A Reinforcement Learning Based Service Scheduling Algorithm for Internet of Drones

Cong Pu
Dept. of CSEE, Marshall University, Huntington, WV 25755, USA. Email: cong.pu@ieee.org

***Abstract*—Originally invented by the military for warfighting, drones have broken through adamant barriers established by traditional commercial and civilian industry and are quickly becoming an accepted part of mainstream. In order to enable drone technology to reach its full potential and integrate heterogeneous drones into existing workflows, Internet of Drones (IoD) has been proposed as a future aerial-ground communication architecture, where drones frequently contact Zone Service Providers (ZSPs) for up-to-date information. When many drones intend to access data through a ZSP concurrently, service scheduling plays a significant role in improving data accessibility. In practice, however, the limited bandwidth and coverage range of ZSP and the high speed of drones make the problem of service scheduling challenging. In this paper, we propose a reinforcement learning based service scheduling algorithm, also called *RELESS*, to optimally satisfy the service requests of drones in the IoD. In *RELESS*, the interaction between the ZSP and drones is formulated as a Markov decision process (MDP) which will be solved by the Q-learning algorithm to produce an optimal service scheduling policy. During this process, the ZSP adopts an ϵ-greedy exploration method to continuously fine-tune its service scheduling policy with various system states, which is guaranteed to converge to an optimal policy. We develop a discrete-event driven simulation framework using OMNeT++, implement *RELESS* and its counterparts, and conduct simulation experiments for performance evaluation and comparison. Numerical results demonstrate that *RELESS* can improve service request satisfaction ratio, service request satisfaction latency, as well as data size satisfaction ratio, indicating a superior service scheduling approach in the IoD.**

***Index Terms*—Service Scheduling, Reinforcement Learning, Drones, Internet of Drones**

I. Introduction

The global COVID-19 pandemic affects every one of us in some way during the last two years. Even as ordinary business shut down, critical industries and facilities such as utilities, mass transit, telecommunications, and oil and gas production were under great pressure to maintain regular operations continuously. All of a sudden, drone technology became the ultimate tool to boost efficiency and accuracy of everyday operations to combat COVID-19. For example, drones are being used extensively for medicine and grocery deliveries, disinfectant spraying, temperature check, and warning citizens to wear masks [1]. Thus, we argue that the global COVID-19 pandemic became an inflection point for drone industry. In addition, as stated in the ''Drone Technology and Global Markets'' report from BCC Publishing, the global drone market is estimated to be worth approximately $55 billion in 2025, with a compound annual growth rate (CAGR) of around 13% for the period of 2020-2025 [2]. With the support and promotion of other advanced technologies (e.g., Artificial Intelligence (AI) and fifth-generation (5G) mobile communications), we anticipate that the use of drone technology will be even more widespread during the post-pandemic period [3].

To unlock the full potential of drones and integrate them into existing workflows, a novel aerial-ground communication architecture, Internet of Drones (IoD) [4], has been proposed and attracted extensive attention from both the scientific community and industry. A great deal of research work has been produced in the realm of IoD, beginning with the physical layer and expanding to the application layer. In addition, in the era of Industry 4.0, drones are widely used as flying sensors and have secured a place in almost all kinds of industrial ecosystems [5]. More specifically, in the IoD, a set of stationary Zone Service Providers (ZSPs) are deployed to administrate their designated upper airspace and provide wireless communications from Internet infrastructure to drones. For instance, a delivery drone can communicate with a nearby ZSP to obtain an optimal flight trajectory [6]. For predicting the spread of COVID-19 disease, surveillance drones can patrol target area, observe crowds, and deliver observational data to a nearby ZSP [7].

As the number of drones in the air gradually increases, service scheduling plays a significant role in improving data accessibility when many drones intend to access data through a ZSP concurrently. In practice, however, the inherent characteristics of IoD environment pose a great challenge to service scheduling at the ZSP. First, the limited communication range of ZSP and the high mobility of drones make the communication contact time between the ZSP and drones extremely short. As a result, each service request is associated with a tight deadline to be met by the ZSP. Second, the wireless communication bandwidth between the ZSP and drones is constrained. Thus, when the ZSP receives service requests from multiple drones, which service request should be satisfied first becomes a tricky problem. Ideally, both of the aforementioned problems can be easily solved by densely deploying ZSPs on the ground. If that were so, we would have to embrace a relatively high deployment, operational, and maintenance costs. Based on the above points of view, we argue that it would be a wise move to design a service scheduling algorithm to resolve the challenges of IoD system as well as improve scheduling efficiency.

