Research Article

Cluster-Based Cooperative Cache Deployment and Coded Delivery Strategy in C-V2X Networks

Haizhou Bao,1 Yiming Huo,2 Chuanhe Huang,1 Xiaodai Dong,2 and Wanyu Qiu1

1School of Computer Science, Wuhan University, Wuhan 430072, China
2Department of Electrical and Computer Engineering, University of Victoria, Victoria, Canada V8P 5C2

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

Received 29 August 2020; Revised 8 April 2021; Accepted 19 April 2021; Published 3 May 2021

Copyright © 2021 Haizhou Bao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cellular vehicle-to-everything-(C-V2X-) based communications can support various content-oriented applications and have gained significant progress in recent years. However, the limited backhaul bandwidth and dynamic topology make it difficult to obtain the multimedia service with high-reliability and low-latency communication in C-V2X networks, which may degrade the quality of experience (QoE). In this paper, we propose a novel cluster-based cooperative cache deployment and coded delivery strategy for C-V2X networks to improve the cache hit ratio and response time, reduce the request-response delay, and improve the bandwidth efficiency. To begin with, we design an effective vehicle cluster method. Based on the constructed cluster, we propose a two-level cooperative cache deployment approach to cache the frequently requested files on the edge nodes, LTE evolved NodeB (eNodeB) and cluster head (CH), to maximize the overall cache hit ratio. Furthermore, we propose an effective coded delivery strategy to minimize the network load and the ratio of redundant files. Simulation results demonstrate that our proposed method can effectively reduce the average response delay and network load and improve both the hit ratio and the ratio of redundant files.

1. Introduction

In recent years, cellular communications, edge computing, artificial intelligence (AI), etc., have greatly promoted the development of the cellular vehicle-to-everything (C-V2X) networks. At present, C-V2X is widely used in many scenarios such as road multimedia-related entertainment messages [1, 2], autonomous driving [3, 4], and emergency services [5]. Mobile devices, such as smartphones [6] and onboard navigation devices with wireless connectivity, have become more popular in recent years. However, these devices occupy considerable network bandwidth while providing users with a comfortable and enjoyable experience, especially when high-definition videos and games with significant network access requirements are usually the desired application scenarios. Furthermore, latency is more sensitive in most aforementioned applications and will largely impact the quality of service (QoS) and vehicle (user) experiences, especially for some delay-sensitive scenarios. How to reduce the network load and ensure network QoS is a major challenge in C-V2X networks with limited bandwidth resources.

On the other hand, edge cache technology can effectively alleviate the pressing issues stemming from the increasing data traffic in C-V2X networks. For example, it can also effectively reduce the response time to user requests, especially when users repeatedly request the same file from a remote server; a lot of network resources are wasted [7]. To avoid this problem, network edge nodes can prefetch part of the file from a remote server to cache [8], when a requested file is cached at the edge nodes by directly downloading from the edge node (user or eNodeB) to reduce the traffic load transmitted in the entire network, which can also reduce the response delay.

Furthermore, the centralized architecture of the cellular network and its limited transmission capacity makes it difficult to cope with growing traffic demands. Also, due to the high velocity of vehicles in C-V2X networks and frequent changes of network topology, it is difficult to obtain reliable links among vehicle users and receive the desired data from...
neighbor vehicles or the nearby eNodeBs reliably and efficiently in C-V2X networks. There have been some research works on addressing this topic, for example, vehicle users on the road are divided into multiple clusters based on certain similar attributes [9, 10], i.e., velocity and position. Cluster members and cluster heads are continuously updated according to certain conditions. Furthermore, the dynamic network can be transformed into a static network as presented in research [11]. To satisfy users’ requests and ensure high QoS, it is necessary to improve the request hit ratio and reduce the delay for users to obtain the required data. Due to the various request probabilities of different files in the system, current research widely suggests that caches with high request probabilities should be preferentially deployed in edge nodes [12, 13] to improve the hit ratio as different cache deployment strategies will lead to different request delay. Moreover, a smaller request delay and a higher hit ratio can be obtained by optimizing the cache deployment strategy, which can also reduce the network load of the backhaul link. To reduce the network bandwidth resources consumed by user requests, utilizing a coded delivery method based on subfile coding [14, 15] can eliminate redundancy among multiple requested files and also reduce redundancy among subfiles. Due to ignoring the impact of cache deployment strategy on response delay and bandwidth [16], we combine the cache strategy and the transmission strategy to discuss their impacts on C-V2X networks in this article.

In this paper, we investigate a novel cluster-based two-level cooperative cache deployment and coded delivery strategy for C-V2X networks to improve the cache hit ratio, reduce the response time, and improve the bandwidth usage efficiency. The main contributions of this paper are summarized as follows:

(i) To make the communication in C-V2X networks efficient and robust, we design a stable and reliable vehicle cluster, which includes the CH selection, the cluster formation, and the cluster merging. Based on the constructed cluster, we propose a two-level cooperative cache deployment strategy to reduce the response time and improve the hit ratio, where the response time for all user requests is defined as a function of the cache deployment decision, and then derive the optimal cache decision for the CH and eNodeB in C-V2X network

(ii) We propose a coded delivery strategy, i.e., file coding and subfile coding, on the edge cache nodes when they receive the users’ request. At this stage, the formulation of the coded delivery problem is formulated as a multiobjective set cover problem (MOSCP), which is an NP-hard problem. Eventually, we propose an efficient heuristic algorithm to minimize the transmission network load and the ratio of redundant files and satisfy users’ requests at the same time.

The rest of this paper is organized as follows: Section 2 investigates and summarizes both the cache deployment scheme and coded delivery scheme on the Internet of Vehicles (IoV). Section 3 proposes a network model and process-
and analyzed the method of coded delivery in a hybrid network, proposed a subfile coding method, and considered that the traffic data of users within a hop range was transmitted by coding, but they ignore redundant files of user requests. Liu et al. in [27] considered designing an optimal coding scheme under cache constraints. However, transmitting the same data within one hop range would still cause users to waste bandwidth resources. Ramakrishnan et al. in [28] proposed to divide a file into subfiles and code the subfiles and then transmit them to users, which can effectively reduce the traffic on the shared link. Asghari et al. studied the optimal coded delivery scheme for arbitrary file deployment [14, 15] and formulated the problem as a set cover problem, which however ignored the redundant data among subfiles in the packet. Such data packets may cause repeated transmission of the same subfiles and still waste a lot of network bandwidth.

Because of the limitations of the above methods, we focus on investigating the cluster method to make the communication more reliable, designing a two-level cooperative caching approach to improve the hit ratio and reduce the request delay, and coded delivery method to reduce the network load and further to improve the network performance.

3. Network Model and System Model

3.1. Network Model. We consider a C-V2X network scenario, where there are $K$ eNodeBs, i.e., $\{\text{eNB}_1, \text{eNB}_2, \ldots, \text{eNB}_K\}$, and several vehicle users are randomly deployed in the network. The system structure diagram is depicted in Figure 1, in which the CH is a cluster head, the green dot circle represents a cluster, and vehicles in the cluster except the CH are cluster members. Each vehicular node uses the LTE-V standard supporting the side link or the vehicle-to-vehicle (V2V) communications by adopting the PC5 interface in the LTE system. The yellow dash line is a side link that is used for V2V communications with the PC5 interface, while the blue dash line is the link used for vehicle-to-infrastructure (V2I) communication with $U_u$ interface.

3.2. System Model. The system model includes two parts: cluster-based two-level cooperative cache deployment strategy and coded delivery strategy. In cluster-based two-level cooperative cache deployment, we first propose a vehicle cluster model according to the vehicle speed, the position, and other information for reliable communication and then design an optimal cache deployment strategy to reduce the response delay and improve the hit ratio. Furthermore, we propose a coded delivery method to reduce the transmission load. From an overall standpoint, our system model is depicted in Figure 2.

