Cluster-based Selective Cooperative Caching Strategy in Vehicular Named Data Networking

Wanying Huang*, Tian Song† Yating Yang† and Yu Zhang*
*School of Information and Electronic, Beijing Institute of Technology
†School of Computer Science and Technology, Beijing Institute of Technology
Email: {h_wanying, songtian, yangyating, yuzhang}@bit.edu.cn

***Abstract*—Vehicular named data networking (VNDN) is a potential candidate to be deployed in information rich applications of vehicular communication. Due to vehicle mobility, it is difficult to construct stable and reliable connections between nodes. Consequently, VNDN is suffering low quality of experience and heavy network load in data transmission. In order to solve these problems and improve network performance, we propose a Cluster-based Selective Cooperative Caching (CSCoC) strategy in VNDN. The main idea of CSCoC is to cache popular data at a set of vehicles that may establish stable communication with others. In our strategy, cluster dividing method is used to divide vehicles and select cluster heads as caching vehicles based on their mobility patterns. To mitigate the impact from fast vehicle mobility, we design a cooperative caching strategy which allows cluster members to request and fetch data from their cluster heads. Additionally, we put forward a method of computing the local data popularity, and a cache replacement strategy based on the data popularity. Compared with existing strategies, simulation results demonstrate how CSCoC increases VNDN performance by around 70% in terms of average access delay, average hop count and cache hit ratio.**

***Index Terms*—VANET, NDN, cooperative caching, cluster**

I. Introduction

VANETs have been rigorously studied by communities from industries and academia in recent decades. As a special type of mobile ad hoc network, they generally consist of road side units (RSUs) and regular vehicles. Nodes in VANETs usually depend on unique IP address to locate the destination and to establish/maintain end-to-end communication. However, based on the TCP/IP, VANETs are facing some typical challenges, which include, but not limited to dynamic topology due to fast and varying mobility, intermittent connection, temporal and spatial network density, etc.

Named data networking, one of the future Internet architecture, is an extension of content-centric networking. NDN changes the semantics of network service from delivering packets to a given destination address to fetching the Data packet identified by a given name [1]. It retrieves a given Data directly using the Data name instead of referring to the IP address of the node [2]. This implementation could mitigate the problems brought by the traditional network architecture.

Vehicular named data networking (VNDN), which merges VANETs and NDN, has attracted many researchers to study theories and technologies for providing efficient data retrieval in mobile environment. In comparison to regular networks, VNDN allows nodes to communicate directly prior to the knowledge of any identification or IP address and asynchronous data exchanges are supported. Therefore, node mobility has less impact on vehicular communication in VNDN.

Corresponding author: Tian Song, songtian@bit.edu.cn

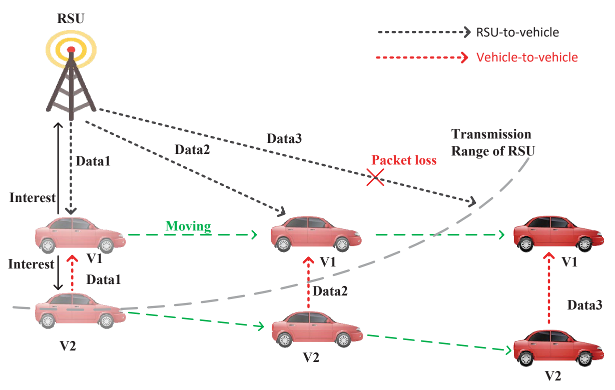


Fig. 1. Unstable RSU-to-vehicle communication and relatively stable vehicle-to-vehicle communication in a cluster

Although VNDN can reduce influence brought by vehicle mobility to a certain extent, it is sometimes difficult to continuously and efficiently request and fetch Data from RSUs in dynamic vehicular network. For example, as shown in Fig. 1, the RSU-to-vehicle communication may be unstable due to the fast and varying vehicle mobility (e.g. the link between RSU and V1), while the connection between regional vehicles is relatively stable in this situation. Unstable connections lead to low QoE and heavy network load in data transmission. Therefore, allowing vehicles to directly obtain Data from neighbors can solve these problems brought by node movement.

