Two-layer Distributed Content Caching for Infotainment Applications in VANETs

Zheng Xue, Yang Liu, Guojun Han, *Senior Member, IEEE,* Ferheen Ayaz, *Student Member, IEEE,* Zhengguo Sheng, *Senior Member, IEEE,* and Yonghua Wang, *Member, IEEE*

***Abstract*—For vehicular ad-hoc networks (VANETs), edge caching has attracted considerable research attention to maximize the efficiency and reliability of infotainment applications. In this paper, we propose a two-layer distributed content caching scheme for VANETs by jointly exploiting the cache at both vehicles and roadside units (RSUs). Specifically, we formulate content caching problem to minimize the overall transmission delay and cost as a nonlinear integer programming (NLIP) problem and propose an alternate dynamic programming search (ADPS) based algorithm to solve it. In ADPS, we divide the original problem into three sub-problems, then we use the dynamic programming (DP) method to solve each sub-problem separately. To reduce the complexity, we further propose a cooperation-based greedy (CBG) algorithm to solve the large scale original problem. Both numerical simulation results and experiments in testbed show that the proposed caching scheme outperforms existed caching schemes, the transmission delay and cost can be reduced by 10% and 24% respectively, while the hit ratio can be increased by 30% in a practical environment, as compared to popularity-based caching scheme.**

***Index Terms*—VANETs, Infotainment Application, content delivery delay, content delivery cost, edge caching scheme**

I. Introduction

HIGH-QUALITY vehicular infotainment applications in VANETs, such as in-car entertainment or mobile advertising, have gained considerable attention in recent years. As an example, using interactive screens installed on vehicles, on-board passengers can browse the list and request interested media contents. This is an attractive value-added service to transportation operators such as taxi companies or peer-to-peer vehicle sharing companies like Uber. For these type of applications, the primary concern is to meet the stringent requirements of quality of service (QoS). Unlike other applications, such as secure message dissemination or adaptive traffic management, infotainment application can significantly impact the amount of data exchanged in VANETs. The high volume of exchanged data will degrade QoS, such as delivery delay. Moreover, from the perspective of operator, the high randomness and high concurrency feature of infotainment application raise challenges for some performance metric of the network, such as content hit ratio. And this may also result in QoS degradation.

Besides, for infotainment applications, the cost of data transmission is also very considerable due to the excessive amount of data to be transferred. Compared with investment into the construction of VANETs, content storage cost, which corresponds to the cost of updating the contents, can be negligible. Hence for operators, it is expected that the cost of wireless mobile communications will become a major source of operational expenses of infotainment application service. The operators need to maximize the revenue and reduce the expense. Hence, taking a commercial perspective, how to reduce the communication cost for infotainment applications has become a crucial problem. Clearly, the cost per unit data transmission varies when the vehicle fetch contents from different nodes in VANETs. If a vehicle always need to fetch content from a remote server, then the total cost of the operator's service will be high.

To address the two problems above, VANET edge caching is proposed to facilitate content delivery. That is, each RSU can selectively store media contents and distribute them to vehicles [1]. However, the caching resource of RSU is constrained. Meanwhile, when vehicles drive through different RSUs, the vehicular connections are always intermittent due to the limited coverage area of RSU. Hence, compared with sole RSU caching, using vehicles for cooperative caching is helpful and reasonable. This cooperation among RSUs and vehicles can provide seamless connections within the interlaced coverage of RSUs. Moreover, the contents for infotainment applications always have different request probability. Hence, a more precise content caching scheme is required, i.e., determining whether to cache a specific content on a specific network node (RSU or vehicle). With this caching scheme, users can access their required contents directly from a much closer RSU or vehicle, which can remarkably decrease the content retrieval delay and transmission cost. The limited storage capacity of vehicles and RSUs can be fully utilized to improve network performance in terms of average content delivery delay, average content delivery cost, and cache hit ratio.

There has been a number of researches focusing on caching in device-to-device (D2D) mobile networks [2]–[13]. Sun *et al*. [2] propose mobility-aware caching strategies in D2D

Manuscript received February 15, 2021; revised May 4, 2021; accepted June 7, 2021. This work was supported in part by Natural Science Foundation of China (Nos. U2001203, 61871136, 62071131), Natural Science Foundation of Guangdong Province (2014A030310266). ($Corresponding\ author: Yang\ Liu.$)

Zheng Xue, Yang Liu and Guojun Han are with School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China. (e-mail: xuezheng@mail2.gdut.edu.cn; liuyang@gdut.edu.cn; gjhan@gdut.edu.cn).

Ferheen Ayaz and Zhengguo Sheng are with the Department of Engineering and Design, University of Sussex, Brighton, BN1 9RH, U.K. (e-mail: f.ayaz@sussex.ac.uk; z.sheng@sussex.ac.uk).

Yonghua Wang is with School of Automation, Guangdong University of Technology, Guangzhou 510006, China. (e-mail:wangyonghua@gdut.edu.cn).

networks to minimize the network delay. Wang *et al.* [5] take advantage of the user mobility pattern by the inter-contact times between different users, and propose a mobility-aware caching placement strategy to maximize data offloading ratio. Some of these works study edge caching for conventional cellular networks. However, they have different emphases from the works for VANETs. Firstly, in conventional cellular networks, the network topology is relatively stable. So that server switching is not very frequent. Secondly, the power and storage of a mobile device in a conventional cellular network are limited, so it is inappropriate to choose mobile devices as caching servers or relays. Thirdly, the works always use inter-contact model to characterize the mobility pattern, but this model is insufficient in dealing with practical vehicle mobility.

In vehicular networks, most researches focus on caching at RSU layer. Ding *et al.* [14] propose a centralized RSU caching algorithms to minimize the average time that an onboard unit (OBU) downloads a file. Hu *et al.* [15] study the scenario of RSU caching, where multiple content providers try to improve the data dissemination efficiency of their own contents by utilizing the storage of RSUs. Su *et al.* [16] propose a model to determine whether and where to obtain the replica of content upon the request of a vehicle. They propose a cross-entropy based dynamic content caching scheme. Qin *et al.* [17] propose a hierarchical end-edge framework with the aid of computation offloading and content caching to minimize network overheads. AlNagar *et al.* [18] propose a cooperative proactive caching scheme between RSUs to minimize the communication latency and enhance QoS in VANETs. Ma *et al.* [19] propose a caching placement policy based on the cloud-based vehicular ad-hoc network (VANET) architecture to minimize the average latency of content retrieve. Hou *et al.* [20] propose a Q-learning-based proactive RSU caching strategy under the support of long short-term memory (LSTM) neural network, aiming at enhancing the QoS for non-safety related services in vehicular networks.

Caching at vehicular layer via vehicle-to-vehicle (V2V) communications reduces transmission delay and bandwidth competition, as compared to RSU caching. Hu *et al.* [21] propose an in-vehicle caching framework based on an innovative integration of distributed storage with cached content relay facilitated by one-hop V2V links. They design a dynamic distributed storage relay (D^2SR) mechanism, which ensures the survival of cached contents with a high probability within the time duration of interest. Yao *et al.* [22] propose a cooperative caching scheme based on social attributes and mobility prediction for vehicular content centric network with higher cache hit ratio and lower content access delay compared to other state-of-the-art schemes. Deng *et al.* [23] propose an optimal retention-aware caching scheme to minimize network cost. Zhang *et al.* [24] model the interactions between caching vehicles and mobile users as a two-dimensional Markov process and propose an online vehicular caching scheme by optimizing network energy efficiency. Han *et al.* [25] develop a dynamic pricing-based incentive mechanism for content sharing in cellular vehicle-to-everything (C-V2X) based vehicular network to improve the successful delivery ratio and energy efficiency. Yang *et al.* [26] propose a secure caching placement and delivery algorithm to minimize the maximum vulnerability among all vehicular users.

A few schemes also studied cross-layer cooperative caching [27], [28] (i.e., Cooperative caching in RSUs and/or vehicles and/or base stations (BSs)). Chen *et al.* [27] propose a cooperative edge caching scheme based on the heterogeneous vehicular networks (H-VNets) with multi-tier edge caching servers (i.e. at BS and RSUs). Qiao *et al.* [28] design a novel edge caching framework based on the cooperation among base station, RSUs, and vehicles. Then, the joint vehicle scheduling and bandwidth allocation scheme is designed to minimize the content access cost while satisfying the constraint on content delivery latency.

Existing literature is mostly based on the VANETs with simple one-layer structure (e.g., vehicular layer or RSU layer). Their edge caching schemes usually consider vehicular downloading applications with a single QoS metric. At the same time, few research addressed the commercial perspective and investigated the data transmission cost issue. In this paper, we consider joint caching of vehicles and RSUs. The paper aims to fulfill two main objectives. Firstly, full utilization of limited caching capacity of vehicles and RSUs to satisfy as many content requests as possible. Secondly, determining appropriate caching scheme to minimize the overall transmission delay and cost. We form the two-layer caching scheme design as a NLIP problem. Because the problem is neither linear nor convex, we propose an ADPS based algorithm to solve the NLIP problem. In ADPS, we divide the original problem into three sub-problems, then we use DP method to solve each sub-problem separately. Furthermore, we propose a low-complexity CBG algorithm to solve the large scale original problem. The results of numerical simulation and test bed verification show our proposed caching scheme outperforms existed caching schemes in transmission delay, cost and hit ratio. The main contributions of this paper are listed as follows.

- We propose a novel two-layer cloud-based VANET edge caching model, consisting of RSU cloud (RC) layer and vehicular cloud (VC) layer. In this model, the networks nodes (e.g., vehicles, RSUs) have different communication interfaces as well as heterogeneous computation, communication and storage capacities. Vehicles may submit multimedia content request, which are offloaded to different network nodes based on a certain scheduling algorithm.
- Considering vehicle mobility and content retrieve process in our model, the average transmission delay, cost and hit ratio are analytically derived. Then, considering the limited caching capacity of different nodes, the joint vehicle and RSU optimized caching problem is formulated as a NLIP problem.
- We propose an ADPS algorithm to solve the formulated NLIP problem. This algorithm divides the original problem into three sub-problems. The sub-problem is reformulated as a multi-stage decision problem and be solved stage-by-stage recursively. The result of this algorithm is called ADPS caching scheme.
- Inspired by the result of ADPS, a CBG algorithm with linear complexity is proposed. This algorithm can give

a good enough sub-optimal solution, at the same time, reduce the complexity of computing significantly. So it is more perfect to very large scale network model. The result of this algorithm is called CBG caching scheme.

- In addition to numerical simulation, the system model is fully implemented in a prototype VANET using real OBU and RSU devices and real cloud computing service. Both simulation results and test bed evaluation show that our proposed caching schemes have the advantage over other caching schemes.

The rest of the paper is organized as follows. Section II gives the system model and problem formulation. Section III and IV respectively propose the ADPS and CBG caching scheme. The simulation results are provided in Section V. In Section VI, test bed evaluation is shown. Finally, Section VII concludes this study.

Notation: Bold lower case letters and bold upper case letters are used to denote column vectors and matrices, respectively.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we introduce the network model, communication model, and content distribution model, respectively. We analytically derive the average transmission delay, cost and hit ratio, and then formulate the NLIP problem. This problem is key for caching scheme decision. The primary notations used in this section are summarized in Table I.

A. Network Model

As shown in Fig. 1, we consider a bidirectional road scenario, including vehicle, RSU and base station (BS). The whole VANETs can be divided into two layers: RSU cloud (RC) layer and vehicular cloud (VC) layer. In RC, the RSUs are connected via backbone network to form a computing cloud and all RSUs are termed as RC members. VC is formed by moving vehicles within a specific area. Each VC has a VC controller (VCC) which manages caching resource and other vehicles are termed as VC members. VCC can be selected through appropriate algorithms for a VC. In the network, BS and content provider (CP) connect to the Internet through the backhaul link. CP is generating a different set of contents. Based on user interest, if the requested content has been cached at VC and/or RC, it can be delivered to the vehicle by VC and/or RC, depending on the trajectory of the vehicle and the locations of cached contents.