4. Cluster-Based Two-Level Cooperative Cache Deployment Strategy

4.1. Vehicle Cluster Model. We assume that vehicles in a cluster can communicate with a CH directly, and the CH can also communicate with its connected eNodeB directly. The status of a vehicle in a cluster includes Undecided Node (UN), CH, and cluster member (CM) as stated in [29, 30]. The clustering procedure includes the CH selection strategy, the cluster maintenance strategy, and the cluster merging strategy.

Vehicles can exchange and collect necessary information about their one-hop vicinities reactively with each other through periodic beacon messages. Each vehicle $V_i$ constructs its neighbor list $\text{NL}_i$; here, we assume the neighbor refers to the relative distance $\Delta d_{ij}$ between any two vehicles $V_i$ and $V_j$ is less than the communication range $R$. After the necessary information from the disseminated beacon messages has been received, each vehicle can calculate three important factors, such as the relative distance, the relative speed, and the link reliability for one-hop neighbors within the communication range $R$. 
4.2. Average Relative Speed and Average Relative Distance. The position and the velocity of \( V_i \) at time \( t \) can be presented as \( p_i(t) = (p_{ix}(t), p_{iy}(t)) \) and \( v_i(t) = (v_{ix}(t), v_{iy}(t)) \), respectively. Similarly, the position and the velocity of \( V_j \) at time \( t \) are \( p_j(t) = (p_{jx}(t), p_{jy}(t)) \) and \( v_j(t) = (v_{jx}(t), v_{jy}(t)) \), respectively. Note that vehicles may be driving in the inverse direction, where the velocity is negative. The relative distance is \( \Delta d_{ij}(t) = \| (p_{ix}(t) - p_{jx}(t))^2 + (p_{iy}(t) - p_{jy}(t))^2 \|_2 \); similarly, the relative velocity is \( \Delta v_{ij}(t) = \| (v_{ix}(t) - v_{jx}(t))^2 + (v_{iy}(t) - v_{jy}(t))^2 \|_2 \). Therefore, the average relative distance \( \Delta d_i(t) \) and average relative velocity \( \Delta v_i(t) \) between vehicle \( V_i \) and its neighbors are given as

\[
\Delta d_i(t) = \frac{1}{|N_L_i|} \sum_{j \in N_L_i} \Delta d_{ij}(t),
\]

\[
\Delta v_i(t) = \frac{1}{|N_L_i|} \sum_{j \in N_L_i} \Delta v_{ij}(t),
\]

where \( N_L_i \) is the number of vehicle nodes in the neighbor list of vehicle \( V_i \).

4.3. LLT and Link Reliability. The LLT describes the ability to maintain the connection of two vehicles. When two vehicles are moving in the same or opposite directions, the LLT between vehicle \( V_i \) and vehicle \( V_j \) can be given as [31]

\[
\text{LLT}_{ij}(t) = \frac{\sqrt{R^2(\Delta v_{ix}(t) + \Delta v_{iy}(t))^2 - (\Delta p_{ix}(t) \Delta v_{iy}(t) - \Delta p_{iy}(t) \Delta v_{ix}(t))^2}}{\Delta p_{ix}(t) + \Delta p_{iy}(t)},
\]

where

\[
\Delta v_{ix}(t) = v_{ix}(t) - v_{jx}(t),
\]

\[
\Delta v_{iy}(t) = v_{iy}(t) - v_{jy}(t),
\]

\[
\Delta p_{ix}(t) = p_{ix}(t) - p_{jx}(t),
\]

\[
\Delta p_{iy}(t) = p_{iy}(t) - p_{jy}(t).
\]

Link reliability is the probability that a direct communication link \( l_{ij} \) will be available between two vehicles \( V_i \) and \( V_j \) over a particular time \( t \). We assume that the velocity of vehicle \( V_j \), \( v_j \), follows the Gaussian distribution \( N(\mu, \sigma^2) \) [11]. Furthermore, the probability density function (PDF) of the link communication duration \( T \) between vehicle \( V_i \) and vehicle \( V_j \) is [11]

\[
f(T) = \frac{4R}{\sigma \Delta v_i \sqrt{2\pi} T^2} e^{-\frac{(2R/T - \mu_{\Delta v_i})^2}{2\sigma^2}}, \quad \forall T > 0,
\]

where \( R_{ij} \) represents the relative transmission range, and \( \mu_{\Delta v_i} \) and \( \sigma_{\Delta v_i} \) denote the mean and standard variation of relative velocity between vehicle \( V_i \) and vehicle \( V_j \), respectively.

The probability of the link \( l_{ij} \) between vehicle \( V_i \) and \( V_j \) is available during \( \text{LLT}_{ij} \) [32], which also represents the reliability of link \( l_{ij} \) between vehicle \( V_i \) and \( V_j \) during \( \text{LLT}_{ij} \) and can be given as

\[
r_t(l_{ij}) = \int_{\text{LLT}_{ij}} f(T) dt = \text{erf} \left( \frac{2R/(t + \text{LLT}_{ij}) - \mu_{\Delta v_i}}{\sigma_{\Delta v_i} \sqrt{2}} \right) - \text{erf} \left( \frac{2R/(t - \text{LLT}_{ij}) - \mu_{\Delta v_i}}{\sigma_{\Delta v_i} \sqrt{2}} \right), \quad \text{LLT}_{ij} > 0,
\]

where \( \text{erf}() \) is the Gauss error function [11].

The average link reliability is given as

\[
r_t(l_{ij}) = \frac{1}{|N_L_i|} \sum_{j \in N_L_i} r_t(l_{ij}).
\]

According to the above definition, the larger \( r_t(l_{ij}) \) is, the more reliable the links between the vehicle \( V_i \) and its neighbor nodes are. Furthermore, the smaller the \( \Delta d_i \) and \( \Delta v_i \), the more stable the topology is. Therefore, we use a score to evaluate the stability of each vehicle \( V_i \) as follows:

\[
S_i = \alpha_i \left( 1 - \frac{\Delta d_i}{\Delta d_{\text{max}}} \right) + \beta_i \left( 1 - \frac{\Delta v_i}{\Delta v_{\text{max}}} \right) + \gamma_i r_t(l_{ij}_{\text{max}}),
\]

where \( \Delta d_{\text{max}} \), \( \Delta v_{\text{max}} \), and \( r_t(l_{ij}_{\text{max}}) \) denote the maximum relative distance, maximum relative speed, and maximum link reliability, respectively. \( \alpha_i \), \( \beta_i \), and \( \gamma_i \) are the weight factor, and \( \alpha_i + \beta_i + \gamma_i = 1 \). We can easily find that the larger the score \( S_i \) is, the more stable the cluster is. After the score is calculated for each vehicle, we can select an appropriate \( \text{CH} \) and then construct a cluster according to the score \( S_i \).

4.4. Procedure of Vehicle Cluster. The vehicle cluster construction includes three steps as follows:

(i) CH Selection. Each vehicle (UN) broadcasts a beacon message to all surrounding vehicles periodically, which includes its location, velocity, and direction information. To start with, each user exchanges beacon messages with its neighbors. Secondly, each user updates its neighbor list \( N_L_i \) after receiving the beacon messages from its neighbors and then calculates the score \( S_i \) for itself and its neighbors. The vehicle with the highest score in the neighbor list (NL) is
chosen as the CH, and it will broadcast the message CH_ACK to all its neighbors to announce itself as the CH and sets the timer CH_TIMER for its neighbors to finish a cluster construction.