Caching, as one of the cores in NDN, is helpful to improve QoE and reduce the network load in data transmission but still lacks the investigation to improve performance of VNDN. Depending on whether the data routers cooperate with each other, caching mechanisms can be divided into non-cooperative caching schemes [11] and cooperative caching schemes. NDN still adopts in-network caching in a non-cooperative caching manner, increase access latency and query overhead in data delivery [4]. Benefiting with high cache hit ratio and low access delay, cooperative caching can significantly increase cache efficiency. By leveraging cache cooperation, vehicles are able to cooperatively cache Data with RSUs, which can provide more stable vehicle-to-vehicle communication in vehicular network. In order to mitigate the impact from vehicle mobility and take the advantages of cooperative caching, we put forward a Cluster-based Selective Cooperative Caching (CSCoC) strategy in VNDN. Our CSCoC strategy uses cluster dividing method to divide vehicles based on the relative movement. Additionally, we adopt cooperative caching method to cache popular contents at a set of selected vehicles. The main contributions of this paper are listed as follows:

- We use a cluster dividing method to divide vehicles into different clusters based on their locations and velocity. Stable connections are established between cluster heads and cluster members;
- In the designed cooperative caching strategy, we allow cluster heads to cooperatively cache Data with RSUs, and cluster members can directly fetch Data from their cluster heads in a certain hop count;
- We propose a popularity-aware cache replacement strategy to ensure the most popular Data is cooperatively cached in VNDN;
- According to experiment results, we prove that CSCoC is superior in improving network performance in terms of average access delay, average hop count, and cache hit ratio by comparing with other existing caching strategies.

The rest of this paper is organized as follows: First of all, relative work and application scenario are presented in Section II. The details of the proposed CSCoC strategy are described in Section III. Then in Section IV, we evaluate CSCoC and analyze the experiment results. Finally, we conclude our work in Section V.

II. Background

A. Related Work

In NDN, non-cooperative caching is that nodes independently decide whether to cache the coming Data or not. This characteristic of non-cooperative caching is not helpful to solve frequent cache updates, over-caching or other issues.

In order to address the issues which can not be well solved by non-cooperative caching, researchers have proposed a lot cooperative caching strategies in NDN to improve network performance. Some researchers divide nodes into different ASs or clusters and allow nodes to cooperatively cache with each other [6, 7]. X Hu et al. reduced access cost by fetching Data from local cache or neighbor caches [8]. Z Ming et al. developed an age-based cooperative caching scheme that adaptively pushed popular contents to the network edge [9]. W Gao et al. proposed a scheme that enabled the sharing and coordination of cached data among multiple nodes and reduced data access delay [10]. In these strategies, the effect of improving cache efficiency and network performance is stronger than non-cooperative ones. However, all above contributions do not consider or explore the characteristic of vehicular environment.

Furthermore, cooperative caching for VNDN has also been investigated by researchers. In [4], two novel social cooperation schemes, namely *partner-assisted* and *courier-assisted* were leveraged to improve QoE in VNDN. In [5], a cooperative caching scheme on the basis of prediction on trajectory of vehicles was proposed. In [11], CCBSR was presented to solve the disconnection caused by the mobility of vehicles. In [12], Y Liu et al. made the arbitrary topology become two-level hierarchy and proposed a collaborative cache placement. In these works, caching vehicles were chosen and cooperatively cache with neighbor vehicles in line with certain designed rules. However, the dynamic topology led to heavy control overhead, which limited the effect of cooperative caching.

To bridge the gap, our work leverages the precious lessons gained from the past to mitigate the impact from vehicle mobility, and improve QoE of users and reduce network load in VNDN.