In this paper, we propose the first machine learning based service scheduling algorithm aiming to bring significant system performance enhancement in the IoD environment. In summary, our contribution is summarized in the following:

978-1-6654-2671-8/22/$31.00 ©2022 IEEE

- We propose a **re**inforcement **le**arning based **s**ervice **s**cheduling algorithm (*RELESS*) to optimally satisfy the service requests of drones in the IoD. The basic idea of *RELESS* is that the interaction between the ZSP and drones is formulated as a Markov decision process (MDP) which will be solved by the Q-learning algorithm to produce an optimal service scheduling policy. During this process, the ZSP adopts an ϵ-greedy exploration method to continuously fine-tune its service scheduling policy with various system states, which is guaranteed to converge to an optimal policy [8].
- In order to satisfy the need of future extension and widen the application range, we adopt multiple attribute decision making theory [9] to evaluate each service request with multiple scheduling parameters during the process of forming the state of the system. The rationale behind this design is that new scheduling parameters can be conveniently included in *RELESS* for various system requirements.
- We select service deadline, data size, and data popularity as scheduling parameters and develop a discrete-event driven simulation framework using OMNeT++ [10]. In addition, we implement *RELESS* and its counterparts, e.g., Psched [11], serve-in-random-order (SIRO), and first-come-first-serve (FCFS), in the simulation framework, conduct extensive simulation experiments, and analyze the results for performance evaluation.

The rest of the paper is organized as follows. We review the existing literature in Section II. We present the system model in Section III, and then propose the scheduling algorithm in Section IV. In Section V, we provide and analyze experimental results. Lastly, we conclude the paper in Section VI.

II. Related Work

The authors in [12] investigate the issue of charging scheduling for flying ad hoc networks, where a limited number of charging stations are installed to provide battery recharging service. When drones run out of battery, they can request a charging time slot from a nearby charging station. Then, the charging station schedules all charging requests according to hashgraph consensus algorithm and allocates the charging time slot to each drone using game-theoretic approach. However, the authors failed to adopt coefficient to control the impact of each scheduling parameter for subjective preference. In their design, all scheduling parameters have equal weight in the calculation of recharge scheduling priority. In [13], a drone-assisted data collection strategy is proposed to capture data in wireless sensor networks, where an absorbing Markov chain is formulated to represent the system dynamic consisting of the residual energy of drones, the size of data queue, the time-to-be-alive of ground sensor, and the channel condition. In addition, the drone is equipped with an experience replay memory which stores the training experiences of data collection schedule at each time step. However, the onboard deep deterministic policy gradient algorithm and experience replay memory create a huge burden for resource-constrained drones.

The authors in [14] propose a data service request scheduling scheme for vehicular ad hoc networks, where fuzzy logic is being used to predict the deadline of request according to instantaneous vehicular network information. For each service request, a priority index is calculated based on the information of service type, request deadline, and vehicle status using fuzzy logic system. Then, the road side unit (RSU) satisfies each data service request based on the priority index value. However, the work fails to consider that several vehicles might request the same data during a scheduling window. Thus, in order to conserve the limited wireless communication bandwidth, the data broadcast can be postponed until the deadline so that several data requests can be satisfied by a single broadcast. In [15], a cluster-enabled scheduling algorithm based on reinforcement learning is proposed to improve the information accessibility in vehicular networks. First, a cluster head vehicle is chosen based on the vehicle distance, relative mobility, and bandwidth efficiency as the most suitable data forwarder. Then, a schedule of data transmission is created with the number of data packets and the probability of successful transmission. In addition, a reinforcement learning algorithm is adopted to select auxiliary vehicles to store data in the cluster, which can help to enhance the reliability of data transmission. However, the major weakness is that the cluster head selection procedure incurs non-negligible communication overhead.

The authors in [16] concentrate on the joint problem of content prefetching and transmission scheduling in RSU-enabled vehicular networks, where the storage capacity of RSU and the capacity of wireless channel are constrained. An integer linear programming technique is adopted to solve the joint problem and achieve the goal of improving content delivery. However, the proposed transmission scheduling algorithm only considers the data popularity, i.e., the number of vehicles is interested in the data, which cannot be applied in the IoD environment with the constraint of service deadline. In [11], the authors propose a priority-based scheduling algorithm (Psched) for the IoD environment. In Psched, the ZSP first receives the service requests from drones at the beginning of service cycle. Then, it allocates a weight to each pre-determined scheduling parameter and computes the priority for each service request. Finally, the service requests will be satisfied one by one based on the priority value. The proposed solution in our paper has some similarities with the work in [11] on scheduling service requests in the IoD environment, since both consider multiple parameters to make a scheduling decision. However, our approach in this paper provides ZSPs with artificial intelligence to exploit an optimal service scheduling policy and maximize overall system performance.

