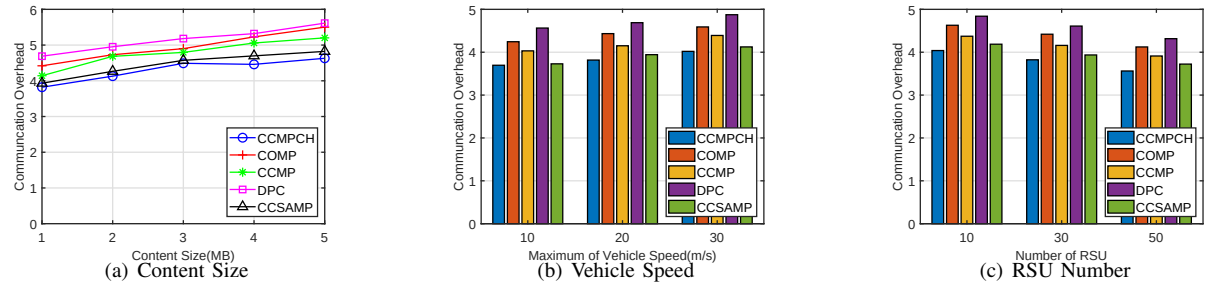
(a) Content Size (b) Vehicle Speed (c) RSU Number

Figure. 17 Comparison of Average Hop Count



(a) Content Size (b) Vehicle Speed (c) RSU Number

Figure. 18 Comparison of Average Delivery Ratio



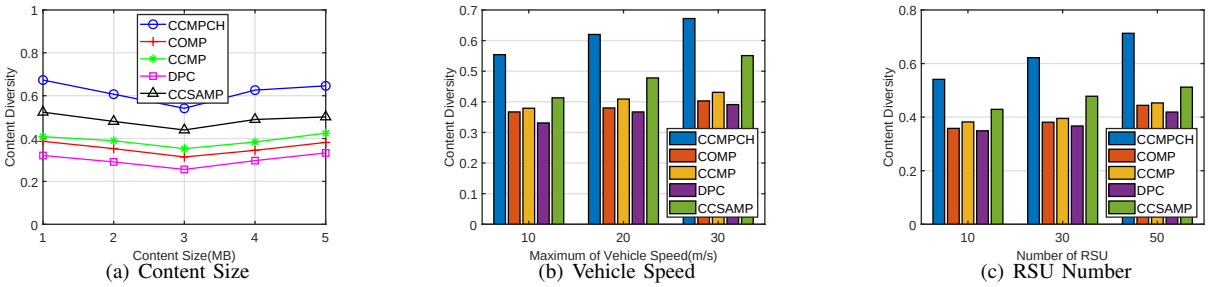
(a) Content Size (b) Vehicle Speed (c) RSU Number

Figure. 19 Comparison of Communication Overhead

to function as cooperative ones. Then, consistent hashing is adopted to allocate cooperative contents. Besides, a popularity-based cache replacement is proposed to give priority to cooperative contents. Extensive performance evaluations have proven its better performance. In the future, we plan to extend our algorithm to achieve energy-efficient caching for VCCN.

ACKNOWLEDGMENT

This research is sponsored in part by the National Natural Science Foundation of China (61872053), the Key Basic Research project of Basic Strengthening Program (2020-JCJQ-ZD-126-11), and the Key-Area Research and Development Program of Guangdong Province (2019B010136001,



(a) Content Size (b) Vehicle Speed (c) RSU Number

Figure. 20 Comparison of Content Diversity

TABLE V: Complexity Evaluation

Scheme	Complexity	Processing Time	CPU Usage	Memory Usage
CCMPCH	$O(n^2)$	0.243s	23.2%	358.37 MB
COMP	$O(n^3)$	0.505s	41.7%	512.35 MB
CCMP	$O(n^3)$	0.491s	37.0%	501.62 MB
DPC	$O(n^2)$	0.325s	24.6%	375.922 MB
CCSAMP	$O(n^2)$	0.272s	23.9%	369.21 MB

2020B0101360001).

REFERENCES

[1] S. H. Bouk, S. H. Ahmed, and D. Kim, "Vehicular content centric network (vccn): A survey and research challenges," in *Acm Symposium on Applied Computing*. ACM, 2015, pp. 695–700.

