IEEE ComSoc IEEE Communications Society | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 29 June 2023; revised 29 January 2024; accepted 1 February 2024.
Date of publication 12 February 2024; date of current version 16 February 2024.

The associate editor coordinating the review of this article and approving it for publication was J. J. P. C. Rodrigues.

Digital Object Identifier 10.1109/TMLCN.2024.3365420

Buyers Collusion in Incentivized Forwarding Networks: A Multi-Agent Reinforcement Learning Study

MOSTAFA IBRAHIM[1], SABIT EKIN[1,2] (Senior Member, IEEE), AND ALI IMRAN[3,4] (Senior Member, IEEE)

[1]Department of Engineering Technology, Texas A&M University, College Station, TX 77843 USA
[2]Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA
[3]James Watt School of Engineering, University of Glasgow, G12 8QQ Glasgow, U.K.
[4]AI4Networks Research Center, School of Electrical and Computer Engineering, The University of Oklahoma, Tulsa, OK 74135 USA

CORRESPONDING AUTHOR: M. IBRAHIM (mostafa.ibrahim@tamu.edu)

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award DE-SC0023957; and in part by the U.S. National Science Foundation under Grant 2323300 and Grant 1923669.

ABSTRACT We present the issue of monetarily incentivized forwarding in a multi-hop mesh network architecture from an economic perspective. It is anticipated that credit-incentivized forwarding and relaying will be a simple method of exchanging transmission power and spectrum for connectivity. However, gateways and forwarding nodes, like any other free market, may create an oligopolistic market for the users they serve. In this study, a coalition scheme between buyers aims to address price control by gateways or nodes closer to gateways. In a Stackelberg competition game, buyer agents (users) and sellers (gateways) make decisions using reinforcement learning (RL), with decentralized Deep Q-Networks to buy and sell forwarding resources. We allow communication links between the buyers with a limited messaging space, without defining a collusion mechanism. The idea is to demonstrate that through messaging, and RL tacit collusion can emerge between agents in a decentralized setup. The multi-agent reinforcement learning (MARL) system is presented and analyzed from a machine-learning perspective. Moreover, MARL dynamics are discussed via mean field analysis to better understand divergence causes and make implementation recommendations for such systems. Finally, the simulation results show the results of coordination among the users.

INDEX TERMS Multi-agent, reinforcement learning, IoT, incentivized forwarding.

LIST OF SYMBOLS

The following list describes the symbols used within the paper

- L Gateway load.
- α Learning rate.
- η Load induced latency penalty.
- γ Discount rate mapping to future rewards.
- $\mathbb{F}$ Reward matrix.
- $\mathbb{G}$ Set of gateways.
- $\mathbb{P}_z$ Probability vector of all actions of the population of agents.
- $\mathbb{U}$ Set of users.
- $\mathbf{Q}$ State-action Q point in the Mean Field Theory analysis.
- $\mathbf{v}$ Speed vector of agents in the Mean Field Theory analysis.
- ψ The flow density of the agents.
- ε Exploration factor.
- ζ_x The suggestions vector.
- A Set of all valid actions in a Markov Decision Process.
- a An action of an agent.
- b Size of forwarded packets in bytes.
- C_{fj} Communication and processing costs with user j.
- C_j The message transmission cost from or to user j.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

dS	A differential hyper rectangle face in the Mean Field Theory analysis.
dV	A differential hyper-rectangle volume in the Mean Field Theory analysis.
e_j	A unit vector in the direction of the joint action of the population.
e_z	A unit vector in the direction of a_j.
G_j	The gain that a user is getting from transferring its message.
k_i	Set of nodes on route gateway i.
P_{fx}	Price of packet forwarding per byte at node x.
Q	Action-value function of a learning process.
R	The reward function for a Markov Decision Process.
R	The reward function for a Markov Decision Process.
S	Set of all valid states in a Markov Decision Process.
s	A state of an agent.
T	Transition probability between states in a Markov Decision Process.
U_{GW_i}	Utility function of gateway i.
U_{user_j}	Utility function of User j.
V	Value function of a learning process.

I. INTRODUCTION

INCENTIVIZED relaying in communication networks is a method proposed and studied in the literature to enable cooperative communication and forwarding in specific application networks [1]. It has potential applications in IoT networks, future cellular networks, smart cities, smart homes, wearable devices, and smart grids, where multi-hop mesh network architectures can enhance network spectral and energy efficiencies [2].

In the future generations, radio access networks are expected to become more open to a diverse set of wireless technologies [3]. This openness may also extend to individual or third-party operators who can run and maintain their services on the radio access plane. Consequently, monetizing forwarding services will be crucial for ensuring smooth operations with third-party-owned relay devices. In such scenarios, credit-incentivized forwarding and relaying become a straightforward approach for trading transmission power and spectrum in exchange for connectivity. However, similar to any free market, gateways (GWs) and forwarding nodes can establish an oligopolistic market for the users they serve.

An oligopolistic market occurs when a small number of sellers have control over market prices and the produced commodity [4]. In our case, gateways and hops in proximity to them exert control over the forwarding price. Individual users must decide whether to purchase a resource at a specific price or forgo it. However, due to the limited number of gateways, users may have a specific optimal route in terms of energy consumption and delay. In some cases, users may not have viable alternative options, allowing gateways to increase their forwarding prices to maximize their utility, and users are left with no choice but to comply. Alternatively, users may seek other gateways, which may not offer the optimal route and may not justify the price difference.

In this study, we propose a coalition scheme among buyers to address the issue of price control by gateways or nodes in close proximity to the gateways. The objective is to determine the parameters and communication messages between users that lead to beneficial coalition results. To achieve this, we employ reinforcement learning (RL) as the decision-making method for buyer agents (users) and sellers (gateways) in a Stackelberg[1] competition game. Implementation-wise, we utilize Deep Q-learning, which combines RL with deep neural networks to represent state-value functions.

In this paper, we tackle a multi-agent reinforcement learning (MARL) problem that involves the behavior of multiple learning agents coexisting in a shared environment. Each agent is driven by individual rewards and acts in its own self-interest, which often conflicts with the interests of other agents, leading to complex group dynamics.

The agents in the system are represented as independent learners utilizing decentralized Deep Q-Networks (DQN). Both competing sides employ DQN reinforcement learning to engage in the Stackelberg game. One notable challenge in this setup is the absence of a central entity, such as a critic or referee, which poses difficulties for system convergence. We delve into the dynamics of the distributed MARL system, the evolution of Q-functions, and the challenges arising from the inherent nonstationarity of the system. Furthermore, we provide implementation recommendations for such systems.

To facilitate secure and accurate storage of information, a distributed ledger is utilized, leveraging Distributed Ledger Technology (DLT). DLT enables the establishment of a decentralized digital database, reducing reliance on a centralized authority to guard against manipulation. By utilizing encryption and cryptographic signatures, DLT ensures the safe storage of any type of information within the distributed network.

We outline the novelty of our contribution in this paper as follows:

- We propose a MARL-based coalition approach to address the problem of incentivized forwarding. Our

[1]The Stackelberg leadership model is a strategic game in economics that describes a hierarchical structure within an oligopoly market. In this model, firms are classified into two groups: leaders and followers. The leader firm, or Stackelberg leader, acts first by setting its output or price, anticipating the reactions of the follower firms. Once the leader has made its decision, the follower firms respond sequentially, taking the leader's action as given and adjusting their own output or price accordingly. The Stackelberg model is particularly useful for examining situations where one firm possesses a clear advantage or dominant position in the market, allowing it to influence the behavior of its competitors. By anticipating the followers' responses to their actions, the leader firm can strategically set its output or price to maximize its profit. This game-theoretic model provides valuable insights into the dynamics of competition in markets with asymmetric information or power.

system architecture assumes that the users can form a coalition to exert control over the gateways' prices. We motivate the use of DLT as a technology enabler for transactions and the exchange of value. The ledger is used in our study as an enabler of the forwarding method.
- Our approach employs a fully distributed MARL framework, where there is no central critic or referee to regulate the learning process. Additionally, the agents operate without any predefined behavioral rules and rely solely on DQNs. This distinguishes our work from most MARL literature, which often relies on central critics or referee agents for guidance.
- The proposed collaboration architecture is implemented using a two-level learning scheme. The first level of DQN facilitates communication between users through shared suggestions. The second level of DQN handles trading decisions while considering the suggestions received from the first level. We also introduce a customized learning schedule for this scheme.
- To gain a deeper understanding of the dynamics of our MARL system, we conduct an extensive analysis using Mean Field Theory [5]. Furthermore, we discuss the reasons for divergence from a statistical differential formulation perspective. Additionally, we propose several practices that increase the likelihood of MARL convergence.

This paper is organized as follows. In the next section, we briefly present the related work from the literature. The proposed system architecture is introduced in Section III. RL formulation is shown in Section IV, followed by our MARL system formulation in Section V. The system dynamics are discussed in Section VI. We present implementation notes in Section VII. Finally, the simulation & results are in Section VIII.

II. RELATED WORK

Several applications benefit from incentivized forwarding because it can be a solution for uncooperative behavior in ad-hoc, multi-hop, and device-to-device (D2D) communication networks. In [6], the use of wireless power transfer is provided in exchange for sensed data to incentivize the owners of wearables to participate in collaborative data collection. In [7], sensing and signal forwarding activity is the commodity being traded in a wireless sensor network. The incentives assumed in this study are the bandwidth assigned to the collaborating nodes. Also in [8], in order to optimize the transmit energy in a multi-hop network virtual cost incentives are paid to encourage cooperation to forward messages throughout the network.

A. INCENTIVIZED FORWARDING MECHANISMS

In this study, our focus is the credit-based mechanisms where virtual credits are used to reward forwarding. In the literature, however, the forwarding methods are not limited to credit-based and there are other studies that introduces Reputation based methods [9], [10], [11], [12], Barter based methods [13], [14], [15], [16], [17] and variations of them coupled with game theoretic methods [18], [19], [20], [21]. In Table 1, we present a brief comparison between the methods, with credit-based methods. Credit-based methods offer clear, tangible rewards (credits) [22], [23], making them effective where trust-based or reciprocal systems might fail. They also provide flexible, dynamic resource allocation, adapting to the varying contributions of network participants. These methods come with challenges. Implementing Credit-Based Methods requires complex accounting and robust security to prevent fraud and maintain fairness. Furthermore, they may demand advanced hardware or software, which could be a limitation in environments with limited resources. Despite these obstacles, the direct incentivization and adaptability of Credit-Based Methods make them an attractive option for enhancing cooperative behavior in wireless networks.

TABLE 1. Comparing incentivized forwarding mechanisms.

Criterion	Barter Methods	Reputation-Based Methods	Credit-Based Methods
Incentive Alignment	Direct service exchange leads to immediate and tangible benefits.	Long-term benefits through maintained reputation.	Direct and quantifiable benefits through virtual credits.
Overhead & Complexity	Low, simple direct exchanges.	High, need to track and update reputations.	Moderate to high, need for secure credit accounting.
Robustness to Selfish Behavior	Limited, as not useful for infrequent service exchanges.	High, as reputation loss is a deterrent.	High, as denying service leads to no credits.
Adaptability & Scalability	Limited, best for small, stable networks.	Moderate, depends on information dissemination.	High, especially in dense networks.
Fairness	Direct but may not consider varying costs or asymmetric links.	Depends on accurate and fair reputation assessment.	Fair if credits are managed impartially.
Trust & Security	Low, relies on pairwise trust.	Moderate to high, vulnerable to false reporting.	High, requires mechanisms for secure credit transactions.

The credit-based schemes can be classified as Stackelberg games, auction-based games, bargaining-based games, and coalition-based games [1]. A Stackelberg game is an economic game with a market leader/s and followers, and both are competing for the maximum profits. In [24], in a two-stage Stackelberg game, the network operator pays a population of mobile users a reward in exchange for their collaboration in data delivery, and users decide whether to collaborate or not. In [25], incentivized users forwarding to one another is studied in a wireless ad-hoc video-on-demand system using a Stackelberg game. In [26], the base station aims at minimizing its cost while the users maximize their utility by choosing a caching policy. In [27], a Stackelberg game allows the source IoT devices to maximize the power purchased from the forwarding relays and allows the relays to improve the price of transmit power they use for forwarding. In [28], a forwarding pricing mechanism is proposed taking the battery levels into consideration. The payment can be in the form of currency or credits in a multimedia application.

Auction-based games [29], [30], [31], [32] and bargaining games [33], [34], [35], [36] have also been presented as frameworks for studying the incentivized forwarding

problem. But they are not closely related to the framework in our study because their exchange of information to decide the pricing is different from our work. In our work, we study the coalition of users or source nodes in a Stackelberg competition repeated game.

Coalition formation game-based credit scheme is also a part of the incentivized forwarding literature. A coalition game is a game where a set of players act as a single entity to gain a higher payoff, called coalition value. The work in [46] proposed a coalitional game with transferable utilities for mobile user-provided networks. In [47], boundary and backbone nodes form a coalition for data relaying across the network. The coalition is implemented via a proposed coalition routing protocol. In [48], in a coalition game, the users try to enhance their delivery delay and relaying cost by deciding to enter or to opt-out of a coalition. In [49], a coalition graph game helps incentive data dissemination with a minimum power consumption over the network. In [50] and [51], social trust and reciprocity help form stable coalitions between devices to share network resources. The novelty of our paper is the application of pure multi-agent distributed reinforcement learning in a credit-incentivized forwarding problem, where users learn to collude without the help of any external referees, critiques, or rules.