III. System Model

The system model is shown in Fig. 1, where a set of drones with an unique identifier are deployed and flying in the sky to execute a task of interest (e.g., express shipping and delivery, law enforcement and surveillance, etc.). ZSPs are connected to the Internet through either wired Ethernet or other networking technology. Thus, when a drone flies into an airspace which

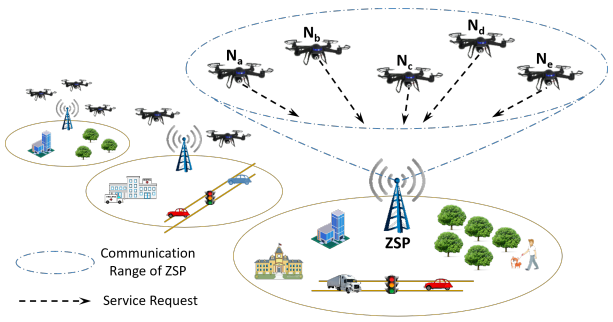

Fig. 1. A system model where a dash-dotted line of ellipse indicates the communication range of ZSP and the drone service request is marked as a dash line with an arrowhead.

is administrated by a ZSP, it can communicate with the ZSP directly and retrieve up-to-date information by sending a service request packet. After receiving the service requests from drones, the ZSP intelligently selects service requests to satisfy according to *RELESS* in such a way that maximizes long-term and large rewards. We assume that ZSPs do not have energy constraint since they are furnished with energy harvesting modules to refill their rechargeable batteries [17], [18]. Due to the limited coverage area (i.e., communication range) of ZSP and the high mobility of drones, each service request is inherently associated with a tight deadline. As a result, the service request needs to get satisfied before the deadline expires (i.e., the drone flies out of the coverage area of ZSP). We also presume that drones are armed with the global positioning system as well as inertial measurement unit to observe their real-time position and speed data [19]. Therefore, each drone can adequately estimate its sojourn time in the communication range of ZSP (i.e., the time when it will fly out of the communication range of ZSP) when it just enters the coverage area. As a matter of course, the estimated sojourn time is regarded as the deadline of drone's service request. In this paper, we do not consider the scenario where drones hover in the air and wait for their service requests to be satisfied.

IV. The Proposed Algorithm

First, the time is divided into a sequence of fixed-lengthed service windows, each is denoted $\mathbb{W}_{svc}$. In addition, each service window $\mathbb{W}_{svc}$ is further divided into a submission sub-window ω_{sub} and a response sub-window ω_{rsp}. As the name suggests, during ω_{sub}, drones can submit their service requests to the ZSP, whereas within ω_{rsp}, the ZSP satisfies service requests through data broadcast. In order to notify incoming drones of the end of ω_{sub}, the ZSP regularly broadcasts a beacon packet. In this paper, $\mathbb{W}_{svc}$, ω_{sub}, and ω_{rsp} are system parameters, which can be adjusted in a fine-grained manner to better suit drones' needs in various scenarios. For instance, the length of $\mathbb{W}_{svc}$ and ω_{sub} can be extended accordingly if a larger drone density is observed within the coverage area of ZSP. Moreover, if a large-sized data is being requested by drones, a larger ω_{rsp} can be allocated proportionately. After entering the coverage area of ZSP, drones continuously monitor the wireless channel for beacon packets to determine whether ω_{sub} is open or not. If drones are still welcome to submit service requests, they can send service request packets to the ZSP. Otherwise, they will need to postpone their service request submissions until next $\mathbb{W}_{svc}$.

