Online Service Request Duplicating for Vehicular Applications

Qing Li, Xiao Ma, *Member, IEEE*, Ao Zhou, *Member, IEEE*, Changhee Joo, *Senior Member, IEEE*, and Shangguang Wang, *Senior Member, IEEE*

Abstract—Vehicles on roads have increasingly powerful computing capabilities and edge nodes are being widely deployed. They can work together to provide computing services for onboard driving systems, passengers, and pedestrians. Typical applications in vehicular systems have service requirements such as low latency and high reliability. Most studies in vehicular networks concerning latency and reliability focus on vehicular communication at the network level. Based on these fundamental works, an increasing proportion of vehicles boast complex applications that require service-level end-to-end performance guarantees. Several works guarantee service-level latency or reliability while new and innovative applications are demanding a joint optimization of the above two metrics. To address the critical challenges induced by the joint modeling of latency and reliability, system uncertainty, and performance and cost trade-off, we employ service request duplication to ensure both latency and reliability performance at the service level. We propose an online learning-based service request duplication algorithm based on a multi-armed bandit framework and Lyapunov optimization theory. The proposed algorithm achieves an upper-bounded regret compared to the oracle algorithm. Simulations are based on real-world datasets and the results demonstrate that the proposed algorithm outperforms the benchmarks.

Index Terms—Vehicular edge computing, service-level latency, service-level reliability, service request duplication

1 Introduction

Vehicles on the road have increasingly powerful computing capabilities. One global forecast is that total electric vehicle sales would secure approximately 32% of the total market share for new car sales by 2030.[1] Electric vehicles usually have powerful computing capabilities together with dozens of cameras, sensors, and storage systems, which makes them supercomputers on wheels. Nivida, a manufacturer of chips used in autonomous vehicles, says a self-driving car can have the equivalent computing power of 200 laptops.[2] What if vehicles, in addition to serving as transportation, also offer computing services? Meanwhile, the key enabler of intelligent transportation, 5G infrastructure has developed rapidly. For example, China so far has already built more than 1.15 million 5G base stations.[3] Besides, many companies invest heavily in edge computing as a key pillar for their overall 5G rollout [1], [2]. Thus, service requests from driving systems, passengers, and pedestrians can get computing services from nearby vehicles and edge servers.

The prime focus of vehicular applications is to guarantee quality of service (QoS) requirements such as latency or reliability. Moreover, the joint optimization of both latency and reliability has become the norm in the context of 5G Ultra-Reliable and Low-Latency Communication (URLLC) for vehicular networks. Existing studies in [3], [4], [5], [6], [7], [8] addressing these requirements focus on vehicular communication at the network level, which are fundamental. They aim to complement physical-level techniques such as automatic repeat request scheme, and its hybrid version at the medium access layer. However, an increasing proportion of vehicles boast complex applications such as autonomous driving, navigation, and other safety diagnostics, which have service-level end-to-end performance requirements [9]. Due to vehicular mobility and dynamic computation resource availability, the service-level performance may still suffer from critical uncertainty despite reliable network-level performance. Such observation leads to the service-level guarantees for vehicular applications [10], which requires looking into the whole service process of both the communication part and computation part. Also, recent works enhance service reliability [11] and minimize service latency [12], [13], [14], which guarantee service-level performance very well. Yet increasingly new and innovative vehicular applications require both service-level reliability and latency guarantee [10]. Hence, we are motivated to guarantee service-level latency and reliability jointly, which is non-trivial due to the following challenges.

1. https://www2.deloitte.com/us/en/insights/focus/future-of-mobility/electric-vehicle-trends-2030.html
2. https://www.wired.com/story/use-self-driving-cars-supercomputers/
3. https://www.statista.com/topics/6705/5g-technology-in-china

- *Qing Li, Xiao Ma, Ao Zhou, and Shangguang Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: {q_li, maxiao18, aozhou, sgwang}@bupt.edu.cn.*
- *Changhee Joo is with the Department of Computer Science and Engineering, Korea University, Seoul 02841, South Korea. E-mail: changhee@korea.ac.kr.*

Manuscript received 14 February 2021; revised 30 November 2021; accepted 27 January 2022. Date of publication 4 February 2022; date of current version 5 June 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1805500, in part by the National Science Foundation of China under Grants 62032003, 61922017, 61921003, and 61902036.
(Corresponding author: Shangguang Wang.)
Digital Object Identifier no. 10.1109/TMC.2022.3148170

First, there can be conflicts between latency and reliability performance in vehicular networks [15] because reliability enhancement mechanisms such as retransmissions can increase latency. It is hard to jointly enhance both latency and reliability performance. Second, service-level latency and reliability characterize different statistical properties of the latency distribution. Service-level latency represents the expectation of the latency distribution while reliability focuses on extreme events with low occurrence probabilities. It is challenging to quantify the correlation between service-level latency and reliability. Third, optimizing service-level latency and reliability for vehicular applications suffers from inherent uncertainty due to unpredictable vehicle mobility, the fluctuating wireless environments, and heterogeneous vehicle computing capabilities.

For the first challenge, we adopt an intuitive solution, i.e., service request duplication, to jointly improve service-level latency and reliability. If we duplicate a service request and send them to multiple server vehicles, the service request can be completed more quickly when the vehicles have a different amount of idle resources. A service request is considered completed if one of the duplications finishes and such an At-Least-One rule can improve service-level reliability when one or several requests fail. Service request duplication has its superiority especially in vehicular systems with fluctuating wireless environment and dynamic computation resource accessibility. The idea is to guarantee performance at the cost of resource redundancy. With increasing powerful computing capabilities in vehicles, service request duplication becomes feasible because the huge array of onboard capabilities are often underutilized [11]. For the second challenge, we construct a joint model of service-level latency and reliability and investigate the correlation between them based on the joint model. For the third challenge, we propose an online learning-based service request duplication algorithm to address the exploitation-exploration tradeoff in the face of system uncertainty as well as the performance and cost tradeoff incurred by service request duplication.

The main contributions are summarized as follows:

- We present a joint model of service-level latency and reliability for vehicular applications and investigate the correlation between them based on the joint model.
- We formulate the problem as a combinatorial Multi-armed Bandit (MAB) problem with long-term cost and reliability constraints and adopt the Lyapunov optimization technique to properly tradeoff the QoS guarantee and system resource cost.
- We propose an online learning algorithm to minimize service-level latency under high-reliability and system resource cost constraints. Further, we rigorously prove that the proposed algorithm has a cumulative *regret* (learning loss) of $O(\sqrt{T\log T})$.
- We carry out extensive simulations using the real-world datasets of Shanghai Taxi Trace and Shanghai Telecom's Base Station. The proposed algorithm outperforms benchmark algorithms in the simulations.

The remainder of this paper is organized as follows. We introduce the related work in Section 2. We describe the system model in Section 3. And we formulate the problem in Section 4. In Section 5, we design the online learning algorithm. The simulation results are shown in Section 6. We conclude in Section 7.

2 Related Work

In this section, we analyze the related works which can be divided into three categories: ultra-reliability and low-latency communication, service duplication, and task offloading in Vehicle Edge Computing (VEC).

Ultra-Reliable and Low-Latency Communication. Most VEC studies [3], [4], [5], [6], [7], [8] addressing latency and reliability focus on vehicular communication at the network level. Samarakoon *et al.* [3] propose a resource allocation algorithm that minimizes the network-wide power consumption of vehicular users subject to high reliability in terms of probabilistic queuing delays. By exploiting the benefits of the massive multiple-input multiple-output concept, Yang *et al.* [4] propose a two-stage radio resource allocation algorithm based on a novel twin timescale perspective to avoid the frequent exchange of near-instantaneous channel state information. Abdel-Aziz *et al.* [5] propose an age of information-aware transmission power and resource block allocation algorithm to balance a tradeoff between minimizing the probability that the vehicles' age of information exceeds a predefined threshold and maximizing the knowledge about the network dynamics. They [6] further develop a novel framework to characterize and optimize the tail of the age of information in vehicular networks. Liu *et al.* [7] propose a power minimization algorithm based on extreme value theory to satisfy second-order statistical constraints on reliability. These works investigate algorithms of improving the reliability or latency of vehicular networks. Moreover, Ge [8] proposes a joint function to evaluate the joint impact of latency and reliability in vehicular networks. As service-level performance is inherently dependent on network-level performance, these works are fundamental to guarantee service-level requirements. Base on these works, we focus on the uncertainty in the whole service process including both the communication and computation parts.

Service Duplication. Previous works focus on service duplication to reduce latency. Vulimiri *et al.* [16] exploit redundancy to achieve reduced latency (especially the tail of the latency distribution) by using extra computing capacity. Joshi *et al.* [17] use queue theory to analyze the latency and cost for queueing tasks in cloud computing systems, which provide significant insight to combat latency variability in various servers. Niknam *et al.* [18] design an algorithm to determine the replication factor for each task in the acyclic SDF graph of streaming applications to improve the utilization of processors. Chang *et al.* [19] dynamically allocate server replicas based on the number of read/write operations for mobile edge computing. Choudhury *et al.* [20] adopt proactive sensing to detect the necessity of duplication before developing the service duplication scheme. Li *et al.* [21] exploit computation duplication in mobile edge computing networks to speed up the results downloading. These works study how to reduce service latency in cloud computing or mobile edge computing while we focus on the latency and reliability in VEC scenarios which are more challenging with inherent system uncertainty.