In Table 2, we infer a brief comparison between the credit-based schemes mentioned above. The decision-making process in Stackelberg games is streamlined, with leaders determining strategies that followers react to, simplifying the communication of strategies and responses. This process reduces the complexity inherent in other game-theoretic approaches, such as the iterative bidding in auction-based games or the continuous negotiations in bargaining-based games. In terms of implementation, while Stackelberg games require the leader to develop decision algorithms that take network dynamics into account, they avoid the elaborate setup necessary for auctions, the ongoing negotiations of bargaining, and over head communication for coalition formations. This can result in lower overhead, especially as the complexity of the network increases.

B. REINFORCEMENT LEARNING COALITION GAMES

Reinforcement learning methods have been used in coalition games for purposes other than incentivized forwarding. In [54], Bayesian reinforcement learning coalition formation is used to solve the problem of distributed resource sharing in device-to-device enabled heterogeneous networks. In [55], micropower grids co-schedule their demands and generate energy levels to minimize the overall costs using Bayesian coalition reinforcement learning. While in [56], micropower grids form a coalition to save power losses of energy exporting from the distant grid. In [57], base stations from different service providers create coalitions as a hierarchical model to solve the computational offloading of a mobile edge computing network.

C. MACHINE LEARNING AND BLOCKCHAIN

In [58] and [59], joint consideration of blockchain and ML may bring significant benefits as it can achieve decentralized, secure, intelligent, and efficient network operation and management. Blockchain can significantly facilitate training data and ML model sharing, decentralized intelligence, security, privacy, and trusted decision-making of ML. The authors in [59] provide a brief survey on blockchain applications for artificial intelligence (AI), which includes ML and other intelligent techniques. They discuss the existing blockchain applications, platforms, and protocols targeting AI area. Decentralized Intelligence: The decentralized nature of blockchain provides the fundamental protocols to enable decentralized ML applications. By applying blockchain, ML could learn, train, and derive decision-making on various end devices in decentralized and distributed networks. In particular, smart contracts and decentralized applications (DAPPs) may provide new opportunities to model the interactions between different entities in an ML application.

In [60], distributed blockchain-based scheme is used to create virtual wireless networks (VWNs) where primary wireless resource-owners PWROs sublease their resources (e.g., a slice of RF spectrum) to mobile virtual network operators MVNOs using machine-to-machine communications based on their Service Level Agreements (SLAs). In [61], cryptocurrency is used in distributed peer-to-peer applications to incentivize users to cooperate. A pricing strategy is proposed to guarantee the security of the incentive mechanism.

In the next section, we present our system architecture.

III. SYSTEM ARCHITECTURE

The network architecture employed in this study is a mesh network consisting of inner nodes and edge gateways, as in Fig. 1. The inner nodes can represent various entities, such as wireless sensors, IoT devices, or human users. The gateways are nodes owned by third parties other than the formal service providers. The gateways forward packets to adjacent clusters or a service provider's core network for the inner users in the uplink and downlink directions. The gateways are incentivized by users by collecting monetary rewards from the adjacent hops.

In the proposed multi-hop network, each forwarding, or group of forwarding, operations will be paid off from the inner hop to the hop closer to the gateways. To hold these monetary transactions, we assume a locally distributed ledger. This ledger serves as a means of keeping track of the debts owed by nodes to one another, and at specific intervals, the debts can be settled through practical means. The distributed ledger is managed by a subset of the nodes with the computational capabilities of keeping consensus and power capabilities necessary for communication overhead related to the ledger. In a real-world scenario, these nodes would be incentivized by imposing a small tax on each transaction they handle. It is important to note that the detailed analysis of

TABLE 2. Comparison of credit-based schemes.

Criterion	Stackelberg Games [24]–[28], [37]	Auction-Based Games [29]–[32], [38]–[40], [40]–[42]	Bargaining-Based Games [33]–[36], [43]–[45]	Coalition-Based Games [46]–[51], [42], [52], [53]
Incentive Structure	Hierarchical, leader sets rules, followers optimize responses. Efficient for networks with central resource management.	Competitive bidding for resources; efficient allocation via market-driven mechanism. Suited for dynamic resource demands.	Direct negotiation for mutual agreements; adaptable to individual needs, ideal for personalized arrangements.	Collaborative, groups formed for shared goals; emphasizes cooperation, suitable for collaborative networks.
Decision-Making Process	Leader sets strategy, followers react strategically. Effective communication of strategy is key.	Dynamic bidding with strategic balance of cost and benefit. Multiple rounds for optimal resource allocation.	Interactive negotiations with offers and counteroffers; requires effective communication and compromise.	Collective decisions on strategies and resource allocation; focus on optimizing performance and fairness.
Implementation Complexity	Requires leader's decision algorithms considering network dynamics. Effective strategy communication is essential.	Complex auction design for fair and efficient outcomes; robust platform needed for secure bidding.	Mechanisms for negotiation and communication; iterative process with continuous adjustments.	Forming and maintaining coalitions with communication and shared resource management.
Overhead and Cost	Information processing and decision-making by the leader; costs increase with network complexity.	Significant for auction setup and bid processing; justified in environments with varied resource demands.	High for continuous communication and negotiation; time-intensive with iterative adjustments.	Coordination and communication among members; managing shared resources and monitoring agreements.
Scalability	Effective with a capable central authority; scalability depends on leader's adaptability.	Challenging with more participants; efficient auction design crucial for maintaining allocation efficiency.	Challenges increase with participant numbers; time and complexity of negotiations are limiting factors.	Moderate; suitable for networks where small-group collaboration enhances performance.
Fairness	Depends on leader's strategy fairness; risk of bias towards leader, requiring transparency.	Fair with well-designed auctions; risks of resource-rich participant dominance.	High potential if negotiations are balanced; risks of disparity in negotiation power or information.	Fair distribution aimed based on contributions; challenges in ensuring fairness in diverse contributions.

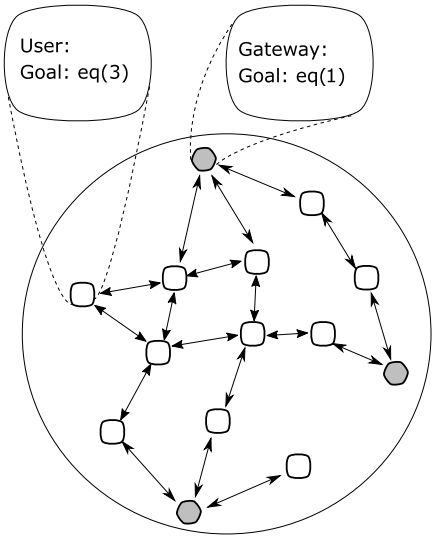
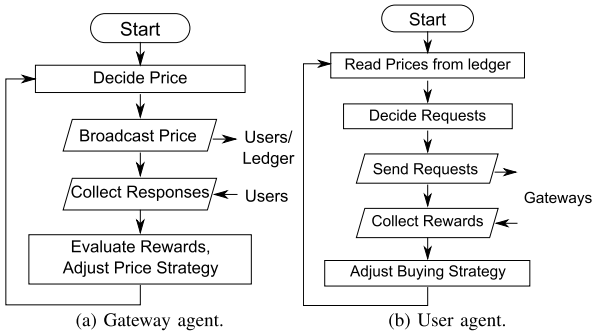


FIGURE 1. An example of the network of the user agents and the gateway agents.



(a) Gateway agent. (b) User agent.

FIGURE 2. Stackelberg game agents.

this aspect of the network is beyond the scope of this study, and the existence of a functional ledger capable of storing transactions and forwarding prices is assumed.

The network structure in this study represents an oligarchy of gateways, where a small group holds the power to set the forwarding price. This hierarchy results in the gateways being able to raise prices, while the individual inner users have limited power to negotiate or bargain individually by adjusting their demands. To address this issue, a control plane is introduced to enable information exchange among the users, which can help them form coalitions if they choose to do so. The purpose of these coalitions is to bring the forwarding prices to their lowest possible values.

In this context, the utility function of a gateway takes into consideration the communication and processing costs C_f for the forwarded packets and the price P_f it can set for the users. It is formulated as:

$$U_{GW_i} = \sum_{j \in J} b\,(P_{fi} - C_{fj}), \tag{1}$$

where the subscript i refers to the i^{th} gateway, C_{fj} is the forwarding cost for b occupied resource blocks at the GW's resource grid, for the j^{th} user. P_{fi} is the forwarding price set by the gateway. It is assumed that the offered price is the same for all the users. The forwarding can occur in the uplink or downlink direction, and the same price is assumed for both directions.

A user who chooses to get its packet forwarded by gateway i has the following utility function:

$$U_{user_j} = b\left[G_j - (P_{fi} + \sum_{x \in k_i} P_{fx}) - C_j\right] - \eta L(\%), \quad (2)$$

where G_j represents the gain that a user is getting from transferring its message, and it is the reason the user wants to communicate with the outer world. Hence, G_j reflects the communication value for each user, which is something specific to the mission of the device or its service. P_{fx} is the forwarding price for a node on the route k_i that leads to the gateway i. C_j is the message transmission cost from or to user j; for simplicity, we assume it is the same for up and downlinks. Queuing latency may occur when a gateway experiences high demand, resulting in requests being placed in a queue for processing. To account for the negative impact of latency on user rewards, $L(\%)$ represents the load of the serving gateway in percentage, and η represents the load-induced latency penalty. The parameter η captures the combined effects of latency resulting from loads and the inconvenience experienced by the user due to latency.

During a specific duration, the user j will be served by a set of gateways $\mathbb{G}$, as it may choose to diversify its routes. Therefore, the utility function over this duration is:

$$U_{user_j} = \sum_{i \in \mathbb{G}} b_j^{(i)} \left[G_j - (P_{fi} + \sum_{x \in k_i} P_{fx}) - C_j\right] - \eta L(\%). \quad (3)$$

Fig. 2 shows the decision-making workflows for both gateway and user agents within the proposed network. For the gateway agent, the process begins with deciding on a price for forwarding services, followed by broadcasting this price to users and the ledger. Responses from users are then collected, enabling the gateway to evaluate rewards and adjust its price strategy accordingly. Meanwhile, the user agent workflow initiates by reading prices from the ledger, which informs their subsequent decisions on requests, managing transactions with gateways, collecting rewards, and adjusting their buying strategy. These flowcharts are central to understanding how agents dynamically interact within the repeated Stackelberg game framework, which encapsulates the strategic economic interactions in our model.

The economic formulation of the interaction in our model can be described as repeated Stackelberg games, where the gateways act as leaders who make decisions regarding the setting of forwarding prices. On the other hand, the users act as followers and make decisions regarding the number of bytes to request, which gateway to purchase from, and what information to share with neighboring users. The gateways play the repeated game and aim to learn the optimal forwarding prices, while the users learn to maximize their profits through collaboration. The means by which the agents make decisions are reinforcement learning methods. Further details about the formulation of RL are mentioned in the next section.

IV. REINFORCEMENT LEARNING FORMULATION

RL is a framework where agents make decisions based on their states and the decisions of other agents. Gateways adjust the prices based on user demand, and users, in turn, adjust their demand based on their utility functions as mentioned earlier. We employ a model-free RL framework, where agents learn the behavior of their neighboring agents and adapt their own behavior accordingly.

The utility of each agent's interactions with the environment corresponds to the rewards. The agents aim to increase their rewards as much as possible. Moreover, every time an agent makes a learning step, their value functions are altered, which modifies their policies. The main contribution of this study is to show that cooperative communication between the users in a multi-agent RL system can introduce gains as the agents learn on their own to negotiate to drop the prices of the gateways collaboratively. In this section, we present the formulation of a single-agent RL system.

A. SINGLE AGENT RL

Single-agent RL studies an agent's interaction with an environment and its learning from its actions and experiences through trial and error. RL depends on rewards as positive or negative feedback to a behavior. The goal is to find a suitable state-action model that maximizes cumulative rewards. The RL problem is well described by a Markov Decision Process (MDP). MDP systems obey the Markov property where the transitions of the process depend only on the current state and actions and not the prior history. The MDP is described by the tuple $< S, A, R, T, \gamma >$, where the elements of the tuple are:

- S is the set of all valid states that can be discrete or continuous.
- A is the set of all valid actions.
- $R : S \times A \times S \mapsto \mathbb{R}$ is the reward function, with $r_t = R(s_t, a_t, s_{t+1})$, and t is the time step unit.
- $T : S \times A \mapsto T(s')$ is the transition probability to state s' if action a taken at state s.
- $\gamma \in [0, 1]$ is the discount rate mapping to the future rewards.

The trial and error process of the agent is supported by exploration, and its purpose is to try states searching for higher rewards. The process of using the knowledge already collected is called exploitation. Between exploration and exploitation the agent tries to find the balance for optimal learning and rewards maximization.

1) ACTION-VALUE FUNCTIONS

The action-value function (Q-function) represents the expected return of the state-action pair (s, a), under policy π.

