FedRDR: Federated Reinforcement Distillation-Based Routing Algorithm in UAV-Assisted Resilient Networks for Communication Infrastructure Failures

Jie Li , Anqi Liu , Guangjie Han * , Shuang Cao , Feng Wang , Xingwei Wang *

Posted Date: 18 October 2023

doi: 10.20944/preprints202310.1118.v1

Keywords: Routing Decision; UAV-MEC; Software Defined Network(SDN); Federated Reinforcement Distillation

Article

FedRDR: Federated Reinforcement Distillation-Based Routing Algorithm in UAV-Assisted Resilient Networks for Communication Infrastructure Failures

Jie Li [1], Anqi Liu [1], Guangjie Han [2,*], Shuang Cao [3], Feng Wang [1] and Xingwei Wang [1,*]

[1] School of Computer Science and Engineering, Northeastern University, Shenyang 110167, China
[2] The Department of Information and Communication Systems, Hohai University, Changzhou, 213022, China
[3] School of Software College, Northeastern University, Shenyang 110167, China
* Correspondence: hanguangjie@gmail.com (G.H.); wangxw@mail.neu.edu.cn (X.W.)

Abstract: Traditional IoT networks have limited coverage and may experience failures due to natural disasters affecting critical IoT devices, making it difficult for them to provide communication services. To address this issue, this study constructs a hierarchical multi-domain data transmission architecture for an emergency network, with unmanned aerial vehicles (UAVs) employed as core communication devices. This architecture expands the functionality of UAVs as key network devices and provides a theoretical basis for their feasibility as intelligent network controllers and switches. Firstly, the UAV controllers perceive the network status and learn the spatio-temporal characteristics of air-to-ground network links. Secondly, a multi-domain routing algorithm based on federated reinforcement distillation (FedRDR) is developed, which enhances the generalization capability of the routing decision model by increasing the training data samples. Simulation experiments are conducted, and the results show that the average communication data size between each domain controller and the server is approximately 45.3KB when using the FedRDR algorithm. Compared to the transmission of parameters through federated reinforcement learning algorithms, FedRDR reduces the transmitted parameter size by approximately 29%. Therefore, the FedRDR routing algorithm facilitates knowledge transfer, accelerates the training process of intelligent agents within domains, and reduces the communication cost of UAV networks.

Keywords: routing algorithm; UAV-assisted networks; reinforcement learning; federated learning

1. Introduction

In emergency situations and natural disaster conditions, traditional IoT communication infrastructure may be damaged, including facilities such as sensors. This can lead to the interruption of communication services. An urgent issue that needs to be addressed is the ability to quickly transition from a faulty state to a normal state and ensure that ongoing communication services can be continued smoothly without being affected by network failures. The flexible communication characteristics of drones enable them to perform auxiliary communications in the IoT. This enables the entire IoT system to resume its normal communication status.

Unmanned aerial vehicles (UAVs) have the ability to create collaborative air-to-ground networks above disaster areas, regions with infrastructure challenges, or densely populated communication hotspots. They support and enhance existing cellular infrastructure to connect previously unconnected networks.[1,2] Due to their high mobility, strong line-of-sight signal transmission, rapid deployment, and strong adaptability to various environments, UAVs play a critical role in 6G networks. For instance, after Hurricane Ida hit Louisiana, AT&T used UAV-mounted cellular on wings (COW) to restore LTE coverage for cellular users.[3] Moreover, He et al. utilized multiple UAVs to provide services to a wide range of users.[4] To extend the overall duration of a communication network, load balancing is implemented between adjacent UAVs, and resource consumption is shared, significantly increasing the overall coverage time of the UAV network. In addition, UAV cluster networks are a crucial component

 doi:10.20944/preprints202310.1118.v1

of air-to-ground collaborative networks and provide an essential foundation for achieving aerial networking and data transmission.[5,6]

The privacy protection of user data should take priority in the construction of an air-to-ground collaborative network established by UAV clusters. Federated learning (FL) offers an effective means of training machine learning models while ensuring user privacy. In the FL system, clients possess diverse computing and communication resources.[7] In recent years, researchers have worked to improve the communication efficiency of federated learning. One commonly employed approach is gradient compression, which directly reduces the size of model updates. Another widely adopted method is collaborative learning, where clients share local model predictions instead of transmitting model updates, thereby reducing communication costs. Knowledge distillation (KD) has also been introduced in FL to enable efficient and low-cost information exchange, especially when dealing with heterogeneous models and aiming to reduce communication expenses.[8]

The networking of distributed and heterogeneous nodes can be challenging to manage, and—due to dynamic changes in network conditions—links are prone to interruption, making their transmission reliability difficult to guarantee. This paper focuses on the application of air-to-ground collaborative networking and utilizes mobile edge computing technology to study network transmission strategies and routing mechanisms in networks with high dynamic changes. To address the issues of low intra-domain data volumes and poor local decision model generalization ability, a federated strategy is proposed to aggregate multi-domain data and accelerate intra-domain training while simultaneously protecting intra-domain privacy, as well as avoiding the leakage of intra-domain network information.