(ii) Cluster Formation. After a neighboring vehicle $V_j$ receives a message CH_ACK or a beacon message from the CH $V_i$ (abbreviated as CH_i), if it is not a cluster member of the cluster $V_i$ which CH_i belongs to, it will send a message JOIN_REQ to the CH_i to join the cluster $V_i$. When the CH_i receives a message JOIN_REQ from a neighbor node $V_j$, the CH_i will judge whether the neighbor vehicle $V_j$ can join the current cluster. If the size of the current cluster NC Mi of cluster $V_i$ exceeds the maximum threshold value $NCM_{max}$, vehicle $V_j$ is not allowed to join in the cluster; otherwise, CH_i will send a joining message JOIN_RESP to the vehicle $V_j$ and add the vehicle $V_j$ to its cluster member list CMLIST_i. If vehicle $V_j$ receives a message JOIN_RESP within the duration JOIN_TIMER, it will change its status to a cluster member as CM_j; otherwise, it will still keep its status as UN. The CH will broadcast the message that a new neighbor node joins the cluster to all the cluster members. If the vehicle cannot join another cluster, and if the score $S_i$ of the vehicle is the largest value within the neighbor list, it will change its status to the CH and broadcast CH_ACK to its neighbor vehicles.

(iii) Cluster Merging. When two clusters are too close that they overlap with each other, the overlapped area may consume more resources; consequently, we can merge these two clusters. When a CH $V_i$ receives a beacon message from another CH $V_j$, if the potential size of the merged cluster is not larger than the threshold $NCM_{merge} \leq NCM_{max}$ and $LLT_{i,j} \geq \omega$ ($V_i, V_j$ are the CH node), the CH with a smaller score will change its status to CM. Simultaneously, another CH with a higher score keeps the status as the CH.

4.5. Algorithm Description for Vehicle Cluster. In Algorithm 1, lines 1-19 are the description of CH selection and cluster formation, and cluster merging procedures is described from lines 20 to 24. From line 2 to line 3, each vehicle calculates scores based on the obtained beacon message and the vehicle with the highest score is a CH, which can send CH_ACK and set the timer CH_TIMER. Lines 4-18 describe that the cluster member joins the cluster and the CH responds to the vehicle

---

**Algorithm 1: Cluster method.**
to join. During the CH timer $CH\_TIMER$, if the vehicle receives a message $CH\_ACK$, and it is not a member of the CH, it will unicast the message $JOIN\_REQ$ and then set the timer $JOIN\_TIMER$. During the $JOIN\_TIMER$, if the vehicle receives the message $JOIN\_RESP$, it will change its status to $CM_i$, i.e., the cluster member of CH. Otherwise, it will keep UN status. On the other hand, lines 15–17 describe the response of the CH when it receives a message. If the CH receives the message $JOIN\_REQ$ and the potential number of CM, $N_{CM_i}$, + 1, is still smaller than the maximum number of CM $NCM_{max}$, the CH will respond with the message $JOIN\_RESP$ to the vehicle $V_j$. From lines 20 to 24, it is the procedure of cluster merging. If the distance of any two CH is smaller than the threshold $L_c$, the link duration between the two CHs is larger than a threshold $\omega$, the total potential cluster member of the two clusters is not larger than the maximum value $NCM_{max}$, and then the CH with smaller score will give up its CH status and then change its status to $CM_{i}$ and add all its members to the CMLIST.$i$.

4.6. Two-Level Cooperative Cache Model

4.6.1. Cluster Size and Cluster Number. Assume that the arrival rate of vehicles follows a Poisson process, the distance between two vehicles follows an exponential distribution; the average number of vehicles in a cluster can be denoted as $[29]$}

$$N_c = E(N_c) = \frac{1}{e^{\lambda R_c E(1/V)^2}}, \quad (9)$$

where

$$E\left(\frac{1}{V}\right) = \int_{\min}^{\max} \frac{1}{v} f_V(v) dv, \quad (10)$$

where $[\cdot]$ is a floor function and $R$ is the vehicle’s communication radius. Consequently, the average number of vehicles under the coverage of eNodeB is

$$N_k = 2R_c \times \hat{\rho} = 2R_c [\hat{\rho}], \quad (11)$$

where $\hat{\rho} = \lambda E(1/V)$ is the average density of vehicles $[30]$, and $R_c$ is the transmission radius of eNodeB. Meanwhile, we can also obtain the average number of clusters under the coverage of the eNodeB, i.e., $N_{ch} = N_k / N_c$.

4.6.2. Request and Response Procedure. In this paper, the response mode for file request has four types as shown in Figure 3, where the purple solid line and the dashed line are the request and response action, respectively. First of all, a file request from a CM is forwarded to a CH in the cluster; if the requested file is cached by the CH, the CH will respond to the request by transmitting the file to the user directly (e.g., steps (1) → (2)); otherwise, the request will be forwarded to the nearby eNodeB. Besides, if the eNodeB has cached the file, the eNodeB will immediately broadcast the file to the user (e.g., steps (3) → (4)); otherwise, the eNodeB will forward the request to its neighboring eNodeB. Furthermore, if the neighboring eNodeBs have cached the requested file, the neighboring eNodeB will respond to the request of the user through the original eNodeB (e.g., steps (3) → (5) → (6) → (4)); otherwise, the request will be forwarded to a remote server. Eventually, the server will transmit the file to the eNodeB and then forward it to the user (e.g., steps (3) → (7) → (8) → (4)).

4.6.3. Delay of File Request. According to procedure 4.6.2, we summarize the response delay as follows:

(i) **Response by the CH.** The delay for a user to obtain a file from CH in a cluster is

$$t_{v2c} = \frac{s_q}{B_r \log_2 \left(1 + \left(P_c |h_{c,n}|^2 (\xi^2 + \sum_{n=1}^{N_n} |P_n |^2 |h_{c,n}|^2)\right)\right)}, \quad (12)$$

where $s_q$ is the file size, the denominator represents the transmission rate between the vehicle $V_j$ and the CH, $h_{c,j}$ is the channel coefficient from cluster head $c$ to other vehicle $V_j$, $N_c$ is the total number of vehicles in the cluster, $P_c$ is the transmit power of the CH, $B_r$ is the channel bandwidth for the vehicle, $\xi^2$ is the noise power, and $P_n$ is the transmit power of other vehicles in the cluster, and the sum term in the denominator is total interferences from other vehicles to the current vehicle $V_j$ in the cluster.

(ii) **Response by eNodeB or Neighboring eNodeB.** If the user cannot download the file from the CH in the cluster, the connected eNodeB or neighboring eNodeB will be scheduled to respond to the request. The response delay can be denoted, respectively, as follows:

$$t_{v2e} = \frac{s_q}{B_r \log_2 \left(1 + \left(P_c |h_{c,n}|^2 (\xi^2)\right)\right)}, \quad (13)$$

where $t_{v2e}$ is the delay from a eNodeB to a user and $t_{v2e}$ is the delay from the user to the nearby eNodeB.
The denominator denotes the transmission rate between the vehicle \( V_j \) and the corresponding eNodeB, \( B_i \), is the subchannel bandwidth allocated to the user \( V_j \), \( h_i \) is the channel coefficient from eNodeB to the user, \( P_i \) is the transmit power of the eNodeB, and \( \xi^2 \) is the noise power. \( T_{\text{hop}} \) is the transmission delay from neighboring eNodeB to current eNodeB with \( N_{\text{hop}} \) hops; here, we assume the \( N_{\text{hop}} = 1 \), and the transmission delay is usually milliseconds