TABLE I
Service Request Table

Row#$^{\otimes}$	Data ID$^{\odot}$	Data Size$^{\circledast}$	Data Popularity$^{\circledcirc}$	Rep. Deadline$^{\ominus}$
1	m_{12}	m_{13}	m_{14}	m_{15}
2	m_{22}	m_{23}	m_{24}	m_{25}
3	m_{32}	m_{33}	m_{34}	m_{35}
...	...	...	...	...
i	m_{i2}	m_{i3}	m_{i4}	m_{i5}
...	...	...	...	...
r_{exp}	$m_{r_{exp}2}$	$m_{r_{exp}3}$	$m_{r_{exp}4}$	$m_{r_{exp}5}$

$^{\otimes}$: i represents the ith service request group, $i \in [1, r_{exp}]$; r_{exp} is the number of expected service request groups.
$^{\odot}$: m_{j2} is the identifier of requested data, $j \in [1, r_{exp}]$.
$^{\circledast}$: m_{k3} is the size of requested data, $k \in [1, r_{exp}]$.
$^{\circledcirc}$: m_{p4} is the popularity of requested data, $p \in [1, r_{exp}]$.
$^{\ominus}$: m_{q5} is the representative deadline of service request group, $q \in [1, r_{exp}]$.

Second, the service request packet, denoted by $pkt[N_{id}, D_{id}, T_{soj}, T_{cur}]$, consists of four components: drone's ID N_{id}, the identifier of requested data D_{id}, the sojourn time T_{soj}, and the current timestamp T_{cur}. Note that T_{soj} indicates how long drone N_{id} can stay in the coverage area of ZSP. Thus, with T_{soj} and T_{cur}, the ZSP is able to calculate the deadline of service request packet as $DL_{[N_{id},D_{id}]}$ = T_{cur} + T_{soj}, which is the time when $pkt[N_{id}, D_{id}]$ has to be satisfied. Otherwise, $pkt[N_{id}, D_{id}]$ has to be discarded by the ZSP. During T_{soj}, we assume that each drone only submits one service request packet to the ZSP. After ω_{sub} finishes, the ZSP groups all received service request packets based on the piggybacked data identifier, where the service request packets piggybacked with the same D_{id} are put in the same group denoted as $SG_{[D_{id}]}$. The rationale behind this design is that the service requests that are interested in the same data can be satisfied through a single data broadcast, which can save the communication bandwidth. In order to ensure all requesting drones can receive the data broadcast before the deadline, the earliest deadline in the service request group $SG_{[D_{id}]}$ is chosen as the representative deadline $DL^{\star}_{[D_{id}]}$ for all service requests. In addition, the number of service requests in the service request group $SG_{[D_{id}]}$ is adopted to represent the popularity of requested data $DP_{[D_{id}]}$. Then, the ZSP builds a service request table as shown in Table I, which is composed of service request group number $R\#$, data identifier D_{id}, data size $SZ_{[D_{id}]}$, data popularity $DP_{[D_{id}]}$, and representative deadline $DL^{\star}_{[D_{id}]}$.

It is worth mentioning that the number of drones present within the coverage area of ZSP is changing from time to time since drones can freely move in the airspace. Therefore, the ZSP might receive varying number of service request packets which can be categorized into different number of service request groups. In order to avoid having different sizes of service request table, we assume that during any arbitrary $\mathbb{W}_{svc}$ the number of expected service request groups is r_{exp}. In this way, the number of rows in the service request table is set to r_{exp}. If the actual number of service request groups r_{act} is larger than r_{exp}, the ZSP randomly selects r_{exp}

service request groups to build service request table. And the remaining service request groups, $[r_{act} - r_{exp}]$, are put on the waiting list so that they can be added into service request table later. However, if $r_{act} < r_{exp}$, the values of metrics (i.e., data identifier, data size, data popularity, and representative deadline) in row $[r_{act}+1, r_{exp}]$ are set to *null*.

Third, after ω_{sub} ends, the ZSP performs value normalization over three service scheduling parameters (i.e., data size, data popularity, and representative deadline) in Table I according to the following min-max normalization process,

$$\begin{aligned} m_{ij}^{\diamond} &= \frac{m_{ij} - min\{m_j\}}{max\{m_j\} - min\{m_j\}} \\ i &\in [1, r_{exp}] \quad j \in [3, 5] \end{aligned} \tag{1}$$

and

$$\begin{aligned} m_{ij}^{*} &= m_{ij}^{\diamond} \times \alpha_j + (1 - \alpha_j). \\ i &\in [1, r_{exp}] \quad j \in [3, 5] \end{aligned} \tag{2}$$