Task Offloading in VEC. Computation tasks in VEC can be divided into two categories: offloading to Edge Nodes (ENs) and offloading to vehicles. For offloading to ENs, the related works are summarized as follows. Tang *et al.* [22] decide the network selection and task offloading simultaneously based on the characteristics of vehicle mobility to minimize the task execution latency. Zhang *et al.* [23] propose a software-defined-networking based load-balancing task offloading scheme in fiber-wireless enhanced VEC to minimize the task execution latency. Batewela *et al.* [24] introduce a concept of risk to measure reliability in VEC and study risk minimization for vehicles' task completion latency. Liao *et al.* [25] design an intent-aware task offloading strategy which can provide QoE and reliable URLLC guarantees. Guo *et al.* [26] design an intelligent task offloading scheme based on deep Q-learning to adapt to fast changing VEC environment. Barbosa *et al.* [27] propose to offload tasks from vehicles using IEEE 802.11p and 5G network interfaces simultaneously. All these works study offloading the computation task to a single edge node, which is different from our duplicating service requests to multiple vehicles.

For offloading to vehicles, tasks can be offloaded from vehicles to vehicles directly [12], [28], [29] or collected by the ENs and then assigned to the server vehicles centralized [11], [13], [30]. Sun *et al.* [12] propose a learning-based task offloading algorithm to minimize the average offloading delay. An adaptive learning-based task offloading algorithm with linear complexity and sublinear regret has been developed [28]. Zhou *et al.* [29] propose a low complexity and stable task offloading mechanism to minimize the total network delay based on the pricing-based matching. The performance of these distributed schemes may be limited by the moving directions and velocities of vehicles and less coordination among task vehicles. With the help of ENs, task offloading in VEC systems can take place in a larger region. An alternative way is to make offloading decisions by the ENs. Chen *et al.* [11] propose an algorithm based on a MAB framework to address the inherent issues in VEC systems including uncertainty of vehicle movement and volatile vehicle members. Jiang *et al.* [13] propose centralized resource allocation schemes based on the Markov decision process with the coordination of ENs. Zhou *et al.* [30] exploit the mobility with opportunistic computation offloading and service request duplication. However, the Markov decision process based approaches usually suffer from the curse of dimensionality.

The aforementioned decentralized and centralized schemes on task offloading in VEC systems focus on reliability and latency. Different from these works, we jointly consider latency and reliability in the same optimization model, which is vital for advanced vehicular applications with high reliability and low latency requirements.

3 System Model

We consider a VEC system with moving vehicles on roads and a set of ENs colocated with base stations covering the main roads. Moving vehicles are classified into Task Vehicles (TaVs) and Server Vehicles (SeVs). TaVs have service requests that need to be offloaded for processing. SeVs have surplus computing resources which are pooled to provide computing service. We divide SeVs into N types where each SeV belongs to a single type based on its mobility information (direction, velocity), computation capabilities, and workload status. Note that the role of each vehicle can change over time, depending on whether it has surplus computing resources. ENs collect service requests from TaVs in its radio range and distribute them to available SeVs.

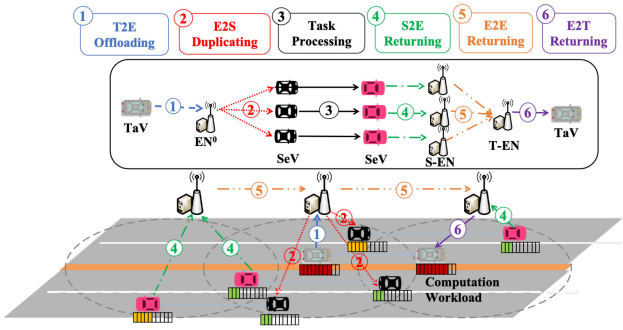


Fig. 1. Service request duplication in VEC systems.

For any EN, let $\{1, 2, \ldots, T\}$ be the service request sequence in the order of their arrival. Each request t is denoted by a tuple $(x_t, y_t, w_t, L_t^{\max})$, where x_t denotes the size of request input data, y_t is the size of the result, w_t is the numbers of CPU cycles required to complete the request and $L_t^{\max}$ is the deadline. A request is completed if the TaV receives the result winin $L_t^{\max}$, otherwise, it fails. For different requests, available SeV types change over time due to vehicle mobility, fluctuating wireless environment, and unpredictable computation resource accessibility. We use $\mathcal{V}_t$ to denote the available SeV types and $P(\mathcal{V}_t)$ the distribution of available SeV types, which is assumed to be i.i.d. for different requests. This distribution is unknown a priori, but the set of available SeV types $\mathcal{V}_t$ will be revealed to the EN at the beginning of the decision round for each task request.

To counter the uncertainty, we introduce the service request duplication technique. The overall procedure is described in Fig. 1: 1) Once an EN receives a service request from a TaV, 2) it selects multiple SeV types (in the available SeV type set $\mathcal{V}_t$) and sends it to them through duplications. 3) Each selected SeV processes the duplicated request using its available computation resources. 4) When the result is ready, the vehicles may move and the SeV is now associated with a different EN, denoted by S-EN. Then the result is sent to S-EN, and 5) relayed to T-EN that is the EN currently associated with TaV. 6) Finally, T-EN forwards it to TaV. We name each request copy as a *duplication* and we use $v, (1 \leq v \leq N)$ to denote the SeV type processing the duplicated request. For simplicity, SeV v refers to the SeV of type v in the following.

3.1 Service-Level Latency

Service-level latency is defined as the time interval from request generating to result returning. Let $L_{t,v}$ denote service-level latency of SeV v. It consists of four parts.

1) *TaV-to-EN (T2E) transmission latency*: when a TaV generates a service request, it connects to a nearby EN, says

EN^0, and offloads its task via the wireless connection. The T2E transmission latency can be given as $L_t^{\mathrm{TE}} = x_t / r_t^{\mathrm{TE}}$, where r_t^{TE} is the transmission rate for request t from TaV to EN^0. Note that EN^0 knows L_t^{TE} by observing the timestamps of data packets defined by network time protocol.

2) *EN-to-SeV (E2S) task transmission latency*: EN^0 identifies the available SeV types in its coverage based on the link condition and selects multiple types of SeVs to offload the task. The E2S transmission latency is $L_t^{\mathrm{ES}} = x_t / r_t^{\mathrm{ES}}$. The proposed algorithm works with different E2S transmission models where the transmission rate r_t^{ES} can be known or unknown to EN^0.

3) *Computation latency*: let $f_{t,v}$ be the available CPU frequency allocated by SeV v for request t, which is unknown to EN^0 a priori. Then, the computation latency can be simply obtained by $L_{t,v}^{\mathrm{C}} = w_t / f_{t,v}$.

4) *Result returning latency*: The SeV sends the result back to the TaV via S-EN and T-EN, as shown in Fig. 1. S-EN and T-EN are determined independently by the mobility of individual vehicles. Let $L_{t,v}^{\mathrm{ST}}$ be the result returning latency of SeV v. It consists of three parts: 1) the transmission latency between SeV v and S-EN $L_{t,v}^{\mathrm{SE}} = y_t / r_{t,v}^{\mathrm{SE}}$, where $r_{t,v}^{\mathrm{SE}}$ is the transmission rate; 2) the backhaul transmission delay between S-EN and T-EN $L_{t,v}^{\mathrm{EE}} = y_t / r_t^{\mathrm{EE}}$, where r_t^{EE} is the backhaul transmission rate. If S-EN and T-EN are the same, then $L_{t,v}^{\mathrm{EE}} = 0$; 3) the latency for transmitting results between T-EN and TaV $L_t^{\mathrm{ET}} = y_t / r_t^{\mathrm{ET}}$, r_t^{ET} is the fixed transmission rate operated by T-EN. The result return latency of SeV v can be obtained as $L_{t,v}^{\mathrm{ST}} = L_{t,v}^{\mathrm{SE}} + L_t^{\mathrm{EE}} + L_t^{\mathrm{ET}}$.

Therefore, service latency of SeV v is $L_{t,v} = L_t^{\mathrm{TE}} + L_t^{\mathrm{ES}} + L_{t,v}^{\mathrm{C}} + L_{t,v}^{\mathrm{ST}}$. It is a black box to EN^0 who makes the duplication decision: some parts of service latency are unknown to EN^0 (e.g., the computation latency $L_{t,v}^{\mathrm{C}}$ and the result return latency $L_{t,v}^{\mathrm{ST}}$), due to the uncertainty in vehicle computation capability, vehicle movement. For each request t, EN^0 chooses a subset of SeV types from $\mathcal{V}_t$ for request t, and we call the subset $\mathcal{A}_t \subseteq \mathcal{V}_t$ the duplication set for request t. In $\mathcal{A}_t$, ENs only choose at most one SeV for each type and send the task request to an arbitrary SeV of the same type. The service-level latency of request t is defined as

$$L_t = \min_{v \in \mathcal{A}_t} L_{t,v}. \tag{1}$$

If TaV receives the result before the deadline, i.e., $L_t \leq L_t^{\max}$, then request t is completed. Other slower SeVs do not cancel the duplication of this request upon its completion for two reasons. On one hand, canceling requests requires TaVs to exchange additional information with SeVs, which introduces additional latency [31]. On the other hand, the latency performance of all the selected SeVs can be exploited to enhance the proposed reinforcement learning algorithm, which will be introduced in Section 4.

3.2 Duplication Cost

Sending each request to all the available SeVs will lead to resource waste. Vehicles running on roads are energy-constraint, especially for electric vehicles. With intensive computations onboard (which consumes almost 10% energy for the current Tesla Model 3), a fully-charged vehicle is estimated to last much less than the standard mileage [32]. Moreover, improper request duplication could congest the VEC system and further degrade the experienced latency of SeV's own tasks. Hence, we limit the duplication cost. We define the duplication cost of request t as the total computation time of all the request duplications [33], i.e.,

$$C_t = \sum_{v \in \mathcal{A}_t} L_{t,v}^{\mathrm{C}}. \tag{2}$$

Our duplication cost is basic, which can be extended to other practical costs such as energy consumption or rent.