B. Application Scenario

VANETs consist of RSUs and vehicles. Each node can communicate with other users within wireless communication range. For vehicle-to-vehicle communication, vehicles can communicate and exchange data with one-hop neighbors. As most vehicular networks, RSUs are deployed throughout the network to relay messages as well as to facilitate Internet connections. We further assume that every vehicle is equipped with a GPS receiver for the location service.

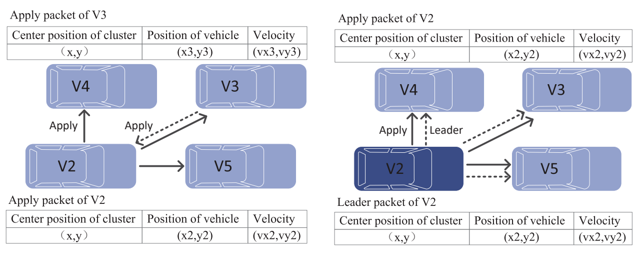
In VNDN, each vehicle equipped with three essential data structures: pending interest table (PIT) for recording Interest packet that has no data packet response, forwarding information base (FIB) for guiding the forwarding of Interest packet and content store (CS) for caching content replicas. Original VNDN takes the pull-based communication that consumer sends Interest packet to producer and retrieves Data. Data is cached in all vehicles on the request path. When the content store is full, the new coming Data packets will replace the existing Data according to the Least Recent Use (LRU) cache replacement strategy.

There is a strong demand to provide drivers and passengers with multimedia streaming services in vehicular network. Fast and varying velocity, intermittent connection in sparse network infrastructures make data delivery become difficult. Therefore, CSCoC strategy is proposed towards multimedia streaming transmission to mitigate the impact from these inherent obstacles.

The application scenario is as follows: Initially, after requirements are generated from users, Interest packets are transmitted to servers. As the Data is queried and delivered by users, it is cached by relayed vehicles and RSUs. Requirements can be satisfied from nodes which have cached the matched Data. Our algorithm focuses on how to choose appropriate vehicles and Data to cache.

III. Cooperative Caching in VNDN

In CSCoC strategy, vehicles are divided into different clusters in terms of their position and velocity. Then cluster heads



Apply packet of V3

Center position of cluster	Position of vehicle	Velocity
（x,y）	(x3,y3)	(vx3,vy3)

Apply packet of V2

Center position of cluster	Position of vehicle	Velocity
（x,y）	(x2,y2)	(vx2,vy2)

Apply packet of V2

Center position of cluster	Position of vehicle	Velocity
（x,y）	(x2,y2)	(vx2,vy2)

Leader packet of V2

Center position of cluster	Position of vehicle	Velocity
（x,y）	(x2,y2)	(vx2,vy2)

(a) (b)

Fig. 2. Cluster dividing example

cooperatively cache Data with RSUs, and cluster members are allowed to request Data from the heads. Finally, a cache replacement strategy based on Data popularity is designed to guarantee the most popular Data is always cached among vehicles.

A. Cluster Dividing

In VNDN, vehicles with the same moving direction are divided into several clusters. All the packets transmitted in this process contain the center position of clusters, and position and velocity of vehicles. The procedure is described as follows:

- When the network is first established, all vehicles are in an orphan state. Each vehicle exchanges Beacon and records the information of one-hop neighbors. Then, vehicles calculate average relative velocity (ARV) [13] and add it to the next Beacon. If ARV of a vehicle is the lowest, it becomes a cluster head and broadcasts *leader* packets. Otherwise, it waits for the *leader* from other vehicles.
- After clusters are established, cluster heads can easily calculate the center position of local clusters. Then those cluster heads periodically broadcast *leader* packets to cluster members.
- If a vehicle V_2 does not receive *leader*, it broadcasts *apply* packets to neighbor vehicles, which is described in Algorithm 1. If a head V_1 receives *apply* from V_2, it replies *leader*. If a cluster member V_3 which near to the center of the cluster receives the *apply* from V_2, it replies *apply* packet (see Fig. 2(a)). Otherwise, the vehicle considers itself as a new head and broadcasts *leader* (see Fig. 2(b)).
- The head broadcasts *leave* when leaving the old cluster. A table of neighbor cluster heads is included in the *leave*. After receiving the packet, cluster members store the table temporarily and select a new head as last rule.
- There may be more than one head during selecting procedure. In order to avoid this situation, a head will abandon its head identity when receiving a *leader* from a vehicle closer to center.