Here, i and j indicate the row and column number in Table I, respectively. m_j denotes all values of column j, and $max\{m_j\}$ and $min\{m_j\}$ represent the maximal and minimal value of column j, respectively. $m_{ij}^{\diamond}$ and m_{ij}^{*} are the normalized value and subjectively adjusted value, respectively. Here, multiplicative factor α_j is adopted to adjust the impact of service scheduling parameter in column j of Table I, where $\sum_{j=3}^{5} \alpha_j$ = 1.0. Then, according to information theory [20], the ZSP calculates the entropy of service scheduling parameter in column j, denoted by EPY_j, as

$$EPY_j = -\frac{1}{\ln r_{exp}} \sum_{i=1}^{r_{exp}} \left(\frac{m_{ij}^{*}}{\sum_{k=1}^{r_{exp}} m_{kj}^{*}} \cdot \ln \frac{m_{ij}^{*}}{\sum_{k=1}^{r_{exp}} m_{kj}^{*}} \right), \quad j \in [3, 5], \tag{3}$$

and the entropy weight of service scheduling parameter in column j, denoted by WT_{EPY_j}, as

$$WT_{EPY_j} = \frac{1 - EPY_j}{\sum_{k=3}^{5} (1 - EPY_k)}, \quad j \in [3, 5]. \tag{4}$$

With the entropy weights of all service scheduling parameters, the ZSP calculates the weight of the ith row (i.e., the ith service request group) WT_i according to multiple attribute decision making theory [9],

$$WT_i = 1 - \frac{\sum_{j=3}^{5} (WT_{EPY_j} \cdot m_{ij}^{*})}{\sum_{k=1}^{r_{exp}} \sum_{j=3}^{5} (WT_{EPY_j} \cdot m_{kj}^{*})}, \quad i \in [1, r_{exp}]. \tag{5}$$

Finally, the ZSP formulates the system state ST as a fix-sized vector of the weights of service request groups, where $ST = \{WT_1, WT_2, \cdots, WT_{r_{exp}}\}$.

Fourth, the ZSP formulates the system dynamics into a finite Markov Decision Process (MDP) framework, which is a straightforward framing of the problem of learning from interaction to achieve a goal. More explicitly, the ZSP and drones interact at each of a sequence of service windows, $\mathbb{W}_{svc}^{x}$, x = 1, 2, 3, ... During the xth service window $\mathbb{W}_{svc}^{x}$, the ZSP observes the system state, $ST_x \in \mathcal{S}$, and on that basis selects an action, $A_x \in \mathcal{A}$. $\mathcal{S}$ is a fix-sized vector of the weights of service request groups during any service window, where $ST \in \mathcal{S}$. $\mathcal{A}$ is the action space and is denoted as $\mathcal{A} = \{1, 2, 3, \cdots, r_{exp}\}$. Therefore, during $\mathbb{W}_{svc}^{x}$, if the ith service request group associated with the weight WT_i is chosen to be satisfied, $A_x = i$. Shortly afterwards, the ZSP receives a numerical value related to reward/penalty in part as a consequence of its action. Within any service window, if the ZSP chooses to satisfy the ith service request group SG, the obtained reward is the number of service requests in SG. However, if a drone flies out of the coverage area of ZSP with the unsatisfied service request, the ZSP receives ρ penalty as a return. It is worth mentioning that the event of unsatisfaction is disclosed in a single service window when the drone flies out of the coverage area of ZSP. However, the ZSP realizes that the sequence of its past actions gives rise to this unsatisfaction event.

Fifth, the ZSP works on finding an optimal service scheduling policy π_* which will maximize the sum of the discounted rewards it receives over the future. For finite MDPs, an optimal policy can be defined that there is always at least one policy that is better than or equal to all other policies. In other words, a policy π is better than or equal to a policy $\pi^{'}$ if and only if $v_{\pi}(ST) \geq v_{\pi'}(ST)$ for all $ST \in \mathcal{S}$, where $v_{\pi}(ST)$ is the value function (i.e., the expected return) when starting in a state ST and following a policy π thereafter. It is worth mentioning that there may be more than one optimal policy, but we denote all optimal policies by π_*. Moreover, π_* shares the same state-value function, called the optimal state-value function v_*, which is defined as

$$v_*(ST) \doteq \max_{\pi} v_{\pi}(ST). \tag{6}$$

Suppose that when the ZSP is following a service scheduling policy π, the action during $\mathbb{W}_{svc}^{k}$ is $A_k = \pi(ST_k)$. In addition, the expected immediate rewards from state ST_k after taking action A_k is represented as $R_k = r(ST_k, A_k)$. Therefore, the sum of the discounted rewards the ZSP receives over the future, denoted G, is given by

$$G = R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots + \gamma^{M-1} R_M = \sum_{k=1}^{M} \gamma^{k-1} R_k, \tag{7}$$

where γ is called the discount rate, $0 \leq \gamma \leq 1$. The discount rate determines the present value of future rewards, where a reward received k service windows in the future is worth only γ^{k-1} times what it would be worth if it were received immediately. With $R_k = r(ST_k, A_k)$, we can write G alternatively as

$$G_{\pi} = \sum_{k=1}^{M} \gamma^{k-1} r(ST_k, A_k). \tag{8}$$

Thus, the optimal policy π_* can be obtained through

$$\pi_* = \underset{\pi \in \Theta}{argmax}\ G_{\pi}, \tag{9}$$

where Θ denotes the set of all policies and $\underset{\pi \in \Theta}{argmax}\ G_{\pi}$ denotes a policy π at which G_{π} takes its maximal value.