The main contributions of this paper are as follows:

(1) First, we design an air–ground collaborative network architecture for UAV-assisted networks. This architecture utilizes SDN's hierarchical multi-domain networking technology, with a MEC service deployed as a super domain controller responsible for obtaining global network state information and cross-domain business requests. The domain controller is responsible for collecting network state information within its domain and aggregating both local and cross-domain business requests, ultimately enhancing the network's information processing capabilities and reducing communication latency.

(2) In order to achieve real-time awareness of network status, we introduce unmanned aerial vehicles (UAVs) for communication transmission, enabling the entire IoT system to handle tasks more flexibly. We adopt a federated reinforcement learning approach, where the server sends the obtained model parameters to each domain, and the intelligent agent in the local domain controller is further trained and updated based on these parameters. This approach allows for the use of independent model parameters within each domain while protecting the privacy of network states within the domain.

(3) Considering the challenges of energy consumption and limited computational resources in UAVs, as well as the need to reduce the burden of handling large parameters, we introduce knowledge distillation and develop a routing algorithm based on federated reinforcement distillation (FedRDR). By combining federated reinforcement learning with knowledge distillation, we can address the issue of transmission overload caused by the presence of a large amount of parameter data in intelligent agent models. This enables us to achieve real-time awareness of the network status and adjust the data forwarding rules accordingly.

(4) To validate our method, we construct a simulation environment for a drone network using a Mininet network simulator and Ryu controller, and conduct algorithm analysis using Python. Through our experiments, we demonstrate that FedRDR outperforms other routing algorithms based on joint learning. Specifically, compared to the federated forced learning routing algorithm, FedRDR can reduce parameter transmission by approximately 29% and significantly accelerate the convergence speed of the model.

 doi:10.20944/preprints202310.1118.v1

2. Related Work

In recent years, natural disasters have had a significant impact on infrastructure, including communication base stations. Essential resources have become scarce and disruptions in communication links have occurred as a result, making the delivery of critical services to individuals challenging.[9] The process of reconstructing a fully operational communication backbone network after a disaster, which is crucial for restarting telecommunications and internet services, can take anywhere from several days to weeks.[10]Hence, it is therefore imperative that we are able to harness the power of emerging artificial intelligence technologies to restore such emergency communication networks.

UAVs are highly effective in search and rescue missions due to their rapid deployment capabilities. By deploying drones in the air, it is possible to establish a local area network (LAN) and backbone network, even in locations without existing network infrastructure. This approach significantly reduces the data transmission time compared to relying solely on satellite links.[11] Once a mission is completed, the retrieval of the drones minimizes the consumption of resources required for network construction. In the context of drone control strategies, G. B. Tarekegn et al. proposed a dynamic and mobile control strategy utilizing deep reinforcement learning (DRL)[12]. The objective is to maximize communication coverage and network connectivity for multiple real-time users within a specified timeframe. Additionally, Zhang et al. studied a communication system assisted by multiple drones equipped with base stations (BSs) to minimize the number of drones required and improve coverage by optimizing the three-dimensional positions of drones[13], user clusters, and frequency band allocation. The above research relies on the perspective of ground users and fully leverages their geographical positions. However, GPS location errors resulting from disasters can pose challenges for air–ground drone communication.

Deploying multiple drones can significantly enhance network coverage within a specific area. Wang et al. utilized a particle swarm optimization algorithm to optimize the deployment positions of multiple drones, with the aim of maximizing network coverage.[14] Shi et al. employed a distributed deep Q network (DQN) approach, in which each drone possesses its own DQN and shares action decisions. This algorithm focuses on maximizing throughput while considering fair service constraints.[15] In a similar vein, Dai et al. proposed a drone network resource allocation algorithm based on multi-agent cooperative environment learning.[16] This method adopts a distributed architecture, where each drone is treated as an independent agent. By dynamically making decisions related to deployment positions, transmission power, and occupied sub-channels, this algorithm enhances the utility of the drone network. However, in the aforementioned research, drones are required to maintain continuous communication to process network information, which can potentially result in reduced work efficiency.