(iii) **Response by Remote Server.** When the edge cache nodes (vehicle users) or neighbor eNodeB have not cached the file that the user requests, the user needs to download the file from a remote server. The latency of the response from a remote server is

\[
t_{v2r} = \frac{s_q}{R_{v2r}} + t_{v2R},
\]

where the first item is the delay from the remote server to the eNodeB and \( R_{v2r} \) is the transmission rate of the backhaul link. The second term is the delay from the eNodeB to the current user

4.6.4. **Hit Ratio of All Users.** After obtaining the file request delay for four modes, we design an efficient cache strategy to reduce the average total delay for all file requests and satisfy the cache capacity for each node. We use \( Q = \{1, 2, 3, \ldots, Q\} \) to represent the file library, where \( Q \) is the total number of files. Each file has \( s_q \) bits with a different request probability; besides, the files are stored on a remote server in order from high to low according to the request probability. We use Zipf distribution to model the request probability of a file \([33]\); as a result, the probability that a file \( q \) is requested is

\[
P_q = \frac{q^{-\beta}}{\sum_{i=1}^{Q} i^{-\beta}},
\]

where \( \beta \geq 0 \) is the bias parameter of Zipf, which indicates popularity. Generally, the larger the value is, the more concentrated the request will be. We assume that all vehicles will just request once (different requests from the same vehicle can be seen as different requests from different vehicles). Obviously, the probability that all files requested by different users satisfy \( \sum_{q=1}^{Q} P_q = 1 \).

We adopt a two-level edge cache strategy based on CH and eNodeB \([24, 34]\) to improve the hit ratio and reduce the response time for all requests. The CH and eNodeB are the first and second layer cache nodes, respectively, and they cache different files. Furthermore, all CHs under the coverage of the eNodeB eNB cache the same files. Let \( V_c \) and \( V_k \) denote the cache capacity of a vehicle and an eNodeB, respectively. On the other hand, we use the matrix \( H_{iq} = [h_{iq}]_{K \times 1 \times Q} \) to indicate whether the \( i (i \in 1, 2) \) layer under the coverage of the eNodeB eNB stores the file \( q \), if affirmative \( h_{iq} = 1 \), otherwise 0. Additionally, the probability that files \( q \) is requested under the coverage of eNodeB eNB is denoted as \( p^c_q \).

The hit ratio of user requests under eNodeB eNB\( h \) coverage is defined as follows:

\[
H_{\text{eNB}_{h}} = \sum_{k=1}^{K} \sum_{q=1}^{Q} p^c_q h^k_{iq} + \sum_{k=1}^{K} \sum_{q=1}^{Q} p^c_q \left( D_q - h^k_{2q} - h^{k}_{1q} \right) + \sum_{k=1}^{K} \sum_{q=1}^{Q} \left( 1 - D_q \right) p^c_q,
\]

where \( D_q = \{0, 1\} \) indicates whether the file \( q \) is cached at least one edge node (eNodeB or CH), and if affirmative, \( D_q = 1 \) (at least one edge node can satisfy the user’s request, that is, \( h^k_{2q} = 1 \) or \( h^k_{1q} = 1 \)), otherwise 0 (\( h^k_{2q} = 0 \) and \( h^k_{1q} = 0 \)).

4.6.5. **Problem Formulation of Optimal Cache Deployment.** Based on the hit ratio definition, we can formulate the problem as a response delay with a different cache strategy. According to the request procedure, the request delay is modeled as a function of the cache strategy. To reduce the overall delay for users to obtain the requested files and satisfy the constraints of cache size for each cache node, the problem can be formalized as follows:

\[
\min_{h_{iq}} \sum_{k=1}^{K} \left( t_{v2c} \sum_{q=1}^{Q} p^c_q h^k_{1q} + t_{v2R} \sum_{q=1}^{Q} p^c_q h^k_{2q} + t_{v2R} \sum_{q=1}^{Q} \left( 1 - D_q \right) p^c_q \right)
\]

s.t. \( \sum_{i=1}^{2} h^k_{iq} \leq 1 \),

\[
\sum_{q=1}^{Q} h^k_{1q} \leq V_c,
\]

\[
\sum_{q=1}^{Q} h^k_{2q} \leq V_k,
\]

\[
\sum_{q=1}^{Q} h^k_{iq} \in \{0, 1\}, \quad \forall i \in \{1, 2\}.
\]

The cache strategy of the CH will directly affect the eNodeB cache strategy; thus, the cache strategy of the CH should be given a priority. In other words, each CH under the coverage of the eNodeB eNB\( h \) caches the most frequently requested files, namely, each CH should cache \( V_c \) files in order from high to low according to the request probability of each file. Consequently, each eNodeB eNB\( h \) selects files from the remaining file list \( [V_c + 1, Q] \) to cache. Problem (17) can be further transformed into the following formation:
\[
\min_{h^k_{2q}} \sum_{k=1}^{K} \left( t_{v2R} \sum_{q=V_{r+1}}^{Q} p^k_q \ast h^k_{2q} + t_{v2R} \sum_{q=V_{r+1}}^{Q} p^k_q \left( D'_q - h^k_{2q} \right) \right) \\
+ t_{v2s} \sum_{q=V_{r+1}}^{Q} p^s_q \left( 1 - D'_q \right)
\]
\[\text{s.t. } \sum_{q=V_{r+1}}^{Q} h^k_{2q} s_q \leq V_k, \]
\[h^k_{2q} \in \{0, 1\}, \]
\]

where \( D'_q = \{0, 1\} \) indicates that the file \( q \) is stored at least one edge node, i.e., CH or eNodeB. Particularly, the sum of the request probabilities for a file \( q \) under the coverage of eNodeB \( eNB_k \) should satisfy
\[
\sum_{k=1}^{K} p^k_q = p_q.
\]

We can convert problem (18) into
\[
\min_{h^k_{2q}} t_{v2R} \sum_{q=V_{r+1}}^{Q} p^k_q h^k_{2q} + t_{v2R} \sum_{q=V_{r+1}}^{Q} p^k_q \left( D'_q - h^k_{2q} \right)
\]
\[+ t_{v2s} \sum_{q=V_{r+1}}^{Q} p^s_q \left( 1 - D'_q \right)\]
\[\text{s.t. } \sum_{q=V_{r+1}}^{Q} h^k_{2q} \leq V_k, \]
\[h^k_{2q} \in \{0, 1\}, \]

where the sum of request probabilities \( p_q \) [12, 22] can be expressed as
\[
f(x) = \sum_{q=V_{r+1}}^{Q} p^s_q \approx \int_{V_{r+1}}^{x} \sum_{q=V_{r+1}}^{Q} p^s_q (1/q^\beta) \, dq = \frac{x^{1-\beta} - (V_c + 1)^{-\beta}}{Q^{1-\beta} - 1}.
\]

4.6.6. Algorithm Design for Optimal Cache Deployment. To reduce the overhead of downloading files from remote servers or neighboring eNodeBs and simplify problem (20), each eNodeB can cache some identical files and a part of different files [35]. We assume that \( aV_k \) of eNodeB capacity is used to cache the same files, and \( K(1-\alpha)V_k \) of eNodeB capacity will be used to cache different files for the total \( K \) eNodeBs; therefore, the total capacity of cached files on the eNodeB is \( V_k + K(1-\alpha)V_k, \alpha \in (0, 1) \). The total hit ratio for all files is the sum of the hit ratio of caching the same file, and the mean hit ratio of caching different file. Consequently, the hit ratio to obtain files from eNodeBs can be expressed as
\[
H_{\text{eNodeB}} = f(\alpha V_k + K(1-\alpha)V_k + V_c) - f(\alpha V_k + V_c)
\]
\[+ f(\alpha V_k + V_c).\]

