IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796717

Multi-Agent Distributed Reinforcement Learning for Making Decentralized Offloading Decisions

Jing Tan
Huawei Technology
Munich, Germany
jingtan@huawei.com

Ramin Khalili
Huawei Technology
Munich, Germany
ramin.khalili@huawei.com

Holger Karl
Hasso Plattner Institute
Berlin, Germany
holger.karl@hpi.de

Artur Hecker
Huawei Technology
Munich, Germany
artur.hecker@huawei.com

***Abstract*—We formulate computation offloading as a decentralized decision-making problem with autonomous agents. We design an interaction mechanism that incentivizes agents to align private and system goals by balancing between competition and cooperation. The mechanism provably has Nash equilibria with optimal resource allocation in the static case. For a dynamic environment, we propose a novel multi-agent online learning algorithm that learns with partial, delayed and noisy state information, and a reward signal that reduces information need to a great extent. Empirical results confirm that through learning, agents significantly improve both system and individual performance, e.g., 40% offloading failure rate reduction, 32% communication overhead reduction, up to 38% computation resource savings in low contention, 18% utilization increase with reduced load variation in high contention, and improvement in fairness. Results also confirm the algorithm's good convergence and generalization property in significantly different environments.**

***Index Terms*—Offloading, Distributed Systems, Reinforcement Learning, Decentralized Decision-Making**

I. Introduction

Vehicular network (V2X) applications are characterized by huge number of users, dynamic nature, and diverse Quality of Service (QoS) requirements [1]. They are also computation-intensive, e.g., inferring from large neural networks [2] or solving non-convex optimization problems [3] [4]. These applications currently reside in the vehicle's onboard units (OBU) for short latency and low communication overhead. Even with companies such as NVidia developing OBUs with high computation power [5], post-production OBU upgrades are typically not commercially viable; and irrespective of local OBU power, the ability to offload tasks to edge/cloud via multi-access edge computing (MEC) devices increases flexbility, protecting vehicles against IT obsolescence. Hence, offloading is a key technique for future V2X scenarios [6]–[9].

Currently, computation offloading decisions are strictly separated between the user and the operating side [10]. Users decide what to offload to optimize an individual goal, e.g., latency [11] or energy efficiency [12]. Apart from expressing their preference through a pre-defined, static and universal QoS matrix [13], users cannot influence how their tasks are prioritized. The operating side centrally prioritizes tasks and decides resource allocation to optimize a system goal that is based on the QoS matrix, but not always the same as the users' goals, e.g., task maximization [14] or load-balancing [15].

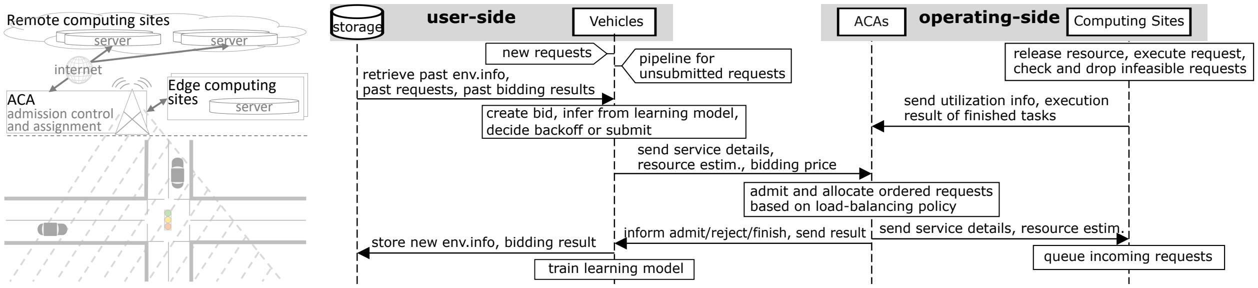
This separation poses problems for both user and operating sides, especially in the V2X context. V2X users have private goals [16], are highly autonomous [17], reluctant to share information or cooperate, and disobedient to a central planner [18]. They want flexible task prioritization and influence on resource allocation without sharing private information [19]. On the operating side, edge cloud computing architecture introduces signalling overhead and information delay in updating site utilization [10]; coupled with growing user autonomy and service customization, traditional centralized optimization methods for resource allocation become challenging due to unavailability of real-time information and computational intractability.

Nonetheless, efforts are made on the operating side to apply centralized methods under such conditions, e.g., using heuristics at run-time [20] or decoupling into smaller problems [21], but these solutions still assume complete information. Other efforts are made to jointly optimize private and system goals through game theoretic approaches—although they naturally deal with *decentralized incentives*, they often require complete information of the game to *centrally execute* the desired outcome. E.g., both [22] and [23] model network resource allocation problems of autonomous users as a game, but [22] assumes users share decision information, and [23] assumes all user and node profiles are known *a priori*. None of these assumptions are plausible in practice.

We, hence, need an interaction mechanism between user and operating sides based on incentives, not rules, and an algorithm that makes decentralized decisions with partial and delayed information in a dynamic environment. There are several challenges with such a mechanism. Users may game the system, resulting in potentially worse overall and individual outcomes [24]—the first challenge **C1** is how to incentivize user behavior such that users willingly align their private goals to the system goal while preserving their autonomy. The second challenge **C2** is finding an algorithm that efficiently learns from partial information with just enough feedback signals, keeping information-sharing at a minimum.

Among learning algorithms for decentralized decision-making, no-regret algorithms apply to a wide range of problems and converge fast; however, they require the knowledge of best strategies that are typically assumed to be static [25]. Best-response algorithms search for best responses to other users' strategies, not for a equilibrium—they are therefore

978-1-6654-5822-1/22/$31.00 ©2022 IEEE


Remote computing sites
server
server
internet
ACA
admission control
and assignment
Edge computing
sites
server
storage
user-side
Vehicles
ACAs
operating-side
Computing Sites
new requests
pipeline for
unsubmitted requests
retrieve past env.info,
past requests, past bidding results
create bid, infer from learning model,
decide backoff or submit
send service details,
resource estim., bidding price
release resource, execute request,
check and drop infeasible requests
send utilization info, execution
result of finished tasks
admit and allocate ordered requests
based on load-balancing policy
inform admit/reject/finish, send result
send service details, resource estim.
store new env.info, bidding result
train learning model
queue incoming requests

(a) Topology

(b) Message sequence

Figure 1: System model

adaptive to a dynamic environment, but they may not converge at all [26]. To improve the convergence property of best-response algorithms, [27] introduces an algorithm with varying learning rate depending on the reward; [26] extends the work to non-stationary environments. However, both these algorithms provably converge only with restricted classes of games. The challenge **C3** still exists to trade off between optimality and convergence, while keeping computation and communication complexity tractable [18].

We propose a decentralized decision-making mechanism based on second-price sealed-bid auctions that successfully addresses these challenges, using the V2X context as an example. Our method is not restricted to V2X applications—it can be applied to other applications facing similar challenges.

Second-price auctions are commonly used to distribute public goods, due to its welfare-maximization property. Typically, in a second-price sealed-bid auction, a bidder has no knowledge of other bidders' bidding prices and it only receives the bidding outcome as feedback signal. Additionally, it receives the final price if it wins the bid—this befits our requirement to limit information-sharing. Our mechanism also utilizes the feedback signal to incentivize cooperative behavior and speed up learning. We prove that in the static case, the outcome of this mechanism is a Nash equilibrium (NE) and a maximization of social welfare; under specific conditions (Sec.III-C), it is also a *Pareto-optimal* allocation of resources (**C1**).

For the dynamic case, we choose to use a multi-agent reinforcement learning (MARL) algorithm, for its ability to learn with partial, noisy and delayed information, and a single reward signal (C2). Specifically, our core RL algorithm learns the best-response strategy updated in a fictitious self play (FSP) method. FSP addresses strategic users' adaptiveness in a dynamic environment by evaluating state information incrementally, and by keeping a weighted historical record [28]; it is easier to implement than the method proposed in [27], especially with a large state and action space (**C3**). Our empirical results show that over time, the best-response strategies stabilize and lead to significantly improved individual and overall outcomes. We compare active (learning-capable) and passive (learning-incapable) agents in both synthetic and realistic V2X setups. Our algorithm demonstrates capability to generalize to very different, unseen environments.