Algorithm 1 Cluster Dividing

```
1:  $V_1$ : cluster head, broadcasts *leader*;
2:  $V_2, V_3$ : cluster members;
3:  **if** $V_2$ receives *leader* from $V_1$ **then**
4:      $V_1$ is the head of $V_2$;
5:  **else**
6:      $V_2$ broadcasts *apply*;
7:      **if** $V_3$ receives *apply* and is nearer to the center of cluster
        than $V_2$ **then**
8:          $V_3$ returns *apply*;
9:      **else**
10:         $V_2$ broadcasts *leader*;
11:     **end if**
12: **end if**
```

Obviously, the relative movement between cluster heads and cluster members is smaller than between RSUs and vehicles. Cluster dividing is helpful to construct more stable communication in high mobile environment.

B. Cooperative Caching Strategy

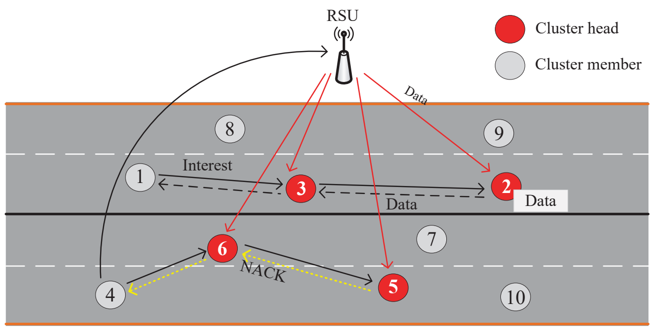


Fig. 3. An example of cooperative caching

After cluster dividing, cluster heads transmit *leader* packets to the closest RSUs and neighbor vehicles. Upon receiving the *leader*, RSUs transmit Data packets to the cluster heads directly, then these heads cache the coming Data in CS. When cluster members generate requirements, they will request and fetch the desired Data from the heads directly instead of RSUs.

Algorithm 2 describes the procedure of cooperative caching strategy in detail. Upon generating requirement, cluster member CV sends Interest to its cluster head H_A and k is updated with $k \leftarrow k-1$. In the case of that H_A does not have the matched Data, if $k \neq 0$, Interest is forwarded to next neighbor cluster head H_B; otherwise, V receives a NACK packet from H_A and fetches Data from the closest RSU. In the case of that H_A has the corresponding Data, it returns the packet to V directly. In this strategy, k is used to limit the hop count and embedded in the Interest packets. The value of k is initialized according to the vehicle density at the establishing stage of network. In sparse scenarios, the proportion of cluster heads in the whole network is higher than that in the dense scenarios. Furthermore, heavy traffic load would impair network ability in the dense scenarios. Therefore, k is set to a small value in the sparse areas and vice versa. For example, if the vehicle density is smaller than $100/km^2$, the value of k is set to 1.