On the other hand, the hit ratio to obtain the requested files from neighbor eNodeBs through cooperation is
\[
H_{\text{coop}} = \frac{K - 1}{K} f(\alpha V_k + K(1-\alpha)V_k + V_c) - f(\alpha V_k + V_c).
\]

After substituting (21), (22), and (23) into problem (20), the problem can be reduced to
\[
\min_{\alpha} \quad t_{v2R} H_{\text{eNodeB}} + t_{v2R} H_{\text{coop}} + t_{v2s} (1 - H_{\text{eNodeB}} - H_{\text{coop}})
\]
\[= t_{v2s} - (t_{v2s} - t_{v2R}) H_{\text{eNodeB}} - (t_{v2s} - t_{v2R}) H_{\text{coop}}\]
\[\text{s.t. } \alpha \in (0, 1).\]

Eventually, this problem can be transformed into the following equation:
\[
\max_{\alpha} \quad (t_{v2s} - t_{v2R}) H_{\text{eNodeB}} + (t_{v2s} - t_{v2R}) H_{\text{coop}}\]
\[\text{s.t. } \alpha \in (0, 1).\]

This problem can be solved by the first derivative for the following two equations:
\[
\frac{dH_{\text{eNodeB}}}{d\alpha} = \frac{(K - 1) V_k (1 - \beta)}{K (Q^{1-\beta} - 1)} \left[ (\alpha V_k + V_c)^{1/\beta} - (\alpha V_k + K(1-\alpha)V_k + V_c)^{1/\beta} \right],
\]
\[
\frac{dH_{\text{coop}}}{d\alpha} = \frac{(K - 1) V_k (1 - \beta)}{K (Q^{1-\beta} - 1)} \left[ (1 - K)(\alpha V_k + K(1-\alpha)V_k + V_c)^{1/\beta} \right].
\]

Combining the two equations in (26), we can obtain the solution (27) to the original problem (17). The procedure can be seen as Algorithm 2, and here, we assume that the merge sort algorithm is adopted at line 3, and the time complexity of Algorithm 2 is \( O(K \ast Q \log Q) \), where \( K \) is the number of eNodeBs and \( Q \) is the total number of files.

\[
\alpha^* \approx \left( t_{v2R} - t_{v2R} \right)^{1/\beta} (KV_k + V_c) - \left( K t_{v2s} - (K - 1) t_{v2R} - t_{v2R} \right)^{1/\beta} V_c.
\]

\[
= \left( K t_{v2s} - (K - 1) t_{v2R} - t_{v2R} \right)^{1/\beta} V_k + \left( t_{v2R} - t_{v2R} \right)^{1/\beta} (K - 1) V_k.
\]


Algorithm 2: Optimal cache strategy.

5. Coded Delivery Strategy

When the cache node (for example, CH, eNodeB, or remote server) receives the user's request, it will transmit the requested file to the users according to the response model. However, it usually costs too many bandwidths and leads to a large delay for message transmitting due to the MAC layer collision, in particular, when many vehicle users exchange multimedia messages on a crowded road in a city scenario, which may lead to higher risk. Usually, when the cache node receives many requests within a certain period, it can respond to asynchronous requests through broadcasting to reduce the response time during a period. To further reduce the consumption of bandwidth resources by cache nodes, we propose an effective coded delivery scheme based on file coding and subfile coding [14, 16], which can minimize network load and reduce the ratio of redundant files or subfiles.

5.1. Subfile Coding. In this section, eNodeB is taken as an example to illustrate the data transmission strategy. According to (11), there are total $N_n$ users under the coverage of eNodeB $eNB_k$. We assume that the $n$th user requests the file $w_n (w_n \in Q)$, and all requests are denoted as a set $\{w_1, w_2, \ldots, w_{N_n}\}$ during a time, where $N_n$ is the real number of users that initiate a request. Obviously, the conditions $n \in [N_n]$ and $N_n \leq N_k$ are satisfied.

The cache file in eNodeB $eNB_k$ may be allocated among $N_n$ users, and the file $w_n$ can be composed of different subfiles $w_{n,S}$, that is, a subfile consisting of a set of bits and these bits are stored only in a specific user group $S$, where $S \subset [N_n]$, and $[N_n]$ represents a set of $N_n$ users [14]. When multiple users initiate requests at the same time, the eNodeB firstly reduces the duplicate files by using XOR coding. The set of files requested by all users under the coverage of eNodeB eNB $eNB_k$ after XOR coding is

$$W = \{w, w \in \oplus_{n \in [N_n]} w_n\}. \quad (28)$$

For the file $w_n$ requested by user $n$, eNodeB eNB $eNB_k$ only need to transmit all subfiles to users as follows:

$$f_n = \{w_{n,S}, S \subseteq W \setminus \{n\}\}. \quad (29)$$

Therefore, the set of subfiles that eNB $eNB_k$ needs to broadcast is

$$F = \cup_{n \in W} f_n. \quad (30)$$

To reduce the number of the subfiles to be transmitted, we can select a part of the subfiles $F' \in F$ and transform all the selected subfiles into a feasible data packet. As a consequence, the feasible data packet can be denoted as

$$p_i = \{\oplus_{w_j \in F'} w_{j,S}, F' \subseteq F\}. \quad (31)$$

Accordingly, we use $P_k = \{p_1, p_2, \ldots\}$ to denote the set of feasible data packets.

5.2. Problem Formulation of Coded Delivery. The construction of a feasible data packet, however, may ignore the fact that the same subfiles may appear repeatedly in different data packets, resulting in repeated data transmission and a waste of bandwidth. To avoid the above problems, a data packet with fewer redundant subfiles should be selected as much as possible when a feasible data packet is generated. For any feasible data packet $p_i \in P_k$, we firstly count the number of repetitions for each subfile. If a subfile $w_{j,S}$ appears $r(w_{j,S})$ times in set $P_k$ and $r(w_{j,S}) > 1$, the subfile is considered as a redundant subfile. In this way, the redundant times of subfile $w_{j,S}$ is recorded as $r(w_{j,S}) - 1$, otherwise 0. The redundancy times for all subfiles in each feasible data packet should be added up. The redundant ratio of the packet is calculated as follows:

$$R(p_i) = \frac{\sum_{w_{j,S} \in p_i} (r(w_{j,S}) - 1)}{\sum_{p_i \in P_k} |\{p_i\}|}, \quad (32)$$

where $|\{p_i\}|$ denotes the number of total subfiles in packet $p_i$.

The size of the feasible packet is defined as [14]

$$B(p_i) = \max_{w_{j,S} \in P_k} B(w_{j,S}), \quad (33)$$

where $p_i$ represents a feasible data packet, and $w_{j,S}$ is a subfile contained in the feasible data packet. We can use $x_i$ to indicate whether the feasible data packet $p_i$ is selected for transmission. If $p_i$ is selected, $x_i = 1$, otherwise 0.
otherwise 0. The vector \( X = \{ x_i \mid p_i \in P_k \} \) represents a set of decision variables \( x_i \). Thus, the problem is denoted as

\[
\min \sum_{p_i \in P} x_i B(p_i),
\]

(34a)

\[
\min \sum_{p_i \in P} x_i R(p_i),
\]

(34b)

\[
s.t. \bigcup_{p_i \in P, x_i = 1} p_i = F,
\]

(34c)

where (34a) and (34b) denote the size of the data to be transmitted and the redundancy ratio of the total transmitted subfiles by using the transmission decision variable vector \( X \), respectively. (34c) indicates that the selected subfiles need to satisfy the user’s request. According to [14], condition (34c) can be converted to a linear condition. To ensure each receiver can successfully decode the files they received, it needs to satisfy \( AX \geq 1, X \in \{0,1\}^{P_1} \), where \( I \) is a \( |F| \times 1 \) column vector. The feasible region of the problem can be defined as

\[
X = \left\{ x_i \in R^{P_1}, \sum_{a_j \in A} a_j x_i \geq 1, x_i \in \{0,1\} \right\},
\]

(35)

where \( A_{[P_1 \times P_1]} \) is a \( 0 \times 1 \) matrix, each row denotes a subfile to be sent, and each column denotes a feasible data packet. If the subfiles can be decoded from the data packet, the corresponding element in the matrix is 1, otherwise 0.