Actions are extracted from the value functions such that we maximize the expected returns.

$$Q^{\pi}(s,a) = E\Big[\sum_{t\geq 0} \gamma^t R(s_t,a_t,s_{t+1}) \mid a_t \sim \pi(\cdot|s_t), s_0 = s, a_0 = a\Big]. \quad (4)$$

The above equation means that the expected return is determined from all of the future state action routes decided by the policy and discounted by the factor γ. The Q-function is updated when transitioning from state-action (s,a) to new state s', and a reward r is received

$$Q(s,a) \longleftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a')], \quad (5)$$

where, α is the learning rate. Next state s' is sampled from the environment's transition rules $s' \sim P := s' \sim P(\cdot|s,a)$. The next action is sampled from the policy rules $a' \sim \pi := a' \sim \pi(\cdot|s')$. The optimal action-value function $Q^*(s,a)$ gives the expected return if the agent starts at (s,a) and keeps acting according to the optimal policy.

$$Q^*(s,a) = \max_{\pi} E\Big[\sum_{t\geq 0} \gamma^t R(s_t,a_t,s_{t+1}) \mid a_t = a^*, s_0 = s, a_0 = a\Big]. \quad (6)$$

The optimal policy takes action a^* that maximizes the expected return

$$a^*(s) = \arg\max_{a} Q^*(s,a). \quad (7)$$

B. MULTI-AGENT RL

The decision-making problem resulting from several agents operating simultaneously in the same environment is addressed by MARL. The multi-agent case can directly use single-agent RL algorithms. The majority of the theoretical guarantees for the single agent are invalid due to the MARL problem of non-stationarity. MARL algorithms are classified in the literature into three types; fully cooperative, fully competitive, and mixed setups. Similar to single-agent RL, MARL representations can be realized via Markov Games (MGs), which is a generalization of MDPs. A Markov Game is defined by the tuple $< N, S, \{A^i\}_{i\in N}, T, \{R^i\}_{i\in N}, \gamma >$. Where

- $N = \{1, \ldots, N\}$ is the set of agents.
- S represents the state space of all agents combined.
- A^i is the action space of agent i, $A := A^1 \times \ldots \times A^N$ is the action space for all agents.
- $T : S \times A \mapsto \Delta(S)$ is the system transition probability from state $s \in S$ to state $s' \in S$ after taking the joint action $a' \in A$.
- $R^i : S \times A \times S \mapsto \mathbb{R}$ is the reward received by agent i for the transition from (s,a) to s'.
- Finally, $\gamma \in [0,1]$ is the discount factor.

The aim of each agent is to optimize its own rewards by adopting the policy $\pi' : s \mapsto \Delta(A^i)$ such that $a^i \sim \pi^i(\cdot|s)$. Consequently the agent value function $V^i : s \mapsto \mathbb{R}$ depends on the joint policy $\pi : s \mapsto \Delta(A)$ which is defined as $\pi(a|s) := \prod_{i\in N} \pi^i(a^i|s)$. This is formulated as,

$$V^i_{\pi^i,\pi^{-i}}(s) := E\Big[\sum_{t\geq 0} \gamma^t R^i(s_t,a_t,s_{t+1}) | a^i_t \sim \pi^i(\cdot|s_t), s_0 = s\Big], \quad (8)$$

where $-i$ denotes the set of agents in N other than agent i. We can observe that an agent's optimal value function is shaped not only by its own policy but also by the policies of other agents. Therefore, the Nash equilibrium is a joint policy $\pi^* = (\pi^{1,*}, \ldots, \pi^{N,*})$ such that

$$V^i_{\pi^{i,*},\pi^{-i,*}}(s) \geq V^i_{\pi^i,\pi^{-i,*}}(s) \ \textit{for any}\ \pi^i. \quad (9)$$

In the literature of MARL, there are cooperative MARL games where agents share a common reward function [62], [63] or team average reward [64], [65], [66]. Competitive MARL games is modeled as zero-sum Markov Games [67], [68], [69], because the sum of the agents' reward matrices is a zero matrix. The mixed games category related to our work is known as general sum games [70], [71]. Each agent is self-interested, and rewards don't have to be aligned for all agents, and they can conflict between some agents.

V. PROPOSED MARL SYSTEM SETUP

A. DEEP Q-LEARNING AGENTS

Our system of networked agents is formalized as MDP-independent learners with decentralized Deep Q-Networks. The agents engage with one another by choosing the actions that maximize their rewards in the future. In order to accomplish an easy-to-realize action-value function, the actions in our design map to gradual changes in the state space, and the states in our architecture are continuous.

1) USERS

The user is defined with two modes of operation. One with no information exchange between the users, where they act individually, trying to maximize their rewards. The second mode is when there is information exchange between the neighboring users.

For the unconnected-users mode, the *'Requests'* DQN is used to map between the states as prices and the current requests of the user. While the actions are the incremental changes in the requests. The requests are formalized as the total requests b, and the preferences vector $< m^{(i)} | \ i \in \mathbb{G} >$. The preference vector holds the ratio of requested resource blocks (RBs) from every GW. The actions are increments in the request δb, $\delta m^{(1)}$, $\ldots$, $\delta m^{(i)}$.

In the connected-users mode, shown in Fig. 3, two state-action DQNs are used. The first one is the requests DQN mentioned above, adding to it the suggestions from neighboring users as input states. The second DQN is the *'Suggestions'* DQN. The states for the suggestions DQN are the prices and the agent's current requests. The actions are the suggestion values with a limited action space. Each request has a corresponding suggestion; therefore, the suggestion related to the

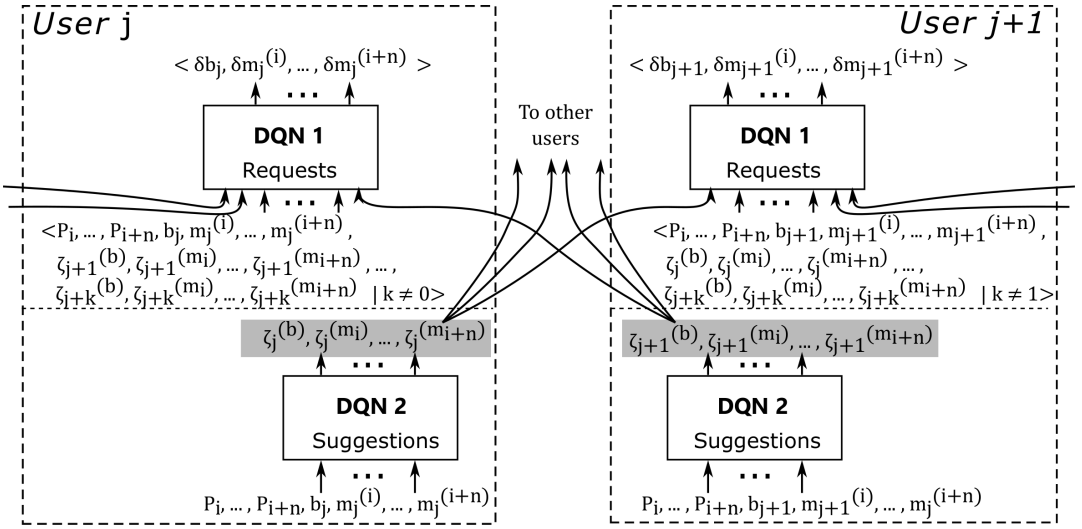


FIGURE 3. Communication Dynamics in Multi-Agent Reinforcement Learning: Sequential decision-making processes of User *j* and User *j+1* across two action stages. First, DQN 2 formulates strategic suggestions that are communicated to adjacent users. Subsequently, these users incorporate the received suggestions into DQN 1 to adjust their request actions.

number of RBs is $\zeta^{(b)}$, and the ones related to the preferences are $< \zeta^{(m_i)} | \ i \in \mathbb{G} >$. The two DQNs will act in different time slots, as shown in the following sections.

The rationale behind adopting two modes of operation is to evaluate the collusion gain later in the Simulations Section. It is presumed that through the connected-users setup, agents can autonomously learn to collude, leveraging their collective bargaining power to influence gateway pricing strategies favorably, all without prior collusion experience. This is in contrast to the conventional Stackelberg game approach, where user agents are isolated and uncoordinated. Our architecture, thus, sets the stage for an innovative exploration of emergent behaviors within multi-agent learning environments. The proposed architecture's parameters are as follows.

DQN 1 (Requests):

- States: $< P_i : i \in \mathbb{G} >$, b, $< m^{(i)} : i \in \mathbb{G} >$, $< \zeta_x^{(b)} : x \in \mathbb{U}^{(-j)} >$, $< \zeta_x^{(m_n)} : x \in \mathbb{U}^{(-j)}, \ n \in \mathbb{G} >$.
- Actions: δb, $< \delta m^{(i)} : \ i \in \mathbb{G}, \ \sum_{i \in \mathbb{G}} m_i = b >$.
- Rewards: U_j (Eq. (3)).

The parameters of **DQN 2 (Suggestions)**:

- State: $< P_i : i \in \mathbb{G} >$, b, $< m^{(i)} : i \in \mathbb{G} >$.
- Actions: $\zeta_y^{(b)} : \ y \in \mathbb{U}^{(-j)}$, $< \zeta_y^{(m_n)} : y \in \mathbb{U}^{(-j)}, \ n \in \mathbb{G} >$.
- Rewards: U_j (Eq. (3)).

Note that we are using the same rewards for both of the networks. For the first mode of operation, DQN 1 runs without the suggestions states. For the second mode of operation, both DQNs are running. Price info is assumed to be gathered from the distributed ledger.

2) GATEWAYS

In our study, we assume the gateways are also represented by RL processes. Gateway i uses a DQN as well to learn the pricing strategies. It maps between the states as load, prices set by the gateway itself and the other GWs, and actions as the increments δP_i. The parameters of **DQN 1 (Pricing)**:

- State: P_i, $< P_x : x \in \mathbb{G} >$, L_i.
- Actions: δP_i.
- Rewards: U_i (Eq. (2)).

In all of the above processes, the action increments can be positive or negative shifts from one of the state parameters. Next, we discuss the training process.

B. DQN TRAINING PROCESS

The training process, according to the Bellman equation, entails updating a neural network from a locally estimated future reward. If the same network generates this reward estimate, it causes feedback and instability in the learning process. A target network is used for more stable training, with a clone of the trained network used for the $Q(s', a')$ estimate to feed the Bellman equation. The main network is trained using environment rewards, and the target network is periodically synchronized with the main Q-network parameter through soft updates.

The training data is pulled from an experienced buffer with the stored transitions (s, a, r, s'). Each row in the buffer represents one transition from s to s' via taking action a and collected rewards r. Typically a batch of experiences $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}')$ is used to train our network. The loss function is calculated from both the Q and the target networks as shown in Fig. 4.

$$L(\theta) = \ [\mathbf{r} + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}' : \theta^{(traget)}) - Q(\mathbf{s}, \mathbf{a} : \theta)]^2, \tag{10}$$

where θ is the Q-Network bias and weight parameters. Then the Q-function is updated by updating the Q-Network

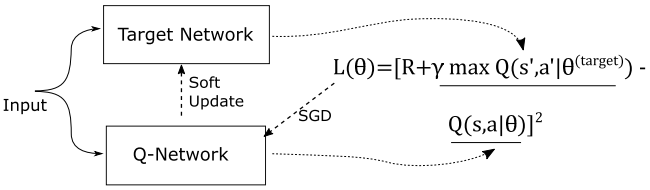


FIGURE 4. Training process.

parameters through stochastic gradient descent (SGD). This is equivalent to minimizing the mean square error represented by the loss function. The θ update equation is

$$\theta^{t+1} = \theta^t + \alpha\Big[[r + \gamma \max_{a'} Q(s', a' : \theta^t) - Q(s, a : \theta^t)]\ \nabla^t_\theta Q(s, a : \theta^t)\Big]. \quad (11)$$

Finally, a soft update of the target network is performed, where the values of the Q-Network are gradually transferred to the target network using a soft update parameter $h \ll 1$.

$$\theta^{(target)} = (1 - h)\,\theta^{(target)} + (h)\,\theta \quad (12)$$

C. LEARNING SCHEDULE

In an RL setup, the learning agent takes action, observes the environment, and collects the rewards. Finally, it stores the transitions to be used for the learning process and the Q-function updates. The transition information includes the actions taken, the previous states, the next states, and the rewards.

In our proposed architecture, we have two levels of decision-making. The first level makes decisions about the ''suggestions'' actions which are fed to the main DQNs to make decisions about the forwarding service buying. Therefore, as shown in Fig. 5, we split the action duration into two parts, one for each DQN level. The main DQN (DQN_1) starts calculating its decisions after DQN_2s take their actions and send their messages to their neighboring agents. Then the collected rewards during the observation duration are all summed in one step at the end of the duration. Finally, the transition is stored in a transition buffer; then, the cycle repeats all over.

There are dependencies between the agents' actions. Hence, we assume that by some means, the collaborating agents have the above-mentioned schedule synchronized and unified among each other. There are lots of protocols in the industry that can make this possible, such as Network Time Protocol (NTP).

The gateways, on the other hand, can have their schedule, especially if the stationarity is not significantly affected by the schedule mismatch. More about stationarity is explained in the next section.

The DQN updates are performed from batches from the transition buffer. The training batch is chosen randomly from the total transitions buffer. The total transitions buffer has a maximum size beyond which it discards all the older

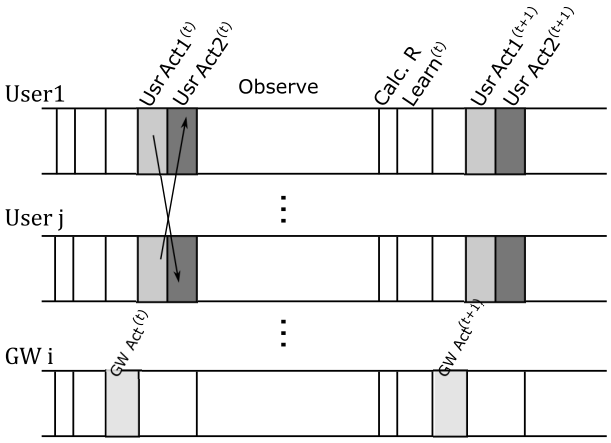


FIGURE 5. Learning schedule: The gateway, acting as the Stackelberg leader, initiates an action, followed by two subsequent action durations undertaken by user agents.

transitions. The DQN update event is not bounded by the above schedule because it only needs the transitions buffer information, which is valid all the time. Therefore, in the implementation, the DQN update process can be performed by a parallel process running at its own pace. Next, we analyze the MARL dynamics and discuss the convergence challenges.