To summarize, our main contributions are:

- We formulate computation offloading as a decision-making problem with decentralized incentive and execution. The strategic players are incentivized to align private and system goals by balancing between competition and cooperation.
- We introduce DRACO, a distributed algorithm that learns based on delayed and noisy environment information and a single reward signal. Our solution reduces the requirement for information-sharing to a great extent.
- We evaluate DRACO in a synthetic setup with randomized parameters, as well as in a realistic setup based on specific mobility model and self-driving applications. Our results show that it significantly increases resource utilization, reduces offloading failure rate, load variation and communication overhead, even in a dynamic environment where information-sharing is limited. The models are easily generalized in different environments.
- The authors have provided public access to their code or data at [29].

Sec.II describes system model and problem formulation, Sec.III proposes our solution, Sec.IV presents empirical results, Sec.V summarizes related work, Sec.VI concludes the paper.

II. System Model and Problem Formulation

A. System model

Our system adopts the classic edge cloud computing architecture: user-side vehicles request services; operating-side admission control and assignment (ACA) units (e.g., road side units or base station) control admission of service requests and assign them to different computing sites, which own resources and execute services [30] (Fig.1a). We propose changes only to: 1) the algorithm deciding admission and assignments, 2) the interaction mechanism. In addition, most signalling needs in our proposed approach are covered by the ISO 20078 standard on extended vehicle web services [31]; additional fields required to pass bidding information are straightforward to implement. Channel security is not the focus of this study.

We first define *service request* in our study; then we explain in detail the user side and the operating side.

1) Service request as bid: The cloud-native paradigm decomposes services into tasks that can be sequentially deployed [32]. A service request comprises 1) a task chain, with varying number, type, order and resource needs of tasks, and 2) a deadline. We consider a system with custom-tailored services placed at different computing sites in the network;

the properties of these services are initially unknown to the computing sites. This enables us to extend use cases into new areas, e.g., self-driving [8] [9]. We consider independent services, e.g., in self-driving, segmentation and motion planning services can be requested independently.

Classic decentralized decision-making mechanisms include dynamic pricing, negotiations, and auctions. Among these mechanisms, auction is most suitable in a dynamic and competitive environment, where the number of bidders and their preferences vary over time and distribution of private valuations is dispersed [33] [34]. Among various forms of auction, second-price seald-bid auction maximizes welfare rather than revenue and has limited information-sharing, hence befitting the requirements in our study. Specifically, our approach is based on Vickrey-Clarke-Groves (VCG) for second-price combinatorial auction [35]. In our case, we use simultaneous combinatorial auction as a simplified version of VCG—each bidder bids for all commodities separately and simultaneously, without having to specify its preference for any bundle [36]. Since it assumes no correlation between commodities, the simplification befits our study of independent service requests.

We conceive of a vehicle's service request as a *bid* in an auction. Besides the service details, a bid includes the bidding price and the vehicle's estimated resource needs.

2) User side: We focus on the behavior of vehicles, conceived of as *agents*. A vehicle acts autonomously and privately: it shares no information with other vehicles and only very limited information with the ACA (Sec.III-A and III-D). Its decision objective is to maximize average utility from joining the auction. If it bids a low price and loses, it suffers costs including transmission delay and communication overhead for bidding and rebidding; if it bids a high price and wins, it has reduced payoff. For lower cost or better payoff, it can decide to join the auction at a later time (i.e. backoff [37]). But if backoff is too long, it has pressure to pay more to prioritize its request. Therefore, it balances between two options: i) back off and try later or ii) submit the bid immediately to ACA. In our approach, 1) vehicles are incentivized to balance between backoff and bidding through a cost factor; 2) backoff time is learned from state information, not randomly chosen.

We study the learning algorithm in each vehicle. We use passive, non-learning vehicles as benchmark, to quantify the effect of learning on performance. Learning essentially sets the priority of a service request, this priority is used by the ACA to order requests; it is simply constant for non-learning agents, resulting in first-in, first-out processing order.

3) Operating side: The ACA unit and computing sites are the operating side (Fig.1b). The ACA unit decides to admit or reject ordered service requests. Upon admission, it assigns the request to a computing site according to a load-balancing policy. Due to information delay, execution uncertainty, system noise, etc., the resource utilization information at different sites is not immediately available to the ACA unit. If all computing sites are overloaded, service requests are rejected, and vehicles can rebid for a maximum number of times. If the request is admitted but cannot be executed before deadline, the computing site drops the service and informs ACA unit. Vehicles receive feedback on bidding and execution outcome, payment, and resource utilization (Sec.III-A).

Table I: Symbol definition

Sym.	Description	Sym.	Description
$k \in K$	service type/commodity	n_k	k's availability
$i \in I$	service request/bid	v	bid value
$m \in M$	vehicle/bidder	p	payment
x	bidding outcome	u	utility
α	backoff decision	b	bidding price
c	lost bid penalty	q	backoff cost
β	utilization	B	budget
$h \in H$	resource types	$\omega_{i,h}$	i's requirement of h
Q	service deadline	ρ	service request details
$\mathbf{e}_m$	m's env. variables	rl_m^t	m's present state for RL
sl_m^t	m's present state for SL	P_{-m}^t	other bidders' state at t
$\mathbf{a}$	action, $\mathbf{a} = (\alpha, b)$	S_m^t	complete state for RL
θ	actor parameters	$\mathbf{w}$	critic parameters

The operating side does not have *a priori* knowledge of the type, priority, or resource requirement of service requests. For example, if at run-time, a site receives a previously unknown service, it uses an estimate of resource needs provided by the vehicle. Over time, a site updates this estimate from repeated executions of the same service. Extension to a more sophisticated form of learning is left to future work.

The total service time of a request is the sum of processing, queueing, and transmission time. Each computing site may offer all services but with different resource profiles (i.e., amount and duration needed), depending on the site's configuration. Site capacity is specified in abstract time-resource units: one such unit corresponds to the volume of a request served in one time unit at a server, when given one resource unit.

B. Problem formulation

Table I summarizes the notation. Let M be the set of vehicles (bidders) and K the set of commodities (service types), each type with total of n_k^t available service slots at time t in computing sites. Bidder $m \in M$ has at most 1 demand for each service type $k \in K$ at t, denoted by $m_k^t \in \{0, 1\}$. It draws its actions for each service type—whether to back off $\alpha_m^t = \{\alpha_{m,1}^t, \cdots, \alpha_{m,|K|}^t\} \in \{0,1\}^{|K|}$, and which price to bid $\mathbf{b}_m^t = \{b_{m,1}^t, \cdots, b_{m,|K|}^t\} \in \mathbb{R}_+^{|K|}$—from a strategy. m's utility is denoted by $u_m(\mathbf{b}_m^t)$. The bidding price $b_{m,k}^t$ is some unknown function of m's private valuation $v_{m,k} \in \mathbb{R}_+$ of the service type, $b_{m,k}^t = f_m(v_{m,k})$. The competing bidders draw their actions from a joint distribution π_{-m}^t based on $(\mathbf{p}^1, \cdots, \mathbf{p}^{t-1})$, where $\mathbf{p}^t \in \mathbb{R}_+^{|K|}$ is the payment vector received at the end of time t, its element p_k^t is the $(n_k^t + 1)^{\text{th}}$ highest bid for k. Bidder m observes the new $\mathbf{p}^t$ as feedback. The auction repeats for T periods. The goal is to maximize the long-term utility: $\mathcal{U} = \frac{1}{T} \sum_{t=1}^{T} \sum_{m \in M} u_m(\mathbf{b}_m^t), T \to \infty$.