3.3 Service-Level Reliability

A widely adopted notion of reliability for wireless communications and standardization bodies as 3GPP is a probabilistic bound over the latency. Following that, service-level reliability is defined as the probability that service-level latency of request t exceeding a threshold given in the service requirement [10]

$$\Pr(L_t > L_t^{\max}) \leq \epsilon, \tag{3}$$

where the outage probability ϵ varies from 10^{-1} to 10^{-9} for different QoS requirements [34].

Note that the above reliability constraints cannot cope with the extreme cases when $L_t > L_t^{\max}$. These extreme cases essentially correspond to the worst cases which are the key determinant of the reliability performance and should be properly addressed. The conventional average based approaches are inadequate for addressing extreme cases and we need to take into account low violation probability, tail (decay) behavior of the complementary cumulative distribution function (CCDF), threshold deviation and its higher-order statistics. To analytically understand these metrics and statistics, extreme value theory [35] is a powerful extreme event control framework.

Let M_t be samples of exceeded value $X_t = L_t - L_t^{\max}$ conditioned on service-level latency $L_{t,v}$ exceeds the deadline $L_t^{\max}$. By enforcing the constraints

$$\lim_{t \to \infty} \sum_{t=1}^{T} X_t \mathbb{I}_{L_t} / \sum_{t=1}^{T} \mathbb{I}_{L_t} \leq \mathbb{E}[M_t], \tag{4}$$

$$\lim_{t \to \infty} \sum_{t=1}^{T} Y_t \mathbb{I}_{L_t} / \sum_{t=1}^{T} \mathbb{I}_{L_t} \leq \mathbb{E}[M_t^2], \tag{5}$$

we can control the fluctuations of service-level latency and maintain its extreme values below the desired threshold. Here, $Y_t = (L_t - L_t^{\max})^2$ and $\mathbb{I}_{L_t}$ is the request completion indicator. If $L_t > L_t^{\max}$, $\mathbb{I}_{L_t} = 1$ (failed), otherwise, $\mathbb{I}_{L_t} = 0$ (completed). The samples M_t can be seen as extreme statistics and characterized by extreme event theory. Assume that service-level latency L_t follows independent and identical distributions. The Pickands-Balkema-de Haan Theorem [35] shows that the conditional CCDF of the exceeded value M_t can be approximated by a generalized Pareto distribution $G(x, \sigma, \xi)$, i.e.,

$$G_M^{\mathbf{d}}(x) = \begin{cases} \frac{1}{\sigma}(1 + \frac{\xi x}{\sigma})^{-1-1/\xi}, & \xi \neq 0 \\ \frac{1}{\sigma}\exp(-\frac{x}{\sigma}), & \xi = 0 \end{cases}, \tag{6}$$

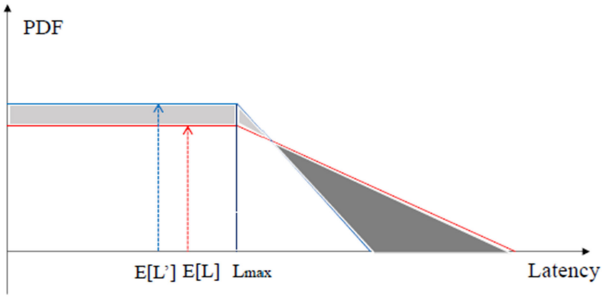


Fig. 2. The correlation between service-level latency and reliability.

where $\mathbf{d} = (\sigma, \xi)$, $\sigma(>0)$ and ξ are the scale and shape parameters designed by ENs, and $x \geq 0$ if $\xi \geq 0$, otherwise, $0 \leq x \leq -\sigma/\xi$. Hence, constraints (4), (5) are rewritten as

$$\lim_{t\to\infty} \sum_{t=1}^{T} X_t \mathbb{I}_{L_t} / \sum_{t=1}^{T} \mathbb{I}_{L_t} \leq \sigma/(1-\xi), \tag{7}$$

$$\lim_{t\to\infty} \sum_{t=1}^{T} Y_t \mathbb{I}_{L_t} / \sum_{t=1}^{T} \mathbb{I}_{L_t} \leq 2\sigma^2/(1-\xi)(1-2\xi). \tag{8}$$

3.4 Correlation of Service-Level Latency and Reliability

As both service-level latency and service-level reliability are related to latency, then it inevitably raises a question: What's the correlation between them? The answer is reliability is consistent but not identical with the latency and the reason is analyzed as follows. Service-level latency focuses on the average value of latency while service-level reliability concerns extreme latency values. In specific, due to the uncertainty of VEC systems, service-level latency can be treated as a random variable with unknown distribution. The time-average latency is the expectation while the reliability restricts the tail of the distribution. They work on different statistics metrics of the latency distribution.

When we restrict exceeded values, the latency distribution is shaped as a more light-tailed distribution. We give an example to show how reliability contributes to reducing the average latency. In Fig. 2, the red curve represents the original latency distribution $f_{PDF}(X)$ with the expectation $E[L]$. The area enclosed by the red curve and coordinate axis is 1. When we add constraints on the exceeded value, the latency distribution is shaped as $f'_{PDF}(X)$ represented by the blue curve with expectation $E'[L]$. Since the area enclosed by the blue curve and coordinate axis is also 1, the area of the two shadows are the same. So we have $E'[L] < E[L]$. Similarly, minimizing the average latency contributes to enhancing the reliability. Our analysis above is supported by Markov's inequality, which says: the heavier the tail, the larger the expectation.

Lemma 1. *(Markov's inequality). Let $x > 0$ be a non-negative random variable. Then, for all $b > 0$*

$$\Pr(x > b) \leq \frac{\mathbb{E}[x]}{b}. \tag{9}$$

4 Problem Formulation and Analysis

Given a total of T requests, our objective is to minimize the average service-level latency of requests under the cost and reliability constraints by deciding the duplication set $\mathcal{A}_t$ without a priori information of service-level latency and duplication cost. The problem is formulated as

$$\begin{aligned} \text{P1}: \quad & \min \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} L_t \\ & s.t.\ \text{C1}: \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} C_t \leq C^B, \\ & \text{C2}: (3), (7), (8). \end{aligned} \tag{10}$$

The constraint C1 is the long-term average cost constraint with maximum cost, C^B. C2 is the service-level reliability constraint based on extreme event theory. There is a significant challenge to directly solve P1 since the long-term cost and reliability constraints couple the duplication decisions across different requests: using more duplication cost (violating the reliability requirement) for the current request will potentially reduce the cost (make the reliability requirement more stringent) for future requests, and yet the decisions have to be made without foreseeing the future. To address this challenge, we leverage Lyapunov optimization [36] to solve a deterministic problem for each request, while adaptively balancing the latency performance and cost over requests.

4.1 Problem Transformation

By following the Lyapunov optimization framework, we construct a virtual duplication cost queue as

$$Q_0(t+1) = Q_0(t) + \max(C_t(\mathcal{A}_t) - C^B, 0). \tag{11}$$

We set $Q_0(0) = 0$ as the system begins at $t = 0$. From the queue evolution, the cost queue length increases by $C_t(\mathcal{A}_t)$ if the duplication set $\mathcal{A}_t$ is made for request t, and it decreases by C^B. Then, we recast the reliability constraint in (3) as $\mathbb{E}[\mathbb{I}_{L_t}] \leq \epsilon$ and define the virtual queues for (3), (7) and (8) as

$$Q_1(t+1) = Q_1(t) + (1-\epsilon)\mathbb{I}_{L_t}, \tag{12}$$

$$Q_2(t+1) = Q_2(t) + \left(X_t - \frac{\sigma}{1-\xi}\right)\mathbb{I}_{L_t}, \tag{13}$$

$$Q_3(t+1) = Q_3(t) + \left(Y_t - \frac{2\sigma^2}{(1-\xi)(1-2\xi)}\right)\mathbb{I}_{L_t}. \tag{14}$$

We set $Q_i(0) = 0, i = 1, 2, 3$ as the system begins at $t = 0$. From the queue evolution, the queue length of Q_1 (Q_2, Q_3) increases by 1 (X_t, Y_t) if request t fails, and it decreases by the parameter ϵ ($\frac{\sigma}{1-\xi}$, $\frac{2\sigma^2}{(1-\xi)(1-2\xi)}$). The queue length of Q_i does not change if request t is completed. Thus, we transform the problem of satisfying a time average inequality constraint into a pure queue stability problem.

Let $\mathbf{Q}(t) = (Q_0(t), Q_1(t), Q_2(t), Q_3(t))$, its Lyapunov function $\mathcal{L}(\mathbf{Q}(t)) = \frac{1}{2}\sum_{i=0}^{3} Q_i^2(t)$, and the drift of the Lyapunov function is defined as

$$\Delta\mathcal{L}_t = \mathcal{L}(\mathbf{Q}(t+1)) - \mathcal{L}(\mathbf{Q}(t)). \tag{15}$$

The upper bound of the Lyapunov drift is given by $\Delta\mathcal{L}_t \leq \Delta_B + Q_0(t)(C_t - C^B) + Q_1(t)\ (1-\epsilon)\mathbb{I}_{L_t} + Q_2(t)(X_t - \frac{\sigma}{1-\xi})\mathbb{I}_{L_t} + Q_3(t)(Y_t - \frac{2\sigma^2}{(1-\xi)(1-2\xi)})\mathbb{I}_{L_t}$, where $\Delta_B = (C^B)^2 + (1-\epsilon)^2 + (M_{\max} - \frac{\sigma}{1-\xi})^2 + (M_{\max}^2 - \frac{2\sigma^2}{(1-\xi)(1-2\xi)})^2$, where $M_{\max} = \max M_t$.