The problem formulation (34) is a multiple objective set covering problem (MOSCP), which is also an NP-hard problem and difficult to solve. Usually, the MOSCP is converted into a single objective problem [36, 37] to simplify the problem. We propose to use the weighted-max-ordering method [38] to convert problem (34) into the following form:

\[
\min \max \lambda_q z_q(x)
\]

s.t. \( x \in X \),

(36)

where \( \lambda_1, \lambda_2 \) represents the weight coefficients of the first and second objective functions of equations (34a) and (34b), respectively, \( \lambda_1 + \lambda_2 = 1 \), and \( z_q(x) \) is the \( q^{th} \) objective function in (34), \( q \in \{1,2\} \).

5.3. Algorithm Design. To reduce the network load and computing complexity, Asghari et al. [14, 15] proposed an efficient method to avoid obtaining the set of feasible data packets, namely, data packets with smaller bits and more subfiles are selected from the subfile set as much as possible. However, for a multiobjective optimization problem, this method cannot be used directly. To solve the multiobjective problem, many authors have proposed to use the cost/coverage ratio as the metric to select items when they face a multiobjective cover problem. The metric is usually defined as metric \( c_j^f \lambda_q |S_j| \) to select data from the original set, where \( |S_j| \) is the number of items in the selected subset, \( c_j^f \) is the coefficient of the \( q^{th} \) objective function, and \( \lambda_q \) is the weight factor of the \( q^{th} \) (\( q = \{1, 2\} \)) objective function [36–39].

The main idea of the algorithm is to find the subset (packet) that can cover as more as possible items (subfiles) from the original set \( E \) and select the subset with \( \lambda_2 c_j^f \) as small as possible. Similarly, the cost/coverage ratio can be used as a measure of our problem. Whenever a data packet is generated, a packet with a relatively small cost/coverage ratio should be selected as much as possible. We define the cost/coverage ratio vector \( \alpha_i \) of the packet \( p_i \) as

\[
\alpha_i = [\alpha_1^i, \alpha_2^i]^T = \left[ \frac{\lambda_1 B(p_i)}{|\{p_i\} \cap F|}, \frac{\lambda_2 R(p_i)}{|\{p_i\} \cap F|} \right]^T,
\]

(37)

where \( B(p_i) \) and \( R(p_i) \) are the size and ratio of redundant files for packet \( p_i \) in the objective function, \( \{p_i\} \) is the set of subfiles in the data packet \( p_i \), and \( \{p_i\} \cap F \) and \( |\{p_i\} \cap F| \) are the set of subfiles and the number of elements in the set \( \{p_i\} \) \( \cap F \), respectively. The lower the packet cost/coverage, the more likely that a packet is to be selected. For two packets \( p_i, p_j \) if the following conditions are satisfied:

\[
\max \left\{ \alpha_i = [\alpha_1^i, \alpha_2^i]^T \right\} \leq \max \left\{ \alpha_j = [\alpha_1^j, \alpha_2^j]^T \right\},
\]

(38)

the packet \( p_i \) will be chosen as a feasible packet.

5.4. Algorithm Description and Analysis. Based on the above analysis, we propose an efficient heuristic algorithm to solve problem (34) as shown in Algorithm 3. The lines 1-2 in Algorithm 3 denote the packets that the eNodeB needs to transmit and the remaining set of subfiles in subfile set \( F \), respectively. As observed from lines 3-11, it firstly uses the function Genpacket (\( \varepsilon \)) to generate the appropriate packets. From line 5 to line 10, it will judge whether each subfile of the potential packet exists in the remaining subfile set. If affirmative, the subfile will be added to the data packet and deleted from the remaining subfile set \( \varepsilon \). This procedure will continue until the remaining subfile set is empty. In the function Genpacket (\( \varepsilon \)), we can obtain an appropriate packet to transmit from...
requests from all users under the coverage of line 2 to line 11. The second line illustrates the number of possible subfiles we can build a set of feasible packets with the smallest possible size from line 2 to line 11. From line 12 to line 19, we will select the packet with the lowest cost/coverage ratio. Furthermore, the eNodeB can calculate the set of packets to be transmitted based on the subfile set. Finally, all these calculated data packets will be sent to users from eNBs, and the user that has made a request can decode the files they need based on the received files and the cached file by themselves. As Algorithm 3 will be terminated until \( \epsilon = \emptyset \), we can see that Algorithm 3 needs \( O(|F|N_{rk}^{2N+i}) \) operation, where \( F \) is the number of subfiles in the subfile set, and \( N_{rk} \) is the number of users that really initiate a request under the coverage of eNBk.

6. Cluster-Based Two-Level Cooperative Cache Deployment and Coded Delivery Strategy

In this section, we summarize the cache deployment and coded delivery as Algorithm 5. The first line shows the vehicle cluster procedure, and the second line is to cache the files on the edge node (CH or eNodeB). From line 3 to line 9, it illustrates the transmission strategy for the cache nodes. If the cache node CNi received several requests during \( T_r \) and has cached the requested files, it will first calculate the subfile

```plaintext
Function GenPacket \( \epsilon \)
\[ p = \emptyset; \]
Let \( M = \{ i \mid i \in [N], \exists w_{i,j} \in \epsilon, S \subseteq [N] \setminus \{i\} \}; \]
for \( T \subseteq M \&\& |T| \neq 0 \) do
  for \( j \in T \) do
    \( L_{i,j} = \{ w_{j,i} \mid w_{j,i} \in \epsilon, T \setminus j \subseteq S \}; \)
    Calculate \( V_{j,T} = \arg \min_{w \in L_{i,j}} B(w); \)
    \( P_T = P_T \cup \{ V_{j,T} \}; \)
  end
  \( P = P \cup \{ P_T \}; \)
end
\( \sigma_{p} = [\alpha_{p}, \sigma_{p}]^T = (+\infty,+\infty)^T; \)
for \( p_j \in P \) do
  \( \alpha_j = [\alpha_j^1, \alpha_j^2]^T = [(\lambda_p B(p_j) / |\{ p_j \} \cap \epsilon|), (\lambda_{p} R(p_j) / |\{ p_j \} \cap \epsilon|)]^T; \)
  if \( \max \{ \alpha_j^1, \alpha_j^2 \} \leq \max \{ \alpha_{p}^1, \alpha_{p}^2 \} \) then
    \( \alpha_{p} = \alpha_j; \)
    \( p = p_j; \)
  end
end
return \( p; \)
```

**Algorithm 4**

**Input:** Velocity \( v_i \), position \((x_i, y_i)\), number of files \( Q \), file size \( s_q \), cache capacity of vehicle \( V_c \), cache capacity of eNodeB \( V_k \), file request probability \( p_{k}^q \) under eNodeB \( k \), bias parameter of Zipf \( \beta \), transmission power \( P_t \), \( P_n \), \( P_r \), bandwidth \( B_j \) and \( B_l \), and cluster size \( N_k \).