In disaster areas, there is a significant demand for wireless communication, and the privacy of various types of sensitive data must be protected. Federated learning (FL) is a privacy-preserving approach that transmits only the model instead of the raw data during the training process. However, if model updates contain a large number of parameters, they can become excessively large, resulting in high communication costs that burden clients. To address this issue, Wu et al. proposed a federated learning method called FedKD,[17] which utilizes adaptive knowledge distillation and dynamic gradient compression techniques. This approach affords enhanced communication efficiency and effectiveness. Wang et al. introduced ProgFed.[18] Furthermore, Yang et al. proposed another privacy-preserving method called partial variable training (PVT), [19]which trains only a small subset of variables on edge devices to reduce memory usage and communication costs. However, all of these cost-reduction methods have been proposed for their respective scenarios and do not consider improving the overall workload problem from the perspective of the transmitted parameter data volume.

Therefore, we propose a UAV-assisted software-defined networking framework to enhance the network's information processing capability and reduce communication latency between the air

 doi:10.20944/preprints202310.1118.v1

and ground. Additionally, we design a state-aware routing algorithm based on deep reinforcement learning to achieve the real-time perception of the network status and adjust the data forwarding rules accordingly. Finally, we develop FedRDR to address the issue of transmission link overload caused by the large volume of intelligent agent model parameter data, while also ensuring the privacy of the network status within the domain.

3. System Model

Based on the unique characteristics of UAV edge computing networks, this paper presents a hierarchical multi-domain network architecture based on SDN. This architecture aims to establish network connections for mobile nodes, thereby creating a mobile edge computing network system with UAVs acting as edge controllers. The network architecture consists of two main components: the control plane and the data plane. The control plane comprises a global server and high-performance UAVs acting as domain controllers. We constructed a set $N = \{1, 2, 3, ..., n\}$ to represent the domain controllers. The data plane is composed of ordinary UAVs. As a switch in the data plane, ordinary UAVs provide mobile node access and data forwarding services. We also created a set $M = \{1, 2, 3, ..., m\}$ to represent the switches. The global controller maintains the domain controller information for all high-performance UAVs. We assume that all high-performance UAVs can communicate with the global controller deployed on the ground or satellite. In order to protect the privacy of information about different domain network states (such as their topology state, link state, node state, or network traffic state), the overall decision-making model deploys the federal learning architecture on the domain controller and global controller, so as to improve the data privacy and security of the network and reduce network transmission overheads.

The domain controller plays a crucial role in collecting status information within its coverage domain and ensuring information consistency throughout the network. This helps to reduce the delay of business control within the domain and reduces reliance on the server for the control architecture. When the domain controller receives an intra-domain traffic flow request, it processes the request, identifies the forwarding path for the traffic flow, and then sends control messages to the switches within the domain to modify their status and complete the forwarding of the request. The server interacts with the domain controller to establish global data and achieves the establishment of a logically centralized control plane with global knowledge using a physically distributed hierarchical approach.

Multi-domain routing is facilitated by the domain controller and the global server through the federation model. The domain controller is responsible for maintaining the routing information within its domain and updating it within the global server as a parameter for federated learning. The global server collects parameters from all domain controllers and maintains a global federated learning model. As the model parameters are transmitted during the learning process, the specific routing path information within the domain is protected, ensuring privacy preservation. The multi-domain intelligent routing system model, based on the SDN hierarchical multi-domain data transmission architecture, is illustrated in Figure 1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

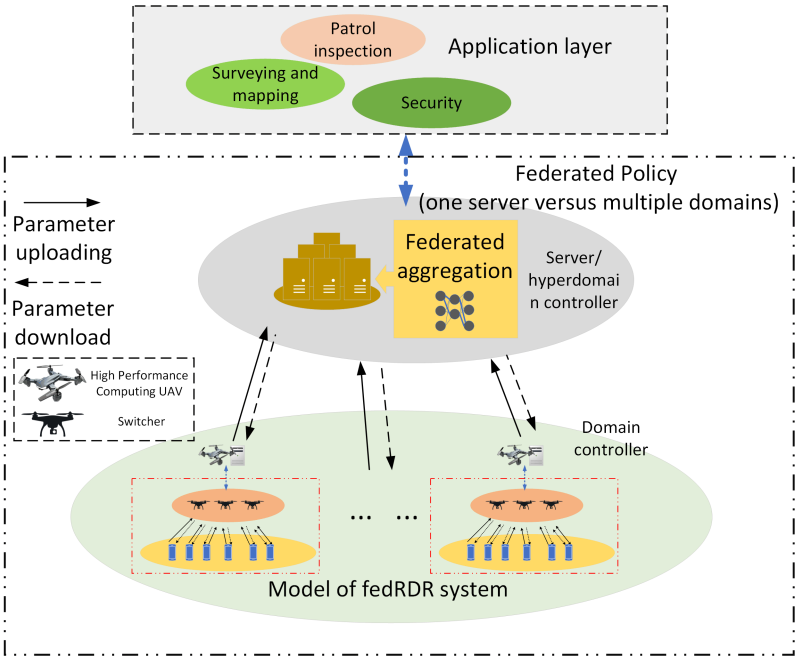

Figure 1. The multi-domain intelligent routing system model.