By controlling the upper bound, the algorithm can ensure the stability of virtual queues. The conditional expected Lyapunov drift of request t is defined as $\mathbb{E}[\mathcal{L}(\mathbf{Q}(t+1)) - \mathcal{L}(\mathbf{Q}(t))|\mathbf{Q}(t)]$. We define η as a parameter that controls the tradeoff between the virtual queue length and the accuracy of the optimal solution of problem P1. We then introduce a penalty term $\eta\mathbb{E}[L_t|\mathbf{Q}(t)]$ to the expected drift and minimize the upper bound of the drift plus penalty, $\mathbb{E}[\mathcal{L}(\mathbf{Q}(t+1)) - \mathbb{L}(\mathbf{Q}(t))|\mathbf{Q}(t)] + \eta\mathbb{E}[L_t|\mathbf{Q}(t)]$. As a result, P1 can be transformed into P2 in (16).

$$\text{P2}: \min_{\mathcal{A}_t \subset \mathcal{V}_t} \mathbb{E}[Q_0(t)C_t + (Q_1(t) + Q_2(t)X_t + Q_3(t)Y_t)\mathbb{I}_{L_t} + \eta L_t|\mathbf{Q}(t)]. \tag{16}$$

There are two major challenges to solving P2. First, optimally solving P2 requires complete information in the system, including parameters of all requests, TaVs, and SeVs, which is hard to obtain in advance. Furthermore, P2 is a nonlinear integer programming problem. Even if the complete future information is known as a priori, it is still difficult to solve with low complexity.

4.2 Oracle Solution

In this section, we obtain the duplication decision by an Oracle where the EN knows the complete information. Assume a genie-aided scenario that the expectations of service-level latency $\mathbb{E}[L_t]$, duplication cost $\mathbb{E}[C_t]$, request completion indicator $\mathbb{E}[\mathbb{I}_t]$, the exceeded value $\mathbb{E}[X_t]$, and its square $\mathbb{E}[Y_t]$ are known. We can get the optimal duplication set $\mathcal{A}_t^*$ to the per-request problem for request t. Therefore, the optimal solution for P2 is $\{\mathcal{A}_t^*\}_{t=1}^T$ which is called *Oracle solution* and the corresponding average latency is L^*. Although it is not realistic due to the uncertainty in vehicle movement and network conditions, the Oracle scenario provides insights into the subsequent algorithm design. Let $\mathcal{A}_t$ be the duplication set derived by a certain algorithm. The performance of this algorithm is evaluated by its loss compared with the Oracle solution. The expected loss is called *regret*, which is formally defined as

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T}(L_t(\mathcal{A}_t) - L^*)\right]. \tag{17}$$

Note that minimizing the regret is equivalent to minimizing the service-level latency. In the next section, we map the service request duplication problem into a combinatorial MAB framework. Thus, an EN can learn the average service-level latency of service request duplications over requests by observing the feedback.

5 Algorithm Design

Sequential decision-making problems under uncertainty are studied under the MAB framework and efficient learning algorithms that provide strong performance guarantees have been developed. Our service request duplication problem fits well in a combinatorial MAB framework. In this framework, SeV types are arms, service-level latency L_t is the reward, and duplication cost L_t^C is the cost to choose each arm, both of which are stochastic variables following certain distributions. For each request, ENs choose a duplication set $\mathcal{A}_t$ and observe the feedback.

We aim to design a service request duplication algorithm to minimize the objective of P2 in (16). In the formulation of P2, service-level latency L_t, the duplication cost C_t, the request completion indicator $\mathbb{I}_t$, the exceeded value X_t and its square Y_t are all unknown. Learning-based algorithms are necessary to strike a balance between exploitation (i.e., choosing the SeV set that gave the lowest latency in the past) and exploration (i.e., seeking new SeV sets that might give lower latency in the future). In the combinatorial MAB framework, the overall duplication cost is a linear summation of each duplication cost and we can the Upper Confidence Bound (UCB) algorithm [37] to estimate the duplication cost. However, service-level latency is a nonlinear function of each duplication latency, where the UCB algorithm doesn't work. We introduce the Stochastically Dominant Confidence Bound (SDCB) algorithm to address the nonlinear combination challenge. The algorithm is stated in Algorithm 1.

5.1 UCB Based Duplication Cost Estimation

Recall that the duplication cost C_t is the summation of all $L_{t,v}^C$ ($v \in \mathcal{A}_t$), i.e., $C_t = \sum_{v\in\mathcal{A}_t} L_{t,v}^C$. We modify UCB algorithm to estimate C_t. We can observe $L_{t,v}^C$ when request t is completed. If for request t, the computation time of SeV v exceeds $L_t^{\max}$, we regard that the request fails in SeV v, and set the observed latency $L_{t,v}^C = L_t^{\max}$ for learning purposes. We set $L_{t,v}^C = L_t^{\max}$ if SeV $v \notin \mathcal{A}_t$. Let $\tilde{L}_{t,v}^C = \frac{L_{t,v}^C}{L_t^{\max}}$ be the normalized computation time of SeV v. Thus, $\tilde{L}_{t,v}^C \in [0,1]$. Let k_v be the number of occurrences of SeV $v \in \mathcal{V}_t$ and let $h_{t,v}$ be the number of times SeV v has been chosen by the completion of request t. Let $\hat{L}_{t,v}^C$ be the normalized sample mean of the observed computation time of SeV v by the completion of request t, i.e., $\hat{L}_{t,v}^C = \sum_{i=1}^{t} \tilde{L}_{i,v}^C / h_{t,v}$. We use $\bar{L}_{t,v}^C$ to denote the UCB estimate of computation time of SeV v for request t, which is given as follows:

$$\bar{L}_{t,v}^C = \max\left\{\hat{L}_{t-1,v}^C - \sqrt{\frac{3\log(t-k_v)}{2h_{t-1,v}}}, 0\right\}, \tag{18}$$

where $\hat{L}_{t,v}^C$ and $\sqrt{\frac{3\log(t-k_v)}{2h_{t-1,v}}}$ correspond to exploitation and exploration, respectively. Similarly, we set $\bar{L}_{t,v}^C = 0$ if $h_{t-1,v} = 0$. The padding term $\sqrt{\frac{3\log(t-k_v)}{2h_{t-1,v}}}$ considers the number k_v occurrences of each SeV v such that the newly appeared SeVs can be better explored. Next, we will estimate L_t following a similar idea.

5.2 SDCB Based Service-Level Latency Estimation

Recall that service-level latency L_t of request t is a nonlinear function of $L_{t,v}$ with $v \in \mathcal{A}_t$, i.e., $L_t(\mathcal{A}_t) = \min_{v\in\mathcal{A}_t} L_{t,v}$. The duplication set $\mathcal{A}_t$ depends on the entire latency distribution of available SeVs, rather than the mean of them. As UCB doesn't work directly on the nonlinear function, another learning algorithm named SDCB [38] can estimate the distribution of $L_{t,v}$ and its stochastically dominant confidence bounds. Hence, we extend SDCB for $L_{t,v}$ estimation. We can observe $L_{t,v}$ when request t is completed. Let $\tilde{L}_{t,v} = L_{t,v}/L_t^{\max}$ be the normalized latency. Denote D_v, $\hat{D}_v$ and $\bar{D}_v$

as the true distribution, the empirical distribution, and the estimated distribution of the normalized latency $\tilde{L}_{t,v}$. Let F_v, $\hat{F}_{t,v}$, and $\bar{F}_{t,v}$ denote the CDF of D_v, $\hat{D}_v$ and $\bar{D}_v$ respectively. Notice $\hat{D}_v$ have finite supports (feasible in practice) and it can be fully described by a finite set of supports (i.e., $\{\tilde{L}_{1,v}, \ldots, \tilde{L}_{t,v}, \ldots, \tilde{L}_{T,v}\}$) and the values of its CDF $\hat{F}_{t,v}$ on the supported points is $\hat{F}_{t,v}(\tilde{L}_{t,v}) = \Pr_{x\sim\hat{D}v}(x \le \tilde{L}_{t,v}), \forall t$. The value of $\hat{F}_{t,v}(\tilde{L}_{t,v})$ is just the fraction of the observed outcomes from SeV v that are no larger than $\tilde{L}_{t,v}$. Therefore it suffices to store these discrete points as well as the values of $\hat{F}_{t,v}$ at these points to store the whole function.

The SDCB estimates the service-level latency distribution of SeV v as follows:

$$\bar{F}_{t,v}(\tilde{L}_{t,v}) = \begin{cases} 0, & \text{if } L_{t,v} = 0 \\ [(\hat{F}_{t-1,v} \oplus \sqrt{\frac{3\log(t-k_v)}{2h_{t-1,v}}})(\tilde{L}_{t,v})]^{-}, & \text{if } L_{t,v} > 0, \end{cases} \tag{19}$$

where $[\cdot]^{-} = \min(\cdot, 1)$ and $\hat{F}_{t-1,v} \oplus c$ is the estimated CDF obtained by applying the element-wise addition of scalar c to the values of the empirical CDF $\hat{F}_{t-1,v}$ supported by $\{\tilde{L}_{1,v}, \tilde{L}_{2,v}, \ldots, \tilde{L}_{t-1,v}\}$. We set $\bar{F}_{t,v}(L_{t,v}) = 1$ if $h_{t-1,v} = 0$. Similar to the UCB estimation, SDCB can balance the exploration-exploitation tradeoff during the learning process. As X_t, Y_t and $\mathbb{I}_{L_t}$ are functions of L_t and can be further derived by $\bar{F}_{t,v}$. Thus, for each request t, we can observe the set of available SeVs $\mathcal{V}_t$ and select an SeV set $\mathcal{A}_t \subset \mathcal{V}_t$ that minimize the objective as (20), where $\bar{\mathbf{D}} = D_1 \times D_2 \times \cdots \times D_{|\mathcal{A}_t|}$ is joint distribution of $\bar{D}_v$

$$\min_{\mathcal{A}_t \subset \mathcal{V}_t} Q_0(t) \sum_{\mathcal{A}_t} \bar{L}_t^C + \mathbb{E}_{\bar{\mathbf{D}}}[(Q_1(t) + Q_2(t)X_t + Q_3(t)Y_t)\mathbb{I}_{L_t} + \eta L_t | \mathbf{Q}(t)]. \tag{20}$$

5.3 Obtaining the Duplication Set

Since service-level latency is continuous, Algorithm 1 may suffer from large storage usage and computational complexity for constructing the latency distribution as t grows. Specifically, the observed service-level latency $L_{t,v}$ of each SeV v might be different at each time, and thus the required storage for each empirical CDF $\hat{F}_{t,v}$ is $O(t)$. Meanwhile, it takes $O(t)$ time to calculate the numerical upper confidence bound $\bar{F}_{t,v}$. To reduce the storage usage and computational complexity of the algorithm, the empirical CDF $\hat{F}_{t,v}$ is specified over m values, $0 \le b_1 \le b_2 \le \cdots \le b_m \le V$ and if $b_{j-1} < \tilde{L}_{t,v} < b_j$, the normalized latency is b_j. Calculating $\mathcal{A}_t$ is a minimum element problem, which is NP-hard [39] even under the discrete distribution.