Algorithm 2 Cooperative Caching

```
1: $H_A$ : a cluster head;
2: $H_B$ : neighbor cluster head of $H_A$;
3: $V$ : a cluster member belongs to $H_A$;
4: $k$ : limited hop count;
5: while $V$ need Data do
6:     $V$ sends Interest to $H_A$;
7:     $k \leftarrow k - 1$;
8:     if $H_A$ does not have matched Data then
9:         if $k \neq 0$ then
10:            $H_A$ forward Interest to $H_B$;
11:            $k \leftarrow k - 1$;
12:        else
13:            $H_A$ forwards NACK to $V$;
14:            $V$ sends Interest to RSU;
15:        end if
16:    else
17:        $H_A$ returns Data;
18:    end if
19: end while
```

Fig. 3 shows an example of the cooperative caching with initial value $k = 2$. Upon receiving Interest from consumer V_1 and V_4, k is updated to V_1 and cluster head V_3 and V_6 find out that matched Data does not exist in CS. Therefore, Interests are forwarded to neighbor heads V_2 and V_5, respectively. Then head V_2 returns the matched Data to vehicle V_1 through the relay vehicle V_3. Head V_5 which has not the matched Data can not forward the Interest to next neighbor head due to $k = 0$. Therefore, requesting vehicle V_4 receives Nack packets from the head V_5, then request the Data from the closest RSU.

The proposed cooperative caching strategy allows Data retrieval through vehicle-to-vehicle communication within limited hop count. Fetching Data from neighbors is conducive to establish stable connection in vehicular network, and limiting the hop count can avoid producing extra network load.

C. Popularity-aware Cache Replacement

In LRU, Data replacement depends on vehicles themselves. Because Data popularity calculated by RSUs is more accurate than by vehicles, a popularity-aware cache replacement strategy is proposed. Only popular Data, with a large number of requests, should be cached; less ones should be replaced by popular ones.

Every node contains a new Data structure named Data Tag as shown in Table 1. For each Data, Data Tag records its name, time of the last request Δt and popularity value P.

TABLE I
DATA TAG

Notation	Description
name	content name
time	time of the last request
value	popularity value

Algorithm 3 Popularity-aware Cache Replacement

```
1: $D_i$ : Data pushed from RSUs to cluster heads;
2: $P_i$ : popularity value of $D_i$, recoded in Data Tag;
3: while $D_i$ is pushed to caching vehicles do
4:     if $CS$ has enough space for $D_i$ then
5:         cache $D_i$ in CS;
6:     else
7:         replace Data with lowest popularity value;
8:         $D_i$ is sorted according to $P_i$;
9:     end if
10: end while
11: while $D_i$ is requested do
12:     $P_i \leftarrow P_{highest}$;
13: end while
```

For example, when a new Data packet i arrives at an RSU, its popularity value P_i equals to the highest popularity value $P_{highest}$ in CS of the RSU. The popularity value P_i declines with time, which is calculated as follows:

$$P_i \Leftarrow P_i \cdot e^{-\lambda \Delta t}$$

where

λ donates an exponential decay constant;

Δt donates the time interval between the current time and the last time receiving the same request on Data i, which is strongly relative to request frequency;

If Data i is requested from RSU, popularity value is updated with the highest value:

$$P_i \Leftarrow P_{highest}$$

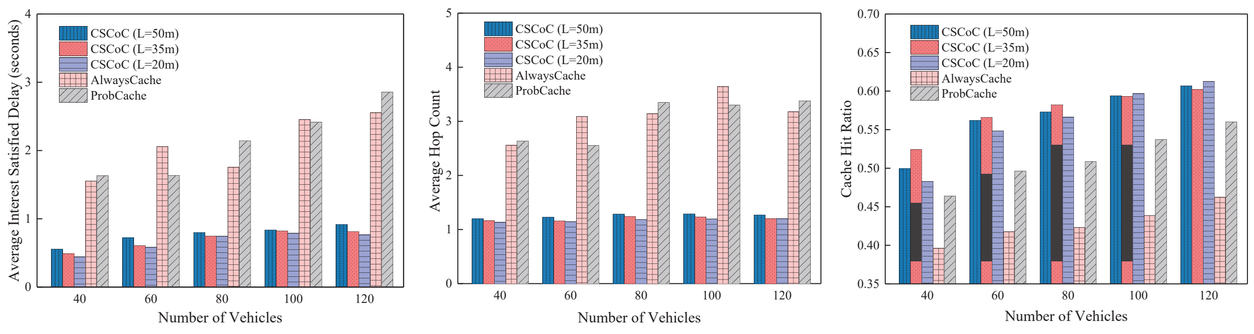
Algorithm 3 elaborates the procedure of this cache replacement strategy. In CSCoC, cluster heads send $leader$ packets to the closest RSUs to inform their status, then RSUs transmit popular Data with the corresponding popularity values to them. If the CS of cluster head has enough space to save the coming Data, then the Data is cached directly. Otherwise, Data with the lowest popularity value in the CS of head is replaced by the coming ones. The popularity value of the new Data is updated according to the above procedure to ensure only popular Data cooperatively cache with RSUs.