For any k, when availability $n_k^t < \sum_{m \in M} m_k^t$, there is more demand than available service slots and we call it "high contention". When $n_k^t \geq \sum_{m \in M} m_k^t$, we call it "low contention". In a dynamic environment, n_k^t depends on utilization at $t-1$ and existing demand at t. Due to noise and transmission delay in a realistic environment, this information is inaccurate and outdated when it becomes available to the ACA unit for admission control (Fig.1b).

Ideally, an auction is incentive-compatible. Unfortunately, with budget constraint and costs, the second-price auction considered here is no longer incentive-compatible. But we still use this type of auction as we can show (in Sec.III-B and III-C) that it maximizes social welfare and optimally allocates resources. We also use the payment signal as additional feedback to aid the bidders' learning process (Sec.III-E).

III. Proposed Solution

To solve the problem described in Sec.II-B, we propose DRACO, a **D**istributed **R**einforcement-learning algorithm with **A**uction mechanism for **C**omputation **O**ffloading. In Sec.III-A we define bidder's utility function; in III-B and III-C, we prove the existence of NE, maximization of welfare and Pareto optimality in the static case, under both low and high contentions. We introduce our algorithm for dynamic environment in Sec.III-D and III-E. Notations are in Table I. For readability, we drop notation for time t in the static case.

A. Utility function

In this section, we first build up the utility function based on the payoff of classic second-price auction. Then we add costs for backoff and losing the bid, incentivizing tradeoff between higher chance of success and lower communication overhead. Finally, we add the system resource utilization goal to the individual utility.

In each auction round, if a bid i for service type k is admitted, its economic gain is $(v_{i,k} - p_{i,k})$. Each bidder has a given $v_{i,k}$ that is 1) linear to the bidder's estimated resource needs for k and 2) within the budget. The first condition guarantees Pareto optimality (Corollary III.2.1); the second avoids overbidding under rationality (Theorem III.2). Our study does not consider irrational or malicious agents, e.g., whose goal is to reduce social welfare even if individual outcome may be hurt. ACA records $b_{j,k}$ of the highest losing bid j for each k, which is also the (n_k+1)th highest bidding price. For $n_k = 1$ this would be the second highest price, hence the name "second-price auction". If i is admitted, the payment $p_{i,k} = b_{j,k}$ is signaled back to the bidder. If i is rejected, it has a constant cost of $c_{i,k}$. The bidder's utility so far:

$$\sqcap_{i,k} = x_{i,k} \cdot (v_{i,k} - p_{i,k}) - (1 - x_{i,k}) \cdot c_{i,k} \tag{1}$$

$x_{i,k} = 1$ means bidder wins bid i for a service slot of service type k, which implies $b_{i,k}$ is among the highest n_k bids for k. Ties are broken randomly.

We add $\alpha_{i,k} \in \{0,1\}$ for backoff decision: bidder submits the bid if $\alpha_{i,k} = 1$, otherwise, it backs off with a cost $q_{i,k}$:

$$u_{i,k} = \alpha_{i,k} \cdot (\sqcap_{i,k} - 1_{p_{i,k}=0} \cdot v_{i,k}) + (1 - \alpha_{i,k}) \cdot q_{i,k} \tag{2}$$

Especially in high contention, more rebidding causes communication overhead, but less rebidding reduces the chance of success. With $c_{i,k}$, the utility incentivizes less rebidding to reduce system-wide communication overhead (**C1**). Together with $q_{i,k}$, the bidder is incentivized to trade off between long backoff time and risky bidding. In our implementation (Sec.IV), α is continuous between 0 and 1 that also indicates the length of backoff time.

To further align the bidder's objective with system overall objective (**C1**), we include system resource utilization β in the utility. This incentivizes bidders to minimize system utilization. Hence, the complete utility definition is:

$$u_i = \sum_{k \in K} u_{i,k} + W \cdot (1 - \beta) \tag{3}$$

W is a constant that weighs the utilization objective. In low contention, there is adequate resource to accept all bids, bidding price is less relevant, and backoff decision becomes more important.

To calculate Eq.3, the bidder needs only these feedback signals: bidding outcome $x_{i,k}$, final price $p_{i,k}$ and system utilization β, addressing **C2**.

B. Low contention

Low contention is much more common in networking and presumably also in future V2X applications as abundant resources are often available. We show that in low contention, the interaction mechanism is a potential game with NE. We use the concept of potential functions to do so [38]:

Definition III.1. $G(I, A, u)$ is an exact potential game if and only if there exists a potential function $\phi(A) : A \longrightarrow \mathbb{R}$ s.t. $\forall i \in I$, $u_i(b_i, b_{-i}) - u_i(b'_i, b_{-i}) = \phi_i(b_i, b_{-i}) - \phi_i(b'_i, b_{-i}), b \in A$.

Remark III.1. Players in a finite potential game that jointly maximize a potential function end up in NE.

Proof. See [38]. □

Theorem III.1. Bidders with utility as Eq.3 participate in a game as described in Sec.II-B in low contention, the game is a potential game, and the outcome is an NE.

Proof. In low contention, $p_{i,k} = 0$, as all bids are accepted. u_i is reduced to: $u_i(\alpha_i, \alpha_{-i}) = \sum\limits_k q_{i,k} - \sum\limits_k \alpha_{i,k} q_{i,k} + W\Big(1 - \sum_j \alpha_j \cdot \frac{\omega_j}{C}\Big)$, where $-i$ denotes bidders other than i. $\omega_j \in \mathbb{R}^{|K|}$ is each bid's resource requirement, C is system capacity. Thus, the auction is reduced to a potential game with discrete action space $\alpha_i \in \mathbb{R}^{|K|}$, and potential function $\phi(\alpha_i, \alpha_{-i}) = \sum\limits_{j,k} q_{j,k} - \sum\limits_{j,k} \alpha_{j,k} q_{j,k} + W\Big(1 - \sum_j \alpha_j \cdot \frac{\omega_j}{C}\Big), \forall i, j \in I, \forall k \in K$.

We prove in Appendix A that $u_i(\alpha_i, \alpha_{-i}) - u_i(\alpha'_i, \alpha_{-i}) = \phi(\alpha_i, \alpha_{-i}) - \phi(\alpha'_i, \alpha_{-i})$, and hence it is a potential game, and bidders maximizing their utilities u_i also maximize the potential function ϕ. Since $\alpha_i \in \mathbb{R}^{|K|}$, it is a finite potential game. According to Remark III.1, the outcome is an NE. □

In low contention, our computation offloading problem becomes a potential game. This enables us to use online learning algorithms such as in [39] that converge regardless of other bidders' behaviors. The NE is a local maximization of the potential function: each bidder finds a balance between its backoff cost and the incentive to reduce overall utilization.

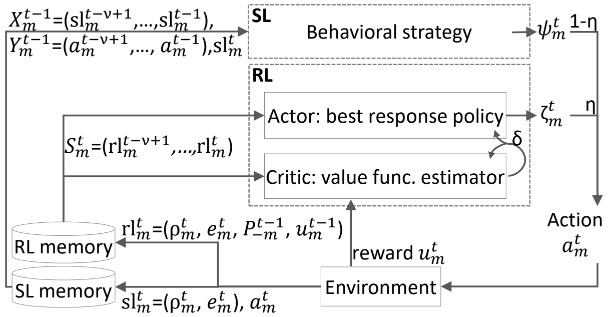

X_m^{t-1}=($\text{sl}_m^{t-\nu+1}$,...,sl_m^{t-1}),
Y_m^{t-1}=($a_m^{t-\nu+1}$,..., a_m^{t-1}),sl_m^t
SL
Behavioral strategy
ψ_m^t
1-η
RL
Actor: best response policy
ζ_m^t
η
δ
S_m^t=($\text{rl}_m^{t-\nu+1}$,...,rl_m^t)
Critic: value func. estimator
Action
a_m^t
rl_m^t=(ρ_m^t, e_m^t, P_{-m}^{t-1}, u_m^{t-1})
RL memory
reward u_m^t
Environment
SL memory
sl_m^t=(ρ_m^t, e_m^t), a_m^t

Figure 2: RL and SL algorithms

Empirical results in Sec.IV confirm that over time this results in a more balanced load.