We start analyzing the structure of the objective function in (20) denoted by $f^{\text{obj}}(\mathcal{A}_t) = f_1^{\text{obj}}(\mathcal{A}_t) + f_2^{\text{obj}}(\mathcal{A}_t)$, where $f_1^{\text{obj}}(\mathcal{A}_t) = Q_0(t)\sum_{\mathcal{A}_t} \bar{L}_t^C$ is a linear summation function and $f_2^{\text{obj}}(\mathcal{A}_t) = \mathbb{E}_{\bar{\mathbf{D}}}[(Q_1(t) + Q_2(t)X_t + Q_3(t)Y_t)\mathbb{I}_{L_t} + \eta L_t | Q(t)]$ is known as a submodular function [39]. Thanks to the special structure of $f_1^{\text{obj}}(\mathcal{A}_t)$ and $f_2^{\text{obj}}(\mathcal{A}_t)$, we can efficiently solve (20) (Lines 11-14). Specifically, we iteratively select the best duplication such that

$$v^* = \arg\max_{v \in \mathcal{V}_t \setminus \mathcal{A}_t} \frac{f_2^{\text{obj}}(\mathcal{A}_t) - f_2^{\text{obj}}(\mathcal{A}_t \cup \{v\})}{f_1^{\text{obj}}(\{v\})}. \tag{21}$$

For any λ, the algorithm for obtaining the duplication set (Lines 11-14) achieves a $(1+\lambda)$ approximation to the optimal value $f^{\text{obj}}(\mathcal{A}_t^*)$ with cost of $C_t(\log \frac{m}{\min_{v\in\mathcal{V}_t} L_{t,v}} + \log \frac{1}{\lambda})$ [39]. The EN offloads service request duplications to all the selected SeVs $v \in \mathcal{A}_t$, waits for their feedback to observe the latency, and finally updates the empirical duplication cost $\hat{L}_{t,v}^C$ and latency distribution CDF $\hat{F}_{t,v}$, and selected times $h_{t,v}$ (Line 19). The time complexity of Algorithm 1 is dominated by while loop (Lines 11-14), which is in the order of $O(\max_t|\mathcal{V}_t| \cdot \max_t|\mathcal{A}_t|)$.

Algorithm 1. Learning Based Service Request Duplication

1: Initialization: $h_{0,v} = 0, k_v = 0, Q_i(0) = 0, i = 0, 1, 2, 3$
2: **for** $t = 1, \ldots, T$ **do**
3: Update $Q_i(t)$ according to (12)-(14).
4: $\mathcal{A}_t = \emptyset$.
5: **for** Each $v \in \mathcal{V}_t$ **do**
6: $k_v = k_v + 1$.
7: Update estimations $\bar{L}_{t,v}^c$ and $\bar{F}_{t,v}$ as (18) and (19).
8: **if** $\exists k_v = 1$ **then**
9: $\mathcal{A}_t \leftarrow \mathcal{A}_t \bigcup \{v\}$.
10: **end if**
11: **end for**
12: **while** $\exists v^*$ s.t $f^{\text{obj}}(\mathcal{A}_t) \ge f^{\text{obj}}(\mathcal{A}_t \cup \{v^*\})$ **do**
13: $\mathcal{A}_t \leftarrow \mathcal{A}_t \cup \{v^*\}$.
14: Choose v^* according to (21).
15: **end while**
16: Duplicate the request to each SeV $v \in \mathcal{A}_t$.
17: /*computing and information transmissions*/
18: **for** Each SeV $v \in \mathcal{A}_t$ **do**
19: Observe latency $L_{t,v}^C$ and $L_{t,v}$.
20: Update $h_{t,v}$ and statistics $\hat{L}_{t,v}^C$, $\hat{F}_{t,v}$.
21: **end for**
22: **end for**

5.4 Algorithm Performance Analysis

In this subsection, we prove an upper bound on the time-average regret $\frac{R(T)}{T}$ under the proposed algorithm by following a similar line of regret analysis in [40]. This upper bound is achieved uniformly over time (i.e., for any finite time horizon T) rather than asymptotically when T goes to infinity. We state this result in Theorem 1.

Theorem 1. *Under the proposed algorithm, the time-average regret $\frac{R(T)}{T}$ has the following upper bound:*

$$\frac{R(T)}{T} \le \frac{\Delta_B}{\eta} + \frac{1}{T\eta}\sum_{i=1}^{5} 2\bar{M}_i N\left(4\sqrt{3T\log T} + 1 + \frac{5\pi^2}{12}\right), \tag{22}$$

where $N = \max_t|\mathcal{V}_t|$, $\bar{M}_1 = \eta L_{\text{ext}}$, $\bar{M}_2 = 1/2 L_{\max} Q_0^{\max}$, $\bar{M}_3 = Q_1^{\max}$, $\bar{M}_4 = (L_{\text{ext}} - L_{\max})Q_2^{\max}$, $\bar{M}_5 = ((L_{\text{ext}} - L_{\max})^2 Q_3^{\max}$, and $L_{\text{ext}} \ge \max_{t,v} L_t, v$.

Proof. For request t, consider an optimal strategy $\mathcal{A}_t^*$ and its corresponding service-level latency $L^*(\mathcal{A}_t^*)$. We rewrite the time-averaged regret of the proposed algorithm as

$$R(T)/T = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\Delta L(t)], \tag{23}$$

where $\Delta L(t) = L_t(\mathcal{A}_t) - L_t^*(\mathcal{A}_t^*) = \min_{v\in\mathcal{A}_t} L_{t,v} - \min_{v\in\mathcal{A}_t^*} L_{t,v}$. Using the Lyapunov-drift analysis [36], we can bound the expected drift-plus-regret as

$$\mathbb{E}[\Delta\mathcal{L}_t + \eta\Delta L(t)|\mathbf{Q}(t)] \le \Delta_B + \mathbb{E}[D_1(t)|\mathbf{Q}(t)], \tag{24}$$

where $D_1(t) = Q_0(t)(C_t - C_t^*) + Q_1(t)(\mathbb{I}_{L_t} - \mathbb{I}_{L_t^*}) + Q_2(t)(X_t - X_t^*)\mathbb{I}_{L_t^*} + Q_3(t)(Y_t - Y_t^*)\mathbb{I}_{L_t^*} + \eta(L_t(\mathcal{A}_t) - L_t^*(\mathcal{A}_t^*))$. Summing (24) for all t, using the trick of telescoping sum, and dividing both sides of the inequality by $T\eta$ with $\mathcal{L}(\mathbf{Q}(T)) > 0$ and $\mathcal{L}(\mathbf{Q}(1)) = 0$, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\Delta L(t)|\mathbf{Q}(t)] \le \frac{\Delta_B}{\eta} + \frac{1}{T\eta}\sum_{t=1}^{T}\mathbb{E}[D_1(t)|\mathbf{Q}(t)]. \tag{25}$$

Then, it remains to show the following bound:

$$\sum_{t=1}^{T}\mathbb{E}[D_1(t)] \le \sum_{i=1}^{5} 2\bar{M}_i N\left(2\sqrt{6T\log T} + 1 + \frac{5\pi^2}{12}\right). \tag{26}$$

Consider a strategy $\mathcal{A}_t'$, for each request t, follows:

$$\mathcal{A}_t' \in \min_{\mathcal{A}_t\subset\mathcal{V}_t} \mathbb{E}[Q_0(t)\sum_{\mathcal{A}_t} L_t^C + (Q_1(t) + Q_2(t)X_t + Q_3(t)Y_t)\mathbb{I}_{L_t} + \eta L_t]. \tag{27}$$