Without loss of generality, we focus on the coverage area of one BS, within which N_{RC} RSUs are deployed along the road using dedicated short-range communication (DSRC) access technology to communicate with vehicles. The coverage areas of different RSUs are assumed to be overlapping and the communication range of RSU is denoted by r. RSUs and vehicles are equipped with caching capacity, with the storage capacity denoted by S^R and S^V, respectively. A network management controller is deployed at BS, which collects information from vehicles and RSUs and makes decisions on content caching [29].

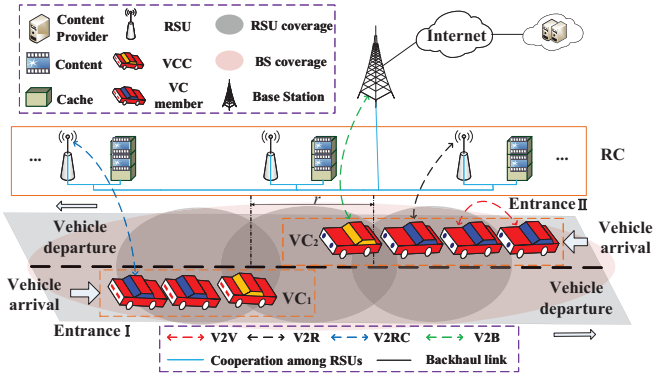

Fig. 1. System model.

TABLE I
PRIMARY NOTATIONS

Notation	Definition
System Elements	
$V_j(t)$	Number of vehicles entering entrance j
$VC_j(t)$	Number of VC_j entering entrance j
λ_j	Average arrival rate of entrance j
μ_j	The expectation of $VC_j(t)$
S_q	Data size of content l
L	Number of contents in the networks
$p(l)$	Request probability of content l
N_{VC_j}	Number of vehicles in VC_j
N_{RC}	Number of RSUs in RC
S^V	Caching capacity of vehicle
S^R	Caching capacity of RSU
r	The coverage range of RSU
v	The average velocity of vehicles
R_{VV}	Data transmission rate for delivering the requested content between vehicles
R_{RV}	Data transmission rate for delivering the requested content from nearest RSU
R_{RCV}	Data transmission rate for delivering the requested content from RC members
R_{PV}	Data transmission rate for delivering the requested content from CP through remote server
P_{VV}	Price for delivering unit data between vehicles
P_{RV}	Price for delivering unit data from nearest RSU
P_{RCV}	Price for delivering unit data from RC members
P_{PV}	Price for delivering unit data from CP through remote server
Analytical Symbols	
$\tau_{j,l}$	The download delay for entire VC_J receiving the whole requesting content l
τ_{av}	The average delay incurred by an arbitrary content request
$\langle f(.)\rangle_0$	$\langle f(.)\rangle_0$ equals 1 if $f(.) > 0$ or 0 if $f(.) = 0$
$\tau_{i,j,l}$	Delay to obtain content l for vehicle V_i in VC_j
$R_{i,m,l,j}$	Data transmission rate for vehicle V_i in VC_j to download content l at RSU_m
$\delta_{i,m,l}^{VC_j}$	Whether vehicle V_i in VC_j has completely obtained the content l from RSU_m
$D_{i,m,l}^{VC_j}$	The cumulative download of the content l when the vehicle V_i in VC_j leaves the cover area of RSU_m.
$Cost_{av}$	The average cost incurred by an arbitrary content request
$Cost_{i,j,l}$	Cost to obtain content l for vehicle V_i in VC_j
$Cost_{i,m,l,j}$	Cost for vehicle V_i in VC_j to download unit data from content l at RSU_m
$\boldsymbol{C}^{VC_j}$	Binary cache placement matrix for VC_j and the dimension is $N_{VC_j} \times L$
$\boldsymbol{C}^{RC}$	Binary cache placement matrix for RC and the dimension is $N_{RC} \times L$

B. Vehicle Mobility Model

In our model, vehicles take platoon-based driving pattern [30], which is a cooperative driving pattern for a group of vehicles with common interests. The speed of the platoon leader VCC is within $[v_{min}, v_{max}]$ [31]. Other vehicles adjust their speed with the goal of tracking the speed of VCC and keeping a constant inter-vehicular space gap [32]. The average velocity of vehicles is $v = (v_{min} + v_{max})/2$. We assume there is one entrance at each end of the road. The number of vehicles entering entrance j $(j = 1, 2)$, denoted as $V_j(t)$, is time-varying and follows a Poisson distribution with the parameter λ_j, which denotes the average arrival rate [14], [21], [33]. Then we have:

$$P\left(V_j\left(t\right) = k\right) = \frac{\left(\lambda_j t\right)^k}{k!} e^{-\lambda_j t}, k = 1, 2, \cdots. \tag{1}$$

The expectation of $V_j(t)$ is $E\left[V_j(t)\right] = \lambda_j t$. $VC_j\left(t\right)$ denotes the number of VC_j entering entrance j. N_{VC_j} denotes the number of vehicles in VC_j. Apparently, $VC_j\left(t\right) = V_j\left(t\right)/N_{VC_j}$. μ_j denotes the expectation of $VC_j(t)$. Then:

$$\mu_j = E\left[VC_j(t)\right] = E\left[\frac{V_j(t)}{N_{VC_j}}\right] = \frac{\lambda_j}{N_{VC_j}} t, j = 1, 2. \tag{2}$$

C. Content Distribution and Retrieve Process Model

$Contents = \{1, 2, \cdots, L\}$ denotes the set of content piece, with the size S_q of each content, which is constantly updated by adding new contents. Considering that some content may be requested more frequently than others, we assume the popularity of content follows a Zipf distribution [34]. The probability that content l is requested is given by:

$$p(l) = \frac{l^{-\gamma}}{\sum_{i=1}^{L} i^{-\gamma}}, \tag{3}$$

where γ is the parameter of the Zipf distribution.

For vehicle V_i $(1 \le i \le N_{VC_j})$ in VC_j and RSU_m $(1 \le m \le N_{RC})$ in RC, we denote binary cache placement matrices at vehicles and RSUs by $\boldsymbol{C}^{VC_j}$ and $\boldsymbol{C}^{RC}$ respectively, where

$$\boldsymbol{C}^{VC_j} = \left[c_{i,l}^{VC_j}\right]^{N_{VC_j} \times L}, \boldsymbol{C}^{RC} = \left[c_{m,l}^{RC}\right]^{N_{RC} \times L}, \tag{4}$$

$c_{i,l}^{VC_j}$, $c_{m,l}^{RC}$ denote the discrete cache placement indicator for content l in vehicle i and RSU_m respectively.

$$c_{i,l}^{VC_j}, c_{m,l}^{RC} = \begin{cases} 1, & \text{if the content } l \text{ is decided to be cached} \\ & \text{in vehicle } V_i \text{ of } VC_j \text{ or } RSU_m, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The content request process is shown in Fig. 2. It should be noticed that each RSU can cooperate with other RC members to fulfill the content request. That means, if a piece of content is cached in one RC member, then all RC members can use it to respond a vehicle's request. We say a hit occurs when the requested content l from moving vehicle V_i in VC_j can be fulfilled by its cache, VC_j or RC. If that still fails, the request can only be fulfilled by CP through the remote server and no hit occurs in this case. So, hit occurrence depends on whether the content requested by the vehicle matches the content

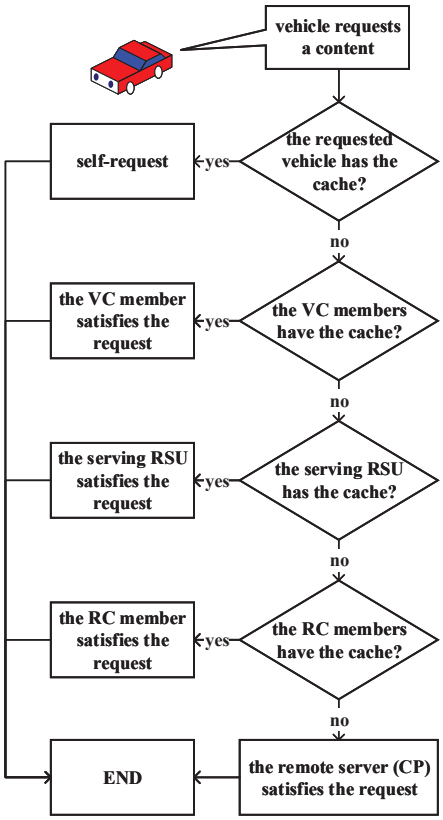

Fig. 2. The process of obtaining the requested content when a vehicle enters the coverage of RSU.

cached in VC and RC. When the content request is satisfied by the VC members, the serving RSU, the RC members, and the CP, the data transmission rates are respectively denoted by $R_{VV}, R_{RV}, R_{RCV}, R_{PV}$. In terms of data transmission cost, the corresponding prices caused by transmitting unit data are denoted by $P_{VV}, P_{RV}, P_{RCV}, P_{PV}$. It is reasonable to assume that $R_{VV} > R_{RV} > R_{RCV} > R_{PV}$, $P_{VV} < P_{RV} < P_{RCV} < P_{PV}$. The transmission delay and cost are 0 if a requesting vehicle obtains the content from its own cache.

D. Average Content Delivery Delay

When VC_j members enter entrance j, vehicles in VC_j will start to download the requested content l. We define $\tau_{i,j,l}$ as the delay for vehicle V_i in VC_j to completely download the requested content l. The download delay for entire VC_j receiving the whole requesting content l is $\tau_{j,l}$. Considering that each content has a different request probability, the expectation of $\tau_{j,l}$ is:

$$E\left[\tau_{j,l}\right] = \sum_{l=1}^{L} p(l) \frac{1}{N_{VC_j}} \sum_{i=1}^{N_{VC_j}} \tau_{i,j,l}. \tag{6}$$

The average delay incurred by an arbitrary content request, denoted by τ_{av}, can be calculated as:

$$\tau_{av} = \sum_{j=1}^{2} \frac{\mu_j E\left[\tau_{jl}\right]}{\mu_1 + \mu_2}. \tag{7}$$

When a vehicle crosses RSU_m, we need to consider the downloading data rate $R_{i,m,l,j}$ of vehicle V_i in VC_j, and

calculate the total downloading data rate for VC_j and RC. We use $R_{i,m,l}^{VC_j}$ to denote the transmission rate satisfying users request within VC_j, then:

$$R_{i,m,l}^{VC_j} = \left(1 - c_{i,l}^{VC_j}\right) \left\langle \sum_{n=1,n\neq i}^{N_{VC_j}} c_{i,l}^{VC_j} \right\rangle_0 R_{VV}(c_{i,l}^{VC_j} \neq 1), \tag{8}$$

where

$$\langle f(.)\rangle_0 = \begin{cases} 0, & \text{if } f(.) = 0, \\ 1, & \text{if } f(.) > 0. \end{cases} \tag{9}$$

In (8), $\left(1 - c_{i,l}^{VC_j}\right) \left\langle \sum_{n=1,n\neq i}^{N_{VC_j}} c_{i,l}^{VC_j} \right\rangle_0$ indicates the event that the desired content is not cached in V_i, but the content can be found in other VC_j members.