VI. MARL DYNAMICS

A. MEAN FIELD ANALYSIS

The work in [72], modeled the dynamics of Q-values in multi-agent learning using the Mean Field Theory. The mean field approximation allowed updating an agent's Q-values as a response to the mean policy of the population. The analysis was made for bimatrix games which enabled the omission of the transition between states-related terms. In the below analysis, we use the Mean Field Theory, but we will consider the state-action Q-value function of agent i in the population N.

First, for a single agent, the expected change of the state-action Q-value function at a time step t is as follows:

$$\mathbb{E}\big[Q^i_{t+1}(s_k, a_j) - Q^i_t(s_k, a_j)\big] = x^i_t(s_k, a_j).\alpha\Big[\mathbb{E}\big[r^i_t(s_k, a_j)\big] + \gamma\ \mathbb{E}\big[\max_{a' \in A} Q^i_t(s', a')\big] - Q^i_t(s_k, a_j)\Big], \quad (13)$$

where $x^i_t(s_k, a_j)$ is the policy of the agent reflected in the probability of taking action a_j at state s_k. This depends on the exploration method used and the current Q-value function of the agent.

For a multi-agent system, we can drop the i index and work with the joint state value function $\mathbf{Q}(s_k, a_j)$, where the state s_k still describes the agent's state and a_j the agent's action. The expected rate of change of the joint state-action Q-value

function can be rewritten as:

$$\mathbb{E}\Big[\frac{d\mathbf{Q}_t(s_k,a_j)}{dt}\Big] = x_t(s_k,a_j)\Big[\alpha\ \mathbb{E}\big[r_t(s_k,a_j)\big] + \alpha\ \gamma\ \mathbb{E}\big[\max_{a'\in A}\mathbf{Q}_t(s',a')\big] - \alpha\ \mathbf{Q}_t(s_k,a_j)\Big]. \tag{14}$$

The state s' is the future state after taking action a_j and the list of actions A is the actions that can follow the action a_j.

The rewards collected from the population of agents for taking action a_j is the multiplication between the action vector of all agents by the reward matrix $\mathbb{F}_R^{(k)}$. This represents the mapping between agents' actions and their corresponding rewards at state s_k.

$$r_t(s_k,a_j) = e_j^T.\mathbb{F}_R^{(k)}.e_z\ , \tag{15}$$

where e_j is a unit vector in the direction of agent j actions, and e_z is a unit vector in the direction of the joint action of the population.

Each element in e_z has a probability that depends on the exploration strategy of the corresponding agent and the corresponding Q-values. If we assume a greedy ϵ exploration, a set of actions that belong to a specific agent i has a Q function-dependent distribution

$$p(a_z^i) = \begin{cases} (1-\epsilon); & Q(s',a_z^i) = \max\limits_{a_z^i} Q(s',a_z^i) \\ \dfrac{\epsilon}{N_i}; & Q(s',a_z^i) < \max\limits_{a_z^i} Q(s',a_z^i),' \end{cases} \tag{16}$$

where N_i is the number of actions of agent i. Taking the probability vector of all actions of the population of agents $\mathbb{P}_z$, we can write the expected value of the received reward as:

$$\mathbb{E}[r_t(s_k,a_j)] = e_j^T.\mathbb{F}_R^{(k)}.\mathbb{P}_z \tag{17}$$

Now, we consider a Q-function space $\mathbb{R}^L$, with L dimensions. L is the number of actions and states. Each agent occupies a point $\mathbf{Q}_t$ in the space L and time t. As in [72], we assume a hyper-rectangle with differential volume:

$$dV = \prod_{\forall a_j\in\mathbf{A}, s_k\in\mathbf{S}} d\mathbf{Q}_t(s_k,a_j), \tag{18}$$

where $\mathbf{A}$ and $\mathbf{S}$ are the set of the combined states and actions of the population of agents. The hyper-rectangle has 2 L faces, and side lengths of $d\mathbf{Q}_t(s_k,a_j)$. Let the agent probability of being in this hyper-rectangle at time t be $p(\mathbf{Q}_t,t)$. At time t, each point $\mathbf{Q}_t$, changes its value with the rate $d\mathbf{Q}_t/d_t$. Therefore, we can treat the movement of this point in the L-dimensional space as the speed vector:

$$\begin{aligned} v(\mathbf{Q}_t) &\triangleq \mathbb{E}[\frac{dQ(t)^i}{dt}] \\ &= x_t(s_k,a_j)\Big[\alpha e_j^T.\mathbb{F}_R^{(k)}.\mathbb{P}_z \\ &\quad + \alpha\ \gamma\ \mathbb{E}\big[\max_{a'\in A^i}\mathbf{Q}_t(s',a')\big] - \alpha\ \mathbf{Q}_t(s_k,a_j)\Big] \end{aligned} \tag{19}$$

As time progresses, the agents will flow in the Q space, and consequently, a change occurs in the density function $p(\mathbf{Q}_t,t)$. The rate of change is governed by the process of the agents entering and exiting the hyper-rectangle through its faces $dS_{(k,j)}$. The flow density of the agents at time t, state s_k, and action a_j can be represented as:

$$\psi(\mathbf{Q_t}(s_k,a_j),t) = p(\mathbf{Q}_t,t)\ v(\mathbf{Q}_t)\ dt\ dS_{(k,j)}. \tag{20}$$

Then the change in the agent's density in volume dV during dt can be found from:

$$\begin{aligned} &p(\mathbf{Q}_t,\ t+dt)dV - p(\mathbf{Q}_t,\ t)dV \\ &\quad = \sum_{s_k}^{\mathbf{S}}\sum_{a_j}^{\mathbf{A}} \psi[\mathbf{Q}_t(s_k,a_j),\ t] \\ &\qquad - \psi[\mathbf{Q}_t(s_k,a_j) + d\mathbf{Q}_t(s_k,a_j),\ t]. \end{aligned} \tag{21}$$

By substituting Eq. (20) in Eq. (21), the density difference can be rewritten as:

$$\begin{aligned} &\frac{p(\mathbf{Q}_t,\ t+dt) - p(\mathbf{Q}_t,\ t)}{dt} \\ &= \sum_{s_k}^{\mathbf{S}}\sum_{a_j}^{\mathbf{A}} \\ &\quad \times \frac{p(\mathbf{Q}_t,t)\ v(\mathbf{Q}_t)dS_{(k,j)} - p(\mathbf{Q}_t+d\mathbf{Q}_t,t)\ v(\mathbf{Q}_t+d\mathbf{Q}_t)dS_{(k,j)}}{dV}. \end{aligned} \tag{22}$$

Finally, the rate of change of an agent's probability $p(\mathbf{Q}_t,t)$ is

$$\begin{aligned} \frac{\partial p(\mathbf{Q}_t,t)}{\partial t} &= -\sum_{j=1}^{L}\frac{\partial}{\partial Q_t(s,a)}[p(Q_t,t).v(Q_t))] \\ &= \nabla.(p(Q_t,t).\mathbf{v}(Q_t)). \end{aligned} \tag{23}$$

This stochastic differential equation is in the form of a Fokker Plank equation which describes the probability density function time evolution of the state value function. The speed vector $\mathbf{v}$ acts as the drift vector of the Fokker Plank equation.

Although the diffusion term in the Fokker Plank equation is absent, and we only have the drift term, there are still some diffusion effects. This diffusion comes from the noise added to the drift process, which disturbs the population's Q-function evolution toward the highest rewards. Next, we discuss the divergence in light of the flow equation above, Eq. (23).

B. DIVERGENCE CAUSES

1) EXPLORATION ERRORS

By inspecting the speed vector $v(\mathbf{Q})_t$, we can understand that the learning process is updating the Q-function to reflect the collected rewards corresponding to the actions of the agent population plus the future expected rewards from the experienced horizon of the learning agent. However, the process is affected by what can be considered exploration noise.

Exploration, when performed by a single agent, is a good process as it allows collecting rewards for actions that were not recommended by the agent's Q-function. But in a multi-agent system, if the population of agents that we are collecting rewards from are also not following their local Q-function's recommendations, the collected rewards will be interfered with by the suboptimal explorative actions.

Hence, there should be a few agents that are exploring suboptimal action at the same time, or this would result in Q particles diffusion from the maximumizing rewards path.

2) NON-STATIONARY SYSTEM

Moreover, there is the problem of non-stationarity in the learning process. For a specific agent, there are observable states of the system and other non-observable states because of the distributed nature of the multi-agent system. In large populations, the non-observable states are typically more than the observable states.

The reward matrix $\mathbb{F}_R^{(k)}$ maps the actions of an agent to their observable states, which is the horizon that the agent is aware of. As time progresses, the learning process gets shaped by the rewards matrix. But if the non-observable states change the environment and the corresponding rewards matrix for the same observable states, the Q-function will be shaped with a nonstationary rewards matrix. This effect can range from being considered a mild noise added to the learning process to being a totally noisy process causing progressive diffusion of the $\mathbf{Q}$ points from the maximum rewards path, as shown in Fig. 6

Moreover, the size of the action step is related to the system's nonstationarity. As mentioned above, whether it is changing the price or the number of requested RBs, the actions are incremental changes in the agents' state. The size of the increment will affect the speed of state change of the user, which, if passed a certain limit, the other agents will not be able to follow up, and the system will be deemed nonstationary. Next section, we will discuss some notes necessary for our proposed system implementation.

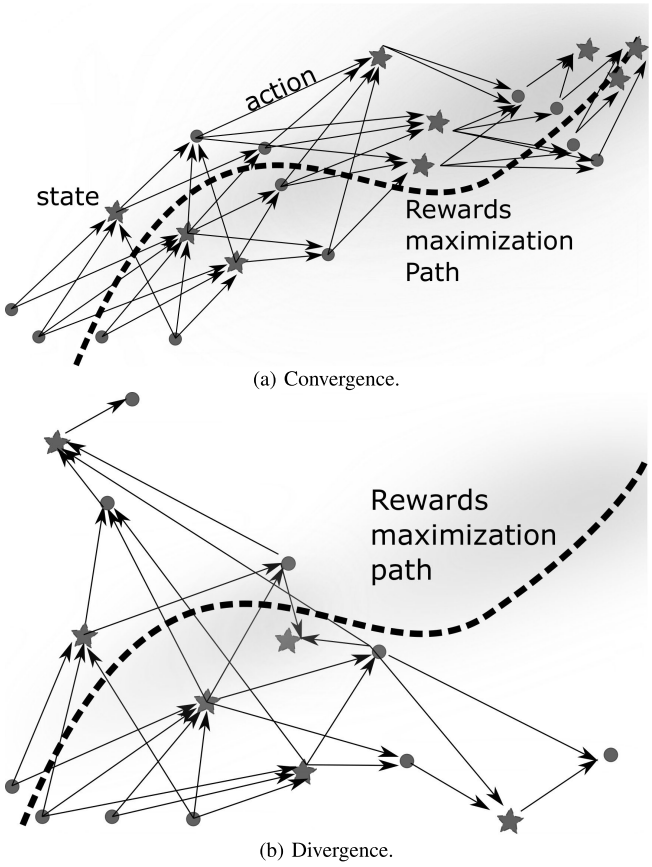


FIGURE 6. Population's Q-function evolution toward maximum rewards.

VII. DISTRIBUTED LEDGER PRACTICAL CONSIDERATIONS

Although the algorithm below will not be simulated along the multi-agent learning system, This section is intended to give insight to the ledger consensus duration and delays. Distributed ledgers revolutionize transaction processing by eliminating the need for a central authority, thereby enhancing democracy and decentralization. At the core of these systems is consensus, a democratic agreement process ensuring all network members concur on the ledger's state even after new transactions or blocks are introduced.

Among the most prominent consensus algorithms are Proof of Work (PoW) [73], Proof of Stake (PoS) [74], Byzantine Fault Tolerance (BFT) [75], and Directed Acyclic Graph (DAG) systems [76]. Each has its merits and demerits, typically balancing between efficiency, security, and decentralization. PoW, the backbone of many cryptocurrencies, is criticized for its energy-intensive nature. PoS and its variant DPoS offer more energy-efficient alternatives, promoting greater decentralization and faster consensus, but still face challenges in network control and security. PoA and BFT prioritize speed and efficiency by relying on a smaller, more trusted group of validators, trading off some degree of decentralization for performance.

DAG-based systems like Tangle in IOTA [77] represent a leap in scalability and transaction throughput, processing multiple transactions concurrently without the need for miners, thereby eliminating transaction fees and achieving faster confirmations. However, it's Hashgraph's [78] unique approach that stands out for our purposes. Hashgraph, while embodying the DAG's graph-like structure, introduces a novel consensus through its "Gossip about Gossip" algorithm. Nodes rapidly disseminate information across the network, ensuring each node has a comprehensive history of all transactions and interactions. This method allows for virtual voting, where nodes predict others' votes without direct communication, leading to quick, efficient consensus. We can summarise the Hashgraph's advantages in comparison with other technologies:

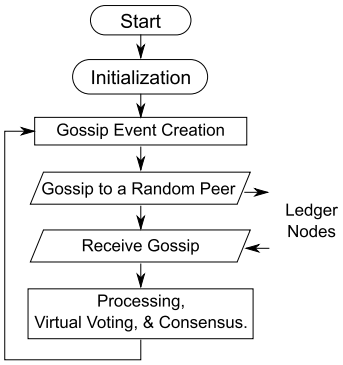


FIGURE 7. Hashgraph's Gossip-about-Gossip/ Consensus flow chart.