[2] X. Liu, Z. Li, P. Yang, and Y. Dong, "Information-centric mobile ad hoc networks and content routing: a survey," *Ad Hoc Networks*, vol. 58, pp. 255–268, 2017.

[3] M. F. Majeed, S. H. Ahmed, and M. N. Dailey, "Enabling push-based critical data forwarding in vehicular named data networks," *IEEE Communications Letters*, vol. 21, no. 4, pp. 873–876, 2016.

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

[5] D. Rossi and G. Rossini, "On sizing ccn content stores by exploiting topological information," in *IEEE INFOCOM Workshops*. IEEE, 2012, pp. 280–285.

[6] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, pp. 286–299, 2013.

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

[8] I. Psaras, W. K. Chai, and G. Pavlou, "In-network cache management and resource allocation for information-centric networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2920–2931, 2014.

[9] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, "Content popularity prediction and caching for icn: A deep learning approach with sdn," *IEEE access*, vol. 6, pp. 5075–5089, 2017.

[10] D. Gang, L. Wang, F. Li, and R. Li, "Distributed probabilistic caching strategy in vanets through named data networking," in *Computer Communications Workshops*, 2016.

[11] W. Huang, T. Song, Y. Yang, and Y. Zhang, "Cluster-based cooperative caching with mobility prediction in vehicular named data networking," *IEEE Access*, vol. 7, pp. 23 442–23 458, 2019.

[12] A. S. Gomes, B. Sousa, D. Palma, V. Fonseca, Z. Zhao, E. Monteiro, T. Braun, P. Simoes, and L. Cordeiro, "Edge caching with mobility prediction in virtualized lte mobile networks," *Future Generation Computer Systems*, vol. 70, pp. 148–162, 2017.

[13] 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 Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5435–5444, 2017.

[14] L. Yao, Y. Wang, X. Wang, and G. Wu, "Cooperative caching in vehicular content centric network based on social attributes and mobility," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[15] M. K. Denko and J. Tian, "Cross-layer design for cooperative caching in mobile ad hoc networks," in *Communications and NETWORKING Conference*. IEEE, 2008, pp. 375–380.

[16] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 8, pp. 1791–1805, 2017.

[17] W. Gao, G. Cao, A. Iyengar, and M. Srivatsa, "Supporting cooperative caching in disruption tolerant networks," in *International Conference on Distributed Computing Systems*, 2011, pp. 151–161.

[18] ——, "Cooperative caching for efficient data access in disruption tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 3, pp. 611–625, 2014.

[19] F. J. González-Cañete, E. Casilari, and A. Triviño-Cabrera, "A cross layer interception and redirection cooperative caching scheme for manets," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, p. 63, 2012.

[20] G. Deng, F. Li, and L. Wang, "Cooperative downloading in vanets-lte heterogeneous network based on named data," in *INFOCOM WKSHPS*. IEEE, 2016, pp. 233–238.

[21] J. Cui, L. Wei, H. Zhong, J. Zhang, Y. Xu, and L. Liu, "Edge computing in vanets-an efficient and privacy-preserving cooperative downloading scheme," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1191–1204, 2020.

[22] X. Zhuo, Q. Li, G. Cao, Y. Dai, B. Szymanski, and T. La Porta, "Social-based cooperative caching in dtns: A contact duration aware approach," in *8th International Conference on Mobile Adhoc and Sensor Systems (MASS)*. IEEE, 2011, pp. 92–101.

[23] T. Le, Y. Lu, and M. Gerla, "Social caching and content retrieval in disruption tolerant networks (dtns)," in *International Conference on Computing, NETWORKING and Communications*. IEEE, 2015, pp. 905–910.

[24] I. H. Bae and S. Olariu, "Design and evaluation of a fuzzy cooperative caching scheme for manets," in *International Conference on Wireless Communications NETWORKING and Mobile Computing*, 2010, pp. 1–5.

[25] S. Lim, C. Yu, and C. R. Das, "Cache invalidation strategies for internet-based vehicular ad hoc networks," *Computer Communications*, vol. 35, no. 3, pp. 380–391, 2012.