Similarly, if the request is not satisfied within VC_j and is forwarded to RC, then the transmission rate $R_{i,m,l}^{RC}$ is:

$$\begin{aligned} R_{i,m,l}^{RC} = c_{m,l}^{RC} R_{RV} + \left(1 - c_{m,l}^{RC}\right) \left\langle \sum_{k=1,k\neq m}^{N_{RC}} c_{m,l}^{RC} \right\rangle_0 R_{RCV} \\ + \left(1 - \left\langle \sum_{k=1}^{N_{RC}} c_{m,l}^{RC} \right\rangle_0\right) R_{PV}. \end{aligned} \tag{10}$$

Considering (8) and (10), the total download transmission rate is:

$$R_{i,m,l,j} = R_{i,m,l}^{VC_j} + \left(1 - \left\langle \sum_{n=1}^{N_{VC_j}} c_{i,l}^{VC_j} \right\rangle_0\right) R_{i,m,l}^{RC}, \tag{11}$$

where $\left(1 - \left\langle \sum_{n=1}^{N_{VC_j}} c_{i,l}^{VC_j} \right\rangle_0\right)$ indicates that the request fails within VC_j.

Next, we use $\delta_{i,m,l}^{VC_j}$ to indicate whether vehicle V_i in VC_j has completely obtained the content l from RSU_m, where:

$$\delta_{i,m,l}^{VC_j} = \begin{cases} 1, & \text{if the vehicle } V_i \text{ in } VC_j \text{ has completely} \\ & \text{obtained the requested content } l \text{ from } RSU_m, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

When the vehicle V_i in VC_j leaves the cover area of RSU_m, we let $D_{i,m,l}^{VC_j}$ to denote the cumulative download of the content l. We have:

$$\begin{aligned} D_{i,m,l}^{VC_1} &= \sum_{1,2,\cdots,m} R_{i,m,l,1} \frac{r}{v}, \\ D_{i,m,l}^{VC_2} &= \sum_{m,m+1,\cdots,N_{RC}} R_{i,m,l,2} \frac{r}{v}. \end{aligned} \tag{13}$$

Based on above analysis, we have:

$$\tau_{i,1,l} = \begin{cases} 0, & \text{if } C1, \\ \frac{S_q}{R_{i,m,l,1}}, & \text{if } C2, \\ \frac{r(m-1)}{v} + \frac{S_q - D_{i,m-1,l}^{VC_1}}{R_{i,m,l,1}}, & \text{if } C3, \\ \frac{rN_{RC}}{v} + \frac{S_q - D_{i,N_{RC},l}^{VC_1}}{R_{i,m,l}^{VC_1} + \left(1 - \left\langle \sum_{n=1}^{N_{VC_1}} c_{i,l}^{VC_1} \right\rangle_0\right) R_{PV}}, & \text{if } C4, \end{cases} \tag{14}$$

$$\begin{aligned} &C1: c_{i,l}^{VC_1} = 1; \quad C2: \delta_{i,m,l}^{VC_1} = 1, m = 1, c_{i,l}^{VC_1} \neq 1; \\ &C3: \delta_{i,m,l}^{VC_1} = 1, m \in \{2,3,\cdots N_{RC}\}, c_{i,l}^{VC_1} \neq 1; \\ &C4: \forall m \in \{1,2,3,\cdots N_{RC}\}, \delta_{i,m,l}^{VC_1} = 0, c_{i,l}^{VC_1} \neq 1. \end{aligned} \tag{15}$$

In (14) and (15), $C1$ means the vehicle can get the content in local cache, and the delay is 0. $C2$ means the vehicle complete the download from the first RSU it passed. In $C3$ situation, the vehicle cannot complete the download from the first RSU, but it can complete the download before the last RSU. $C4$ means when the vehicle passed the last RSU, the download still can not be completed and the unfinished content is obtained through the remote server. The analysis process of $\tau_{i,2,l}$ is similar to that of $\tau_{i,1,l}$.

E. Average Content Delivery Cost

The total average delivery cost can be calculated as:

$$Cost_{av} = \sum_{j=1}^{2} \sum_{l=1}^{L} \sum_{i=1}^{N_{VC_j}} \frac{\mu_j}{\mu_1 + \mu_2} \frac{p(l)}{N_{VC_j}} Cost_{i,j,l}, \tag{16}$$

when the vehicle V_i in VC_j passes RSU_m, the unit data transmission cost $Cost_{i,m,l,j}$ is:

$$\begin{aligned} Cost_{i,m,l,j} = &\left(1 - c_{i,l}^{VC_j}\right) \left\langle \sum_{n=1,n\neq i}^{N_{VC_j}} c_{i,l}^{VC_j} \right\rangle_0 P_{VV} \\ &+ \left(1 - \left\langle \sum_{n=1}^{N_{VC_j}} c_{i,l}^{VC_j} \right\rangle_0\right) [c_{m,l}^{RC} P_{RV} + \left(1 - c_{m,l}^{RC}\right) \\ &\left\langle \sum_{k=1,k\neq m}^{N_{RC}} c_{m,l}^{RC} \right\rangle_0 P_{RCV} + \left(1 - \left\langle \sum_{k=1}^{N_{RC}} c_{m,l}^{RC} \right\rangle_0\right) P_{PV}]. \end{aligned} \tag{17}$$

When the vehicle V_i in VC_j passes RSU_m, the total transmission cost $Cost_{i,m,l}^{VC_j}$ is

$$Cost_{i,m,l}^{VC_j} = \frac{r}{v} R_{i,m,l,j} Cost_{i,m,l,j}. \tag{18}$$

The transmission cost sum $Cost_{i,m,l,j}^{sum}$ can be calculated as:

$$\begin{aligned} Cost_{i,m,l,1}^{sum} &= \sum_{1,2,\cdots,m} Cost_{i,m,l}^{VC_1}, \\ Cost_{i,m,l,2}^{sum} &= \sum_{m,m+1,\cdots,N_{RC}} Cost_{i,m,l}^{VC_2}. \end{aligned} \tag{19}$$

To completely download the content q_l, the total cost of the vehicle V_i in VC_1 is:

$$Cost_{i,1,l} = \begin{cases} S_q Cost_{i,m,l,1}, & \text{if } C1, \\ Cost_{i,m-1,l,1}^{sum} + \left(S_q - D_{i,m-1,l}^{VC_1}\right) Cost_{i,m,l,1}, & \text{if } C2, \\ C4, & \text{if } C3, \end{cases} \tag{20}$$

where

$$C1: \delta_{i,m,l}^{VC_1}=1, m=1; \quad C2: \delta_{i,m,l}^{VC_1}=1, m\in\{2,3,\cdots N_{RC}\};$$
$$C3: \forall m\in\{1,2,3,\cdots N_{RC}\}, \delta_{i,m,l}^{VC_1}=0;$$
$$C4: Cost_{i,N_{RC},l,1}^{sum}+\left(S_q-D_{i,N_{RC},l}^{VC_1}\right)[\left(1-c_{i,l}^{VC_1}\right)$$
$$\left\langle\sum_{n=1,n\neq i}^{N_{VC_1}} c_{i,l}^{VC_1}\right\rangle_0 P_{VV}+\left(1-\left\langle\sum_{n=1}^{N_{VC_1}} c_{i,l}^{VC_1}\right\rangle_0\right)P_{PV}]. \tag{21}$$

The analysis process of $Cost_{i,2,l}$ is similar to that of $Cost_{i,1,l}$.

F. Average Content Delivery Hit Ratio

When the vehicle V_i in VC_j has completely downloaded the content l, we can get the number of hits and the total number of RSUs that V_i has passed. Let $hitratio_{i,l,j}$ denote their ratio, then the average hit ratio can be calculated as:

$$hitratio_{av}=\sum_{j=1}^{2}\sum_{l=1}^{L}\sum_{i=1}^{N_{VC_j}}\frac{\mu_j}{\mu_1+\mu_2}\frac{p(l)}{N_{VC_j}}hitratio_{i,l,j}. \tag{22}$$

G. Problem Formulation

In infotainment applications for VANETs, delay is a critical indicator of user experience to obtain desired content. Another key point is the cost. The caching scheme will affect delay and cost significantly. In this paper, we aim to make full use of the limited storage capacity of vehicles and RSUs to reduce the overall transmission delay and cost. Then the joint vehicle and RSU caching optimization problem can be formulated as:

$$\begin{aligned} P_C: \min \quad & U(\boldsymbol{C}^{VC_1},\boldsymbol{C}^{VC_2},\boldsymbol{C}^{RC}) \\ & s.t. \\ & \textstyle\sum_{l=1}^{L} c_{i,l}^{VC_1}\le S^V, c_{i,l}^{VC_1}\in\{0,1\}, i=1,2,\cdots N_{VC_1}, \\ & \textstyle\sum_{l=1}^{L} c_{i,l}^{VC_2}\le S^V, c_{i,l}^{VC_2}\in\{0,1\}, i=1,2,\cdots N_{VC_2}, \\ & \textstyle\sum_{l=1}^{L} c_{m,l}^{RC}\le S^R, c_{m,l}^{RC}\in\{0,1\}, m=1,2,\cdots N_{RC}, \\ & l=1,2,\cdots L. \end{aligned} \tag{23}$$

$$\begin{aligned} U(\boldsymbol{C}^{VC_1},\boldsymbol{C}^{VC_2},\boldsymbol{C}^{RC}) &= \alpha\frac{\tau_{av}}{\tau_{max}}+\beta\frac{Cost_{av}}{Cost_{max}} \\ &= \sum_{l=1}^{L}\sum_{j=1}^{2}\sum_{i=1}^{N_{VC_j}}\frac{\mu_j}{\mu_1+\mu_2}\frac{p(l)}{N_{VC_j}}(\alpha\frac{\tau_{i,j,l}}{\tau_{max}}+\beta\frac{Cost_{i,j,l}}{Cost_{max}}), \end{aligned} \tag{24}$$

where α and β are the weight parameters which can be adjusted according to user's preference. τ_{max} and $Cost_{max}$ are the maximum tolerable delay and download cost when the content is completely downloaded. In the worst case, the requested content can only be found in a remote server, and then $\tau_{max}=S_q/R_{PV}$, $Cost_{max}=S_qP_{PV}$. The constraint in (23) indicates that the content in each vehicle and RSU cache cannot exceed its capacity. In the objective function of (24), we use $\frac{\tau_{av}}{\tau_{max}}$ and $\frac{Cost_{av}}{Cost_{max}}$ for normalization.

Noting that problem P_C is a 0-1 NLIP problem, which is usually NP-hard and difficult to solve. The objective function of problem P_C (24) is very complicated, which makes problem P_C impossible to be handled by any convex relaxation techniques.[1] In the following sections, we will propose effective algorithms to solve it. The problem of finding the global optimal $\boldsymbol{C}^{VC_1}, \boldsymbol{C}^{VC_1}$ and $\boldsymbol{C}^{RC}$ can be treated as a future extension of this paper.

III. ADPS Joint Vehicle And RSU Caching Scheme

In this section, inspired by alternate convex search (ACS) algorithm [35] [36], an alternate dynamic programming search (ADPS) algorithm is proposed to solve problem P_C. At first we divide the original problem into three sub-problems. Then we solve each sub-problem using DP method. At the separation step, each time we choose two matrices and fix them, then a sub-problem is created. This strategy will generate three simplified sub-problems for problem P_C:

$$\begin{aligned} P_{C1}: \min \quad & U(\boldsymbol{C}_{fix}^{VC_1},\boldsymbol{C}_{fix}^{VC_2},\boldsymbol{C}^{RC}) \\ s.t. \quad & \textstyle\sum_{l=1}^{L} c_{m,l}^{RC}\le S^R, \quad c_{m,l}^{RC}\in\{0,1\}, \\ & m=1,2,\cdots N_{RC}, \quad l=1,2,\cdots L, \end{aligned} \tag{25}$$

$$\begin{aligned} P_{C2}: \min \quad & U(\boldsymbol{C}^{VC_1},\boldsymbol{C}_{fix}^{VC_2},\boldsymbol{C}_{fix}^{RC}) \\ s.t. \quad & \textstyle\sum_{l=1}^{L} c_{i,l}^{VC_1}\le S^V, \quad c_{i,l}^{VC_1}\in\{0,1\}, \\ & i=1,2,\cdots N_{VC_1}, \quad l=1,2,\cdots L, \end{aligned} \tag{26}$$

$$\begin{aligned} P_{C3}: \min \quad & U(\boldsymbol{C}_{fix}^{VC_1},\boldsymbol{C}^{VC_2},\boldsymbol{C}_{fix}^{RC}) \\ s.t. \quad & \textstyle\sum_{l=1}^{L} c_{i,l}^{VC_2}\le S^V, \quad c_{i,l}^{VC_2}\in\{0,1\}, \\ & i=1,2,\cdots N_{VC_2}, \quad l=1,2,\cdots L. \end{aligned} \tag{27}$$

Problem P_{C1}, P_{C2}, P_{C3} are still NLIP problem and have similar structure. Next, an optimal DP algorithm will be proposed to solve these three sub-problems separately.