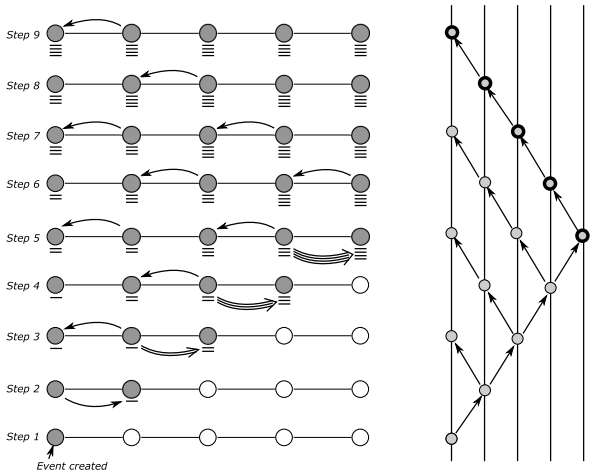


FIGURE 8. Hashgraph's Gossip-about-Gossip time.

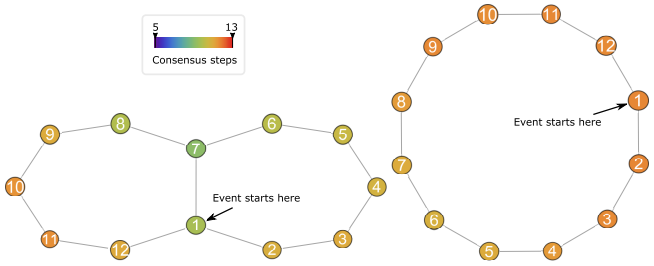


FIGURE 9. 12 Nodes Gossip-about-Gossip/ Consensus Example.

- Against PoW: Hashgraph is markedly more efficient and scalable, reducing both computational and energy demands.
- Against PoS: Hashgraph competes favorably in efficiency and scalability, bypassing the need for stake-based validator selection.
- Against BFT: Hashgraph enhances traditional BFT with its gossip protocol, improving scalability and maintaining robust security.

For our mesh network in a multiagent system reinforcement learning setup, the rapid consensus of Hashgraph is crucial. It ensures that the learning duration is appropriately aligned with the consensus time, maintaining consistency across the network. By adopting Hashgraph, we leverage its speed and efficiency, ensuring that the agents operate with the latest, unanimously agreed-upon data, thereby optimizing the learning and decision-making process. Gossip-about-gossip and virtual voting are presented in [78] and evaluated in terms of speed in Hedera's white paper [79], but evaluated over different network sizes the consensus latency is presented in seconds and not so clear how does that relate to the shape of the network. Therefore, we attempt to give some insight into how virtual voting spreads in a network and how long it takes in the following subsection.

A. GOSSIP-ABOUT-GOSSIP AND VIRTUAL VOTING

The Gossip about Gossip protocol spreads information about an event or about the nodes that have gossiped about that event, thus enabling a node to track all the routes leading to the event's origin. Virtual voting is accomplished by counting the number of nodes that have reshared the event, and if this count passes a certain limit, e.g., two-thirds of the mesh nodes, the event is then validated. Fig. 7 represents a simplified flow chart of the gossiping of a node in a mesh network to its surrounding neighbors. We assume that the gossiping step is composed of several substeps of back-and-forth communication to check for knowledge differences, and lastly, gossip is shared for the incrementation of knowledge.

In Fig. 8, we show an example of how an event spreads through the network. The examples shown in the original Swirlds paper [78] show several events created at several nodes in the same round. Here, we focus on a single event, offering how it spreads and how consensus is reached at different instances and nodes. Then, the spread of other events can be superimposed on the pattern we are showing without interference.

On the right side of the figure, We use the same graph notations the original paper is using to show the gossip and how information spreads over time. On the left side of the figure, we present the connectivity of the mesh network and we use our notation of arrows, shadings and underbars " _ " to demonstrate the spread of information over time. The node shading represents knowledge of the event, the number of bars under the node represents the number of nodes creating the different routes to the event, and the number of lines composing the arrows represents the number of extra nodes gossiped along with the event.

In step 1, the event is created at node A. In step 2, node A gossips to node B about the event, hence node B is shaded and has one under-bar to represent knowledge of the event through one node. In step 3, node B gossips to node C about the event and about the route to node A, and it does not gossip to node A because there is no difference between their knowledge. The number of lines drawing the gossip arrow from B to C represents the number of nodes in the routes leading to the event that node C did not already know about.

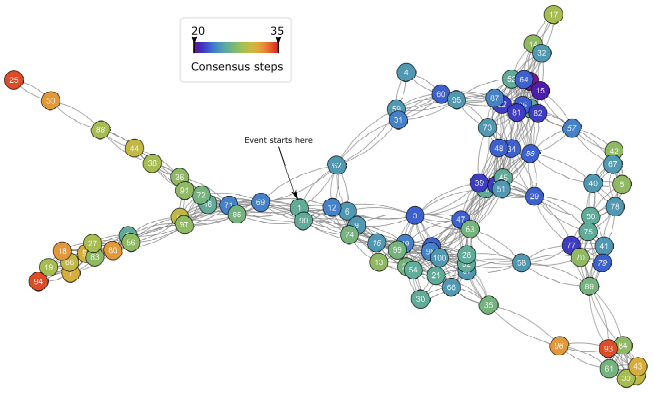


FIGURE 10. 100 Nodes Gossip-about-Gossip/Consensus Example.

The process continues in the same manner as we move from the nodes on the left to the rightmost node. Additional route nodes are added to every gossip until they reach node E, and then the gossip flows back to include the route to node E. Assuming consensus is reached when all nodes are included in the routes list, we can observe that nodes start gradually reaching consensus in the following order: Node E at step 5, Node D at step 6, Node C at step 7, Node B at step 8, and Node A gets the acknowledgment of network consensus at step 9. The consensus event is shown in the graph on the right as a bold circle.

Tracking the nodes and the routes to the event origin ensures virtual voting. We realize that consensus is reached at different steps depending on the position of the node and the event position in the network.

Next, we show that the network shape and its connectivity make a difference in the number of steps it takes to reach consensus. Fig. 9 shows an example of two networks of 12 nodes, with different connectivity. The event is created at node 1, and the colors represent the number of steps taken to reach consensus, assuming a step involves a gossip-about-gossip interaction. We can observe that the connection between node 1 and node 7 shortened the consensus time as it created shorter routes for the gossip to spread. On the other hand, the ring network has the longest routes, therefore taking an average of 12 to 13 steps to reach consensus. We can also observe a different number of steps depending on the nodes' location in the network in relation to the event location.

As we increase the number of nodes, as in Fig. 10, we observe that the consensus durations are longer on average, ranging from 20 to 35 steps, with the nodes clustered close to more neighbors reaching consensus faster, and those at the edges of the network having longer consensus durations.

In relation to the learning duration and information horizon in a multi-agent reinforcement learning system, we want the learning episodes' durations to be much longer than the longest consensus duration of a specific network to allow for unified knowledge at the nodes. This alignment ensures that all agents in the network operate with a synchronized understanding of the environment, crucial for the effectiveness of collective decision-making and learning processes. Longer learning episodes provide a buffer, ensuring that agents' decisions are based on a stable and consistent view of the network state, rather than transient or partial information. Otherwise, we risk the agents making decisions based on outdated or incomplete data, which could lead to suboptimal or even counterproductive actions.

VIII. IMPLEMENTATION RECOMMENDATIONS

This section presents the details of implementing and connecting the learning networks of the proposed scheme. In the first subsection, we present the flow charts of the ledger entity and the users and GWs agents. We emphasize that our outlined processes flowchart is for our simulations reproducibility but is not exclusive in deploying our proposed work. After that we discuss other implementation aspects related to parallel processing, and action increment sizes.

A. SIMULATION FLOWCHARTS

1) THE LEDGER

The ledger, in an actual setup, should be a distributed process managed by a consensus protocol that is responsible for keeping different versions of the same information at different agents. The ledger can run on a subset of the network agents. In our implementation for the simulation environment, we treat the ledger as a separate agent, and we don't emphasize the distributed aspects of that agent because the ledger is not the focus of our study.

The network agents can communicate with the ledger to store their transactions and the information they want to broadcast. The ledger is responsible for keeping the transactions and the forwarding prices of the gateways. So, whenever a gateway changes its price, it sends it to the ledger, and the info is then updated; when the users make a decision based on the current prices, they query the Ledger agent for the price information.

The ledger entity, represented as a flow chart, is shown in Fig. 11. We include it to facilitate reproducibility. This flow chart operates as a continuous loop, responsible for maintaining the global learning state, labeled as "LearningStates." These states alternate between "Observe," "Learning," "GW Act," "USR Act1," and "USR Act2," as detailed in Section [reference section here].

Each state persists for a predetermined duration before the system transitions to the next state. This transition is governed by checking whether the previously assigned target time has been reached. Once reached, the state is updated, and the target time is adjusted by adding the corresponding state duration to the current time.

In a Hashgraph ledger, these time durations can be substituted with a number of ledger rounds. Rounds in a Hashgraph ledger represent the intervals at which the consensus algorithm synchronizes and finalizes the order of events

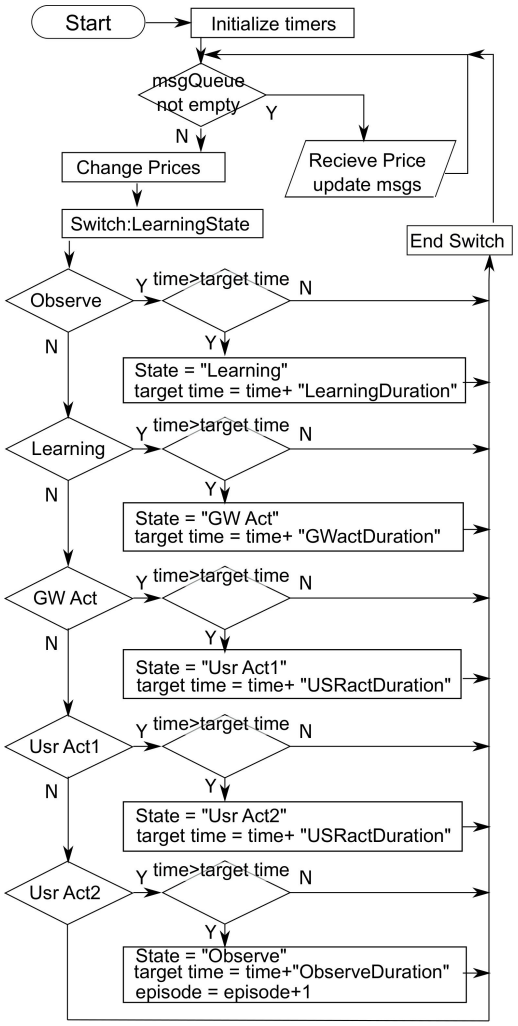


FIGURE 11. **A flow chart of the ledger as an entity in the simulation environment.**

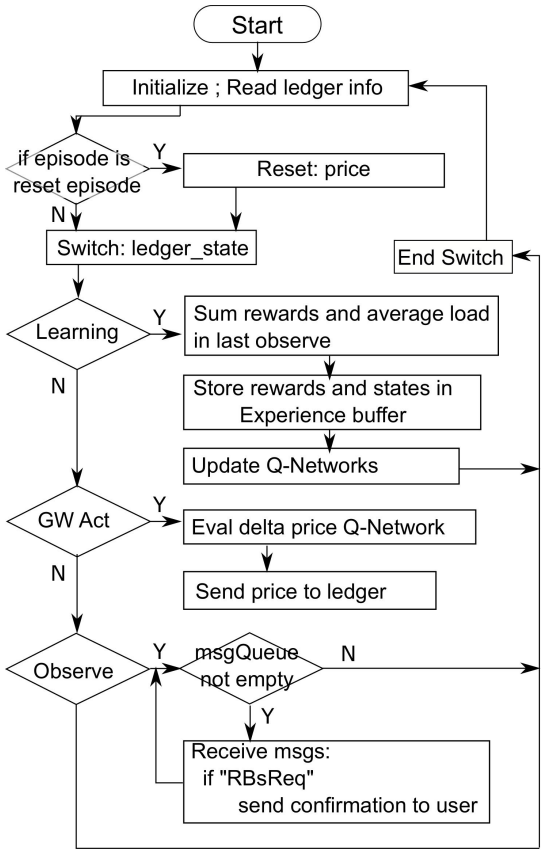


FIGURE 12. **A flow chart of the gateways' process.**

or transactions. Each round signifies a consensus instance, ensuring that all nodes in the network agree on the transaction order and state of the ledger. This synchronization is crucial for maintaining the integrity and consistency of the ledger, especially in environments with rapid transaction updates or state changes.

A parameter ''episode'' is created and incremented with each learning cycle. This parameter assists the agent in tracking the end of the learning epoch and determining when to reset. This tracking mechanism can also be aligned with Hashgraph rounds. During each loop, the ledger receives price update messages from the gateways and subsequently updates its internal parameter values. In our simulation environment, the variables used are prices, learning states, and episodes, while the constants are the durations of each state.