[26] R. Tiwari and N. Kumar, "Cooperative gateway cache invalidation scheme for internet-based vehicular ad hoc networks," *Wireless Personal Communications*, vol. 85, no. 4, pp. 1789–1814, 2015.

[27] P. Parvathy and K. A. Kumar, "2tiercocs: A two-tier cooperative caching scheme for internet-based vehicular ad hoc networks," *Procedia computer science*, vol. 46, pp. 1079–1086, 2015.

[28] R. Soua, E. Kalogeiton, G. Manzo, J. M. Duarte, M. R. Palattella, A. Di Maio, T. Braun, T. Engel, L. A. Villas, and G. A. Rizzo, "Sdn coordination for ccn and fc content dissemination in vanets," in *Ad Hoc Networks*. Springer, 2017, pp. 221–233.

[29] D. R. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. S. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web caching with

consistent hashing," *the web conference*, vol. 31, no. 11, pp. 1203–1213, 1999.
[30] H. Nakazato, M. Nishio, and M. Fujiwara, "Data allocation method considering server performance and data access frequency with consistent hashing," in *2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2012, pp. 1–8.
[31] X. Fu, C. Peng, and W. Han, "A consistent hashing based data redistribution algorithm," *International Conference on Intelligent Science and Big Data Engineering*, pp. 559–566, 2015.
[32] J. Zhou, W. Xie, Q. Gu, and Y. Chen, "Hierarchical consistent hashing for heterogeneous object-based storage," in *Trustcom/BigDataSE/I SPA*. IEEE, 2016, pp. 1597–1604.
[33] W. Xie and Y. Chen, "Elastic consistent hashing for distributed storage systems," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017.
[34] Q. Li, K. Wang, S. Wei, X. Han, L. Xu, and M. Gao, "A data placement strategy based on clustering and consistent hashing algorithm in cloud computing," in *9th International Conference on Communications and Networking in China*. IEEE, 2014, pp. 478–483.
[35] K. Thar, S. Ullah, and C. S. Hong, "Consistent hashing based cooperative caching and forwarding in content centric network," in *Network Operations & Management Symposium*. IEEE, 2014, pp. 314–319.
[36] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet of Things Journal*, 2019.
[37] J. Silva, N. Varela, O. B. P. Lezama, V. Álvarez, and B. de la Hoz, "Vehicle flow prediction through probabilistic modeling," in *Developments and Advances in Defense and Security*, Á. Rocha, M. Paredes-Calderón, and T. Guarda, Eds. Singapore: Springer Singapore, 2020, pp. 409–417.
[38] F. N. Neto, C. Baptista, and C. Campelo, "Predicting routes and destinations of urban trips using ppm method," in *Anais do VII Simpsio Brasileiro de Computao Ubqua e Pervasiva*. SBC, 2015, pp. 121–130.
[39] V. Mirrokni, M. Thorup, and M. Zadimoghaddam, "Consistent hashing with bounded loads," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2018, pp. 587–604.
[40] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker *et al.*, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE Infocom*. IEEE, 1999, pp. 126–134.
[41] A. Keranen, J. Ott, and T. Karkkainen, "The one simulator for dtn protocol evaluation," in *International Conference on Simulation TOOLS and Techniques*, 2009, p. 55.
[42] F. Ekman, A. Keränen, J. Karvo, and J. Ott, "Working day movement model," in *Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models*. ACM, 2008, pp. 33–40.
[43] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *ACM SIGCOMM workshop on Delay-tolerant networking*, 2005, pp. 1–8.

Lin Yao is a professor in School of DUT-RU Information Science and Engineering, China. Her research interests include data caching and data security.

Xiaoying Xu is an M.E. candidate in School of Software, Dalian University of Technology. Her research interests include security and privacy in VCCN.

Jing Deng (M'02, SM'13, F'17) is a professor in the Department of Computer Science at UNC Greensboro, Greensboro, NC, U.S.A. Dr. Deng's research interests include online social networks, wireless networks, and network security.

Zhaoyang Li is an M.E. candidate in School of Software, Dalian University of Technology. His research interests include security and privacy in VCCN.

Guowei Wu is a professor in School of Software, Dalian University of Technology (DUT). His research interests include embedded real-time system, cyber-physical systems (CPS), wireless sensor networks, and NDN.