**Output:** Cluster, cache decision, transmitted packet.

// Cache Deployment
Divide all vehicles into clusters and calculate the CH and CM according to Algorithm 1;
Cache the frequently requested file at all CHs and eNodeBs according to Algorithm 2;

// Coded Delivers
for any cache node \( CN_i \) received several requests within \( T_r \) do
  if node \( CN_i \) has cached the requested files then
    Set \( N \) users’ requests as \( \{ w_1, w_2, \ldots, w_N \} \);
    Calculate the subfile set \( F \) according to (30);
    Calculate the packet set \( P_j \) transmitted by \( CN_i \) according to Algorithm 3;
  end
end

**Algorithm 5:** Cluster-based two-level cooperative cache deployment and coded delivery strategy.
set $F$ according to the users’ requests and then calculate a set of packets $P'_i$ that needs to be transmitted by cache node $CN_i$. As there are $N_k$ vehicles under the coverage of eNB, Algorithm 1 needs $O(KN_k)$ operations, and Algorithm 2 needs to operate $O(KQ \log Q)$ times. Furthermore, as there are $N_{ck}$ clusters under the coverage of each eNodeB, thus, there are total $KN_{ck}$ cluster heads, it needs to operate $O(KN_{ck} |F|N_{ck}^{2N_{ck}})$ times, and the $K$ eNodeBs need to operate $O(K |F|N_{ck}^{2N_{ck}})$ times from step 3 to step 9 in Algorithm 5 according to Algorithm 3, where $N_{rk}$ is the total number of users that initiate a request in a cluster under the coverage of eNB, and $|F|$ is the number of subfiles in a cluster. Therefore, the total time complexity of Algorithm 5 is $O(K(N_{rk} + Q \log Q + N_{ck} |F|N_{rk}^{2N_{rk}} + F'N_{rk}^{2N_{rk}}))$.

7. Performance Evaluation

7.1. Parameter Setting. We simulate our algorithms by using NS-3 simulation software to compare other methods. The arrival process of vehicles follows a Poisson distribution, and the average velocity $\mu$ of vehicles is 40, 60, 80, 100, and 120 km/h, respectively, and the corresponding standard deviation $\delta$ is 10, 16, 22, 28, and 34. The minimum speed and maximum speed are $\mu - 3\delta$ and $\mu + 3\delta$ according to [33], respectively. Vehicle arrival rates are 600, 700, 800, 900, and 1000 vehicles/h, and the total number of files is 160, i.e., $Q = 160$. The cache capacity of eNodeB is set to 8, 16, 24, 32, and 40, and the cache capacity of eNodeB is set to 4, 8, 12, 16, and 20, respectively. The size of each file is set to 1, 2, 3, 4, and 5 MB. The vehicle communication radius is 200 m while the eNodeB transmission radius is 500 m. Furthermore, the road networks are generated by the SUMO software, we choose one road with length 2 km to conduct the simulation, and the parameter of the Zipf distribution is 0.5. The path loss among vehicles is $128 + 36.7\log_{10}(d)$ dB, where $d$ is the distance between the vehicle and CH or eNodeB, $d$ in km, and the power of noise power density is -174 dBm/Hz. Bandwidth $B_v = B_c$ is 10 MHz, and the transmitting power is set as 6 dBm. The detailed simulation parameters can be seen in Table 1.

We compare and validate our proposed method with some state-of-the-art schemes in terms of the following performance metrics:

1. **Cache Hit Ratio.** The ratio of the files satisfied by all cache nodes to all requests to the cache nodes
2. **Average Response Delay.** The average time interval from the user sending the request to receive data (short for)
3. **Network Load.** The network load that is downloaded from edge cache nodes and a remote server
4. **Ratio of Redundant Files.** The ratio of the received packets that contain redundant subfiles to the total received subfiles in all packets, we use redundant ratio in figures to represent the ratio of redundant files to simplify the expression

On the one hand, we compare our coded delivery method with COMP [23], “RSU-based cache only” [40], and “FSA-based cache” [41] in terms of a cache hit ratio, average response delay, and average network load. On the other hand, we compare our proposed method with “Optimal coded cache” [14] and “No coded-based method”, which includes COMP, “RSU-based cache only”, and “FSA-based cache” in terms of the ratio of the redundant files. Last but not the least, we will analyze the impact of cache size, size per file, velocity, and arrival rate on the four metrics.

7.2. Effect of Cache Size. As can be seen from Figure 4, we compare our proposed method with COMP, “FSA-based cache”, and “RSU-based cache only”. In addition, we set the cache size of eNodeB $V_k$ [8, 16, 24, 32, 40] and $V_v$ [4, 8, 12, 16, 20], respectively. The average velocity is 40 km/h, and the arrival rate of vehicles is set to 600 vehicles/h; the size per file is set as 1 MB. The number of the x axis on y axis represent the total cache size, i.e., $V_k + V_v$, and the performance metrics, respectively.

Figure 4(a) illustrates that the network load of the other three methods keeps constant and higher than our proposed method despite different cache sizes. Because our proposed coded delivery method does not only reduce redundant files requested by users but also eliminates the redundancy among subfiles, however, the total requests of the other three methods do not change due to the constant vehicle number given the fixed velocity and arrival rate.

Furthermore, in Figure 4(b), we can find that the ratio of redundant files of all methods will almost decrease slightly
with the increase of cache size on the edge nodes because the
probability of different users request the same file will
decrease with the increase of cache size.

From Figure 4(c), it is easily observed that our proposed
method outperforms the other three methods in terms of
hit ratio. It can be explained that our proposed method con-
siders a two-level edge cache; for instance, cluster head and
eNodeB can cache the most requested files according to the
request probability. COMP also adopts cluster heads to cache
the most popular files and part of less popular files coopera-
tively, but there is no cooperation among RSUs; thus, the hit
ratio is lower than ours. The “FSA-based cache” method and
“RSU-based cache only” are worse than our proposed
method and COMP due to the lack of edge cache. On the
other hand, the hit ratio “FSA-based cache” is higher than
that of “RSU-based cache only” because RSU also considers
the cooperation cache among RSUs though it ignores the
cooperation among cluster heads in “FSA-based cache.”

In Figure 4(d), the average response delay of the four
methods is compared. On the one hand, the average response
delay of all methods will decrease with the increase of cache
size, because more requests will be satisfied by the cache node
if the cache size is large enough. On the other hand, it can be
observed that our proposed method can obtain the least aver-
age response delay because we cache the popular file on the
edge node as much as possible, where the response delay is
smaller compared to the response delay from the remote
server. The average response delay of COMP decreases
quickly because more requests will be satisfied by cluster
heads and eNodeB when the cache size increase although
the request is mainly satisfied by a remote server at the begin-
ing. As for “FSA-based cache” and “RSU-based cache only”,

Figure 4: Effect of cache size $V_k$ and $V_c$ on system performance.
the average response delay is much larger than our proposed method, because they do not consider caching files on cluster head cache, all the requests will be satisfied from eNodeB or remote server. Furthermore, the average response delay of the “FSA-based cache” outperforms that of “RSU-based cache only”, because “FSA-based cache” has also considered the cooperation among eNodbBs. From the above figures, we can easily find that cache size has a great impact on the hit ratio and response delay, and the impact of cache size on the network load and redundant ratio is not so obvious.

7.3. Effect of Size per File. In this subsection, the capacity of eNodeB and CH is $V_k = 40$ and $V_c = 20$, respectively. The average velocity is 40 km/h, and the arrival rate of vehicles is set to 600 vehicles/h, and $Q = 160$. The size per file changes from 1 MB to 5 MB.