2) THE GATEWAYS

The GWs flow chart is depicted in Fig. 12. The GW cycles through learning states, which are orchestrated by the ledger. The agent resets the forwarding price values at a ''reset episode,'' a number predetermined by consensus among all agents or set as a simulation parameter. The agent's behavior in the learning states is as follows: At the ''Observe'' state, the agent receives user requests specifying the number of resource blocks (RBs) needed, followed by sending back response confirmations to the users. Note that sending and receiving messages can be implemented using HTTP messaging if a REST API is integrated with each agent. A REST API (Representational State Transfer Application Programming Interface) allows for standardized HTTP calls between clients and servers, facilitating seamless communication and data exchange. In our implementation, we utilized message queuing. Message queues provide a reliable way to asynchronously communicate or exchange data between different system components, ensuring that messages are processed in the order they are received.

During the ''Learning'' state, the average load in the last observe duration is calculated as $avg.\ Load = \frac{avg.\ offered\ RBs}{available\ RBs}$, and then the rewards accumulated during the observe duration are summed up. The rewards, states, and actions from the previous ''GW Act'' duration are then concatenated into the experience buffer. Finally, the Q-Networks are updated from the experience buffer using a mini-batch process.

In the ''GW Act'' state, the price increment, denoted as δ price, is determined from the Q-Network by evaluating

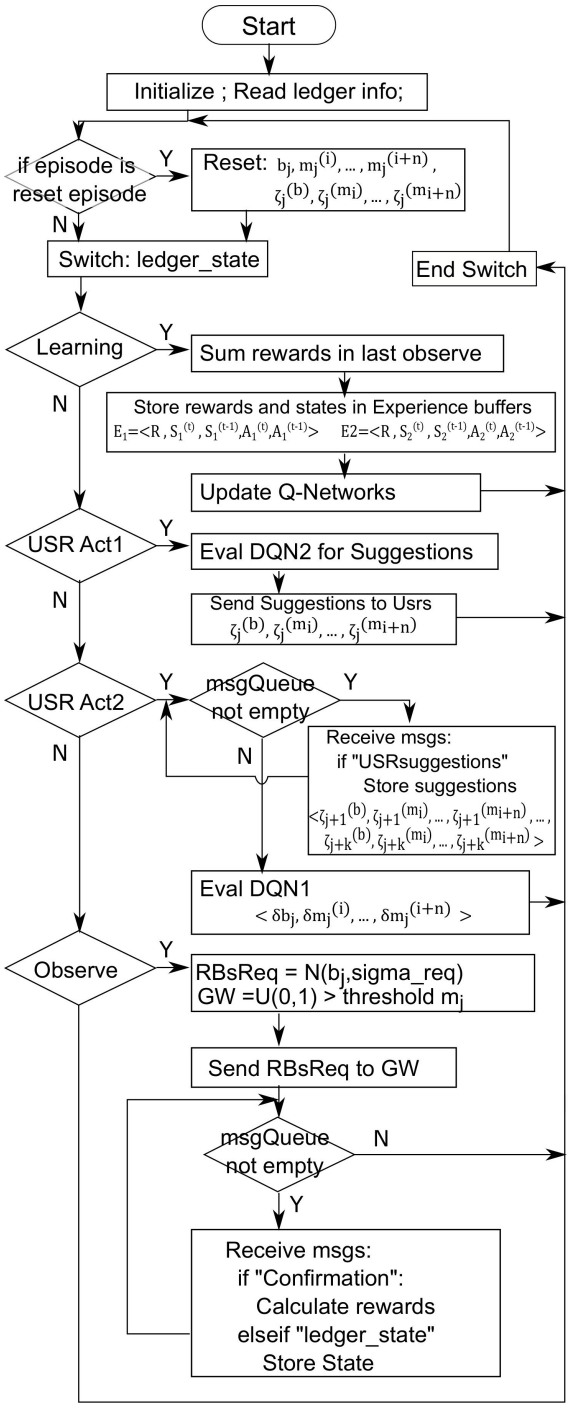


FIGURE 13. A flow chart of the users' process.

the GW prices inputs. This increment is then added to the previous price and communicated to the ledger.

3) THE USERS

The users flow chart is shown in Fig. 13. Same as the GW, the user cycles through the learning states managed by the ledger. The only difference is that the users follow different acting steps, depicted as "USR Act1," and "USR Act2." Also, the agents follow the same reset procedure of the GW agents. The behavior in the learning states is as follows:

At the "Observe" state, the user generates a random RB request with average mu_{req}, and standard deviation $sigma_{req}$, and sends it to a GW with probability that depends on the parameter *"threshold."* The requests are sent to the designated GW, then the agent waits for confirmation, which when received, the agent pauses for the duration of the request and calculates the rewards related to that cycle's request. Also, at this step can receive ledger state. At the "Learning" state, the rewards accumulated during the observe duration are summed up. The rewards, states, and actions from the previous "USR Act1" duration are then concatenated into the experience buffer of the first level Q-Networks, and corresponding states and actions of "USR Act2" are concatenated into the other experience buffer of the second level Q-Networks. Then all the Q-Networks are updated via mini-batch processing.

For the acting durations, at "USR Act1", the agent evaluates the Q-Network of layer 1 to create the suggestions for the neighboring users, then it is sent. At "USR Act1", suggestions are received and then used in the evaluation of layer 2 Q-Networks to get the increments δb_i and δm_i.

B. PARALLEL PROCESSING AND RACE HAZARDS

In our environment, the agents run as separate modules, making use of Python's multiprocessing library [80]. The multiprocessing package allows the spawning of concurrent subprocesses, which codes run in parallel. We used queues between the processes to send the RB request messages and to communicate with the ledger process. Queues are one of the multiprocessing package-supported ways of exchanging information between the subprocesses. Messages are sent through the queues using *put* commands and pulled through *get* commands.

Race hazard is possible in parallel implementation. This may happen if a message between agents arrives several cycles later than its intended cycle, due to queuing buffers or delays, for example. Therefore, conditions and checks should be built within the agents' logic to discard faulty or late messages and skip taking action when such violations happen. We used a global number for each learning cycle as a check; if the agent receives a message with a wrong learning cycle stamp, then this means that we have a wrong message.

Moreover, the learning and tacking action durations of the learning schedule should not be shorter than the capabilities of the hardware running the experiment. Otherwise, we will have a higher probability of race conditions and, consequently, faulty learning.

C. ACTION INCREMENTS AND TRANSITIONS BUFFER SIZES

The agents' action increment sizes determine the speed of state variation. This plays a role in the stationarity of the learning process. In our setup, we design state speeds

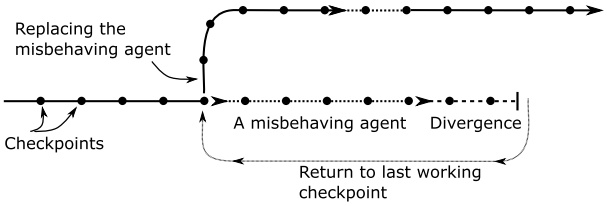


FIGURE 14. Killing rogue agents.

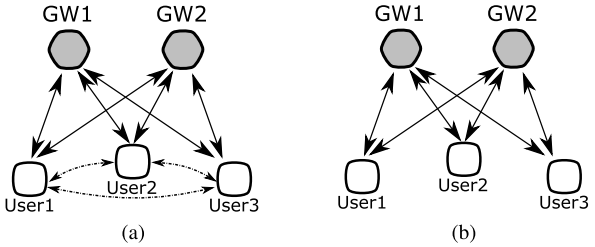


FIGURE 15. Simulated scenarios: a) Connected users setup b) Unconnected users setup.

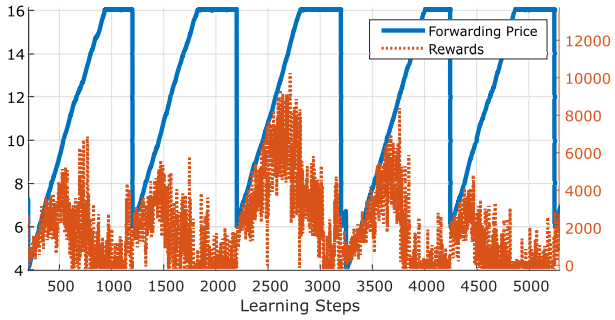


FIGURE 16. GW prices and rewards in early training epochs.

that mimic a nested learning operation. The GWs are the slow-moving process with small action increments, and the users are the fast-moving processes with large action increments. The users are learning from a transition buffer with a size within which the GWs are approximately stationary, hence it should be slowly varying. The population of users who learn the local state of the GWs adjusts their behavior to increase their expected rewards. At the same time, the gateway experiences the collective actions of the population of agents and learns the rewards corresponding to their prices.

D. EVOLUTIONARY PRACTICES (KILLING THE ROGUE AGENTS)

In a multi-agent learning system, if we attempt to gather a group of agents with no prior experience or training, some agents may diverge in the early training phases. Especially when agents are learning and acting in isolation, with no central entity or referee that helps to formulate a shared common knowledge.

Divergence from local behavior can be perceived as an agent in a maze stuck in a wall for several epochs. Or an agent that is gathering negative rewards for a long time until it reflects on its Q-function. An agent can diverge from logical behavior for many epochs. Although in a single-agent scenario, this can be tolerated, because with enough time it may eventually behave logically, in a multi-agent scenario this is dangerous.

In a multi-agent learning scenario, a misbehaving agent affects the rewards the rest of the population collects. As this misbehavior lasts longer, other agents may diverge as well. How we should treat a divergent agent is a question that can be examined in future studies, especially when the other agents are stable and not having problems. There are better solutions than starting the whole simulation from scratch. In our implementation, we choose to borrow the concept of an agent's death from evolutionary game theory [81].

We create checkpoints that save the population DQNs every epoch, as shown in Fig. 14. Then if an agent is misbehaving for a long duration and it started to affect the rest of the population. We kill the divergent agent, replace it with a new unbiased agent, and return back to the checkpoint where the misbehavior started to appear and continue the simulation.

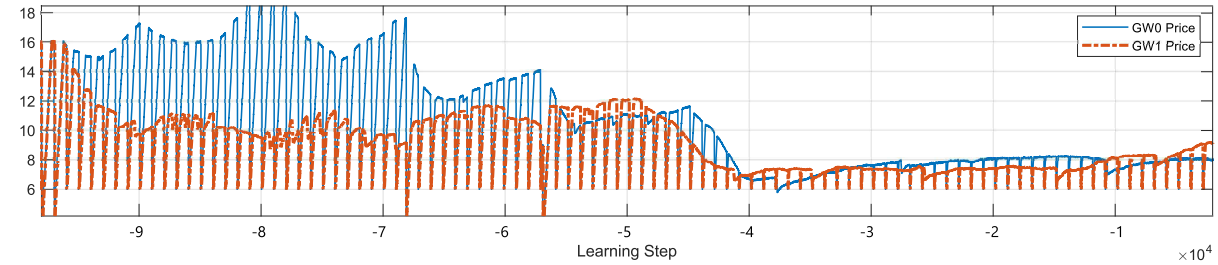
IX. SIMULATION & RESULTS

This section presents a simulation of our proposed architecture involving a single tier of user agents served by two GW agents. The users collaborate by sharing suggestions, while the GWs aim to maximize their rewards by increasing their prices. The learning Stackelberg game parameters are provided in Table 3.

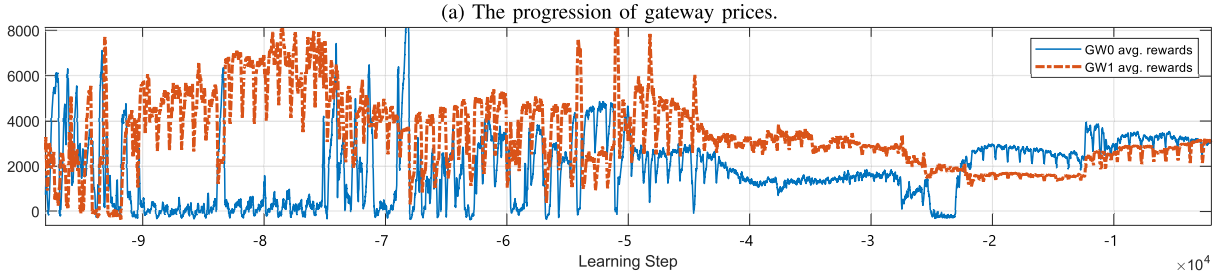
TABLE 3. Simulation parameters.