IV. RESULTS

A. Simulation Setup

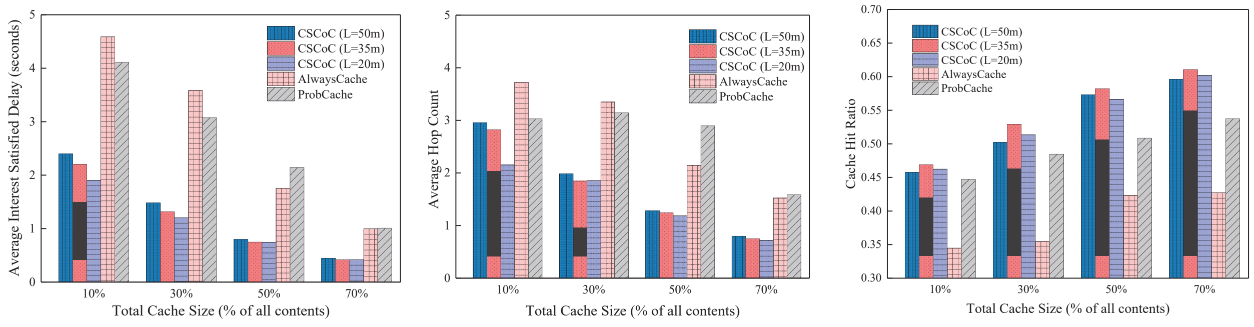
In order to evaluate merits of the proposed strategy, CSCoC over VNDN is simulated in the ndnSIM [15], a NS-3 based simulator for NDN. A mobility model consists of a 5Km single way with four lanes and 10 RSUs is used. RSUs uniformly distribute along the way and can communicate with each other. In addition, we also vary the network size ranging from 40 to 120 nodes (i.e. NDN equipped Vehicles). The rest simulation parameters are summarized in Table 2.

During the simulation, the longest length of each cluster is $R = 50m$, which is the half of transmission range of vehicles. Therefore, vehicles can communicate with each other in both local and neighbor clusters. Vehicles only request multimedia



(a) Average Access Delay (b) Average Hop Count (c) Cache Hit Ratio

Fig. 4. Impact of number of vehicles on different schemes



(a) Average Access Delay (b) Average Hop Count (c) Cache Hit Ratio

Fig. 5. Impact of cache size on different schemes

streaming service and the limited hop count k is set to 1 to evaluate CSCoC in a convenient way. All vehicle nodes have the same size of CS and the total size of each CS are ranged from 10% to 70% of the whole contents. Simulation assumes that there are 1000 content items with different popularity value and same size 20M. Different network size is considered to simulate different vehicle density scenarios. The total number of vehicles are ranged from 40 to 120. We conduct 100 independent simulations for each scenario and the simulation time is set to 120s.

The performance of CSCoC with different length of cluster L is compared against: i) the regular cache placement scheme that cache Data in all nodes on the request path, termed as AlwaysCache, ii) cache Data in nodes with a certain probability constant ($p = 0.5$), termed as ProbCache [14]. In contrast strategies, vehicles directly request and fetch desired data from the closest RSUs rather than other vehicles.