According to [8], the value function of a state ST_k under a policy π satisfies the following Bellman equation,

$$v_\pi(ST_k) = r(ST_k, A_k) + \gamma \sum_{ST_{k+1}} p(ST_{k+1}|ST_k, A_k) v_\pi(ST_{k+1}), \tag{10}$$

where $p(ST_{k+1}|ST_k, A_k)$ is the probability of transition to state ST_{k+1} from state ST_k taking action A_k when following a policy π. In order to solve Eq. (10), the information of state transition $p(ST_{k+1}|ST_k, A_k)$ is needed. However, the formulated MDP framework is lack of the information of state transition probabilities. Thus, Q-learning [8] becomes a feasible method for the ZSP to find the optimal policy through experiencing the consequences of actions without relying on the information of state transition probabilities.

Q-learning is a model-free reinforcement learning algorithm that seeks to learn the best action to take in a particular state. More specifically, Q-learning estimates its optimal policy without the need for any transition function from the environment. In addition, Q-learning updates its value function based on Bellman equation rather than estimating the value function with a greedy policy. Suppose that $Q_*(SK, A)$ is the optimal Q-value function which provides a maximum return achievable from a given state-action pair by any policy. According to [8], $Q_*(SK, A)$ can be defined as

$$Q_*(ST_k, A_k) = r(ST_k, A_k) + \gamma \sum_{ST_{k+1} \in ST} p(ST_{k+1}|ST_k, A_k) v_\pi(ST_k), \tag{11}$$

In addition, the optimal policy $\pi_*(ST_k)$ can be obtained through

$$\pi_*(ST_k) = \underset{A_k}{argmax}\ Q_*(ST_k, A_k), \tag{12}$$

where $v_*(ST_k) = \underset{A_k}{max}\ Q_*(ST_k, A_k)$. Since Q-learning uses Temporal Differences (TD) to estimate the value of $Q_*(ST, A)$, the Q-value can be estimated via the following,

$$Q(ST_k, A_k) = Q(ST_k, A_k) + \psi \cdot \Big(r(ST_k, A_k) + \gamma \cdot \underset{A_{k+1}}{max} \Big(Q(ST_{k+1}, A_{k+1}) - Q(ST_k, A_k) \Big) \Big). \tag{13}$$

Here, ψ is the learning rate, $0 < \psi \leq 1$. Finally, the optimal Q-value function $Q_*(SK, A)$ as well as the optimal policy $\pi_*(ST)$ can be obtained when the Q-learning algorithm [8] converges.

V. Performance Evaluation

To evaluate the performance of *RELESS*, we develop a simulation framework using OMNeT++ [10]. The area of network simulation is 800 $\times$ 800 m^2, and each simulation run lasts 5000 seconds. A typical wireless communication standard for vehicular environments, IEEE 802.11p, is being adopted to provide communications between drones and the ZSP, where the communication range of drone and ZSP is set to 250 and 200 meters, respectively. In addition, the data rate is set to 5 Mbps. 100-200 drones are initially deployed in the network area, where each drone follows the random waypoint mobility model [21] and moves 25 meter/sec with a pause time of zero. When the drone is within the communication range of ZSP, it only submits one service request packet with a 70% probability. The total number of data items is 50, and the size of data item is varying between 50Kb and 5Mb. The learning rate ψ is set to $\frac{1}{x}$, where x indicates the xth service window. Moreover, the discount rate γ is 0.5.

We revisit Psched [11], serve-in-random-order (SIRO), and first-come-first-serve (FCFS), and implement them to work in our OMNeT++ simulation framework for performance comparison and analysis. The original idea of these three benchmark schemes are briefly discussed in the following:

- Psched: The service request is satisfied based on the calculated priority value.
- SIRO: The service request is answered randomly regardless of scheduling parameters.
- FCFS: The service request piggybacked with the earliest timestamp is served first.