C. High contention

In high contention, α is used in a repeated auction to avoid congestion and ensure better reward over time. To simplify the proofs, we consider only the time steps where $\alpha = 1$ (bidder joins auction). We also take a small enough W, such that the last term in Eq.3 can be omitted in high contention, to further simplify the utility function in the proof.

Theorem III.2. *In a second-price auction, where bidders with utility as Eq.3 compete for service slots as commodities in high contention, 1) bidders' best-response is of linear form, 2) the outcome is an NE and 3) welfare is maximized.*

Proof. See Appendix B. □

When bidders bid for service slots, the required resources are allocated. Theorem III.2 guarantees the maximization of welfare (total utility of bidders), but it does not guarantee the optimality of the resource allocation, unless the following conditions are met: if bidders' valuation of the commodity is linear to its resource requirement, and all bidders have some access to resources (fairness).

Corollary III.2.1. *In a second-price auction, where M bidders with utility as Eq.3 compete in high contention, the outcome is an optimal resource allocation, if the bidders' valuation of commodities is linear to resource requirement and all bidders have a positive probability of winning.*

Proof. See Appendix C. □

Our setup meets both conditions.

D. The FSP algorithm

The FSP algorithm addresses the convergence challenge of a best-response algorithm (**C3**). FSP balances exploration and exploitation by replaying its own past actions to learn an average behavioral strategy regardless of other bidders' strategies; then it cautiously plays the behavioral strategy mixed with best response [28]. The method consists of two parts: a supervised learning (SL) algorithm predicts the bidder's own behavioral strategy ψ, and an RL algorithm predicts its best response ζ to other bidders. The bidder has η, $\lim_{t\to\infty} \eta = 0$ probability of choosing action $\mathbf{a} = \zeta$, otherwise it chooses $\mathbf{a} = \psi$. The action includes backoff decision α and bidding price b. If α is above a threshold, the bidder submits the bid; otherwise, the bidder backs off for a duration linear to α. We predefine the threshold to influence bidder behavior: with a higher threshold, the algorithm becomes more conservative and tends to back off more service requests. A learned threshold (e.g., through meta-learning algorithms) is left to future work.

Algorithm 1 FSP algorithm for bidder m

```
Initialize $\psi_m, \zeta_m$ arbitrarily, $t = 1, \eta = 1/t, \nu, P_{-m}^{t-1} = 0, u_m^{t-1} = 0$,
    observe $e_m^t$, create $\text{rl}_m^t, \text{sl}_m^t$ and add to memory
while true do
    Take action $\mathbf{a}_m^t = (1-\eta)\psi_m^t + \eta\zeta_m^t$
    Receive $P_m^t$, calculate $u_m^t$, observe $\rho_m^{t+1}, \mathbf{e}_m^{t+1}$
    Create and add state to RL memory: $\text{rl}_m^{t+1}$
    Create and add state to SL memory: $(\text{sl}_m^{t+1}, \mathbf{a}_m^t)$
    Construct $S_m^t, S_m^{t+1}$, calculate $\zeta_m^{t+1} = \text{RL}(S_m^t, S_m^{t+1}, u_m^t)$
    Calculate $\psi_m^{t+1} = \text{SL}(\text{sl}_m^{t+1})$
    $t \leftarrow t+1, \eta \leftarrow 1/t$
end while
```

Although FSP is only convergent in certain classes of games [40], and in our case of a multi-player, general-sum game with infinite strategies, it does not necessarily converge to an NE, it is still an important experiment as our application belongs to a very general class of games; and empirical results show that by applying FSP, overall performance is greatly improved compared to using only RL. The FSP is described in Alg.1.

Input to SL includes bidder m's service requests—service type, resource amount required, and deadline: $\rho_m^t = \{(k_i, \omega_{i,h}, Q_i)|i \in I, h \in H\}$ (m can create multiple bids, each an independent request for service type k_i; ρ_m^t is the set of all m's bids at t), and current environment information visible to m, denoted e_m^t (e.g., number of bidders in the network and system utilization β^t). SL infers behavioral strategy ψ_m^t. The input $\text{sl}_m^t = (\rho_m^t, e_m^t)$ and actual action $\mathbf{a}_m^t$ are stored in SL memory to train the regression model. we use a multilayer perceptron in our implementation.

Input to RL is constructed from m's present state rl_m^t. rl_m^t includes 1) ρ_m^t; 2) e_m^t; 3) previous other bidders' state P_{-m}^{t-1}, represented by the final price p_k, or $P_{-m}^t = \mathbf{p}^t = \{p_k^t | k \in K\}$; and 4) calculated utility u_m^{t-1} according to Eq.3. To consider historical records, we take ν most current states to form the complete state input to RL: $S_m^t = \{\text{rl}_m^\tau | \tau = t-\nu+1, \cdots, t\}$. RL outputs best response ζ_m (Fig.2). The input consists of bidder's private information and easily obtainable public information, e.g., environment data and past prices, thus addressing **C2**.

E. The RL algorithm

Authors of [41] use VCG and a learning algorithm for the bidders to adjust their bidding price based on budget and observation of other bidders. Our approach is similar in that we estimate other bidders' state P_{-m} from payment information and use the estimate as basis for a policy. Also, similar to their work, payment information is only from the seller.

Our approach differs from [41] in several major points. We use a continuous space for bidder states (i.e., continuous value for payments). As also mentioned in [41], a finer-grained state space yields better learning results. Moreover, we consider

Algorithm 2 RL algorithm for bidder m

```
Initialize $\theta, w$ arbitrarily. Initialize $\lambda$
while true do
    Input $t$ and $S_m^t, S_m^{t+1}$ constructed from RL memory
    Run critic and get $\hat{V}(S_m^t, \mathbf{w}), \hat{V}(S_m^{t+1}, \mathbf{w})$
    Calculate $\bar{u}_m = \lambda \bar{u}_m$ and $\delta$ (utility $u$ is reward $R$)
    Run actor and get $\mu(\theta), \Sigma(\theta)$
    Sample $\zeta_m^{t+1}$ from $F(\mu, \Sigma)$, update $\mathbf{w}$ and $\theta$
end while
```

multiple commodity/service types, which is more realistic, and therefore has a wider range of applications. Further, we do not explicitly learn the transition probability of bidder states. Instead, we use historical states as input and directly determine the bidder's next action.

We use the actor-critic algorithm [42] for RL (Alg.2). The **critic** learns a state-value function $V(S)$. Parameters of the function are learned through a neural network that updates with $\mathbf{w} \leftarrow \mathbf{w} + \gamma^w \delta \nabla \hat{V}(S, \mathbf{w})$, where γ is the learning rate and δ is the temporal difference (TD) error. For a continuing task with no terminal state, the average reward is used to calculate δ [42]: $\delta = u - \bar{u} + \hat{V}(S', \mathbf{w}) - \hat{V}(S, \mathbf{w})$. In our case, the reward is utility u. We use exponential moving average (with rate λ) of past rewards as $\bar{u}$.