Parameter	value
Prices range	$P_f \in [0,\ 25]¢$
Requests range	$b \in [0,\ 15]$
m, and ζ ranges	$m,\ \zeta \in [0, 1]$
Forwarding cost	$C_{f_{gw}} = 2¢/RB$
Transmission gain	$G_{user} = 15¢/RB$
Transmission cost	$C_{user} = 1¢/RB$
Load induced penalty	$L_{GW} = 10¢/load(\%)$
User actions	$\delta\zeta \in [+0.3, -0.3]$
	$\delta m \in [+0.3, -0.3]$
	$\delta b \in [+3, -3]$
GW actions	$\delta P \in [+0.06¢, -0.06¢]$
Buffer memory size	80000 steps
Minibatch size	512
Exploration factor ε	0.2
$\gamma_{GW} = \gamma_{user}$	0.9
Connected Users:	
Epoch size	1000 steps
α_{gw}	1×10^{-4}
α_{usr}	1×10^{-4}
Unconnected Users:	
Epoch size	500 steps
α_{gw}	1×10^{-4}
α_{usr}	1×10^{-3}
Initial/reset states:	
Prices	6 ¢
m, ζ	0.5, 0.5
RBs: (b, σ_{req})	(7, 2)

To compare the proposed colluding agents with a population of users who do not share information, two user agent cases are simulated: connected and unconnected users,

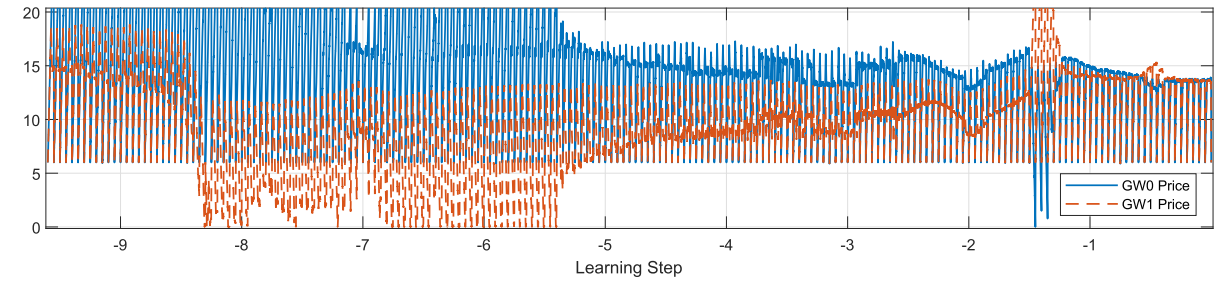


(a) The progression of gateway prices.

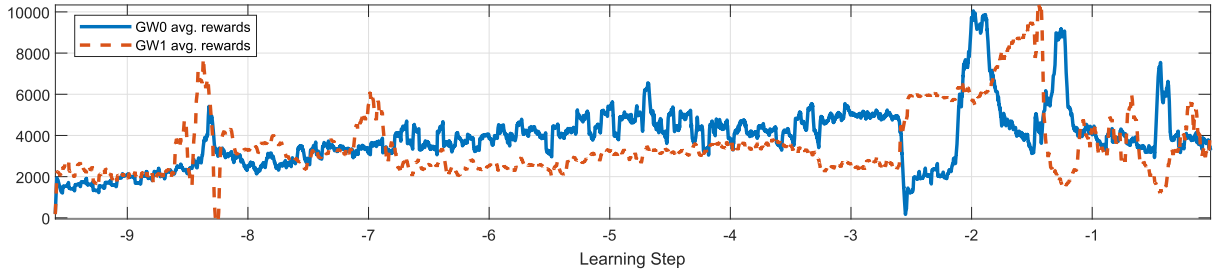


(b) Rewards collected from users.

FIGURE 17. Connected users setup: The evolution of GW0 and GW1 prices, and their corresponding rewards.



(a) The progression of gateway prices.



(b) Rewards collected from users.

FIGURE 18. Unconnected users setup: The evolution of GW0 and GW1 prices, and their corresponding rewards.

as shown in Fig. 15. The GW agents remain the same for both setups. Each Stackelberg game corresponds to a single learning epoch, and the learning buffer can accumulate experience from multiple epochs. The GWs' rewards are based on the users' purchases of the forwarding service (Eq. (1)), while the users' rewards come from the forwarding utility (Eq. (3)). At the end of each epoch, the game parameters are reset to the same price and request values, as specified in Table 3.

For the user agents, two types of actions are considered: the amount of RBs to buy and the preference between the GWs m. To expedite the learning and convergence process, the m value decision is separated from the *RBs* decision. This is achieved by implementing a separate DQN for each action type, running them in parallel for both first- and second-level (suggestion) networks. Consequently, eight separate DQNs are assigned per connected user agent, and four DQNs per unconnected user agent. The neural networks used for the user agents have four layers with 20 ReLU activation functions each. The learning rates are set to low values to allow sufficient time for the Q-functions to be adequately trained and to mitigate the influence of exploration noise and early suboptimal behavior during the initial training phase.

Fig. 16 showcases a captured window during the early stages of training with connected users. At this point, the GW

has developed a Q-function that recognizes raising the price leads to higher rewards, but beyond a specific price, users stop purchasing, preventing the GW from further increasing its price. On the users' side, they experience negative rewards at high prices, resulting in reduced purchases and, subsequently lower rewards for the gateways by the end of each epoch. This user behavior trains the gateway agents and updates their Q-functions, ultimately leading to a new peak across the price range.

In the simulation, a virtual currency denoted as ¢ is employed to represent values for forwarding and transmission costs, as well as load-induced latency penalties. By substituting the simulation parameters from Table 3 into the utility equations (1) and (3), the price ranges where users and GWs receive positive, negative, or zero rewards can be determined. The rewards of the GWs depend on the users' requests and the current price, thus mapping the users' actions to the GWs' rewards. The GWs have a minimum price below which their rewards become negative, while the users' rewards decrease with the price increase, with a maximum price above which users receive negative rewards. The maximum price for users' positive rewards is 14¢, and the lowest price for positive GW rewards is 3¢. The populations of users compete within this price range, aiming to maximize their rewards through repeated gameplay.

We can observe the results of price progression for the connected users' setup in Fig. 17 and the unconnected users' setup in Fig. 18. The coordination among connected users enables efficient load balancing, maximizing their rewards at specific price settings. Moreover, it allows the users to decrease their requests collectively or to be biased to buy from one GW, which compels the other GW to lower its price. This alternating behavior continues through a lengthy bargaining process during GW training until a price is reached where users can no longer push the GWs further due to approaching the lowest price limit. In contrast, such coordination is absent in the unconnected users' setup since a single-user agent cannot negotiate individually when other users are not collaborating.

During the early training phase, some agents may deviate from logical actions for several epochs, which negatively impacts the learning of other agents, especially considering the small number of agents in our setup. In such cases, an agent may get stuck in a state where it consistently receives negative rewards, and a low learning rate can resolve this issue after several epochs. It is preferable to declare the rogue agent as ''dead'' and replace it with a new unbiased agent. Consequently, by the time convergence is achieved, agents have different ages, and the time scale on the figures will have different values for each agent. Hence we choose to show the time axis referred to the end of the simulation, starting with 0 on the right side. Also, for convergence purposes, the price range is limited in the range [0¢, 25¢]

In the case of unconnected users, the simulations converged with user agents having a higher learning rate than the GW agents. They also required more epochs to reach equilibrium. This is due to the lack of connectivity among the users, and the agreement that was easily reached through messaging in the connected users' setup now requires multiple trials. Consequently, we reduce the epoch duration from 1000 to 500 steps to ensure that GW time frames are comparable to those in the connected users' scenario.

In our economic system, connected users resemble a free market where competition between GWs, along with user communication, drives prices to a lower equilibrium point compared to unconnected users. Our results align with the economic study in [82], which indicates that communication among buyers can effectively support collusion. This is also consistent with experiments conducted in [83] and [84] that demonstrate how communication between participants allows for tacit collusion and higher group profits.

Several simplifications were used in our model in order to focus on the collusion effect without having interference from any other asymmetries or heterogeneity in the network. It was assumed that the offered price was the same for all users. The diversity of network ranges, path losses, energy budgets, request rates, and tolerance to latency would add asymmetries in the network, which adds an extra layer of effects in the emergent bargaining. Therefore, these factors were deliberately avoided. Future work can explore the effects of communication assisted with RL on group welfare, including more diverse parameters and inhomogeneous situations, as well as introducing greater complexity as in [85].

Finally, it is important to note that collusion emerged as a behavior due to communication with a limited messaging space, specifically using up and down flags for the requests DQN and the GW preference DQN. We did not define the meaning of the flags or establish rules for interpreting the messages. Instead, users learn the communication language through the learning process. The communication patterns that emerged between users can vary between any two pairs of users, as no specific rules were specified for this interaction.

X. CONCLUSION AND DISCUSSION

This paper explores the impact of communication among user agents in monetarily incentivized forwarding systems and its ability to facilitate tacit collusion behavior. We show that in the absence of communication, users act as separate islands, and an oligopolistic market led by the GWs can emerge. In such a market, the GWs can raise the prices with no pushback from the buyers until they reach a level where users are experiencing negative rewards, then the users stop requesting the forwarding service. The price where the users are about to stop buying the service is where their rewards are slightly positive, and this is the point of equilibrium.

However, in the presence of communication, the user agents possessed means of coordination, which they learned through RL to use to bargain with the GWs. As a result, we observe a significantly lower price equilibrium point compared to the non-communication scenario.

To achieve these dynamics in a MARL system with no prior experience or defined rules, we created a two-level

learning scheme for the user agents. The first DQN level is responsible for the communication between users in the form of shared suggestions with limited messaging space. The second level of DQN focuses on trading decisions, taking into account the suggestions received from the first level. The paper demonstrates that through multiple iterations in a Stackelberg repeated game, the two DQN layers are able to identify the messages and their meanings, enabling coordination. This coordination allows the agents to act collectively, providing them with a bargaining advantage.

The learning process in the distributed system can be affected by nonstationarity resulting from agents' incomplete knowledge. The paper presents a MARL dynamics analysis using mean field theory, providing insights into the causes of divergence in such systems. Additionally, implementation notes are provided to ensure stable learning and eventual convergence.

REFERENCES

[1] B. Jedari, F. Xia, and Z. Ning, "A survey on human-centric communications in non-cooperative wireless relay networks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 914–944, 2nd Quart., 2018.

[2] U. Uyoata, J. Mwangama, and R. Adeogun, "Relaying in the Internet of Things (IoT): A survey," *IEEE Access*, vol. 9, pp. 132675–132704, 2021.

[3] M. W. Akhtar, S. A. Hassan, R. Ghaffar, H. Jung, S. Garg, and M. S. Hossain, "The shift to 6G communications: Vision and requirements," *Hum.-Centric Comput. Inf. Sci.*, vol. 10, no. 1, pp. 1–27, Dec. 2020.

[4] H. Von Stackelberg, *Market Structure and Equilibrium*. Cham, Switzerland: Springer, 2010.

[5] M. Opper and D. Saad, *Advanced Mean Field Methods: Theory and Practice*. Cambridge, MA, USA: MIT Press, 2001.

[6] O. Galinina, K. Mikhaylov, K. Huang, S. Andreev, and Y. Koucheryavy, "Wirelessly powered urban crowd sensing over wearables: Trading energy for data," *IEEE Wireless Commun.*, vol. 25, no. 2, pp. 140–149, Apr. 2018.

[7] J. Khoury, C. T. Abdallah, and J. Crichigno, "Incentivizing cooperation in sensor and control networks," in *Proc. IEEE Int. Symp. Comput.-Aided Control Syst. Design (CACSD)*, Sep. 2011, pp. 19–24.

[8] M. Mousavi et al., "Game-based multi-hop broadcast including power control and MRC in wireless networks," in *Proc. IEEE 26th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Aug. 2015, pp. 1589–1594.

[9] G. Bigwood and T. Henderson, "IRONMAN: Using social networks to add incentives and reputation to opportunistic networks," in *Proc. IEEE 3rd Int. Conf. Privacy, Secur., Risk Trust IEEE 3rd Int. Conf. Social Comput.*, Oct. 2011, pp. 65–72.

[10] L. Wei, Z. Cao, and H. Zhu, "MobiGame: A user-centric reputation based incentive protocol for delay/disruption tolerant networks," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2011, pp. 1–5.

[11] J. Mundinger and J. L. Boudec, "Analysis of a robust reputation system for self-organised networks," *Eur. Trans. Telecommun.*, vol. 16, no. 5, pp. 375–384, Sep. 2005.

[12] L. Buttyán and J.-P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *Mobile Netw. Appl.*, vol. 8, no. 5, pp. 579–592, 2003.

[13] L. Buttyán, L. Dóra, M. Félegyházi, and I. Vajda, "Barter trade improves message delivery in opportunistic networks," *Ad Hoc Netw.*, vol. 8, no. 1, pp. 1–14, Jan. 2010.

[14] Y. Jin, Y. Yi, G. Kesidis, F. Kocak, and J. Shin, "Hybrid client-server and peer-to-peer caching systems with selfish peers," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1744–1752.

[15] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[16] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," in *Proc. 5th ACM Conf. Electron. Commerce*, May 2004, pp. 102–111.

[17] J. A. Pouwelse, J. R. Taal, R. L. Lagendijk, D. H. J. Epema, and H. J. Sips, "Real-time video delivery using peer-to-peer bartering networks and multiple description coding," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2004, pp. 4599–4605.

[18] M. Felegyhazi, J.-P. Hubaux, and L. Buttyan, "Nash equilibria of packet forwarding strategies in wireless ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 5, pp. 463–476, May 2006.

[19] U. Shevade, H. Hee Song, L. Qiu, and Y. Zhang, "Incentive-aware routing in DTNs," in *Proc. IEEE Int. Conf. Netw. Protocols*, Oct. 2008, pp. 238–247.

[20] H. Zhou, J. Chen, J. Fan, Y. Du, and S. K. Das, "ConSub: Incentive-based content subscribing in selfish opportunistic mobile networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 669–679, Sep. 2013.

[21] D. Niyato and E. Hossain, "Competitive spectrum sharing in cognitive radio networks: A dynamic game approach," *IEEE Trans. Wireless Commun.*, vol. 7, no. 7, pp. 2651–2660, Jul. 2008.

[22] M. Jakobsson, J.-P. Hubaux, and L. Buttyán, "A micro-payment scheme encouraging collaboration in multi-hop cellular networks," in *Proc. Int. Conf. Financial Cryptogr.*, Guadeloupe, French West Indies. Cham, Switzerland: Springer, Jan. 2003, pp. 15–33.