Recall that for each request t, the proposed algorithm chooses a duplication set $\mathcal{A}_t$ according to (20). Therefore, we have $Q_0(t)\sum_{\mathcal{A}_t}\bar{L}_t^C + \mathbb{E}_{\bar{\mathbf{D}}}(\mathcal{A}_t)[(Q_1(t) + Q_2(t)X_t + Q_3(t)Y_t)\mathbb{I}_{L_t} + \eta L_t] \le Q_0(t)\sum_{\mathcal{A}_t'}\bar{L}_t^C + \mathbb{E}_{\bar{\mathbf{D}}}(\mathcal{A}_t')[(Q_1(t) + Q_2(t)X_t + Q_3(t)Y_t)\mathbb{I}_{L_t} + \eta L_t]$. Following that, we derive an upper bound on $D_1(t)$ as

$$\begin{aligned}
D_1(t) \le\ & \eta\underbrace{(L_t(\mathcal{A}_t) - \mathbb{E}_{\bar{\mathbf{D}}}[L_t(\mathcal{A}_t)])}_{J_1} + Q_0(t)\underbrace{(C_t - \bar{C}_t)}_{J_2} \\
& + Q_1(t)\underbrace{(\mathbb{I}_{L_t} - \mathbb{E}_{\bar{\mathbf{D}}}[\mathbb{I}_{L_t}])}_{J_3} + Q_2(t)\underbrace{(X_t\mathbb{I}_{L_t} - \mathbb{E}_{\bar{\mathbf{D}}}[X_t\mathbb{I}_{L_t}])}_{J_4} \\
& + Q_3(t)\underbrace{(Y_t\mathbb{I}_{L_t} - \mathbb{E}_{\bar{\mathbf{D}}}[Y_t\mathbb{I}_{L_t}])}_{J_5} + \eta\underbrace{(\mathbb{E}_{\bar{\mathbf{D}}}[L_t'(\mathcal{A}_t')] - L_t'(\mathcal{A}_t'))}_{J_1'} \\
& + Q_0(t)\underbrace{(C_t' - \bar{C}_t')}_{J_2'} + Q_1(t)\underbrace{(\mathbb{E}_{\bar{\mathbf{D}}}[\mathbb{I}_{L_t'}] - \mathbb{I}_{L_t'})}_{J_3'} \\
& + Q_2(t)\underbrace{(\mathbb{E}_{\bar{\mathbf{D}}}[X_t'\mathbb{I}_{L_t'}] - X_t'\mathbb{I}_{L_t'})}_{J_4'} + Q_3(t)\underbrace{(\mathbb{E}_{\bar{\mathbf{D}}}[Y_t'\mathbb{I}_{L_t'}] - Y_t'\mathbb{I}_{L_t'})}_{J_5'}.
\end{aligned} \tag{28}$$

Define $J_i(t)$ and $J_i'(t), (i = 1, 2, 3, 4, 5)$ as in (28). Since $Q_i(t)$ is independent and bounded by a constant $Q_i^{\max}$, we prove the bounds of $J_i(t)$ and $J_i'(t)$ by following Lemmas 1-3 of [38] and Theorem 2 of [40]

$$\mathbb{E}[J_i] \le 2M_i N\left(2\sqrt{6T\log T} + 1 + \frac{\pi^2}{4}\right), i = 1, 2, 3, 4, 5, \tag{29}$$

$$\mathbb{E}[J_i'] \le 2M_i N\frac{\pi^2}{6}, i = 1, 2, 3, 4, 5, \tag{30}$$

where $M_1 = L_{\text{ext}}$, $M_2 = 1/2L_{\max}$, $M_3 = 1$, $M_4 = L_{\text{ext}} - L_{\max}$, $M_5 = (L_{\text{ext}} - L_{\max})^2$. □

The regret upper bound in (22) is quite appealing as it separately captures the impact of the cost and reliability constraints and the impact of the uncertainty in service-level latency for any finite request number T. Note that the regret upper bound in (22) has two terms. The first term $\frac{\Delta_B}{\eta}$ is inversely proportional to η and is attributed to the impact of the cost and reliability constraints. The second term $\frac{1}{T\eta}\sum_{i=1}^{5} 2\bar{M}_i N(4\sqrt{3T\log T} + 1 + \frac{5\pi^2}{12})$ is of the order $O(\sqrt{\log T/T})$. This part of the regret corresponds to the notion of regret in typical MAB problems and is attributed to the loss in the learning/exploration process.

5.5 Limitations of Our Algorithm

Our algorithm has two main limitations. First, each EN makes service request duplication decisions for request t after the completion of all $t-1$ previous requests in the algorithm description. Our work can also apply in scenarios where requests come before the completion of all previous requests. In this case, the EN makes decisions once requests arrive at the EN based on the estimated latency performance of requests that are already completed. The influence on the algorithm performance will reduce as t grows because the estimated latency performance would be more accurate along with the request increase. Second, our algorithm requires 1) plenty of moving vehicles to provide idle computing resources and 2) sufficient revisit times of SeVs in the coverage of each EN to accumulate sufficient historical data for good learning performance. Hence, our algorithm works better on busy city roads than rural roads.

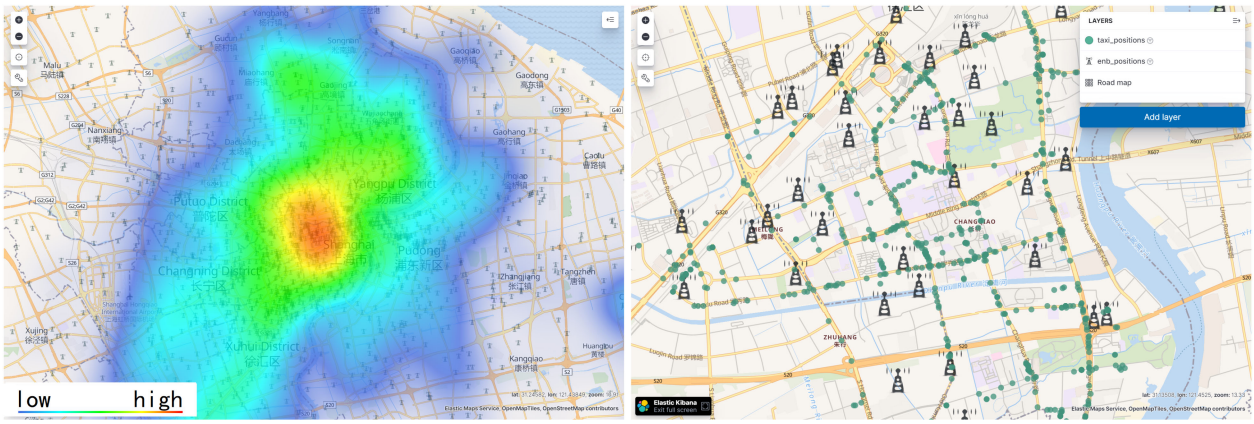
6 Evaluation

In this section, we evaluate the performance of the proposed algorithm by two metrics: average latency and average outage probability. We share source codes[4] for researchers who are interested in our work.

6.1 Simulation Settings

Our simulation uses two real-world datasets: Shanghai Taxi Trace dataset and Shanghai Telecom's Base Station dataset.[5] The Shanghai Taxi Trace dataset contains the traces of 4,328 taxis in Shanghai. Shanghai Telecom's base station dataset contains the exact location information of 3,233 base stations. It also contains more than 7.2 million records of accesses from 9,481 mobile phones during six successive months. Fig. 3a shows the heat map of taxi trace data, Fig. 3b shows an individual taxi trace and the surrounding base stations' deployment more clearly. The green spots are taxi traces and the black tower shapes represent base stations. We combine the two datasets to simulate a scenario where the EN manages service request duplication from TaVs to SeVs. Specifically, we use taxi traces to simulate vehicle movement in the VEC system. We choose base stations along roads to deploy

4. http://sguangwang.com/resources.php
5. http://sguangwang.com/TelecomDataset.html



(a) Taxi trace heat map. (b) Base station deployment.

Fig. 3. Dataset illustrations.

ENs according to the location information in the base station dataset. Vehicles move on roads and can access at least one EN.

The maximum coverage of each EN is set as 500m. We divide the SeVs into 10 types according to their speed and computing capabilities. The available SeV types are uniformly distributed for each request. Requests from TaVs are of the same type with the input data size x_t = 1Mb, the request result size y_t = 0.5Mb and the required CPU cycles w_t = 200M [11]. The deadline $L_t^{\max}$ for each request t is set as 1sec. The EN-to-Vehicle data transmission rate is set as 50 Mbps [41]. The Vehicle-to-EN transmission rate is uniformly distributed in [0.25, 10] Mbps [41]. The backhaul transmission time is set as 50ms. From [12], the maximum CPU frequency of each SeV is uniformly chosen within [2, 8] GHz. Moreover, we set the cost limit C^B as 2sec which would not waste much system computation resource. We set the reliability parameters $\epsilon = 0.01$ which is a typical reliability requirement for VEC applications described in [10]. And we set $\sigma = 1.18$, $\xi = -0.59$ to restrict the tail distribution, which controls the extremely large values [42].

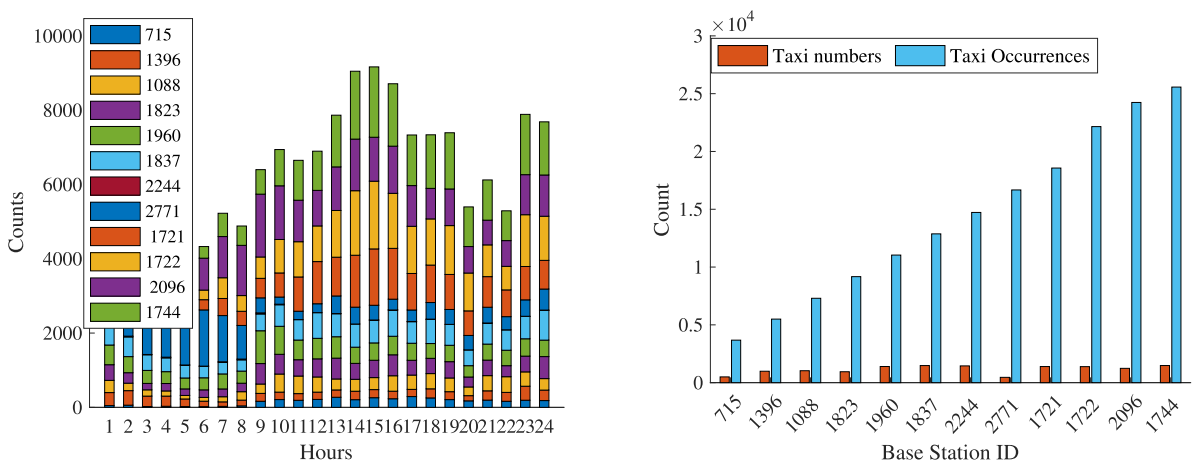
6.2 Benchmarks

We compare the proposed algorithm with five baselines:

1) *Oracle*: the EN obtains the same number of the best service request duplications with the proposed algorithm by knowing expectations of each SeV's latency and cost, which is not realistic in practice.
2) *DATE-V [11]*: a learning algorithm for task offloading proposed in [11] maximizing average reliability.
3) *LTR [12]*: a UCB-based task offloading proposed in [12] aiming to minimize average latency.
4) *RD (Random Duplication)*: the EN randomly selects the same number of SeVs as the proposed algorithm.
5) *ND (No Duplication)*: EN always selects the best SeV for each request following the proposed algorithm.