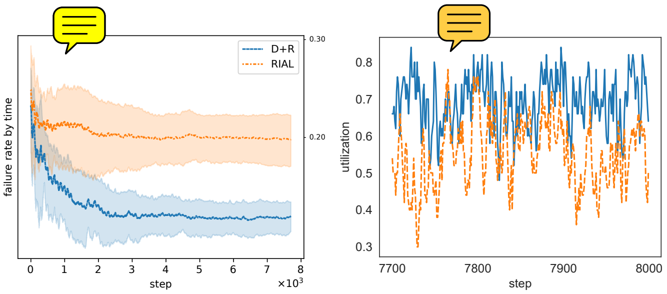
The **actor** learns the parameters of the policy π in a multidimensional and continuous action space. Correlated backoff and bidding price values are assumed to be normally distributed: $F(\mu, \Sigma) = \frac{1}{\sqrt{|\Sigma|}} \exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu))$. For faster calculation, instead of covariance Σ, we estimate lower triangular matrix L ($LL^T = \Sigma$). Specifically, the actor model outputs the mean vector μ and the elements of L. Actor's final output ζ is sampled from F through: $\zeta = \mu + L\mathbf{y}$, where $\mathbf{y}$ is an independent random variable from standard normal distribution. Update function is $\theta \leftarrow \theta + \gamma^\theta \delta \nabla \ln \pi(\mathbf{a}|S, \theta)$. We use $\frac{\partial \ln F}{\partial \mu} = \Sigma(\mathbf{x} - \mu)$ and $\frac{\partial \ln F}{\partial \Sigma} = \frac{1}{2}(\Sigma(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T \Sigma - \Sigma)$ for back-propagation.

The objective is to find a strategy that, given input S_m^t, determines $\mathbf{a}$ to maximize $\frac{1}{T-t} \mathbb{E}[\sum_{t'=t}^{T} u_m^{t'}]$. To implement the actor-critic RL, we use a stacked convolutional neural network (CNN) with highway [43] structure similar to the discriminator in [44] for both actor and critic models. The stacked-CNN has diverse filter widths to cover different lengths of history and extract features, and it is easily parallelizable, compared to other sequential networks. Since state information is temporally correlated, such a sequential network extracts features better than multilayer perceptrons. The highway structure directs information flow by learning the weights of direct input and performing non-linear transform of the input.

In low contention, authors of [39] prove that an actor-critic [42] RL algorithm converges to NE in a potential game. In high contention, although we prove the existence of an NE in the static case, the convergence property of our algorithm in a stochastic game is not explicitly analyzed. We show it through empirical results in Sec.IV.

IV. Evaluation

We develop a Python discrete-event simulator, with varying number of vehicles of infinite lifespan, one MEC with ACA and edge computing site, and one remote computing site (extention to multiple ACA units and computing sites is left to future work). The edge and remote sites have different resource profiles. To imitate a realistic, noisy environment, the remote site is some distance to the ACA unit, such that data transmission would cause non-negligible delay in state information update. We also add a small, normally distributed noise to this delay, as well as to the actual resource required for a service. Each vehicle is randomly and independently initialized with a budget of "high" or "low" with 50% probability. For the operating-side load-balancing policy, we apply state-of-the art resource-intensity-aware load-balancing (RIAL) [45] with slight modifications. The method achieves dynamic load-balancing among computing sites through resource pricing that is correlated to the site's load, and loads are shifted to "cheaper" sites. Finally, we compare the performance of active agents (DRACO on the user side, RIAL on the operating side, or D+R) to passive agents (only RIAL on the operating side), as shown in Fig.1b.



(a) DRACO reduces the overall offloading failure rate.

(b) DRACO learns to better utilize resource in remote computing site.

Figure 3: OFR and resource utilization, capacity=60, MP=1

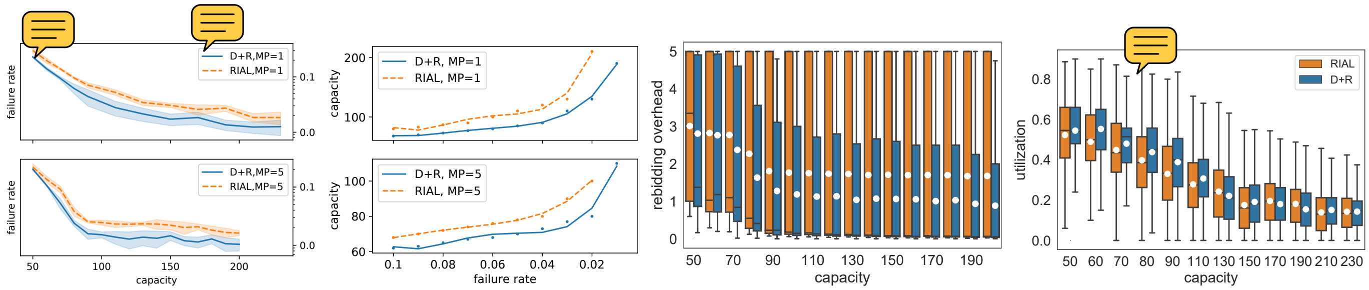
We test our approach in two steps. First, we comprehensively study the performance of active agents in a synthetic setup with randomized inputs and a wide range of environment parameters. Then, we choose a realistic scenario, a 4-way traffic intersection with realistic mobility model for vehicles and with incoming service requests modeled after specific V2X applications, to show the generalization properties of DRACO. We evaluate the following metrics:

- **Offloading failure rate (OFR):** Ratio of failed offloading requests rejected by ACA or not executed before deadline.
- **Resource utilization:** Ratio of resources effectively utilized at computing sites.
- **Rebidding overhead:** If a bid is rejected before deadline, the vehicle can bid again. More rebidding causes communication overhead, but less rebidding reduces the chance of success. We study this tradeoff, comparing the average number of actual rebiddings per vehicle within maximum permitted-rebidding (MP).

A. Synthetic setup

In this setup, we cover a wide range of hypothetical scenarios by varying parameters such as system capacity, service/task types and number of rebidding:

- Task types by resource needs in time-resource units: F1: 3 units, and F2: 30 units.



(a) D+R reduces OFR by 40%, achieves 1% OFR in low contention; RIAL OFR only reaches 2%.

(b) D+R needs less resource for same OFR (e.g., 2% failure and MP=1, 38% less resource needed).

(c) Rebidding overhead vs capacity, MP=5. Overhead reduces by 32% on average.

(d) Resource utilization, MP=1. DRACO better utilizes resource by 18% in high contention.

Figure 4: (a): OFR vs capacity, (b): required capacity to reach OFR≤ 10%, (c): rebidding overhead, (d): utilization by capacity

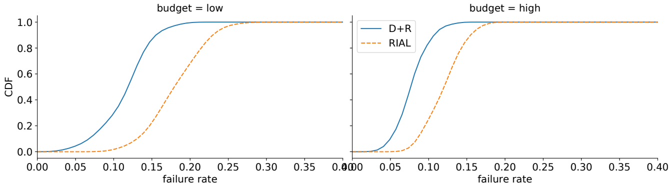
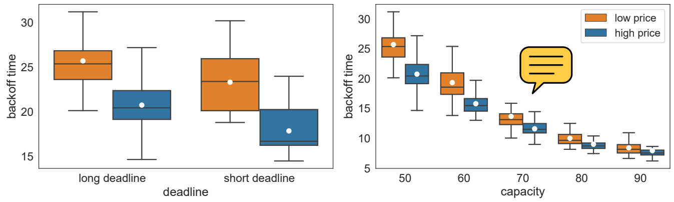


Figure 5: CDF of individual OFRs (capacity=70, MP=1): DRACO reduces individual OFRs.

- Service types by deadline and probability: F1, 300ms: 18.75%; F1, 50ms: 18.75%; F2, 300ms: 6.25%; F2, 50ms: 6.25%; F1-F2, 300ms: 18.75%; F1-F2, 50ms: 18.75%; F2-F1, 300ms: 6.25%; F2-F1, 50ms: 18.75%.
- Service arrival rate per vehicle: randomized according to a two-state Markov modulated Poisson process (MMPP) [46], with $\lambda_{\text{high}} \in (0.48, 0.6), \lambda_{\text{low}} \in (0, 0.12)$ and transition probabilities $p_{\text{high}} = p_{\text{low}} = 0.6$.
- Capacity: 50-230 resource units.
- Maximum permitted rebidding: 1 or 5 times, respectively.
- Vehicle count: constant at 30.
- Vehicle arrival rate: 0, always in the system; speed: 0.
- Data size: uniform random between 2.4-9.6kbit.
- Uplink and downlink latency: 0.