A. Dynamic Programming Algorithm

DP is an effective algorithm to handle integer variables by breaking the original integer problem down into a multiple-stage decision problem, and solves the problem stage by stage. In particular, each stage is associated with multiple states, and the optimal decision in each stage is made based on the optimal decision at previous stage according to a recursive relationship. Thus, with the optimal decision at the first stage and the recursive relationship, the optimal decision can be made stage-by-stage and the optimal solution of original problem can be constructed. In what follows, we solve P_{C1} to illustrate the key steps of the proposed DP algorithm. These steps include reformulating problem P_{C1} as a multi-stage decision problem and finding the corresponding recursive relationship between adjacent stages. The DP algorithm is described in Algorithm 1.

1) *Multi-Stage Decision Problem Reformulation* : We divide problem P_{C1} into L stages and optimize the

[1]Specifically, the step function in (9) makes the problem hard for any convex relaxation techniques.

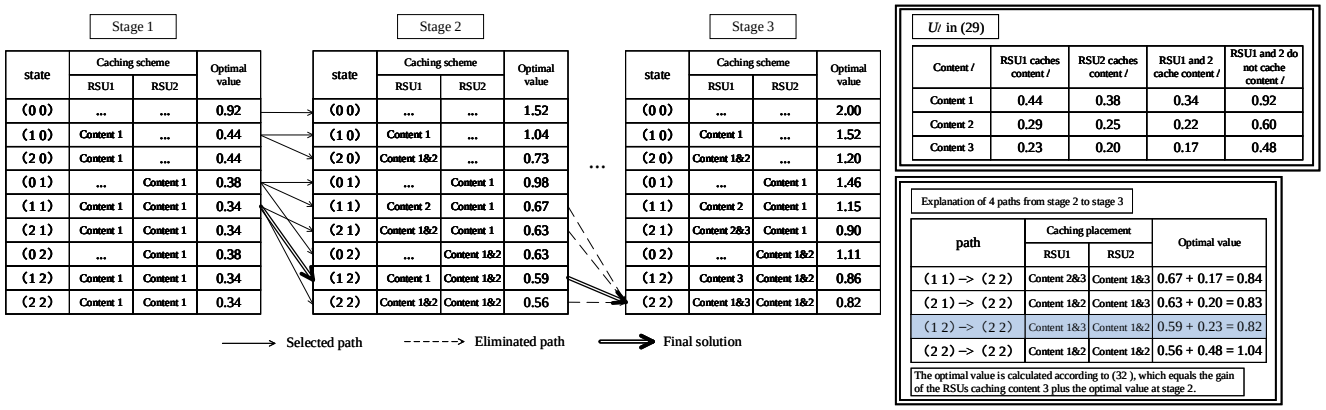

Fig. 3. An illustration of the proposed DP algorithm with $\boldsymbol{C}_{fix}^{VC_1} = \mathbf{0}$, $\boldsymbol{C}_{fix}^{VC_2} = \mathbf{0}$, $N_{RC} = 2$, $S^R = 2$ and $L = 3$. There are 3 stages, where the eliminated paths from stage 1 to stage 2 are omitted. The states at stage l represent the remaining caching capacity to store the first l contents. For example, state (1 2) at stage 2 means that RSU 1 caches content 1 or 2, and RSU 2 caches content 1 and 2 in total. The final solution first caches content 1 at both RSU 1 and 2, then caches content 2 at RSU 2, finally caches content 3 at RSU 1.

Algorithm 1 The DP Algorithm to Solve Problem P_{C1}

```
Initialization;
for all s_1 ≤ [S_m]^{N_RC×1} do
    J_1(s_1) = U_1(min{1, s_1});
end for
for l = 2 to L do
    for all s_l ≤ [S_m]^{N_RC×1} do
        Initialize J_l(s_l) as a large positive constant;
        for all possible c_l^RC do
            if J_l(s_l) > U_l(c_l^RC) + J_l(s_{l-1}) then
                J_l(s_l) = U_l(c_l^RC) + J_l(s_{l-1});
                c_l^{RC*} = c_l^RC
            end if
        end for
    end for
end for
return the optimal cache placement matrix C^{RC*}.
```

caching scheme for the first l contents at the l-th stage. $\boldsymbol{s}_l = [s_{m,l}]^{L\times 1}(0 \le s_{m,l} \le S^R)$ denotes the state at the l-th stage, which represents the remaining caching capacities of all RSUs for storing the first l contents (In P_{C1}, we only need to consider caching the contents in RSUs).

For state $\boldsymbol{s}_l$ at the l-th stage, the cache placement problem for the first l contents is expressed as:

$$\begin{aligned} P_{C1_l}: J_l(\boldsymbol{s}_l) = \min_{\{\boldsymbol{c}_n^{RC}, 1\le n\le l\}} \quad & U_n(\boldsymbol{c}_n^{RC}) \\ s.t. \quad & \textstyle\sum_{n=1}^{l} c_{m,n}^{RC} \le s_{m,l}, m = 1,2,\cdots N_{RC}, \\ & c_{m,n}^{RC} \in \{0,1\}, \forall m, 1 \le n \le l, \end{aligned} \tag{28}$$

where $\boldsymbol{c}_n^{RC} = [c_{m,n}^{RC}]^{N_{RC}\times 1}$ is the caching placement vector for content l, and

$$U_n(\boldsymbol{c}_n^{RC}) = \sum_{j=1}^{2}\sum_{i=1}^{N_{VC_j}} \frac{\mu_j}{\mu_1+\mu_2}\frac{p(n)}{N_{VC_j}}(\alpha\frac{\tau_{i,j,n}}{\tau_{max}} + \beta\frac{Cost_{i,j,n}}{Cost_{max}}). \tag{29}$$

For the last stage decision problem with $l = L$ and $s_{n,L} = S^R$, $\boldsymbol{s}_L = [S_m]^{N_{RC}\times 1} = [\mathbf{1}S^R]^{N_{RC}\times 1}$, problem P_{C1_L} is expressed as:

$$\begin{aligned} P_{C1_L}: J_L(\boldsymbol{s}_L) = & \\ \min_{\substack{\{\boldsymbol{c}_n^{RC},\\ 1\le n\le L\}}} \quad & \sum_{n=1}^{L}\sum_{j=1}^{2}\sum_{i=1}^{N_{VC_j}} \frac{\mu_j}{\mu_1+\mu_2}\frac{p(n)}{N_{VC_j}}(\alpha\frac{\tau_{i,j,n}}{\tau_{max}} + \beta\frac{Cost_{i,j,n}}{Cost_{max}}) \\ s.t. \quad & \textstyle\sum_{n=1}^{L} c_{m,n}^{RC} \le S_m, m = 1,2,\cdots N_{RC}, \\ & c_{m,n}^{RC} \in \{0,1\}, \forall m, 1 \le n \le L. \end{aligned} \tag{30}$$

It is easy to observe that problem P_{c1_L} is equivalent to the original problem P_{C1}. Next we will develop a recursive relationship between adjacent stage problems, the optimal decision of problem P_{C1_l} can be achieved stage-by-stage. Finally, the original problem P_{C1} is optimally solved.

2) *Recursive Relationship* : At the first stage, the optimal decision of RSU_m is to cache the first content if $s_{m,1} \ge 1$. Mathematically, $\boldsymbol{s}_1 = min\{\mathbf{1}, \boldsymbol{s}_1\}$. Thus, we have:

$$J_1(\boldsymbol{s}_1) = U_1(min\{\mathbf{1}, \boldsymbol{s}_1\}). \tag{31}$$

Then, the optimal decision at the l-th stage $(2 \le l \le L)$ is made based on the decision at $l-1$-th stage by using the following recursive relationship, and an illustration is shown in Fig. 3.

$$\begin{aligned} J_l(\boldsymbol{s}_l) = \min_{\{\boldsymbol{c}_l^{RC}\}} \quad & U_l(\boldsymbol{c}_l^{RC}) + J_l(\boldsymbol{s}_{l-1}) \\ s.t. \quad & c_{m,l}^{RC} \in \{0,1\}, \\ & c_{m,l}^{RC} \le s_{m,l}, \\ & m = 1,2,\cdots N_{RC}, \end{aligned} \tag{32}$$

where $\boldsymbol{s}_{l-1} = \boldsymbol{s}_l - \boldsymbol{c}_l^{RC}$ is the state transition equation between adjacent stages.

Specifically, for state $\boldsymbol{s}_l$ at the l-th stage, the optimal decision of $\boldsymbol{c}_l^{RC}$ can be achieved by exhaustively searching and finding the optimal one with the minimum value of $U_l(\boldsymbol{c}_l^{RC}) + J_l(\boldsymbol{s}_{l-1})$, where $J_l(\boldsymbol{s}_{l-1})$ has been obtained at the $l-1$-th stage. Therefore, with $J_1(\boldsymbol{s}_1)$ and the recursive relationship (32), problem P_{C1} can be optimally solved.

3) *Complexity Analysis* : According to Algorithm 1, the computational complexity of stage 1 (i.e., line2-line4) is $O((S^R+1)^{N_{RC}})$. For stage l $(1 < l \leq L)$, each RSU has S^R+1 states, and each state needs to decide whether to cache the content l except for the 0 state, the computational complexity of an RSU is $O(2S^R+1)$. So the computational complexity of N_{RC} RSUs (i.e., line6-line14) is $O((2S^R+1)^{N_{RC}})$. Hence, the whole complexity is $O((S^R+1)^{N_{RC}})+O((L-1)(2S^R+1)^{N_{RC}})$, i.e., $O((L-1)(2S^R+1)^{N_{RC}})$.

B. ADPS Joint Vehicle and RSU Caching Scheme

Problem P_{C1}, P_{C2}, P_{C3} can be solved using DP algorithm separately. Then for problem P_C, an ADPS caching scheme is proposed in this paper. The detail of the algorithm to get ADPS scheme is shown in Algorithm 2.