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Caching Buffer	100M, 300M, 500M, 700M
Network Size	40, 60, 80, 100, 120
No. of Contents	1000
Content Size	10M
Simulation Time	120s
Vehicle Speed	[7m/s, 10m/s]
RSU Transmission Range	500m
Vehicle Transmission Range(R)	100m
Limited Hop Count(k)	1
Length of Cluster(L)	50m, 35m, 20m
Exponential Decay(λ)	0.05

B. Average Access Delay

The average access delay is firstly compared with different strategies as shown in Fig. 4(a) and 5(a). The average access delay is defined as latency from sending Interest packets to fetching the desired data. The decreasing of latency represents the improvement of quality of experiment in data transmission. In Fig. 4(a), the average access delay of the CSCoC, AlwaysCache and ProbCache is shown with varying number of vehicles and 500M total cache size. We can see that as the number of vehicles grows, the average delay increases for all the strategies considered. It is easily understood that more vehicles generate more Interest packets for desired data. Latency is produced in the progress of processing Interest and Data packets. Comparing with AlwaysCache and ProbCache, the proposed strategy respectively reduces delay with 54.4-71.6% and 47.0-73.1%. Fig. 5(a) depicts that with the total cache size increasing, the average access delay declines for all the experimental strategies. The main reason is that larger cache size creates more chances for vehicles to fetch popular data from local or neighbor cluster heads. It is noteworthy that CSCoC has lower access delay than other two existing proposals. The highest decrease of average access delay is up to 73.8% for cache size 10%. That is because the CSCoC enables the cluster members to request data from their own cluster heads or neighbor heads rather than RSUs, which could significantly reduce latency. Smaller access delay state that CSCoC strategy can provide higher QoE for users.

C. Average Hop Count

The average hop count, the number of hops that the retrieved data packet traveled on the way back from producer application or cache. The differences between average hop count for varying number of vehicles and total cache size are respectively depicted in Fig. 4(b) and 5(b). Results show that CSCoC has the lower and stabler average hop count than other two existing proposals, AlwaysCache and ProbCache with 53.2-67.1% and 52.0-64.6%, respectively. Simulation results also show that CSCoC reduces the hop count by up to 68.1%. This is because vehicles can retrieve data from their one-hop cluster heads. As the total cache size increases, the metric decreases for all considered strategies. CSCoC performs the best because the popularity-aware cache replacement can guarantee the most popular data to be cached in the cluster heads. Therefore, most of the requirements can be satisfied from neighbor vehicles, which lead to average hop count decreasing. Also, the results state that CSCoC effectively reduces the network load.

D. Cache Hit Ratio

The cache hit ratio is defined as the ratio of the number of requests served by the cache to the total number of requests arrived at the cache. We study the cache hit ratio for the different number of vehicles and the cache size. The Fig. 4(c) and 5(c) show that CSCoC has a better performance than both AlwaysCache and ProbCache. For example, when the number of vehicles is 120, the cache hit ratio of AlwaysCache and ProbCache are about 0.46 and 0.56, respectively, and the ratio of CSCoC is about 0.61, achieving an improvement of 5-15%. The ratio of CSCoC is always higher than both contrast schemes in different situations. The highest growth rate of the proposed strategy in the cache hit ratio is up to 71.3%. The main reason is that cluster heads always cache the most popular data so that requesting vehicles can directly request and fetch Data from the heads. Through combination of the cooperatively caching strategy and popularity-aware cache replacement, cluster members always can retrieve desired data from local or neighbor heads. The high cache hit ratio implies that CSCoC significantly improves the cache efficiency and reduces the substantial load of data servers.