Fig.3a shows a training example where D+R's OFR is 14% compared to RIAL's 20% at the end of training, or a reduction of 30%. The lines are the mean OFR of serveral simulation runs, and the shaded area marks the standard deviation. Fig.3b shows where the learning is most useful. We depict the remote site's resource utilization. Since the ACA unit's information of site utilization is delayed, with only RIAL, the site is either over-utilized or starved, in distinctive cycles (dotted line). When the vehicles learn with DRACO, they achieve better utilization (solid line).

Overall OFR in all parameter settings is shown in Fig.4a. Evaluation data is collected from additional evaluation runs after the models are trained, with random incoming service requests newly generated by the MMPP. Besides requests that are not admitted by the ACA unit, the failure rate also includes requests that are admitted, but cannot be executed by the operating side before deadline (reliability). We observe that with D+R, reliability is 99% and consistently higher than with RIAL for all results in the paper. We also observe that DRACO significantly reduces OFR (on average 40% reduction), especially when MP is low. In low contention, i.e., capacity≥ 100, by efficient use of resource, D+R achieves the same level of OFR with much less resource (Fig.4b). The improvement becomes more significant as OFR decreases. In particular, D+R reaches 1% OFR with much less resource compared to RIAL regardless of MP.



(a) Capacity=50, MP=5, backoff vs. price tradeoff: for all deadlines, vehicles that bid low (high) use long (short) backoff.

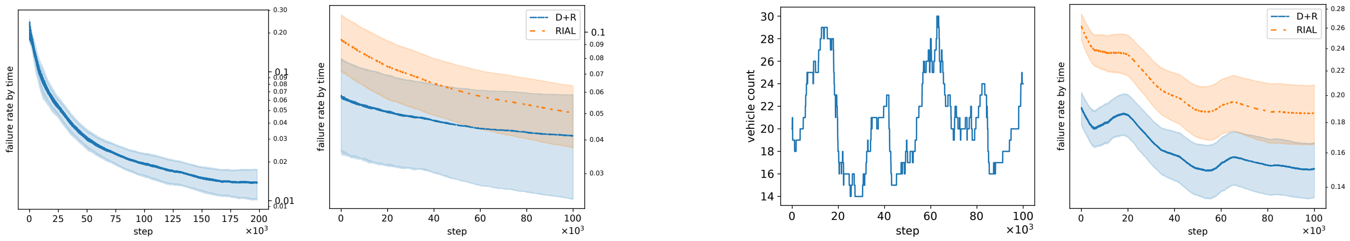
(b) MP=5, long deadline, high contention: backoff time decreases with higher capacity, but the tradeoff with price remains.

Figure 6: Backoff and price tradeoff

We also observe that higher MP reduces D+R's advantage over RIAL. This result is to be expected: when more rebidding is permitted, low OFR can be achieved by trial-and-error, limiting the advantage of DRACO's backoff strategy. However, trial-and-error comes with a cost: Fig.4c compares the rebidding overhead used by both algorithms when MP=5. In high contention, both active and passive agents leverage on rebidding, and the difference in rebidding overhead is small. D+R's advantage becomes more significant as capacity increases. The box plot shows the median (line), mean (dot), 1st to 3rd quartiles (box), and data range (whiskers). D+R imposes on average 32% lower rebidding overhead.

To validate our findings in Fig.3b, we compare resource utilization under different capacities. Fig.4d shows the remote site's utilization when MP=1. In high contention, the increase in utilization is up to 18%—when capacity is limited, D+R achieves lower OFR through more efficient resource usage. In low contention, capacity is less critical, active and passive agents result in similar utilization. Regardless of capacity level, D+R reduces the standard deviation in utilization by up to 21%.

Fig.5 shows the cumulative probability of vehicles' individual OFRs. With DRACO, as system overall OFR reduces, the individual OFRs reduce accordingly: the auction does not cause disadvantage to individual vehicles. Moreover, vehicles with lower budget improve by a greater margin: they learn to utilize backoff mechanism to overcome their disadvantage in initial parameterization. Fig.6a shows how vehicles learn to trade off between bidding price and backoff time. They are



(a) Training env.: low contention with abundant resource, traffic phase=10-40s, low vehicle speed(10km/h), low arrival rate=(1/2.2s), low variation in vehicle count(22-30): OFR in training(left) and evaluation(right).

(b) Test env.: high contention with limited resource, traffic phase=20s, high vehicle speed(30km/h), high arrival rate(1/1s), high variation in vehicle count(14-30): vehicle count over time(left) and OFR(right).

Figure 7: Offloading failure rate (OFR) in training and test environments

separated into two groups: a vehicle is in the "low price" group if it bids on average lower than the average bidding price of all vehicles; otherwise, it is in the "high price" group (here we analyze actual bidding prices instead of the predefined budgets). When service requests have a longer deadline, vehicles in both price groups learn to utilize longer backoff. Regardless of the service request deadline, "low price" vehicles always use longer backoff in their bidding decisions, compared to the "high price" group. Fig.6b demonstrates the tradeoff effect with increasing capacity. As capacity increases, backoff time decreases, but the tradeoff is present in all cases.

To summarize: Fig.3 and 4 demonstrate DRACO's excellent overall system performance. More importantly, Fig.5 shows that system objective is aligned with individual objectives through incentivization (**C1**), and Fig.6 demonstrates where our approach fundamentally differs from previous approaches: differently initialized agents learn to select the most advantageous strategy based on limited feedback signal (**C2**). The capability to learn and behave accordingly makes our agents highly flexible in a dynamic environment. Finally, Fig.3a shows good convergence speed despite computation and communication complexity of the problem (**C3**).

B. Realistic setup

In this setup, we adopt the data patterns of segmentation and motion planning applications extracted from various self-driving data projects [47] or referenced from relevant studies [48] [49]. We also use Simulation of Urban Mobility (SUMO) [50] to create a more realistic mobility model of a single junction with a centered traffic light; the junction is an area downloaded from open street map. Assuming 802.11ac protocol, we place the ACA unit in the middle of the graph and limit the edges to within 65m of the ACA. The net is with two lanes per street per direction, SUMO uniform-randomly creates a vehicle at any one of the four edges.

Parameters of the setup are as follows [47]–[49]:

- Task types: F1: 80 units, and F2: 80 units.
- Service types and deadline: F1: 100ms and F2: 500ms.
- Service arrival rate per vehicle: fixed at F1: every 100ms, and F2: every 500ms.
- Capacity: 20 in high contention, 30 in low contention.
- Maximum permitted rebidding: 1.
- Vehicle count: 14-30 from simulated trace data.
- Vehicle arrival rate: constantly at 1 every 1 or 2.2 seconds; speed: 10 or 30 km/h when driving.
- Data size: uplink: F1: 0.4Mbit, F2: 4Mbit. Downlink: F1: 0 (negligible), F2: 0.4Mbit.
- Latency: we take 802.11ac protocol that covers a radius of 65 meters, and assume maximum channel width of ca. 1.69 Gbps. We model the throughput as a function of distance to the ACA unit: throughput=−26×distance+1690 Mbps [51]. If there are N vehicles transmitting data to the ACA unit, we assume that each gets $1/N$ of the maximum throughput at that distance.

For training, we set the traffic light phases to 10-40s of green for each direction, alternatively. We train our active agents with DRACO in low contention. Fig.7a-left shows convergence to OFR of 2%. Then we evaluate the trained models in the same environment with newly simulated trace data from SUMO, our approach still reaches OFR of 4% and outperforms RIAL (Fig.7a-right). All simulations are repeated several times to take randomness into account.

Finally, we test our trained models in a significantly different environment, changing traffic light phases, vehicle arrival rate and speed to make the environment more volatile and dynamic, and reducing capacity to create a high-contention situation. The resulting vehicle count over time (Fig.7b-left) shows a much heavier and more frequent fluctuation compared to the training environment. Note that vehicle count and OFR do not vary synchronously—OFR is determined by vehicle count and numerous other complicating factors such as transmission, queueing and processing time, past utilization, etc. Despite the significant changes to the environment, D+R still outperforms RIAL, reaching low OFRs in high contention without requiring any further training (Fig.7b-right). It shows that DRACO has very good generalization properties—in fact, in the more volatile and dynamic environment, the superiority of active agents becomes more obvious.