From Figure 5(a), we can find that our proposed method can achieve better performance in terms of network load compared with COMP, “FSA-based cache”, and “RSU-based cache only.” The network load of COMP, “FSA-based cache”, and “RSU-based cache only” almost increase with the size per file linearly, because the total number of requests is almost the same given the mean velocity and arrival rate, and the network load will increase with the increase of the size of per file. Our proposed method adopts the coded delivery method to reduce the data traffic so that it can outperform the other three methods.
From Figure 5(b), it can be observed that the ratio of redundant files of our proposed method outperforms the “Optimal coded cache” and “No coded-based method” methods, as it not only considers reducing the ratio of redundant files between files and also reduces the ratio of redundant files between subfiles. “No coded-based method” is the worst in terms of the ratio of redundant files because they do nothing to reduce the redundancy among files.

As can be seen from Figure 5(c), the hit ratio of four methods almost keeps the same under different sizes per file, as the hit ratio has nothing to do with size per file. Furthermore, the hit ratio of our proposed method is larger than the other three methods as our proposed method adopts the two-level caching strategy. COMP is also better than the other two methods as it takes advantage of the RSU cache cooperatively.

From Figure 5(d), it can be observed that the average response delay of the “RSU-based cache only” is larger than any other methods despite different sizes per file. Furthermore, the performance of “FSA-based cache” is better than that of “RSU-based cache only” since it considers the cooperation among RSUs. COMP and our proposed method can almost achieve the same performance and better than “FSA-based cache” and “RSU-based cache only”, as they obtain the request files mainly from edge cache node; thus, the average delay is smaller than that of “FSA-based cache” and “RSU-based cache only.” From the above discussion, we can find that the average response delay linearly grows with the size per file for all methods, since we assume that the response delay for each unit file is fixed. It can also be found that size per file has a large influence on the average response delay and network load, but little impact on the redundant ratio and hit ratio.

7.4. Effect of Velocity of Vehicle. In this subsection, we compare our proposed method with other methods and analyze the impact of the mean velocity of vehicles on the performances. We set the size per file as 1 MB, \( V_k = 8 \), and \( V_v = 4 \). The arrival rate of vehicles is set to 600 vehicles/h, and the average velocity changes from 40 km/h to 120 km/h.

As can be seen from Figure 6(a) that the network load for all methods decreases with the mean velocity because the larger the mean velocity is, the smaller the cluster size and the total number of vehicles on the road is, however, the number of clusters will increase. Thus, the request and response data traffic from all vehicles will decrease with the increase of mean velocity. Though the reduction is not significant for our proposed method, our proposed method outperforms the other three methods in terms of network load due to the coded delivery method.

From Figure 6(b), we can find that the ratio of redundant files of “No coded-based method” cache and “Optimal coded cache” increases slightly with the increase of velocity, and our proposed method increases rapidly but has better redundant performance than the other two methods. The reason is that when the average velocity increases, there are more clusters and smaller cluster sizes under the coverage of eNodeB, which means that there are more CH that can cache popular files. Each CH under the coverage of eNodeB will cache the same files, and our proposed method is more dependent on the edge cache, which causes a sharp increase in the ratio of redundant files in our proposed method when the arrival rate is high.

From Figure 6(c), it can be easily found that our proposed method can achieve the highest hit ratio; the second one is COMP. With the increase of velocity, there will be more clusters and each CH can cache some files, and our proposed method and COMP can take advantage of the edge cache; thus, their hit ratio will grow significantly with the increase of velocity. However, “FSA-based cache” and “RSU-based cache only” mainly depend on the cache of eNodeB; thus, their hit ratio almost keeps constant.

In Figure 6(d), we compare our proposed method with the other three methods in terms of the average response delay. When the mean velocity increases, there will be more clusters and smaller size of the cluster; thus, many files can be cached on the CHs, which will reduce the response time greatly and improve the hit ratio for our proposed method and COMP; of cause, our proposed method still obtains the lowest response delay. Furthermore, we can find the average response delay of “FSA-based cache” and “RSU-based cache only” slowly decreases as the total number of vehicles and requests will decrease with the increase of mean velocity. From the simulation, it can be found that the mean velocity has a great impact on all performance metrics.

7.5. Effect of Arrival Rate. In this subsection, we will compare our proposed method with the other three methods and analyze the impact of the arrival rate of vehicles on the performances. In this subsection, we set the size per file 1 MB, \( V_k = 24 \), and \( V_v = 12 \). The average velocity is 40 km/h, the file number is set as \( Q = 160 \), and the arrival rate of vehicles changes from 600 vehicles/h to 1000 vehicles/h.

From Figure 7(a), it is observed that the network load for the other three methods increases when the arrival rate of vehicles also increase, which can be explained by the fact that the larger the arrival rate of vehicles will lead to more vehicles on the road given the mean velocity; consequently, the request and the total traffic will also increase. However, the arrival rate of vehicles almost does not put any impact on our proposed method in terms of network load, that is because we adopt the coded delivery method.

From Figure 7(b), we can find that the ratio of redundant files in our proposed method is the lowest among all methods. Though the “Optimal coded cache” and our proposed method have adopted coded delivery method to reduce the ratio of redundant files. What is more, our proposed method also focuses on reducing the ratio of redundant files among subfiles; thus, our proposed method can outperform “Optimal coded cache.” “No coded-based method” is still the worst method in terms of the ratio of redundant files as it does nothing to reduce the redundant files.

In Figure 7(c), we compare the hit ratio with the other three methods and analyze the impact of the arrival rate on the hit ratio. First of all, we find that the hit ratio of all four methods will decrease with the increase in the arrival rate. The reason is that when the arrival rate increase when the mean velocity is given, the total number of vehicles and the
Cluster size on the road will also increase; however, the number of the cluster under the coverage of each eNodeB will decrease, which will lead to the decrease of the contents on the edge nodes; therefore, the hit ratio will decrease, especially for our proposed method and COMP. The hit ratio of “FSA-based cache” and “RSU-based cache only” almost keep the constant, because their hit ratio mainly depends on the cached files on eNodeB (RSU); furthermore, we can also find that our proposed method still can obtain the highest hit ratio.

From Figure 7(d), it can be easily found that the average response delay will grow with the increase of arrival rate, and this can also be explained that a larger arrival rate will lead to larger cluster size and less number of cluster, which leads to a decrease in the hit ratio, and more requests will be responded by a remote server. Our proposed method outperforms other methods because it can make the most of the edge cache advantage and cooperatively cache on neighboring eNodeBs, and it will obtain the most files from the edge nodes. CH cache is also considered in COMP; thus, its average response delay is better than that of “FSA-based cache” and “RSU-based cache only.” When the arrival rate increase, more vehicles will compete to request the files from the limited space of eNodeB, and the average response delay for “FSA-based cache” and “RSU-based cache only” will almost linearly grow with the increasing of arrival rate. In summary, we can find that the arrival rate of vehicles has a great impact on the network load, hit ratio, and average response delay.
8. Conclusion

In this paper, we have designed an effective cluster method for reliable communication and proposed a two-level cooperative cache deployment approach based on the CH and eNodeB cache to reduce the response delay and improve the hit ratio. In the proposed approach, eNodeBs can cooperatively cache the same and different files to further improve the hit ratio and reduce the response time, and then, we derive an optimal cache deployment strategy for eNodeB and CH is derived. Lastly, we have proposed a coded delivery method based on file coding and subfile coding to reduce the network load and reduce the response delay. In terms of the simulation results, it is unveiled that a larger cache size, smaller size per file, smaller mean velocity, and arrival rate can improve the system performance. On the other hand, our proposed method can greatly reduce the network load and the response delay and improve the ratio of redundant files and hit ratio. In future work, we will focus on a multihop transmission strategy based on edge cache and a coded delivery strategy in the 5G New Radio (NR) V2X system.

Data Availability

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

The authors declare that there is no conflict of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61772385).

References