We measure the performance of *RELESS*, Psched, SIRO, and FCFS in terms of service request satisfaction ratio, service request satisfaction latency, as well as data size satisfaction ratio by changing the number of drones.

First, we measure the service request satisfaction ratio of *RELESS*, Psched, SIRO, and FCFS by changing the number of drones in Fig. 2(a). Overall, the service request satisfaction ratio of all abovementioned schemes decrease as the number of drones increases in the network. An increasing number of drones deployed in the network causes the number of drones within the communication range of ZSP increase. Thus, more drones can submit their service request packets to the ZSP. Since the number of service windows is constant, the number of satisfied service requests will decline, which results in a decreasing service request satisfaction ratio. However, *RELESS* still outperforms Psched, SIRO, and FCFS with regard to service request satisfaction ratio. In *RELESS*, a drone flying out of the communication range of ZSP with the unanswered service request is considered as an undesired event which the ZSP is trained to avoid in the future. When the ZSP is well trained, more service request packets can be satisfied before the deadline, thus a higher service request satisfaction ratio is observed. Psched achieves a better service request satisfaction ratio than SIRO and FCFS because it adopts multiple scheduling parameters to satisfy the service request packets. However, Psched is not competitive as *RELESS* because Psched does not have intelligence to avoid undesired events. SIRO randomly selects service request packets to satisfy, therefore, more service request packets have to be discarded as the number of drones increases. FCFS shows the worst performance because it only considers the arrival time of service request packets to make scheduling decisions.

Second, the service request satisfaction latency of *RELESS*, Psched, SIRO, and FCFS is obtained with varying number of drones in Fig. 2(b). At first glance, as the number of drones increases, the service request satisfaction latency of all four schemes increase. When there are more drones in the network, more service request packets will also be gen-

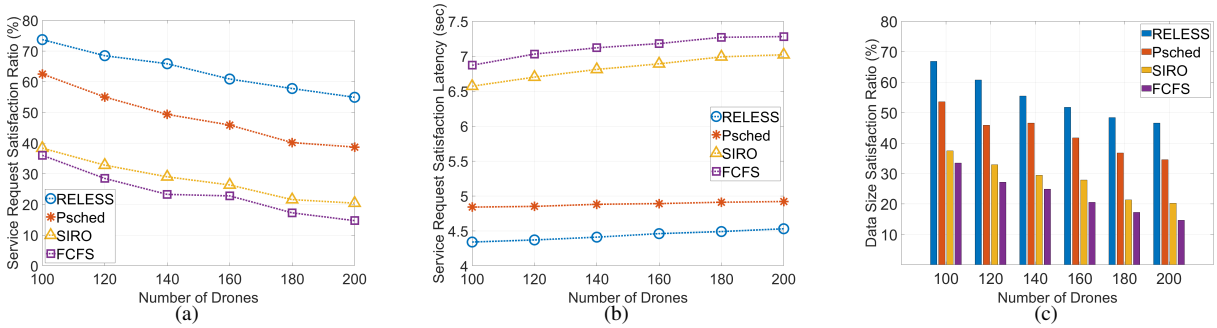

Fig. 2. The performance of service request satisfaction ratio, service request satisfaction latency, and data size satisfaction ratio against the number of drones.

erated. However, due to the constraints of communication bandwidth, more service request packets cannot be answered before their deadlines expire. As a result, those unsatisfied service requests suffer the longest delay, and the overall service request satisfaction latency increases. The most attractive point is that *RELESS* shows much lower service request satisfaction latency than Psched, SIRO, as well as FCFS. This is because *RELESS* considers the deadline of service request as one of its scheduling parameters, and it is trained to satisfy the service requests with a shorter deadline first. Compared to SIRO and FCFS, Psched shows promising performance in terms of service request satisfaction latency. Since Psched considers the deadline of service request packets when making decisions, a lower service request satisfaction latency is maintained. The service request satisfaction latency of SIRO and FCFS are not attractive, because they do not consider time constraint of service request packets during scheduling. As a result, more drones leave the communication range of ZSP with unsatisfied service requests, which causes the service request satisfaction latency increase.

Third, Fig. 2(c) shows *RELESS*'s data size satisfaction ratio compared with other three schemes. Here, the data size satisfaction ratio is calculated as the total size of satisfied data items divided by the total size of requested data items. It is clear that *RELESS* delivers the best performance in terms of data size satisfaction ratio. When the ZSP is trained to satisfy the service request packets from drones, it considers the size of requested data item as one scheduling parameter. Since the service requests that ask for the larger data items will generate more weights, as a result, those service request packets will have more chances to be satisfied and a higher data size satisfaction ratio is observed by *RELESS*.