Algorithm 2 ADPS Joint Vehicle and RSU Caching Scheme

Input: $Acc, Maxiter, \boldsymbol{C}_0^{VC_1}, \boldsymbol{C}_0^{VC_2}, \boldsymbol{C}_0^{RC}$;
Output: $\boldsymbol{C}_j^{VC_1}, \boldsymbol{C}_j^{VC_2}, \boldsymbol{C}_j^{RC}$;
1: $i = 1$
2: $\Delta = \infty$
3: **while** $(\Delta > Acc)$ Λ $(i < Maxiter)$ **do**
4: $\boldsymbol{C}_i^{RC} = \arg\min_{\boldsymbol{C}^{RC}} U(\boldsymbol{C}_{i-1}^{VC_1}, \boldsymbol{C}_{i-1}^{VC_2}, \boldsymbol{C}^{RC})$ [Slove problem P_{C1} with algorithm 1]
5: $\boldsymbol{C}_i^{VC_1} = \arg\min_{\boldsymbol{C}^{VC_1}} U(\boldsymbol{C}^{VC_1}, \boldsymbol{C}_{i-1}^{VC_2}, \boldsymbol{C}_i^{RC})$ [Slove problem P_{C2} with DP algorithm]
6: $\boldsymbol{C}_i^{VC_2} = \arg\min_{\boldsymbol{C}^{VC_2}} U(\boldsymbol{C}_i^{VC_1}, \boldsymbol{C}^{VC_2}, \boldsymbol{C}_i^{RC})$ [Slove problem P_{C3} with DP algorithm]
7: $\Delta = \left| U(\boldsymbol{C}_i^{VC_1}, \boldsymbol{C}_i^{VC_2}, \boldsymbol{C}_i^{RC}) - U(\boldsymbol{C}_{i-1}^{VC_1}, \boldsymbol{C}_{i-1}^{VC_2}, \boldsymbol{C}_{i-1}^{RC}) \right|$
8: $i = i + 1$
9: **end while**
10: $I_{max} = i$; // Total number of iterations;
11: $j = \arg\min_{n=1,2,\cdots i-1} U(\boldsymbol{C}_n^{VC_1}, \boldsymbol{C}_n^{VC_2}, \boldsymbol{C}_n^{RC})$
12: **return** $\boldsymbol{C}_j^{VC_1}, \boldsymbol{C}_j^{VC_2}, \boldsymbol{C}_j^{RC}$

The procedure of ADPS algorithm is to alternately solve problem P_C by fixing two caching matrix variables among all the three variables. The iteration process continues until the objective function converges. It should be noted that different initial points might lead to different convergence points. We set $\boldsymbol{C}_0^{VC_1}$, $\boldsymbol{C}_0^{VC_2}$ and $\boldsymbol{C}_0^{RC}$ as zero matrices and let them be the initial point, which means that no content is cached. The algorithm ends when the performance of U becomes stable after I_{max} iterations. It should be noted that, given all the parameters, the algorithm is quite effective that U becomes stable within $I_{max} = 2$ iterations.

Complexity Analysis : According to Algorithm 2, since the computational complexity of the DP algorithm has been analyzed, the computation complexity using DP algorithm in line 4-6 are $O((L-1)(2S^R+1)^{N_{RC}})$, $O((L-1)(2S^V+1)^{N_{VC1}})$ and $O((L-1)(2S^V+1)^{N_{VC2}})$ respectively. So the whole complexity of ADPS algorithm is $O(I_{max}(L-1)((2S^R+1)^{N_{RC}}+(2S^V+1)^{N_{VC1}}+(2S^V+1)^{N_{VC2}}))$, which increases exponentially with the number of vehicles or RSUs.

IV. Low-Complexity Cooperation-based Greedy Caching Scheme

The computational complexity of ADPS algorithm for Pc increases exponentially with the number of vehicles or RSUs. For large scale problem, in this section, we design a more practical cooperation-based greedy algorithm (CBG) as a complement to ADPS.

A. Cooperation-Based Greedy Caching Scheme

As shown in Fig. 2, a specific vehicle request can be fulfilled by either a specific VC or RC member. If we let the VC members and RC members cooperate with each other, then a specific content only need to be cached in one of VC or RC members, and this content can be fetched by a requesting vehicle through the relay of some VC and RC members. In this situation, the limited storage space could be fruitfully exploited to improve the network hit ratio.

In Table II, we show the results of the caching scheme obtained by using the ADPS algorithm in a small-scale scenario. We set $L = 25$, $S^V = 1$, $S^R = 3$, $S_q = 150$, $\gamma = 0.6$, $N_{VC_1} = 5$ and $N_{VC_2} = 6$, other parameter settings are the same as in Section V. From this result, we can observe that, with the increase of the number of RSUs N_{RC}, VC and RC start caching from the most popular content, and each member caches different content. This caching scheme makes full use of the cache space, so that the requesting vehicle can obtain more content on the edge server. Inspired by this observation and the above analysis, and in order to reduce complexity, we propose the CBG caching scheme to use the idea of mutual cooperation among VC and RC members. Here greedy means the decision of which member the content is cached depends on whether can obtain the lowest objective function value. The algorithm to get CBG scheme is described in Algorithm 3. At first, we start caching the most popular content. For content l, to decide which member of VC or RC to cache it, we consider the cooperative caching behavior of all nodes, calculate an objective value in line 3-5, and select the node whose situation has the optimal value until the vehicle or RSU's cache is exhausted. For vehicles in the VC, we do not need to consider the vehicle number sequence. The vehicle requesting the content can cooperate with the VC member anywhere, which has no effect on the objective value. For RSU, which RSU caches the same content has an impact on the value of the object. We use Fig. 4 to illustrate the concept of line 3-5 in Algorithm 3.

B. Complexity Analysis

In Algorithm 3, the computation complexity is $O(2(S^V * max\{N_{VC_1}, N_{VC_2}\} + S^R * N_{RC}))$, where $S^V * max\{N_{VC_1}, N_{VC_2}\} + S^R * N_{RC}$ is the number of loops. It is linear with respect to the number of nodes.

V. Numerical Simulation Results

In this section, we evaluate the performance of the proposed caching scheme through numerical simulation. Note that both

TABLE II
ADPS CACHING SCHEME WITH $L = 25$, $S^V = 1$, $S^R = 3$, $\gamma = 0.85$, $S_q = 150$, $N_{VC_1} = 5$ AND $N_{VC_2} = 6$.

Caching matrix ⋆	$N_{RC} = 2$	$N_{RC} = 3$	$N_{RC} = 4$	$N_{RC} = 5$
$[\boldsymbol{C}^{VC_1}]^T$	$\left[\begin{array}{ccccc} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \\ \hline \multicolumn{5}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{ccccc} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \\ \hline \multicolumn{5}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{ccccc} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \\ \hline \multicolumn{5}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{ccccc} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \\ \hline \multicolumn{5}{c}{\mathbf{0}} \end{array}\right]$
$[\boldsymbol{C}^{VC_2}]^T$	$\left[\begin{array}{cccccc} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&0 \\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots \\ 0&0&0&0&0&1 \\ \hline \multicolumn{6}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{cccccc} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&0 \\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots \\ 0&0&0&0&0&1 \\ \hline \multicolumn{6}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{cccccc} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&0 \\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots \\ 0&0&0&0&0&1 \\ \hline \multicolumn{6}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{cccccc} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ \hline \multicolumn{6}{c}{\mathbf{0}} \end{array}\right]$
$[\boldsymbol{C}^{RC}]^T$	$\left[\begin{array}{cc} \multicolumn{2}{c}{\mathbf{0}} \\ \hline 1&0 \\ 1&0 \\ 1&0 \\ 0&1 \\ 0&1 \\ 0&1 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ \hline \multicolumn{2}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{ccc} \multicolumn{3}{c}{\mathbf{0}} \\ \hline 1&0&0 \\ 1&0&0 \\ 1&0&0 \\ 0&1&0 \\ 0&1&0 \\ 0&1&0 \\ 0&0&1 \\ 0&0&1 \\ 0&0&1 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ \hline \multicolumn{3}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{cccc} \multicolumn{4}{c}{\mathbf{0}} \\ \hline 1&0&0&0 \\ 1&0&0&0 \\ 1&0&0&0 \\ 0&1&0&0 \\ 0&1&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&1&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 0&0&0&1 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ \hline \multicolumn{4}{c}{\mathbf{0}} \end{array}\right]$	$\left[\begin{array}{ccccc} \multicolumn{5}{c}{\mathbf{0}} \\ \hline 1&0&0&0&0 \\ 1&0&0&0&0 \\ 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&1&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&1&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&1&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \\ 0&0&0&0&1 \\ 0&0&0&0&1 \\ \hline \multicolumn{5}{c}{\mathbf{0}} \end{array}\right]$
U	0.9134	0.7951	0.7217	0.6752

⋆ The rows of all matrices in this table represent content IDs, and **0** represents zero matrix. As N_{RC} increases, the content cached in VC remains almost unchanged. Extra contents are cached by the newly added RSU.

the proposed two caching schemes in this paper are implemented in a centralized way by BS. The optimized two-layer cache placement indicators are distributed by BS to each RSU and VCC, then by VCC to each vehicle. In the simulation, the throughput of entrance 1/2 is set to be [40,36]/min. The size of each content is set to 150 Mb. The coverage range for each RSU is set to 200 meters, and the average speed of vehicle is set to 20 m/s. The transmission rate to transmit content from a vehicle/RSU/RC member/CP to the target vehicle is set to [8,6,4,2] Mbps. The price for downloading unit data is set to [1,4,6,10]. In addition, without loss of generality, we set equal weights of delay and cost in the objective function, i.e. $\alpha = \beta = 1$.

In addition to ADPS and CBG caching schemes, five other caching schemes are also considered for comparison.

- $Popularity\text{-}Based\ Caching\ Scheme$ $(PoBCS)$: Each caching node caches the most popular contents until their storage spaces run out. No coordination is considered.
- $Probability\text{-}Based\ Caching\ Scheme$ $(PrBCS)$: Each caching node caches a particular content with a certain

Algorithm 3 Cooperation-Based Greedy Caching Scheme

Input: $\boldsymbol{C}_0^{VC_1}, \boldsymbol{C}_0^{VC_2}, \boldsymbol{C}_0^{RC}$;
Output: $\boldsymbol{C}_{l_{max}-1}^{VC_1*}, \boldsymbol{C}_{l_{max}-1}^{VC_2*}, \boldsymbol{C}_{l_{max}-1}^{RC*}$;
1: The maximum caching capacity of all vehicles and RSUs is $l_{max} = S^V * max\{N_{VC_1}, N_{VC_2}\} + S^R * N_{RC}$;
2: **for** l = 1 to l_{max} **do**
3: Select vehicles i, j from VC_1, VC_2 and RSU k with extra storage space;
4: In order to cooperate with each other, if the vehicle in the VC caches content l, there is no need to cache content l in the RSU k, and if none of the vehicles in the VC cache content l, then the content l needs to be cached in the RSU k;
5: Considering all the cooperating cache situations, solve (24) and get the optimal $\boldsymbol{C}_{l-1}^{VC_1*}, \boldsymbol{C}_{l-1}^{VC_2*}, \boldsymbol{C}_{l-1}^{RC*}$;
6: **end for**
7: **return** $\boldsymbol{C}_{l_{max}-1}^{VC_1*}, \boldsymbol{C}_{l_{max}-1}^{VC_2*}, \boldsymbol{C}_{l_{max}-1}^{RC*}$

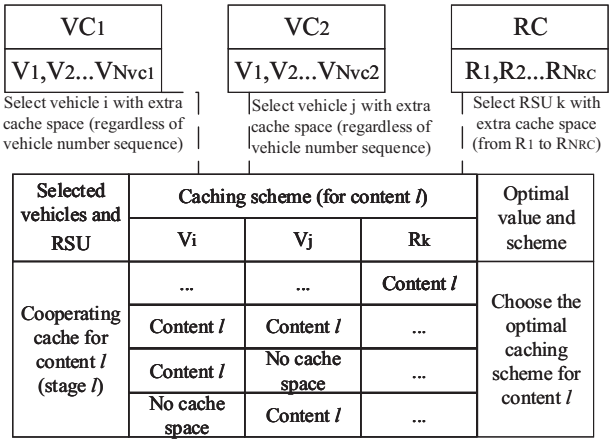

Fig. 4. Conceptual illustration of line 3-5 in Algorithm 3.

probability, and this probability is related to the popularity of the content. The higher probability, the more likely the content is to be stored by vehicles and RSUs.

- *Random-Based Caching Scheme* (*RBCS*): Each caching node determines its cached contents randomly, without considering popularity or other parameters.
- *No Caching* : Each caching node does not consider caching any content.
- *Low Latency Caching Placement* (*LLCP*): This caching scheme is proposed in [19] to minimize the average latency for cloud-based VANETs while satisfying the QoS requirements of vehicles, and the problem is solved effectively via simulated anneal (SA) algorithm. The SA algorithm is also applied to our scheme for comparison. The performance of SA algorithm is related to the initial value.