V. Related Work

Centralized approaches such as [20], [21] for resource allocation, and [14], [52], [53] for offloading, are suited to core-network and data-center applications, when powerful central ACA can be set up, and data can be relatively easily obtained. They are not the focus of our study.

Previous studies of decentralized systems address some of the issues in centralized approaches. [54], [55] propose distributed runtime algorithm to optimize system goals, but disregard user preferences. [56]–[58] only consider cooperative resource-sharing or offloading. [22], [23], [59] require complete information to compute the desired outcome. [60] only considers discrete actions. [19] learns with partial information, but it reduces complexity by assuming single service type and arrival rate. Our approach also differs from [41] as we consider a multi-dimensional continuous action space with multiple service types, and both cooperative and competitive behaviors.

Besides the previously mentioned decentralized learning algorithms [25]–[28] for a dynamic environment, independent learner methods [61] are used to reduce modeling and computation complexity, but they fail to guarantee equilibrium [62], and have overfitting problems [63]. Finally, federated learning [64] is not applicable, as it provides a logically centralized learning framework.

VI. Conclusion

Our algorithm learns how to best utilize backoff option based on its initialization parameters. As a result, the algorithm achieves significant performance gains and very good generalization properties. Our interaction mechanism aligns private and system goals without sacrificing either user autonomy or system-wide resource efficiency, despite the distributed design with limited information-sharing.

We assume there is no "malicious" agent with the goal to reduce social welfare or attack the system. In general, agents with heterogeneous goals is left to future research. In V2X, all devices are potential computing sites; offloading between any devices should be considered. Long-term effect of decisions—e.g. if unused budget can be saved for the future, is also an interesting topic. How initialization and predefined parameters affect agent behavior and the algorithm's convergence property, needs to be studied in detail.

Appendix A
Proof of potential game

Proof. We define player i's utility as $u_i(\alpha_i, \alpha_{-i}) = \sum_{k\in K} q_{i,k} - \sum_{k\in K} \alpha_{i,k} q_{i,k} + W\left(1 - \frac{\sum_j \alpha_j \cdot \omega_j}{C}\right)$, where $\omega_j \in \mathbb{R}^K$ is the resource requirement of each commodity, C is the system capacity.

We define potential function: $\phi(\alpha_i, \alpha_{-i}) = \sum_{j\in I, k\in K} q_{j,k} - \sum_{j\in I, k\in K} \alpha_{j,k} q_{j,k} + W\left(1 - \frac{\sum_j \alpha_j \cdot \omega_j}{C}\right)$.

To simplify, we substitute with $Q_i = \sum_{k\in K} q_{i,k}$, $A_i = \sum_{k\in K} \alpha_{i,k} q_{i,k}$, $A_{-i} = \sum_{j\in I, j\neq i, k\in K} \alpha_{j,k} q_{j,k}$, $B_i = \sum_k \alpha_{i,k}\omega_{i,k}$, $B_{-i} = \sum_{j\in I, j\neq i, k\in K} \alpha_{j,k}\omega_{j,k}$, and rewrite: $u_i(\alpha_i, \alpha_{-i}) = Q_i - A_i + W - \frac{W}{C}(B_i + B_{-i})$ and $u_i(\alpha'_i, \alpha_{-i}) = Q_i - A'_i + W - \frac{W}{C}(B'_i + B_{-i})$; hence, $\phi(\alpha_i, \alpha_{-i}) = \sum_j Q_j - (A_i + A_{-i}) + W - \frac{W(B_i + B_{-i})}{C}$, $\phi(\alpha'_i, \alpha_{-i}) = \sum_j Q_j - (A'_i + A_{-i}) + W - \frac{W(B'_i + B_{-i})}{C}$, which implies $u_i(\alpha_i, \alpha_{-i}) - u_i(\alpha'_i, \alpha_{-i}) = -(A_i - A'_i) - \frac{W}{C}(B_i - B'_i) = \phi(\alpha_i, \alpha_{-i}) - \phi(\alpha'_i, \alpha_{-i})$. □

Appendix B
Second-price auction

Under high contention, as defined in Sec.III-A, u_i is reduced to:

$$u_i = \sum_{k\in K} \Big(x_{i,k} \cdot (v_{i,k} - p_{i,k}) - (1 - x_{i,k}) \cdot c_{i,k}\Big) \quad (4)$$

We prove the theorem for $|M| = 2$ and $|K| = 1$, extension to other settings is straightforward. Our proof is an extension from [65]. Unlike [65], we include in utility the second-price payment and cost for losing a bid. Based on [65], it can also be extended to multiple bidders.

2 bidders receive continuously distributed valuations $v_i \in [l_i, m_i], i \in \{1, 2\}$ for 1 commodity, and choose their strategies $f_1(v_1), f_2(v_2)$ from the strategy sets F_1 and F_2. The resulting NE strategy pair is (f_1^*, f_2^*). Any strategy function $f(v)$ is increasing in v, with $f_1(l_1) = a$, and $f_1(m_1) = b$. We assume users have budgets (B_1, B_2), and that they cannot bid more than the budget. We define cost for losing the bid c_i.

We formulate the problem into a utility maximization problem: $\max_{f_2 \in S_2(f_1)} u_2(f_1, f_2)$. We say f_2 is a best response of bidder 2, if $u_2(f_1, f_2) \geq u_2(f_1, f'_2), \forall f'_2 \in S_2(f_1)$. A NE strategy pair (f_1^*, f_2^*) has the strategies as each other's best responses.

Theorem B.1. Given bidder 1's bidding strategy $f_1 \in F_1, f_1(l_1) = a_1, f_1(m_1) = b_1$, bidder 2's best response has the form $\begin{cases} f_2(v_2) \leq a_1 & \text{for } v_2 \in [l_2, \theta_1] \\ f_2(v_2) = j_2 \cdot v_2 + d_2 & \text{for } v_2 \in [\theta_1, \theta_2] \\ f_2(v_2) \geq b_1 & \text{for } v_2 \in [\theta_2, m_2] \end{cases}$, where $\theta_1, \theta_2 \in [l_2, m_2]$ and $j_2\theta_1 + d_2 = a_1, j_2\theta_2 + d_2 = b_1$.

Theorem B.1 implies that the best response of bidder 1 and 2 are both of the linear form. Using the new best responses function, we similarly extend the proof of the NE outcome and welfare maximization to suit our case. Detailed proof is provided in supplemental meterial [29].

Appendix C
Pareto optimality

Valuation of the service request is a linear function of the resource needed: $v_1 = g_1\omega_1 + k_1, v_2 = g_2\omega_2 + k_2$, g, k are constants, ω is amount of resource required. The allocation rule under NE is: $A^*_{v_1,v_2} = 1$, if $j_1 v_1 + d_1 \geq j_2 v_2 + d_2$, otherwise 2. Form of the condition is from best response form in appendix Sec.B. We also assume that both bidders have at least some access to the resources, as a form of fairness. We define the fairness constraint: $\mathbb{E}[\omega_1|_{A_{v1,v2}=1}]/\mathbb{E}[\omega_2|_{A_{v1,v2}=2}] = \gamma \in \mathbb{R}_{>0}$.

Theorem C.1. The allocation $A^*_{v_1,v_2}$ maximizes overall resource allocation $\omega_1 + \omega_2$, subject to the fairness constraint, when the valuations are linear functions of resources. Or, the NE of the game achieves optimal resource allocation.