V. CONCLUSION

In this paper, Cluster-based Selective Caching strategy (CSCoC) is proposed to mitigate the low QoE and heavy network load resulted from vehicle mobility in VNDN. By using cluster dividing method, CSCoC can divide vehicles based on the relative movement. A set of selected vehicles cache popular Data and provide stable data transmission to others. In the experiment, we evaluate three metrics in the case of the different number of vehicles and cache sizes: average access delay, average hop count and cache hit ratio. Simulation results show that the CSCoC respectively reduces the average access delay and the average hop count by up to 73.8% and 68.1%, and the highest growth rate in the cache hit ratio is up to 71.3% in comparison with AlwaysCache and ProbCache. Our work demonstrates CSCoC strategy can mitigate the impact from vehicle mobility, provide higher QoE and lower network load in VNDN.

ACKNOWLEDGMENT

This work is sponsored by the Science and Technology on Communication Networks Laboratory Foundation Project, the Aerospace Field Pre-research Foundation Project(No.060501) and National Natural Science Foundation of China(No.61672101).

REFERENCES

[1] Lixia Zhang, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang, “Named Data Networking,” ACM SIGCOMM CC Review, vol.44, no. 3, 66-73, July. 2014.

[2] Modesto, Felipe, and A. Boukerche, “A Novel Service-oriented Architecture for Information-Centric Vehicular Networks,” ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems ACM, 2016:136-139.

[3] Ioannou, Andriana, and S. Weber, ”A Survey of Caching Policies and Forwarding Mechanisms in Information-Centric Networking,” IEEE Communications Surveys and Tutorials, 2016, 18(4):2847-2886.

[4] Wei Quan, Changqiao Xu, Jianfeng Guan, Hongke Zhang and Luigi Alfredo Grieco, “Social cooperation for information-centric multimedia streaming in highway VANETs,” World of Wireless, Mobile and Multimedia Networks IEEE, 2014:1-6.

[5] Lin Yao, Ailun Chen, Jing Deng, Jianbang Wang and Guowei Wu, “A Cooperative Caching Scheme Based on Mobility Prediction in Vehicular Content Centric Networks,” IEEE TVT, 2017.

[6] Jason Min Wang, Jun Zhang and Brahim Bensaou, “Intra-AS cooperative caching for content-centric networks” ACM SIGCOMM Workshop on ICN, 2013:61-66.

[7] Jia Ji, Mingwei Xu and Yuan Yang, “Content-hierarchical intra-domain cooperative caching for information-centric networks,” International Conference on Future Internet Technologies ACM, 2014:6.

[8] Xiaoyan Hu, Christos Papadopoulos, Jian Gong and Daniel Massey, “Not So Cooperative Caching in Named Data Networking,” Global Communications Conference IEEE, 2013:2263-2268.

[9] Zhongxing Ming, Mingwei Xu and Dan Wang, “Age-based cooperative caching in information-centric networking,” International Conference on Computer Communication and Networks IEEE, 2012:1-8.

[10] Wei Gao, Guohong Cao, Arun lyengar and Mudhakar Srivatsa, “Cooperative Caching for Efficient Data Access in Disruption Tolerant Networks,” IEEE Transactions on Mobile Computing, 2014:611-625.

[11] LanChao Liu, Dongliang Xie, Siyu Wang and Zhen Zhang, “CCN-based cooperative caching in VANET,” International Conference on Connected Vehicles and Expo IEEE, 2016:198-203.

[12] Yinlong Liu, Dali Zhu and Wei Ma, “A novel cooperative caching scheme for Content Centric Mobile Ad Hoc Networks,” Computers and Communication IEEE, 2016:824-829.

[13] Ni, Minming, Z. Zhong, and D. Zhao. ”MPBC: A Mobility Prediction-Based Clustering Scheme for Ad Hoc Networks.” IEEE Transactions on Vehicular Technology 60.9(2011):4549-4559.

[14] Ioannis Psaras, Wei Koong Chai and George Pavlou, “Probabilistic in-network caching for information-centric networks,” Edition of the ICN Workshop on ICN ACM, 2012:55-60.

[15] Spyridon Mastorakis, Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang, “ndnSIM 2.0: A new version of the NDN simulator for NS-3,” NDN, Technical Report NDN-0028, 2015.