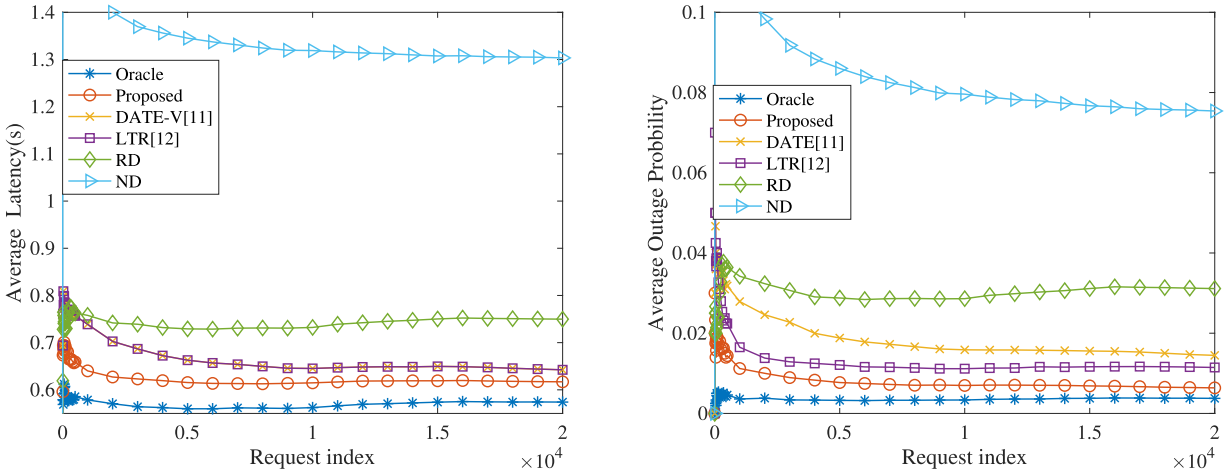
6.3 Evaluation of Request Duplication Feasibility

Before evaluating our algorithm, we analyze the realistic dataset to support the feasibility of service request duplication in VEC scenarios. Our main idea is to illustrate there are enough vehicles in the radio range of each EN to realize request duplications on vehicles. We select twelve typical



(a) Taxi counts over a day. (b) Taxi counts of each EN.

Fig. 4. Evaluation of service request duplication feasibility.



(a) Service-level latency. (b) Service-level reliability.

Fig. 5. Algorithm comparison along with request number.

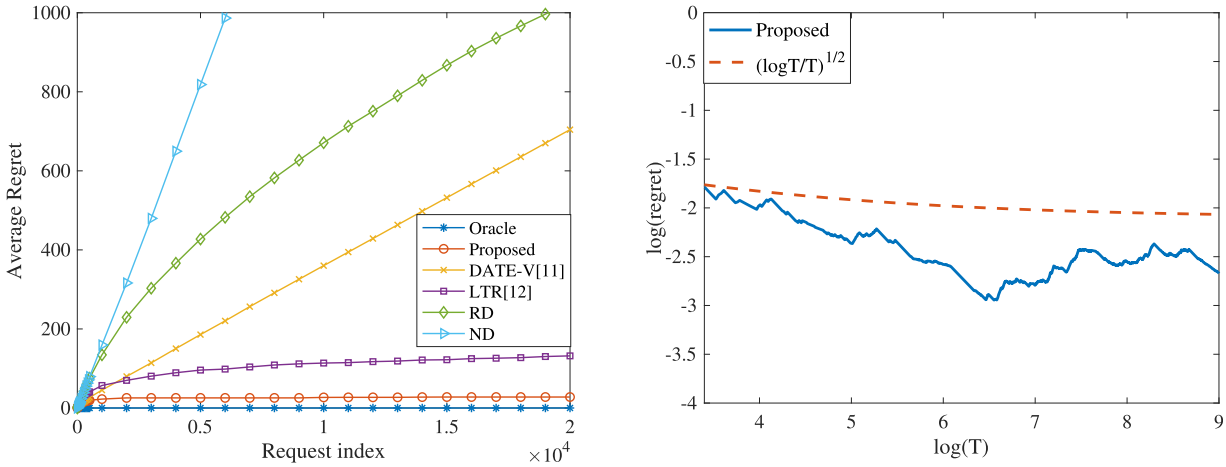
ENs with different vehicle occurrence times from Fig. 3b and count the number of vehicle occurrences every hour, the sum of occurrence counts over one day of each EN, and the sum of taxi number over one day of each EN.

From Fig. 4a, we can observe that the vehicle occurrence time varies during a day. The vehicle occurrence number of EN 715 is less than ten during the small hours. In this case, service request duplication may be not feasible because the vehicle number is small. We can observe this is a small proportion in the illustrated ENs and we find that more than 87% ENs in the dataset have more vehicle occurrence times than EN 715. This can verify the feasibility of service request duplication in the urban areas. Fig. 4b shows the taxi occurrence times and taxi number in the coverage of each EN over one day. As each taxi's occurrence time in the coverage of each EN is around 8.7-25.4 times during a day. We divide the vehicles into ten types and learn the performance of each type. It can be implied that the algorithm would have enough time to learn the performance of each SeV type.

6.4 Performance Comparison Along With Requests

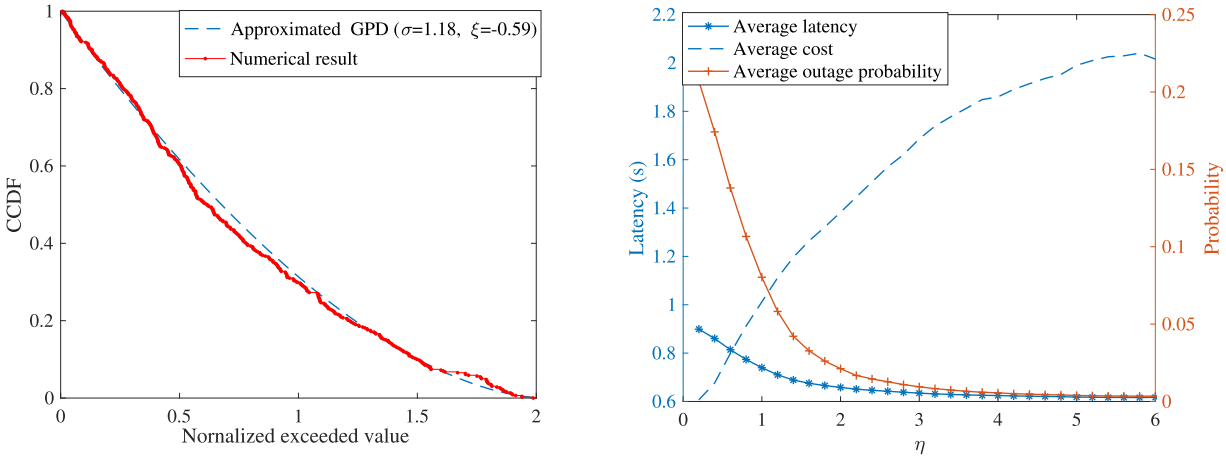
In the simulation, we select EN 1744 and randomly select a vehicle as TaV and the rest as SeVs in its coverage for each request. From Fig. 4, we can observe that the taxi number passing by one EN during one day is large while the passing time of each vehicle is not enough to learn the performance of each vehicle. So we divide the SeVs into ten types and randomly select one from each type as candidate SeV set for each request. The algorithm learns the performance of each type in the simulation.

Fig. 5 shows the time-averaged performance achieved by the proposed algorithm and the other five benchmarks. We select a serial of 2×10^4 request requests from start. We record the average latency, average outage probability along with the request number. In Fig. 5a, the Oracle algorithm achieves the lowest latency of 0.58 sec, which gives a lower bound to other algorithms. The proposed algorithm achieves the latency of 0.61 sec while LTR and DATE-V achieve 0.64 sec, RR algorithm achieves 0.75 sec and NR achieves 1.30 sec. We see that the proposed algorithm



(a) Cumulative regret. (b) Time-averaged regret.

Fig. 6. Regret of our algorithm.



(a) Exceeded value. (b) Performance *v.s.* cost.

Fig. 7. Parameter impact.

decreases the average latency from 0.70 sec to 0.61 sec. This means that the proposed algorithm can learn from latency feedback over time and after sufficiently many requests, it selects duplications almost as well as Oracle does. In Fig. 5b, the Oracle algorithm achieves the highest reliability of 99.62% while the proposed algorithm achieves 99.37%. The baselines of LTR, DATE-V, RR, and NR achieve 98.85%, 98.55%, 96.88%, and 92.46% respectively. We observe that the proposed algorithm has a 5% latency improvement compared with LTR (and DATE-V). Only the Oracle algorithm and the proposed algorithm can satisfy the reliability requirement. Although the performance improvement is not that much, it is significant for applications such as self-driving, which require quite low latency and high reliability (e.g., a subtle drop in reliability may negatively affect the safety of self-driving vehicles).

6.5 Regret of Proposed Algorithm

Fig. 6a shows the regret of the proposed algorithm. The proposed algorithm has much less regret than other benchmarks. The regret of DATE-V, LTR, NR algorithms is sublinear, while that of RR is linear. The regret of DATE-V, LTR are guaranteed by the learning algorithm proposed in [11], [12]. Fig. 6b seems to suggest that the time-average regret $R(T)/T$ can be well bounded by $\sqrt{\log(T)/T}$ as shown in Theorem 1.