Proof. Find the Lagrangian multiplier λ^* that satisfies the fairness constraint with NE allocation $A^*_{v_1,v_2}$. Define g, k as: $g_1 = (1 + \lambda^*)/j_1$, $k_1 = -d_1/j_1$, and $g_2 = (1 - \gamma\lambda^*)/j_2$, $k_2 = -d_2/j_2$. We rewrite the allocation: $A^*_{\omega_1,\omega_2} = 1$, if $\omega_1(1+\lambda^*) \geq \omega_2(1-\gamma\lambda^*)$, otherwise 2. Rest of the proof is same as [65]. □

References

[1] A. Masmoudi *et al.*, "A survey on radio resource allocation for v2x communication," *Wireless Communications and Mobile Computing*, 2019.

[2] M. Hofmarcher *et al.*, "Visual scene understanding for autonomous driving using semantic segmentation," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[3] L. Claussmann *et al.*, "A review of motion planning for highway autonomous driving," *IEEE Trans. on Intelligent Transp. Systems*, 2019.

[4] C. Badue *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, 2020.

[5] C.-s. Oh *et al.*, "Hardware acceleration technology for deep-learning in autonomous vehicles," in *IEEE BigComp*, 2019.

[6] C. J. Bernardos *et al.*, "European vision for the 6g network ecosystem," *The 5G Infrastructure Association*, 2021.

[7] X. You *et al.*, "Towards 6g wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *Science China Information Sciences*, 2021.

[8] "C-v2x use cases: Methodology, examples and service level requirements," *5GAA Automotive Association*, 2019.

[9] "C-v2x use cases volume ii: Examples and service level requirements," *5GAA Automotive Association*, 2020.

[10] P. Mach *et al.*, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Comm. Surveys & Tutorials*, 2017.

[11] S. Baidya *et al.*, "Vehicular and edge computing for emerging connected and autonomous vehicle applications," in *ACM/IEEE DAC*, 2020.

[12] G. Loukas *et al.*, "Computation offloading of a vehicle's continuous intrusion detection workload for energy efficiency and performance," *Simulation Modelling Practice and Theory*, 2017.

[13] M. Masdari *et al.*, "Qos-driven metaheuristic service composition schemes: a comprehensive overview," *Springer AI Review*, 2021.

[14] S. Choo *et al.*, "Optimal task offloading and resource allocation in software-defined vehicular edge computing," in *IEEE ICTC*, 2018.

[15] M. Vondra *et al.*, "Qos-ensuring distribution of computation load among cloud-enabled small cells," in *IEEE CloudNet*, 2014.

[16] S. Shivshankar *et al.*, "An evolutionary game theory-based approach to cooperation in vanets under different network conditions," *IEEE Trans. on Vehicular Technology*, 2014.

[17] F. J. Martinez *et al.*, "Assessing the impact of a realistic radio propagation model on vanet scenarios using real maps," in *NCA*, 2010.

[18] J. Feigenbaum *et al.*, "Distributed algorithmic mechanism design," in *Algorithmic Game Theory*. Cambridge University Press, 2007.

[19] L. Li *et al.*, "Learning-based pricing for privacy-preserving job offloading in mobile edge computing," in *IEEE ICASSP*, 2019.

[20] T.-W. Kuo *et al.*, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Trans. on Networking*, 2018.

[21] S. Agarwal *et al.*, "Joint vnf placement and cpu allocation in 5g," in *IEEE INFOCOM*, 2018.

[22] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. on Parallel and Distributed Systems*, 2014.

[23] V. Cardellini *et al.*, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, 2016.

[24] J. Oh *et al.*, "A few good agents: multi-agent social learning," in *AAMAS*, 2008.

[25] Y.-H. Chang, "No regrets about no-regret," *Artificial Intelligence*, 2007.

[26] M. Weinberg *et al.*, "Best-response multiagent learning in non-stationary environments," in *AAMAS*, 2004.

[27] M. Bowling *et al.*, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, 2002.

[28] J. Heinrich *et al.*, "Fictitious self-play in extensive-form games," in *ICML*, 2015.

[29] "Draco source code," https://github.com/DRACOsource/draco.

[30] M. Whaiduzzaman *et al.*, "A survey on vehicular cloud computing," *Journal of Network and Computer applications*, 2014.

[31] "Iso 20078:2019 road vehicles-extended vehicle (exve) web services," *International Organization for Standardization*, 2019.

[32] "Service-based architecture in 5g: case study and deployment recommendations," 2019.

[33] R. M. Schindler *et al.*, *Pricing strategies: a marketing approach*. sage, 2011.

[34] L. Einav *et al.*, "Auctions versus posted prices in online markets," *Journal of Political Economy*, 2018.

[35] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of finance*, 1961.

[36] M. Feldman *et al.*, "Simultaneous auctions are (almost) efficient," in *ACM Symposium on Theory of Computing*, 2013.

[37] F. Cali *et al.*, "Ieee 802.11 protocol: design and performance evaluation of an adaptive backoff mechanism," *IEEE JSAC*, 2000.

[38] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, 1996.

[39] S. Perkins *et al.*, "Mixed-strategy learning with continuous action sets," *IEEE Trans. on Automatic Control*, 2015.

[40] D. S. Leslie *et al.*, "Generalised weakened fictitious play," *Games and Economic Behavior*, 2006.

[41] M. Khaledi *et al.*, "Optimal bidding in repeated wireless spectrum auctions with budget constraints," in *IEEE GLOBECOM*, 2016.

[42] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[43] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *NeurIPS*, 2015.

[44] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *AAAI*, 2017.

[45] H. Shen *et al.*, "A resource usage intensity aware load balancing method for virtual machine migration in cloud datacenters," *IEEE Trans. on Cloud Computing*, 2020.

[46] K. Wang *et al.*, "Characterizing the impact of the workload on the value of dynamic resizing in data centers," in *IEEE INFOCOM*, 2013.

[47] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016.

[48] B.-k. Chen *et al.*, "Importance-aware semantic segmentation for autonomous driving system." in *IJCAI*, 2017.

[49] A. Broggi *et al.*, "Proud-public road urban driverless test: Architecture and results," in *IEEE Intelligent Vehicles Symposium Proceedings*, 2014.

[50] M. Behrisch *et al.*, "Sumo–simulation of urban mobility: an overview," in *SIMUL*, 2011.

[51] Z. Shah *et al.*, "Throughput comparison of ieee 802.11 ac and ieee 802.11 n in an indoor environment with interference," in *IEEE ITNAC*, 2015.

[52] X. Lyu *et al.*, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. on Vehicular Technology*, 2016.

[53] M. Chen *et al.*, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE JSAC*, 2018.

[54] M. Blöcher *et al.*, "Letting off steam: Distributed runtime traffic scheduling for service function chaining," in *IEEE INFOCOM*, 2020.

[55] S. Schneider *et al.*, "Self-learning multi-objective service coordination using deep reinforcement learning," *IEEE Trans. on Network and Service Management*, 2021.

[56] N. Kumar *et al.*, "Bayesian coalition game as-a-service for content distribution in internet of vehicles," *IEEE IoT Journal*, 2014.

[57] ——, "Coalition games for spatio-temporal big data in internet of vehicles environment: a comparative analysis," *IEEE IoT Journal*, 2015.

[58] X. Chen *et al.*, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. on Networking*, 2015.

[59] Z. Chen *et al.*, "Mechanism design with efficiency and equality considerations," in *Intl Conference on Web and Internet Economics*, 2017.

[60] F. Shams *et al.*, "Energy-efficient power control for multiple-relay cooperative networks using q-learning," *IEEE Trans. on Wireless Comm.*, 2014.

[61] M. Tan, "Multi-agent reinforcement learning: independent vs. cooperative agents," in *ICML*, 1993.

[62] Y. Yang *et al.*, "Mean field multi-agent reinforcement learning," in *ICML*, 2018.

[63] M. Lanctot *et al.*, "A unified game-theoretic approach to multiagent reinforcement learning," in *NeurIPS*, 2017.

[64] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *aistats*, 2017.

[65] J. Sun *et al.*, "Wireless channel allocation using an auction algorithm," *IEEE JSAC*, 2006.