A. The Effectiveness of Proposed CBG Caching Scheme

The effectiveness of proposed CBG caching scheme is demonstrated in this subsection. Both the performance in terms of delay, cost, hit ratio and the computational time of all caching schemes are compared. Because the computational

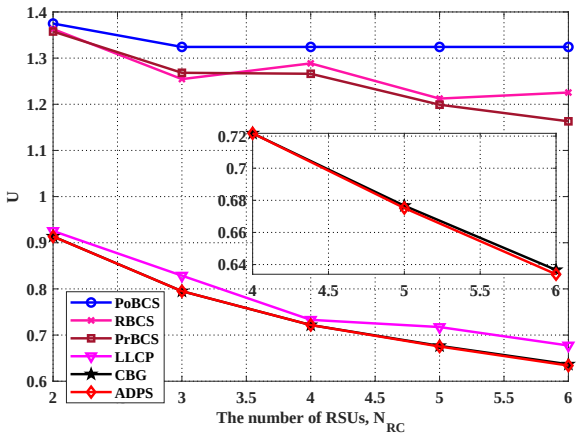

Fig. 5. U versus the number of RSUs with $L = 25, S^V = 1, S^R = 3, \gamma = 0.85, S_q = 150, N_{VC_1} = 5$ and $N_{VC_2} = 6$.

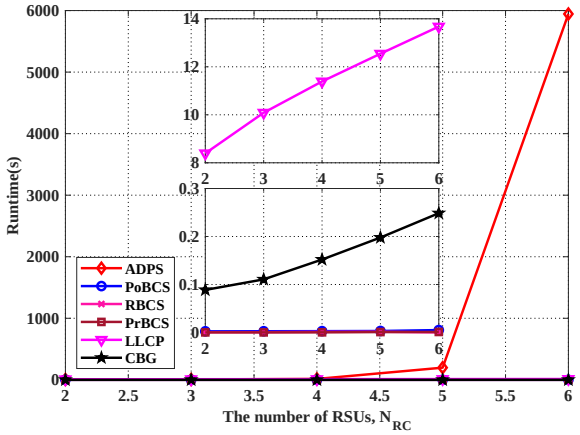

Fig. 6. The runtime of caching schemes versus the number of RSUs with $L = 25, S^V = 1, S^R = 3, \gamma = 0.85, S_q = 150, N_{VC_1} = 5$ and $N_{VC_2} = 6$.

complexity of ADPS algorithm is high, we use a small network scenario. We set $L = 25$, $S^V = 1$, $S^R = 3$, $\gamma = 0.85$, $N_{VC_1} = 5$ and $N_{VC_2} = 6$. Other parameters are the same as before.

Fig. 5 shows how the number of RSUs affects U in (24) for different caching schemes. U keeps decreasing when the number of RSUs increases, except for RBCS and PoBCS. The reason is that, when the number of RSUs increases, the total caching capacity in the network becomes larger and vehicles have more chances to acquire interested contents. For PoBCS, although the total storage space has increased, each RSU caches the same popular content, resulting in non-cooperation between RSUs. When a vehicle requests other content, it can only be obtained from the remote server and cannot be obtained from the RSUs. For RBCS, each RSU determines its cached contents randomly, without considering popularity or other parameters. Due to randomness, the performance of RBCS is quite unstable. Also, both of ADPS and CBG caching schemes outperform other caching schemes, owing to exploiting of mutual cooperation between members in VC and RC. Most importantly, the proposed CBG caching scheme can achieve almost the same performance with ADPS caching scheme.

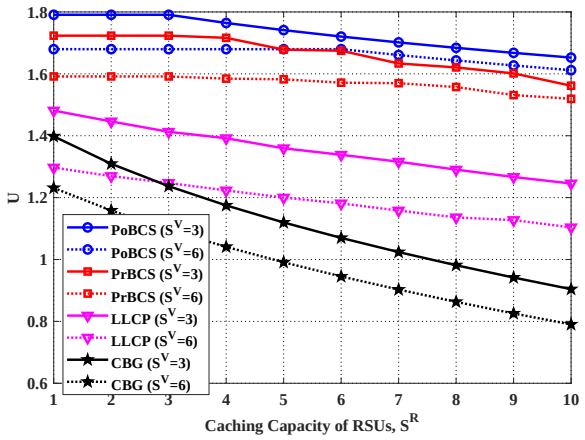

Fig. 7. U versus caching capacity with $L = 250, \gamma = 0.6, S_q = 150, N_{VC_1} = 5, N_{VC_2} = 6$ and $N_{RC} = 20$.

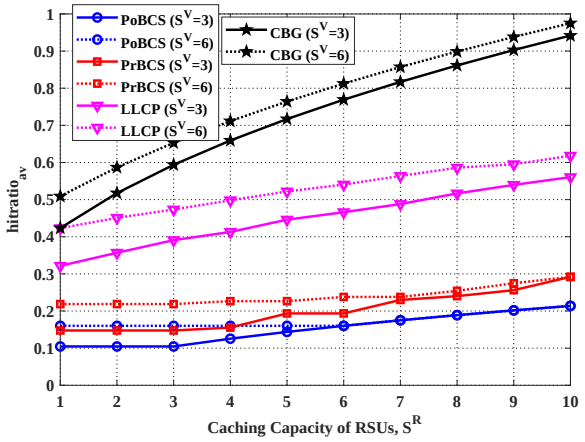

Fig. 9. The average hit ratio versus caching capacity with $L = 250, \gamma = 0.6, S_q = 150, N_{VC_1} = 5, N_{VC_2} = 6$ and $N_{RC} = 20$.

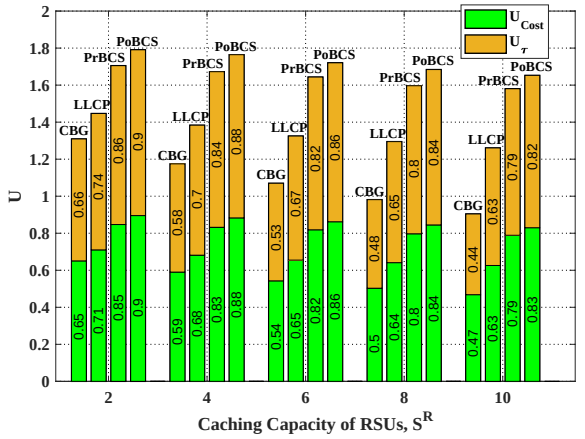

Fig. 8. U versus caching capacity with $L = 250, \gamma = 0.6, S_q = 150, S^V = 3, N_{VC_1} = 5, N_{VC_2} = 6$ and $N_{RC} = 20$.

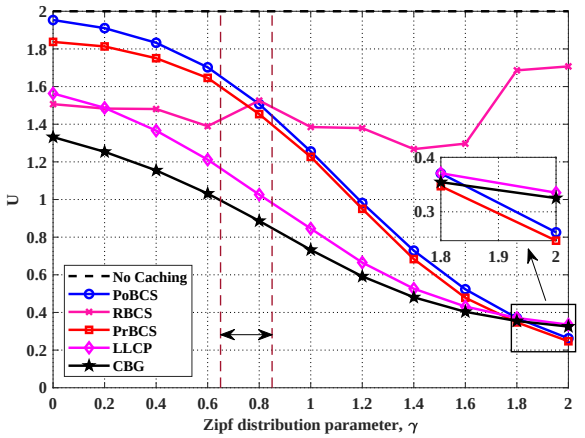

Fig. 10. U versus Zipf distribution parameter with $L = 250, S_q = 150, S^V = 2, S^R = 8, N_{VC_1} = 5, N_{VC_2} = 6$ and $N_{RC} = 20$.

Fig. 6 shows the runtime of all caching schemes on a desktop computer with a 3.20 GHz Intel Core i7 processor and 16 GB installed memory. It is straightforward that the runtime of ADPS caching scheme increases exponentially with the increase of N_{RC}. The CBG caching scheme and LLCP increase almost linearly with respect to N_{RC}, which matches the computation complexity analysis in Section IV. Moreover, with the almost same delay and cost performance, the runtime of CBG caching scheme is much lower than the ADPS caching scheme. For example, when $N_{RC} = 6$, the runtime of the CBG caching scheme is 0.25s, while that of ADPS caching scheme' is 5945s being prohibitively long. This phenomenon shows the effectiveness of the proposed CBG caching scheme. Therefore, in what follows, only the proposed well-salable CBG caching scheme is adopted for the performance comparison in large scale scenarios.

B. The Impact of Network Parameters on Caching Schemes

Fig. 7 shows how the caching capacity of vehicle and RSU affects U of different caching schemes. U decreases when the caching capacity S^R and S^V increase. The reason is that, with enlarged capacity, more and more contents will be stored by VC and RC members and then successfully delivered with faster transmission rate and lower price, resulting in lower delay and cost. We use U_τ and U_{Cost} to express the first part $\alpha\frac{\tau_{av}}{\tau_{max}}$ and the second part $\beta\frac{Cost_{av}}{Cost_{max}}$ in (24), their changes with caching capacity S^R are shown in Fig. 8. It can be clearly seen that the average delay and average cost continue to decrease as the caching capacity of RSUs increases, resulting in an overall decrease in U. In addition, the proposed CBG caching scheme outperforms other caching schemes, the average content delivery delay and cost can be reduced by 30% and 25% respectively, and the overall reduction is 29% with $S^R = 10$, as compared to LLCP scheme. This is due to the fact that the CBG caching scheme makes full use of partnership, storage space and content popularity information.

The impact of caching capacity on average hit ratio is investigated in Fig. 9. As expected, the average hit ratio becomes higher with larger caching capacity. Moreover, the proposed CBG caching scheme achieves higher hit ratio than other caching schemes. The reason is that, VC members and RC members cooperate with each other to avoid caching the same contents between them, and more contents in the limited cache space can be cached. An interesting phenomenon that needs to be noted about PoBCS is in Fig. 7 and Fig. 9. It is that when the vehicle caching capacity S^V is less than the

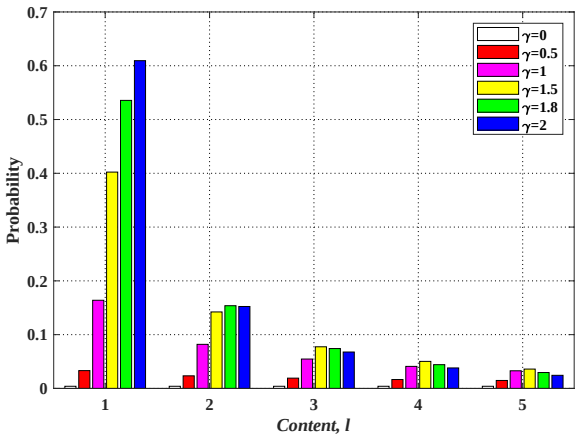

Fig. 11. The probability of request for the top 5 most popular content with $L = 250$.

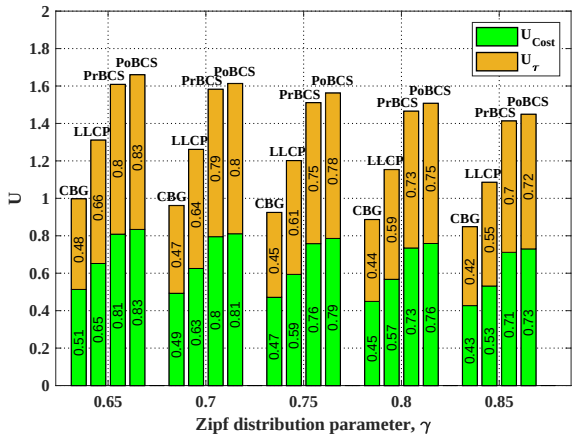

Fig. 12. U versus Zipf distribution parameter with $L = 250, S_q = 150, S^V = 2, S^R = 8, N_{VC_1} = 5, N_{VC_2} = 6$ and $N_{RC} = 20$.