VI. Conclusion

In this paper, we proposed a reinforcement learning based service scheduling algorithm (*RELESS*) to optimally satisfy the service requests of drones in the IoD. *RELESS* formulates the interaction between the ZSP and drones as a Markov decision process (MDP) which will be solved by the Q-learning algorithm to produce an optimal service scheduling policy. During this process, the ZSP adopts an ϵ-greedy exploration method to continuously fine-tune its service scheduling policy with various system states, which is guaranteed to converge to an optimal policy. In addition, we developed a simulation framework using OMNeT++ and compared *RELESS* with three benchmark schemes for performance evaluation and analysis. The experimental results show that *RELESS* provides better performance than its counterparts.

References

[1] *How Drones Can Be Used to Combat COVID-19*, https://www.unicef.org/supply/coronavirus-disease-covid-19.

[2] *Drone Technology and Global Markets*, 2020, https://www.bccresearch.com/market-research.

[3] C. Pu and P. Zhu, "Mitigating Routing Misbehavior in the Internet of Drones Environment," in *Proc. IEEE VTC2022-Spring*, 2022, pp. 1–6.

[4] P. Boccadoro, D. Striccoli, and L. Grieco, "An extensive survey on the Internet of Drones," *Ad Hoc Networks*, vol. 122, p. 102600, 2021.

[5] *The Use of Drones in Industry 4.0*, https://atos.net/en/blog/industry-4-0-the-use-of-drones-in-industry.

[6] C. Pu, I. Ahmed, E. Allen, and K. Choo, "A Stochastic Packet Forwarding Algorithm in Flying Ad Hoc Networks: Design, Analysis, and Evaluation," *IEEE Access*, vol. 9, pp. 162 614–162 632, 2021.

[7] C. Pu and P. Zhu, "Defending against Flooding Attacks in the Internet of Drones Environment," in *Proc. IEEE GLOBECOM*, 2021, pp. 1–6.

[8] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[9] A. Alinezhad and J. Khalili, *New Methods and Applications in Multiple Attribute Decision Making (MADM)*. Springer, 2019.

[10] A. Varga, *OMNeT++*, 2014, http://www.omnetpp.org/.

[11] C. Pu and L. Carpenter, "Psched: A Priority-Based Service Scheduling Scheme for the Internet of Drones," *IEEE Systems Journal*, vol. 15, no. 3, pp. 4230–4239, 2021.

[12] V. Hassija, V. Saxena, and V. Chamola, "Scheduling drone charging for multi-drone network based on consensus time-stamp and game theory," *Computer Communications*, vol. 149, pp. 51–61, 2020.

[13] K. Li, W. Ni, and F. Dressler, "Continuous Maneuver Control and Data Capture Scheduling of Autonomous Drone in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[14] A. Sharma and L. Awasthi, "AdPS: Adaptive Priority Scheduling for Data Services in Heterogeneous Vehicular Networks," *Computer Communications*, vol. 159, pp. 71–82, 2020.

[15] Y. Xia, L. Wu, Z. Wang, X. Zheng, and J. Jin, "Cluster-Enabled Cooperative Scheduling Based on Reinforcement Learning for High-Mobility Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12 664–12 678, 2020.

[16] S. Berri, J. Zhang, B. Bensaou, and H. Labiod, "Content-Prefetching and Broadcast Scheduling in Vehicular Networks with a Realistic Channel Model," in *Proc. IEEE INFOCOM WKSHPS*, 2019, pp. 1–7.

[17] J. Wang, K. Zhu, and E. Hossain, "Green Internet of Vehicles (IoV) in the 6G Era: Toward Sustainable Vehicular Communications and Networking," *IEEE Transactions on Green Communications and Networking*, pp. 1–1, 2021.

[18] C. Pu, S. Lim, B. Jung, and J. Chae, "EYES: Mitigating forwarding misbehavior in energy harvesting motivated networks," *Elsevier Computer Communications*, vol. 124, pp. 17–30, 2018.

[19] C. Pu, "Stochastic Packet Forwarding Algorithm in Flying Ad Hoc Networks," in *Proc. IEEE MILCOM*, 2019, pp. 490–495.

[20] C. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[21] M. Chowdhury, W. Saad, and I. Güvenç, "Mobility management for cellular-connected UAVs: A learning-based approach," in *Proc. IEEE ICC Workshops*, 2020, pp. 1–6.