6.6 Parameter Impact

Fig. 7a illustrates the effectiveness of Pickands-Balkema-de Haan theorem to characterize the exceeded latency. We can observe that the empirical CCDF of exceeded value can be well fitted by the approximated GPD with parameters $\sigma = 1.18$ and $\xi = -0.59$. The shape and scale parameters converge to the value given in the parameter settings. Characterizing the statistics of exceeded value helps to locally estimate the network-wide extreme metrics, e.g., the maximal latency among all SeVs, and enables us to proactively deal with extreme events.

Fig. 7b shows the tradeoff between latency (reliability) and cost, which is controlled by the parameter η. We seek to provide guidelines for selecting η in real implementations: under the cost constraint, one should choose appropriate η that can minimize the average latency performance. By changing η from 0.2 to 6, the proposed algorithm cares more about the latency performance, and thus the average latency decreases. This further contributes to reducing outage probability. However, with less concern on the duplication cost, the average duplication cost increases and will finally reach the maximum cost.

7 Conclusion

In this paper, we investigate online service request duplication for vehicular applications. We present a joint model of service-level latency and reliability. We formulate this problem as a combinatorial MAB problem with long-term cost and reliability constraints and then adopt the Lyapunov optimization technique to properly tradeoff the QoS guarantee and system resource cost. Then, we propose a learning algorithm by extending confidence bound based learning algorithms to deal with the exploitation-exploration tradeoff in face of system uncertainty. Further, we rigorously prove that the proposed algorithm has a sublinear cumulative regret. Simulation results demonstrate that the proposed algorithm outperforms the benchmark solutions. In future work, we will investigate the federated learning among different ENs to estimate the extreme parameters to further improve the latency and reliability performance.

References

[1] China Mobile Edge Computing Open Laboratory, "China mobile edge computing technical white paper," Tech. Rep., 2019.
[2] M. Xu *et al.*, "From cloud to edge: A first look at public edge platforms," in *Proc. 21st ACM Internet Meas. Conf.*, 2021, pp. 37–53.
[3] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1146–1159, Feb. 2020.
[4] H. Yang, K. Zheng, L. Zhao, and L. Hanzo, "Twin-timescale radio resource management for ultra-reliable and low-latency vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 1023–1036, Jan. 2020.

[5] M. K. Abdel-Aziz, S. Samarakoon, M. Bennis, and W. Saad, "Ultra-reliable and low-latency vehicular communication: An active learning approach," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 367–370, Feb. 2020.
[6] M. K. Abdel-Aziz, C.-F. Liu, S. Samarakoon, M. Bennis, and W. Saad, "Ultra-reliable low-latency vehicular networks: Taming the age of information tail," in *Proc. IEEE Global Commun. Conf.*, 2018, pp. 1–7.
[7] C.-F. Liu and M. Bennis, "Ultra-reliable and low-latency vehicular transmission: An extreme value theory approach," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1292–1295, Jun. 2018.
[8] X. Ge, "Ultra-reliable low-latency communications in autonomous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5005–5016, May 2019.
[9] I. Mazzocco, "When cars become computers: The new data challenge for EV companies," 2021. [Online]. Available: https://macropolo.org/electric-vehicles-computers-the-new-data-challenge/
[10] 5GAA Automotive Association, "C-V2X use cases volume II: Examples and service level requirements," Tech. Rep., 2020.
[11] L. Chen and J. Xu, "Task replication for vehicular cloud: Contextual combinatorial bandit with delayed feedback," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 748–756.
[12] Y. Sun, S. Zhou, and Z. Niu, "Distributed task replication for vehicular edge computing: Performance analysis and learning-based algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1138–1151, Feb. 2021.
[13] Z. Jiang, S. Zhou, X. Guo, and Z. Niu, "Task replication for deadline-constrained vehicular cloud computing: Optimal policy, performance analysis, and implications on road traffic," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 93–107, Feb. 2018.
[14] X. Ma, A. Zhou, S. Zhang, Q. Li, A. X. Liu, and S. Wang, "Dynamic task scheduling in cloud-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, early access, Sep. 24, 2021, doi: 10.1109/TMC.2021.3115262.
[15] X. Ge, Z. Li, and S. Li, "5G software defined vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 87–93, Jul. 2017.
[16] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Low latency via redundancy," in *Proc. ACM Conf. Emerg. Netw. Experiments Technol.*, 2013, pp. 283–294.
[17] G. Joshi, E. Soljanin, and G. Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, no. 2, 2017, Art. no. 12.
[18] S. Niknam, P. Wang, and T. Stefanov, "Resource optimization for real-time streaming applications using task replication," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2755–2767, Nov. 2018.
[19] W.-C. Chang and P.-C. Wang, "Adaptive replication for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2422–2432, Nov. 2018.
[20] B. Choudhury, S. Choudhury, and A. Dutta, "A proactive context-aware service replication scheme for Adhoc IoT scenarios," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1797–1811, Dec. 2019.
[21] K. Li, M. Tao, and Z. Chen, "Exploiting computation replication for mobile edge computing: A fundamental computation-communication tradeoff study," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4563–4578, Jul. 2020.
[22] L. Tang, B. Tang, L. Zhang, F. Guo, and H. He, "Joint optimization of network selection and task offloading for vehicular edge computing," *J. Cloud Comput.*, vol. 10, no. 1, pp. 1–13, 2021.
[23] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.
[24] S. Batewela, C.-F. Liu, M. Bennis, H. A. Suraweera, and C. S. Hong, "Risk-sensitive task fetching and offloading for vehicular edge computing," *IEEE Commun. Lett.*, vol. 24, no. 3, pp. 617–621, Mar. 2020.
[25] H. Liao *et al.*, "Learning-based intent-aware task offloading for air-ground integrated vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5127–5139, Aug. 2021.
[26] H. Guo, J. Liu, J. Ren, and Y. Zhang, "Intelligent task offloading in vehicular edge computing networks," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 126–132, Aug. 2020.
[27] A. B. de Souza, P. A. L. Rego, P. H. G. Rocha, T. Carneiro, and J. N. de Souza, "A task offloading scheme for WAVE vehicular clouds and 5G mobile edge computing," in *Proc. IEEE Global Commun. Conf.*, 2020, pp. 1–6.
[28] Y. Sun *et al.*, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.
[29] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8322–8335, Sep. 2019.
[30] S. Zhou, Y. Sun, Z. Jiang, and Z. Niu, "Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 49–55, May 2019.
[31] K. Gardner, M. Harchol-Balter, and A. Scheller-Wolf , "A better model for job redundancy: Decoupling server slowdown and job size," in *Proc. IEEE 24th Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, 2016, pp. 1–10.
[32] L. Lu, X. Han, and M. Ouyang, "A review on the key issues for lithium-ion battery management in electric vehicles," *J. Power Sources*, vol. 226, pp. 272–288, 2013.
[33] D. Wang, G. Joshi, and G. W. Wornell, "Efficient straggler replication in large-scale parallel computing," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 4, pp. 1–23, 2019.
[34] M. Bennis, M. Debbah, and H. V. Poor, "Ultra-reliable and low-latency wireless communication: Tail, risk, and scale," *Proc. IEEE*, vol. 106, no. 10, pp. 1834–1853, Oct. 2018.
[35] L. de Haan and A. F. Ferreira, *Extreme Value Theory: An Introduction*. Berlin, Germany: Springer, 2006.
[36] J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lect. Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[37] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.
[38] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, and P. Lu, "Combinatorial multi-armed bandit with general reward functions," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1659–1667.
[39] A. Goel, S. Guha, and K. Munagala, "How to probe for an extreme value," *ACM Trans. Algorithms*, vol. 7, no. 1, 2010, Art. no. 12.
[40] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1702–1710.
[41] M. H. C. Garcia *et al.*, "A tutorial on 5G NR V2X communications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1972–2026, Jul./Sep. 2021.
[42] C. She, C. Liu, T. Q. S. Quek, C. Yang, and Y. Li, "Ultra-reliable and low-latency communications in unmanned aerial vehicle communication systems," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3768–3781, May 2019.

Qing Li received the MS degree from the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China, in 2017. She is currently working toward the PhD degree in the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. Her research interests include cloud computing and mobile edge computing.

Xiao Ma (Member, IEEE) received the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2018. She is currently a lecturer with the State Key Laboratory of Networking and Switching Technology, BUPT. From October 2016 to April 2017, she visited the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests include mobile cloud computing and mobile edge computing.

Ao Zhou (Member, IEEE) received the PhD degrees from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015. She is currently an associate professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has published more than 20 research papers. She played a key role at many international conferences. Her research interests include cloud computing and edge computing.

Changhee Joo (Senior Member, IEEE) received the PhD degree from Seoul National University, Seoul, South Korea. From 2005, he had been with Purdue University and Ohio State University, USA. In 2010, he joined the Korea University of Technology and Education, Korea, and now he is with the Ulsan National Institute of Science and Technology (UNIST), Korea. His research interests include resource allocation, distributed algorithms, and network economics. He is an editor of the *Journal of Communications and Networks* and an associated editor of the *IEEE/ACM Transactions on Networking*. He has served several primary conferences as a committee member, including IEEE INFOCOM, ACM MobiHoc, and IEEE SECON. He received the IEEE INFOCOM'08 Best Paper Award, the KICS Haedong Young Scholar Award (2014), the ICTC'15 Best Paper Award, and the GAMENETS'18 Best Paper Award.

Shangguang Wang (Senior Member, IEEE) received the PhD degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2011. He is a professor with the School of Computer Science and Engineering, Beijing University of Posts and Telecommunications, China. He has published more than 150 papers. His research interests include service computing, mobile edge computing, and satellite computing. He is currently serving as chair of the IEEE Technical Committee on Services Computing, and vice-chair of the IEEE Technical Committee on Cloud Computing. He also served as general chairs or program chairs of more than 10 IEEE conferences. He is a fellow of the IET. For more information, please visit http://www.sguangwang.com.