RSU caching capacity S^R, U and the average hit ratio remain unchanged. This is because that PoBCS caching the same popular content which leads to non-cooperation between VC members and RC members. Only after the caching capacity of RSU exceeds the caching capacity of vehicle, after caching other popular content, these contents can be obtained from RSU instead of the remote server. Therefore, our proposed CBG caching scheme has significant performance advantage with limited caching capacity, which is the common case in practice.

Fig. 10 shows how the parameter of Zipf distribution, γ affects U of different caching schemes. In Fig. 10, U is decreasing with the increase of γ except for RBCS and No Caching scheme. The reason is that the percentage of requests for popular content goes up when γ increases. As a result, the cache of vehicle and RSU may provide content and satisfy more requests when the percentage of requests for popular content increases. No Caching scheme does not consider caching technique, and all the contents are fetched from the CPs with maximum delay and cost. In RBCS, every vehicle and RSU determines its cached contents randomly, without considering popularity or other parameters, thus its performance is independent of γ. Also, due to randomness,

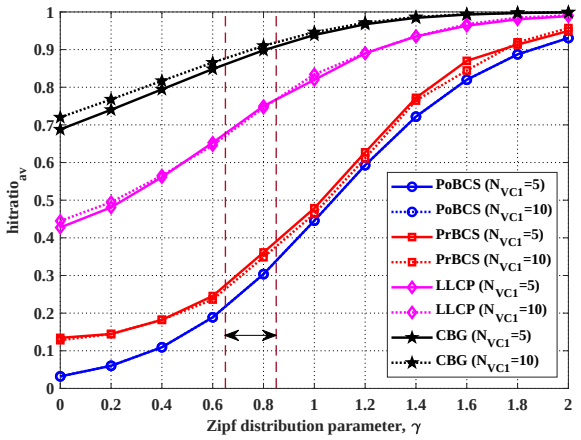

Fig. 13. The average hit ratio versus Zipf distribution parameter with $L = 250, S_q = 150, S^V = 2, S^R = 8, N_{VC_2} = 6$ and $N_{RC} = 20$.

the performance of RBCS is quite unstable, but always outperforms No Caching scheme, which partially indicates the gain achieved by applying caching. In addition, the proposed CBG caching scheme outperforms other caching schemes when γ is changed from 0 to 1.6. However, PoBCS and PrBCS perform better than CBG caching scheme, when $\gamma >= 1.8$. We can find the reason in Fig. 11. The probability of request for the top 5 most popular content can be seen in Fig. 11. We can see that the most popular content request's probability is above 0.5 and that of others are all below 0.2, when $\gamma = 1.8$ and $\gamma = 2$. The reason is that the most popular content has the highest request probability compared to other contents when γ is changed from 1.8 to 2. Therefore, it is worth caching the most popular content, just like the PoBCS and PrBCS, for high performance gains.

Existing studies [16] [34] show that γ may change from 0.65 to 0.85 according to different situations of networks. Fig. 12 shows how γ ($0.65 \leq \gamma \leq 0.85$) affects the average content delivery delay and cost of different caching schemes in a more realistic situation. With γ increasing, a large proportion of requests arises for the popular content. With the proposed CBG caching scheme, the content can be kept in cache of vehicle and RSU with the consideration of change in content popularity. As the requesting vehicle does not need to fetch the most popular content from other places with high delay and cost, the average content delivery delay and cost can be reduced, resulting in an overall decrease in U. From Fig. 12, our proposed CBG caching scheme also outperforms PrBCS, PoBCS and LLCP in terms of U. It can be observed that, when the proposed CBG caching scheme is adopted, the average delay and cost can be reduced by 24% and 19% respectively, and the overall reduction is 21% with $\gamma = 0.85$, as compared to LLCP scheme.

Fig. 13 shows the impact of γ and the number of vehicles in VC on average hit ratio for different caching schemes. The result demonstrates that CBG caching scheme can obtain the highest average hit ratio. Besides, the average hit ratio keeps increasing when γ increases in the aforementioned schemes. When the size of platoon (i.e., N_{VC_1}) increases, the performance of PoBCS when $N_{VC_1} = 5$ and $N_{VC_1} = 10$ totally coincide, and this indicates PoBCS is non-cooperative

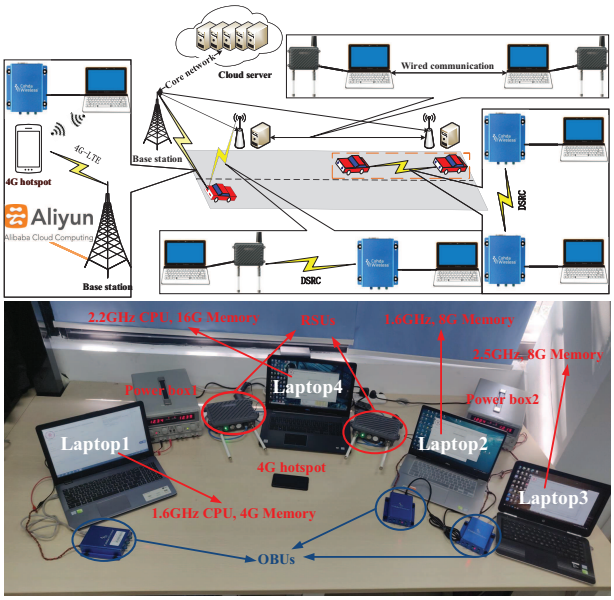

Fig. 14. Prototype implementation.

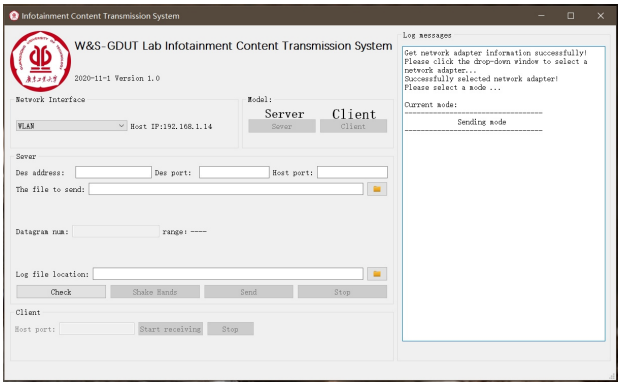

Fig. 15. System interface.

in nature. For the PrBCS and LLCP when $N_{VC_1} = 5$ and $N_{VC_1} = 10$, the increase in average hit ratio is not obvious, The proposed CBG caching scheme can make full use of the partnership, storage capacity, and content popularity information to increase the average hit ratio when γ is changed from 0 to 1. The average hit ratio is mainly affected by request probability rather than the size of platoon when γ is changed from 1 to 2, which is related to the definition of hit ratio in (22). It can be observed that, when the proposed CBG caching scheme is adopted, the average hit ratio can reach over 85% with $\gamma = 0.65 \sim 0.75$.

VI. Test Bed Evaluation

In this section, we design and implement an infotainment content transmission (ICT) system based on the proposed two-layer architecture.

A. System Prototype Implementation

As shown in Fig. 14, we built a small scale VANETs model using commercial OBU and RSU devices. We use Aliyun cloud server as the content provider, which is configured

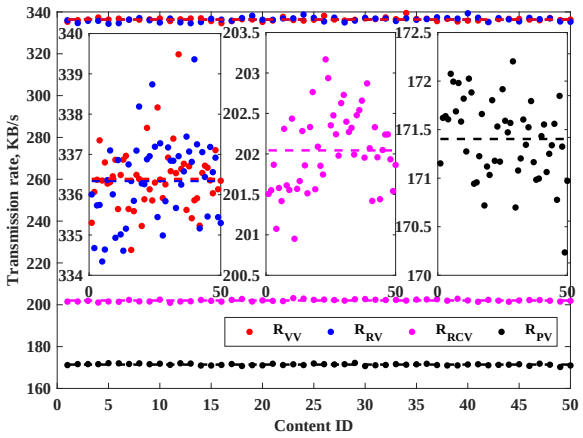

Fig. 16. Transmission rate in real-world data set.

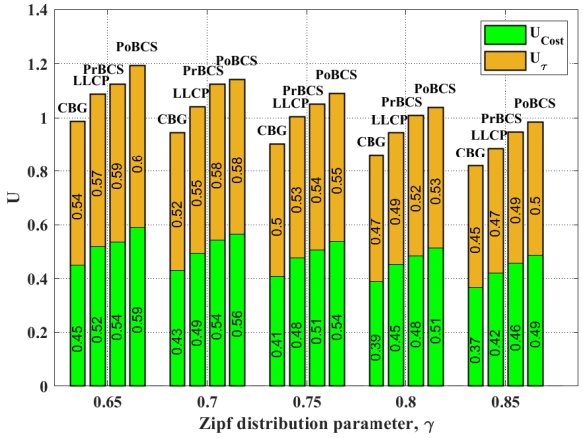

Fig. 17. U versus Zipf distribution parameter in real-world data set with $L = 50, S^V = 5, S^R = 10, N_{VC_1} = 1, N_{VC_2} = 2$ and $N_{RC} = 2$.

with 2.5 GHz CPU and 4G memory. Node vehicle 1 (V1) consists of a Cohda Wireless MK5 OBU [37] and laptop called Laptop1. Laptop1 has 1.6 GHz CPU and 4G memory. Similarly, V2 has 1.6 GHz CPU and 8G memory. V3 has 2.5 GHz CPU and 8G memory. Meanwhile, Laptop4 is connected with two RSUs, with the configuration of 2.2 GHz CPU and 16G memory. V1, V2 and V3 can communicate with RSUs and cloud server through DSRC interface and 4G hot-spot respectively. In addition, we made a specific database containing a video data-set [38]. This data-set consists 50 video clips from 50 Hollywood movies with size of 8968-9790 KB. We program each node so that they can perform the content requesting and caching function. As shown in Fig. 15, the node can be programmed as server or client in terms of requirements. Each laptop in Fig. 14 is able to communicate with RSUs and OBUs via User Datagram Protocol (UDP) [39].

B. Performance of Transmission Rate in Test Bed.

In our experiment setting, we test the real content transmission rate $R_{VV}, R_{RV}, R_{RCV}, R_{PV}$ based on the video data-set. Fig. 16 shows different transmission rates in real vehicular communication environments. The average transmission rate of R_{VV} is very close to R_{RV}, this is caused by the same internal structure of Cohda Wireless MK5 RSU and OBU.

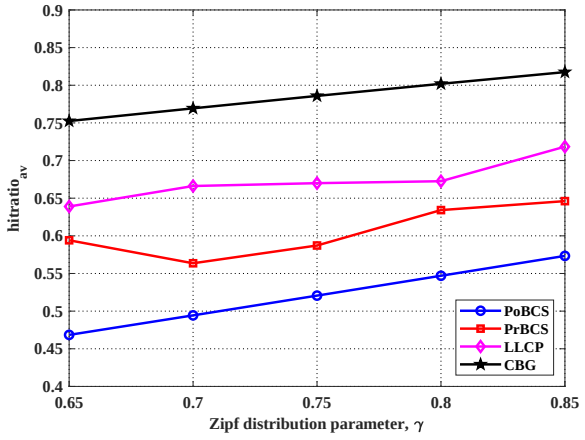

Fig. 18. The average hit ratio versus Zipf distribution parameter in real-world data set with $L = 50, S^V = 5, S^R = 10, N_{VC_1} = 1, N_{VC_2} = 2$ and $N_{RC} = 2$.

In addition, other average transmission rates conform to the rate assumption of the two-layer structure. The actual test data obtained by implementing the system prototype can verify the effectiveness of different caching schemes.