[23] N. B. Salem, L. Buttyán, J.-P. Hubaux, and M. Jakobsson, "A charging and rewarding scheme for packet forwarding in multi-hop cellular networks," in *Proc. 4th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 2003, pp. 13–24.

[24] K. Sugiyama, T. Kubo, A. Tagami, and A. Parekh, "Incentive mechanism for DTN-based message delivery services," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2013, pp. 3108–3113.

[25] W. Wu, J. C. S. Lui, and R. T. B. Ma, "A game theoretic analysis on incentive mechanisms for wireless ad hoc VoD systems," in *Proc. 10th Int. Symp. Modeling Optim. Mobile, Ad Hoc Wireless Netw. (WiOpt)*, May 2012, pp. 177–184.

[26] Z. Chen, Y. Liu, B. Zhou, and M. Tao, "Caching incentive design in wireless D2D networks: A Stackelberg game approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[27] X. Zhang, P. Huang, L. Guo, and M. Sha, "Incentivizing relay participation for securing IoT communication," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 1504–1512.

[28] Y. Guo, L. Duan, and R. Zhang, "Optimal pricing and load sharing for energy saving with cooperative communications," *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 951–964, Feb. 2016.

[29] C. Xu, L. Song, Z. Han, Q. Zhao, X. Wang, and B. Jiao, "Interference-aware resource allocation for device-to-device communications as an underlay using sequential second price auction," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 445–449.

[30] C. Xu et al., "Efficiency resource allocation for device-to-device underlay communication systems: A reverse iterative combinatorial auction based approach," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 348–358, Sep. 2013.

[31] S. Huang, C. Yi, and J. Cai, "A sequential posted price mechanism for D2D content sharing communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[32] M. H. Hajiesmaili, L. Deng, M. Chen, and Z. Li, "Incentivizing device-to-device load balancing for cellular networks: An online auction design," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 265–279, Feb. 2017.

[33] L. Li, Y. Qin, and X. Zhong, "A novel routing scheme for resource-constraint opportunistic networks: A cooperative multiplayer bargaining game approach," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6547–6561, Aug. 2016.

[34] Q. Xu, Z. Su, and S. Guo, "A game theoretical incentive scheme for relay selection services in mobile social networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6692–6702, Aug. 2016.

[35] F. Wu, T. Chen, S. Zhong, C. Qiao, and G. Chen, "A game-theoretic approach to stimulate cooperation for probabilistic routing in opportunistic networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1573–1583, Apr. 2013.

[36] T. Ning, Z. Yang, H. Wu, and Z. Han, "Self-interest-driven incentives for ad dissemination in autonomous mobile social networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2310–2318.

[37] B. Wang, Z. Han, and K. J. R. Liu, "Distributed relay selection and power control for multiuser cooperative communication networks using Stackelberg game," *IEEE Trans. Mobile Comput.*, vol. 8, no. 7, pp. 975–990, Jul. 2009.

[38] L. Anderegg and S. Eidenbenz, "Ad hoc-VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents," in *Proc. 9th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2003, pp. 245–259.

[39] J. Huang, Z. Han, M. Chiang, and H. V. Poor, "Auction-based distributed resource allocation for cooperation transmission in wireless networks," in *Proc. IEEE GLOBECOM Global Telecommun. Conf.*, Nov. 2007, pp. 4807–4812.

[40] A. Bousia, "Energy efficient resource allocation scheme via auction-based offloading in next-generation heterogeneous networks," in *Resource Allocation in Next-Generation Broadband Wireless Access Networks*. Hershey, PA, USA: IGI Global, 2017, pp. 167–189.

[41] X. Wang, Y. Ji, H. Zhou, and J. Li, "A nonmonetary QoS-aware auction framework toward secure communications for cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 7, pp. 5611–5623, Jul. 2016.

[42] C. Singhal and S. De, *Resource Allocation in Next-Generation Broadband Wireless Access Networks*. Hershey, PA, USA: IGI Global, 2017.

[43] Z. Zheng, L. Song, D. Niyato, and Z. Han, "Resource allocation in wireless powered relay networks: A bargaining game approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 6310–6323, Jul. 2017.

[44] Q. Cao, H. V. Zhao, and Y. Jing, "Power allocation and pricing in multiuser relay networks using Stackelberg and bargaining games," *IEEE Trans. Veh. Technol.*, vol. 61, no. 7, pp. 3177–3190, Sep. 2012.

[45] M. M. Umar, S. Khan, R. Ahmad, and D. Singh, "Game theoretic reward based adaptive data communication in wireless sensor networks," *IEEE Access*, vol. 6, pp. 28073–28084, 2018.

[46] D. Syrivelis, G. Iosifidis, D. Delimpasis, K. Chounos, T. Korakis, and L. Tassiulas, "Bits and coins: Supporting collaborative consumption of mobile internet," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 2146–2154.

[47] Z. Han and H. V. Poor, "Coalition games with cooperative transmission: A cure for the curse of boundary nodes in selfish packet-forwarding wireless networks," *IEEE Trans. Commun.*, vol. 57, no. 1, pp. 203–213, Jan. 2009.

[48] K. Akkarajitsakul, E. Hossain, and D. Niyato, "Cooperative packet delivery in hybrid wireless mobile networks: A coalitional game approach," *IEEE Trans. Mobile Comput.*, vol. 12, no. 5, pp. 840–854, May 2013.

[49] Y. Zhao and W. Song, "A coalitional graph game for device-to-device data dissemination with power budget constraints," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6.

[50] Y. Cao, T. Jiang, X. Chen, and J. Zhang, "Social-aware video multicast based on device-to-device communications," *IEEE Trans. Mobile Comput.*, vol. 15, no. 6, pp. 1528–1539, Jun. 2016.

[51] F. Wang, Y. Li, Z. Wang, and Z. Yang, "Social-community-aware resource allocation for D2D communications underlaying cellular networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 5, pp. 3628–3640, May 2016.

[52] D. R. Brown and F. Fazel, "A game theoretic study of energy efficient cooperative wireless networks," *J. Commun. Netw.*, vol. 13, no. 3, pp. 266–276, Jun. 2011.

[53] N. Gupta, V. A. Bohara, and V. K. Singh, "Design and measurement results for cooperative device to device communication," in *Resource Allocation Next-Generation Broadband Wireless Access Networks*. Hershey, PA, USA: IGI Global, 2017, pp. 81–97.

[54] A. Asheralieva, "Bayesian reinforcement learning-based coalition formation for distributed resource sharing by device-to-device users in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5016–5032, Aug. 2017.

[55] M. Sadeghi, S. Mollahasani, and M. Erol-Kantarci, "Cost-optimized microgrid coalitions using Bayesian reinforcement learning," *Energies*, vol. 14, no. 22, p. 7481, Nov. 2021.

[56] M. Sadeghi and M. Erol-Kantarci, "Power loss minimization in microgrids using Bayesian reinforcement learning with coalition formation," in *Proc. IEEE 30th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2019, pp. 1–6.

[57] A. Asheralieva and D. Niyato, "Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8753–8769, Oct. 2019.

[58] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Blockchain and machine learning for communications and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1392–1431, 2nd Quart., 2020.

[59] K. Salah, M. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for AI: Review and open research challenges," *IEEE Access*, vol. 7, pp. 10127–10149, 2019.

[60] D. B. Rawat and A. Alshaikhi, "Leveraging distributed blockchain-based scheme for wireless network virtualization with security and QoS constraints," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 332–336.

[61] Y. He, H. Li, X. Cheng, Y. Liu, C. Yang, and L. Sun, "A blockchain based truthful incentive mechanism for distributed P2P applications," *IEEE Access*, vol. 6, pp. 27324–27335, 2018.

[62] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proc. TARK*, vol. 96, 1996, pp. 195–210.

[63] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 535–542.

[64] S. Kar, J. M. F. Moura, and H. Vincent Poor, "*QD*-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations," 2012, *arXiv:1205.0047*.

[65] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5872–5881.

[66] T. Doan, S. Maguluri, and J. Romberg, "Finite-time analysis of distributed TD (0) with linear function approximation on multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1626–1635.

[67] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. 11th Int. Conf. Int. Conf. Mach. Learn.*, 1994, pp. 157–163.

[68] D. H. Jacobson, "Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games," *IEEE Trans. Autom. Control*, vol. AC-18, no. 2, pp. 124–131, Apr. 1973.

[69] S. Weiland, "H_∞-Optimal control and related minimax design problems: A dynamic game approach-tamer bazar and Pierre bernhard," *IEEE Trans. Autom. Control*, vol. 41, pp. 1397–1399, 1996.

[70] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.

[71] M. L. Littman et al., "Friend-or-foe Q-learning in general-sum games," in *Proc. ICML*, vol. 1, 2001, pp. 322–328.

[72] S. Hu, C.-W. Leung, and H.-F. Leung, "Modelling the dynamics of multiagent Q-learning in repeated symmetric games: A mean field theoretic approach," in *Proc. Adv. Neural Inf. Process. Syst.*, Vol. 32, 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/40afd3a37cca05efe623b7509855c73a-Abstract.html

[73] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bit coin Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton, NJ, USA: Princeton Univ. Press, 2016.

[74] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85727–85745, 2019.

[75] K. Driscoll, B. Hall, H. Sivencrona, and P. Zumsteg, "Byzantine fault tolerance, from theory to reality," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.* Cham, Switzerland: Springer, 2003, pp. 235–248.

[76] H. Pervez, M. Muneeb, M. U. Irfan, and I. U. Haq, "A comparative analysis of DAG-based blockchain architectures," in *Proc. 12th Int. Conf. Open Source Syst. Technol. (ICOSST)*, Jul. 2018, pp. 27–34.

[77] S. Popov, "The tangle," *White Paper*, vol. 1, p. 30, Jan. 2018.

[78] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance," Swirlds, Dallas, TX, USA, Tech. Rep. SWIRLDS-TR-2016-01, 2016, pp. 9–11, vol. 34.

[79] L. Baird, M. Harmon, and P. Madsen, "Hedera: A governing council & public hashgraph network," The Trust Layer Internet, San Francisco, CA, USA, White Paper, V.2.1, 2018, pp. 1–97, vol. 1.

[80] *Multiprocessing Process-Based Parallelism*. Accessed: Feb. 13, 2024. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html

[81] W. J. Ewens, *Mathematical Population Genetics: Theoretical Introduction*, vol. 1. Cham, Switzerland: Springer, 2004.

[82] H.-T. Normann, J. Rösch, and L. M. Schultz, "Do buyer groups facilitate collusion?" *J. Econ. Behav. Org.*, vol. 109, pp. 72–84, Jan. 2015.

[83] D. J. Cooper and K.-U. Kühn, "Communication, renegotiation, and the scope for collusion," *Amer. Econ. J., Microeconomics*, vol. 6, no. 2, pp. 247–278, May 2014.

[84] R. M. Isaac and J. M. Walker, "Communication and free-riding behavior: The voluntary contribution mechanism," *Econ. Inquiry*, vol. 26, no. 4, pp. 585–608, Oct. 1988.
[85] M. Ibrahim, U. S. Hashmi, M. Nabeel, A. Imran, and S. Ekin, "Embracing complexity: Agent-based modeling for HetNets design and optimization via concurrent reinforcement learning algorithms," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4042–4062, Dec. 2021.

MOSTAFA IBRAHIM received the B.Sc. degree in electronics and electrical communication engineering from Ain Shams University, Egypt, in 2010, the M.Sc. degree in electrical engineering from Istanbul Medipol University, Turkey, in 2017, and the Ph.D. degree in electrical and computer engineering from Oklahoma State University, Stillwater, OK, USA, in 2023. He is currently a Post-Doctoral Researcher with the Department of Engineering Technology and Industrial Distribution, Texas A&M University. His research interests include distributed management in wireless communication systems, 6G and beyond waveform design, and air-ground channel modeling and measurements.

SABIT EKIN (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Texas A&M University, College Station, TX, USA, in 2012. He has four years of industrial experience as a Senior Modem Systems Engineer at Qualcomm Inc. He is currently an Associate Professor of engineering technology and electrical and computer engineering at Texas A&M University. Prior to this, he was an Associate Professor of electrical and computer engineering at Oklahoma State University. His research interests include the design and analysis of wireless systems, encompassing mmWave, and terahertz communications from both theoretical and practical perspectives, visible light sensing, communications and applications, noncontact health monitoring, and the Internet of Things applications. He received numerous Qualstar awards for his achievements and contributions to cellular modem receiver design from Qualcomm Inc.

ALI IMRAN (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Engineering and Technology Lahore, Pakistan, in 2005, and the M.Sc. degree (Hons.) in mobile and satellite communications and the Ph.D. degree from the University of Surrey, Guildford, U.K., in 2007 and 2011, respectively. He is a Professor of intelligent systems with the James Watt School of Engineering, University of Glasgow. He is currently on leave with The University of Oklahoma, USA, where he is a Williams Presidential Professor in ECE and the Founding Director of the Artificial Intelligence (AI) for Networks (AI4Networks) Research Center. His research interests include AI and its applications in wireless networks and healthcare. His work on these topics has resulted in several patents and over 150 peer-reviewed articles, including some of the highly influential papers in the domain of wireless network automation. On these topics, he has led numerous multinational projects, given invited talks/keynotes and tutorials at international forums, advised major public and private stakeholders, and co-founded multiple start-ups. He is a member of the Advisory Board to the Special Technical Community on Big Data, the IEEE Computer Society. He is also an Associate Fellow of the Higher Education Academy, U.K.