C. Performance of Caching Schemes in Real-World Data Set

We verify the effectiveness of our proposed caching scheme through real-world data set (i.e., transmission rate, video size) recorded by the test bed. The residence time of each vehicle with each real RSU is set to 10s to imitate the real movement of the vehicle. The performance of different caching schemes using real data set is shown in Fig. 17 and Fig. 18. It is straightforward that the average content delivery delay and cost are decreasing. Average hit ratio increases as γ increases. When the proposed CBG caching scheme is adopted, the average delay and cost can be reduced by 10% and 24% respectively, and the average hit ratio increases 30% with $\gamma = 0.85$, as compared with PoBCS. As expected, cooperation between different nodes plays a key role in the caching scheme design, the proposed CBG caching scheme extremely outperforms other caching schemes in the actual environment, which effectively shows the advantages of our proposed CBG caching scheme.

VII. Conclusion

In this paper, we expand the single-layer caching scheme to two-layers. The average content delivery delay, cost and hit ratio have been analytically obtained for a two-layer cloud based VANET architecture in content caching scheme. Then, a joint vehicle and RSU caching optimization problem has been formulated to reduce both the transmission delay and cost considering limited caching capacity. By solving this optimization problem, we propose ADPS caching scheme and low complexity CBG caching scheme. Numerical simulation together with real test bed verification have shown the advantage of our proposed caching scheme. Specifically, the proposed CBG caching scheme consumes remarkably shorter runtime than ADPS caching scheme, test bed evaluation results show that CBG caching scheme reduces delay and cost by 10% and 24% respectively, and increases hit ratio by 30% as compared to PoBCS.

The analytical framework established in this paper can be extended to study general cases of other QoS metrics. For the future work, the prototype system will be further evolved from the current hardware testing to small scale realistic internet of vehicle environments.

References

[1] K. Xiong, Y. Zhang, P. Fan, H. Yang, and X. Zhou, "Mobile service amount based link scheduling for high-mobility cooperative vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9521–9533, Oct. 2017.

[2] R. Sun, Y. Wang, L. Lyu, N. Cheng, S. Zhang, T. Yang, and X. Shen, "Delay-oriented caching strategies in d2d mobile networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8529–8541, May 2020.

[3] R. Sun, Y. Wang, N. Cheng, L. Lyu, S. Zhang, H. Zhou, and X. Shen, "Qoe-driven transmission-aware cache placement and cooperative beamforming design in cloud-rans," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 636–650, Nov. 2020.

[4] R. Wang, J. Zhang, S. H. Song, and K. B. Letaief, "Mobility-aware caching in d2d networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5001–5015, May 2017.

[5] R. Wang, J. Zhang, S. H. Song, and K. B. Letaief, "Exploiting mobility in cache-assisted d2d networks: Performance analysis and optimization," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5592–5605, Jun. 2018.

[6] J. Ma, L. Liu, H. Song, R. Shafin, B. Shang, and P. Fan, "Scalable video transmission in cache-aided device-to-device networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 4247–4261, Mar. 2020.

[7] Q. Xu, Z. Su, Y. Wang, and K. Zhang, "Secure edge caching for layered multimedia contents in heterogeneous networks," in *Proc. IEEE Global Commun. Conf.*, Waikoloa, HI, USA, Feb. 2019, pp. 1–6.

[8] T. Deng, G. Ahani, P. Fan, and D. Yuan, "Cost-optimal caching for d2d networks with user mobility: Modeling, analysis, and computational approaches," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3082–3094, Feb. 2018.

[9] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Device-to-device coded-caching with distinct cache sizes," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 2748–2762, Jan. 2020.

[10] M. C. Lee and A. F. Molisch, "Individual preference aware caching policy design in wireless d2d networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5589–5604, May 2020.

[11] Y. Chen, X. Gong, R. Ou, L. Duan, and Q. Zhang, "Crowdcaching: Incentivizing d2d-enabled caching via coalitional game for iot," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5599–5612, Jun. 2020.

[12] Y. Fang, P. Chen, G. Cai, F. C. Lau, S. C. Liew, and G. Han, "Outage-limit-approaching channel coding for future wireless communications: Root-protograph low-density parity-check codes," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 85–93, Jun. 2019.

[13] Y. Fang, S. C. Liew, and T. Wang, "Design of distributed protograph ldpc codes for multi-relay coded-cooperative networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7235–7251, Nov. 2017.

[14] R. Ding, T. Wang, L. Song, Z. Han, and J. Wu, "Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery," in *Proc. IEEE Wireless Commun. Netw. Conf.*, New Oleans, LA, USA, Mar. 2015, pp. 1207–1212.

[15] Z. Hu, Z. Zheng, T. Wang, L. Song, and X. Li, "Roadside unit caching: Auction-based storage allocation for multiple content providers," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 10, pp. 6321–6334, Oct. 2017.

[16] Z. Su, Y. Hui, Q. Xu, T. Yang, J. Liu, and Y. Jia, "An edge caching scheme to distribute content in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5346–5356, Jun. 2018.

[17] Z. Qin, S. Leng, J. Zhou, and S. Mao, "Collaborative edge computing and caching in vehicular networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Seoul, Korea, Jun. 2020, pp. 1–6.

[18] Y. AlNagar, S. Hosny, and A. A. El-Sherif, "Towards mobility-aware proactive caching for vehicular ad hoc networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshop*, Marrakech, Morocco, Nov. 2019, pp. 1–6.

[19] J. Ma, J. Wang, G. Liu, and P. Fan, "Low latency caching placement policy for cloud-based vanet with both vehicle caches and rsu caches," in *Proc. IEEE Global Commun. Workshops*, Singapore, Dec. 2017, pp. 1–6.

[20] L. Hou, L. Lei, K. Zheng, and X. Wang, "A q -learning-based proactive caching strategy for non-safety related services in vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4512–4520, Jun. 2019.

[21] B. Hu, L. Fang, X. Cheng, and L. Yang, "In-vehicle caching (iv-cache) via dynamic distributed storage relay (d^2sr) in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 843–855, Nov. 2019.

[22] 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*, vol. 20, no. 2, pp. 391–402, Feb. 2021.

[23] T. Deng, P. Fan, and D. Yuan, "Optimizing retention-aware caching in vehicular networks," *IEEE Trans. Commun.*, vol. 67, no. 9, pp. 6139–6152, Jun. 2019.

[24] Y. Zhang, C. Li, T. H. Luan, Y. Fu, W. Shi, and L. Zhu, "A mobility-aware vehicular caching scheme in content centric networks: Model and optimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3100–3112, Feb. 2019.

[25] X. Han, X. Li, C. Luo, H. Ji, and H. Zhang, "Incentive mechanism with the caching strategy for content sharing in vehicular networks," in *Proc. IEEE Global Commun. Workshops*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[26] H. Yang, K. Zhang, K. Zheng, and Y. Qian, "Secure edge caching placement and delivery for ultra-reliable and low-latency vehicular networks," in *Proc. IEEE Global Commun. Workshops*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[27] J. Chen, H. Wu, P. Yang, F. Lyu, and X. Shen, "Cooperative edge caching with location-based and popular contents for vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10 291–10 305, Jun. 2020.

[28] 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 Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.

[29] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. S. Shen, "Catalyzing cloud-fog interoperation in 5g wireless networks: An sdn approach," *IEEE Netw.*, vol. 31, no. 5, pp. 14–20, Sep. 2017.

[30] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen, "A survey on platoon-based vehicular cyber-physical systems," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 1, pp. 263–284, 1st Quart. 2016.

[31] H. Zhou, B. Liu, T. H. Luan, F. Hou, L. Gui, Y. Li, Q. Yu, and X. Shen, "Chaincluster: Engineering a cooperative content distribution framework for highway vehicular communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2644–2657, Dec. 2014.

[32] S. Ucar, S. C. Ergen, and O. Ozkasap, "Ieee 802.11p and visible light hybrid communication based secure autonomous platoon," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8667–8681, May 2018.

[33] M. Xing, J. He, and L. Cai, "Utility maximization for multimedia data dissemination in large-scale vanets," *IEEE. Trans. Mob. Comput.*, vol. 16, no. 4, pp. 1188–1198, Apr. 2017.

[34] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *Proc. Annu. Joint Conf. IEEE Comput. Commun. Soc.*, New York, NY, USA, Mar. 1999, pp. 126–134.

[35] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: a survey and extensions," *Math. Method Oper. Res.*, vol. 66, no. 3, pp. 373–407, Dec. 2007.

[36] H. Wang, W. Jing, X. Wen, Z. Lu, and S. Zhao, "Segment-based joint caching and recommendation optimization for mobile video transmission," in *Proc. IEEE Global Commun. Workshops*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[37] W. Cohda, "Cohda mk5," Cohda Wireless Pty, ltd., Dec. 2020, [Online]. Available: https://cohdawireless.com/solutions/hardware/.

[38] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic, "Weakly supervised action labeling in videos under ordering constraints," in *Proc. Eur. Conf. Comput. Vis.*, Zurich, Switzerland, Sep. 2014, pp. 1–16.

[39] X. Xu, K. Liu, K. Xiao, H. Ren, L. Feng, and C. Chen, "Design and implementation of a fog computing based collision warning system in vanets," in *Proc. IEEE Symp. PCE*, Shenzhen, China, Dec. 2018, pp. 1–6.

Zheng Xue received the B.E. degree in automotive service engineering from Chongqing Jiaotong University, Chongqing, China, in 2018. He is currently working toward the M.S. degree with the Department of Communication Engineering, Guangdong University of Technology, Guangzhou, China. His primary research interest is vehicular networks.

Yang Liu received his Ph.D. and M.E. degree in Communication and Information System from Sun Yat-sen University, Guangzhou, China, and the B.S. degree in Automation from Beijing Institute of Technology, Beijing China. From November 2015 to November 2016, he was a visiting scholar at the Department of Computer and Information Sciences, University of Delaware, Newark, USA. He is now a lecturer at the School of Information Engineering, Guangdong University of Technology, Guangzhou, China. His research interests include Internet of things, optimization algorithm and machine learning.

Guojun Han received his Ph.D. from Sun Yatsen University, Guangzhou, China, and the M.E. degree from South China University of Technology, Guangzhou, China. From March 2011 to August 2013, he was a Research Fellow at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. From October 2013 to April 2014, he was a Research Associate at the Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology. He is now a Full Professor and Executive Dean at the School of Information Engineering, Guangdong University of Technology, Guangzhou, China. He has been a Senior Member of IEEE since 2014. His research interests are in the areas of wireless communications, signal processing, coding and information theory. He has more than 14 years experience on research and development of advanced channel coding and signal processing algorithms and techniques for various data storage and communication systems.

Ferheen Ayaz (Graduate Student Member, IEEE) received the B.E. and M.E. degrees from NED University of Engineering and Technology, Karachi, Pakistan, in 2010 and 2014, respectively. She is currently pursuing the Ph.D. degree from the University of Sussex, Brighton, UK. Her research interests include blockchain implementation for security and privacy in vehicular networks. She is an Early Careers Officer in IEEE UK & Ireland Cybersecurity group.

Zhengguo Sheng (Senior Member, IEEE) received the B.Sc. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006, and the M.S. and Ph.D. degrees from Imperial College London, London, U.K., in 2007 and 2011, respectively. He is currently a Senior Lecturer with the University of Sussex, Brighton, U.K. Previously, he was with UBC, Vancouver, BC, Canada, as a Research Associate and with Orange Labs, Santa Monica, CA, USA, as a Senior Researcher. He has more than 120 publications. His research interests cover IoT, vehicular communications, and cloud/edge computing.

Yonghua Wang received his B.S. degree in Electrical Engineering and Automation from Hebei University of Technology in 2001, the M.S. degree in Control Theory and Control Engineering from Guangdong University of Technology in 2006, and Ph.D. degree in Communication and Information System from Sun Yat-sen University in 2009. He currently is an associate professor in School of Automation in Guangdong University of Technology. His current research interests include machine learning and cognitive radio networks.