4. Algorithm Model

This paper proposes a routing algorithm based on federated enhanced distillation. In real network environments, each domain generates relatively small amounts of data. Agents deployed in a single-domain controller require extensive training to obtain a stable model, which significantly increases the model training time and computational energy consumption of the UAVs acting as controllers. Moreover, since the agent can only use data within its own domain for training, the trained model exhibits personalized characteristics and is only suitable for local routing decisions. To address these challenges, we design a federated reinforcement learning routing algorithm. This algorithm constructs multiple model training domains and aggregates data from multiple domains. Each domain controller can participate in the training of the complete model, thereby expanding the data samples and avoiding the leakage of network state information within each domain. This approach enhances the universality and generalization ability of the model.

To ensure that the model can better adapt to the task requirements and possess a greater generalization ability, larger-scale models are often utilized. However, the exchange of model parameters incurs significant costs. Communication between the server and the domain controller can lead to excessive link load when uploading large amounts of data. This can result in data loss within the domain, which, in turn, impacts the training of the decision-making model.

Therefore, in order to perform tasks within each domain, complex models need to be deployed, resulting in resource wastage. In the proposed federated reinforcement distillation approach, it is not the model parameters that are transmitted, but rather the proxy experience memory. Through the federated reinforcement distillation algorithm, knowledge is transferred from one domain controller to a heterogeneous model in another domain controller. In complex task domains, agents with larger-scale neural networks are deployed to achieve a better task performance, while smaller-scale models are deployed in simpler task domains. This enables models in each domain to complete tasks more efficiently within their local domain, thus reducing the computational energy consumption of the domain controller.

In this study, the routing path is directly determined by the deep reinforcement learning algorithm, with the aim of achieving the fastest forwarding of traffic from the source to the destination node. Furthermore, quality of service (QoS) is considered by taking into account link delay. After taking an action, the agent modifies the current network environment. Network state parameters are collected

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

through the network state monitoring module and subsequently updated to reflect changes in the network state.

Therefore, we design a federated reinforcement distillation [20] routing—a distributed machine learning architecture that preserves privacy and has high communication efficiency. A proxy of the experiential memory in the domain is constructed and the data are exchanged between the domain controller and the server. The proposed federated enhanced distillation routing algorithm does not cause the leakage of sensitive intra-domain data when combining multi-domain agent experience memory, and the proposed algorithm can reduce the amount of data that needs to be transmitted, thereby reducing communication overheads.

When there are data to be transmitted in the switch queue, the domain controller obtains the network state through the network state monitoring module and calculates the routing strategy, so as to guide the specific forwarding process of the data. For each input state of the algorithm, a strategy is selected to obtain an output result, and an action is selected according to the output result. The input state (state), output action (action), and reward value (reward) of the algorithm are, respectively, defined as:

(1) The input state: At present, the training process of the reinforcement learning routing algorithm in most studies is based on the network traffic matrix. However, in an actual network, it is difficult to obtain a real-time and effective traffic matrix. In this paper, the switch traffic is used as the state input of the agent, so that the actions made are more time-effective. The input state is specifically represented as a matrix. Every time the agent takes an action, the domain controller obtains the current state parameters, including the short-term traffic of the switch. We assume that there are n nodes in the network, in which d_i represents the i-th node. The input status S_t of time t is represented by Equation 1:

$$S_t = \begin{bmatrix} d_1 & : & f_1^t \\ d_2 & : & f_2^t \\ \vdots & \vdots & \vdots \\ d_n & : & f_n^t \end{bmatrix} \tag{1}$$

where f_i^t represents the amount of traffic flowing through d_i from time $t-1$ to time t.

(2) The output action: After the agent receives the input state, the output value is calculated according to the neural network, and the output value is mapped to the corresponding action. According to the authors of [21], the action output is the path of the data flow, but a routing algorithm based on Q-learning is used. When the traffic in the network increases, the state and action space increase sharply, resulting in a poor learning effect on the agent. This study considers all nodes from the source to the destination node as the candidate action set. The space size of this method is fixed, which facilitates the training process of the agent. The action candidate set of the current state contains all the next hop nodes connected to the current traffic node, which are expressed according to Equation 2:

$$A = a_i -> [a_{ij} ... a_{ik}] \tag{2}$$

where a_i represents the current traffic node, and $[a_{ij} ... a_{ik}]$ represents all the next hop nodes of node i, from node j to node k.

(3) The reward value: The reward value is an important parameter that affects the training of the agent. After the agent selects an action in the current state, the environment is subsequently affected. A reward value is then obtained based on the feedback of the environment to evaluate the quality of the action. At the same time, the network transitions to the next state. Common reward values in network traffic are designed for link bandwidth, latency, and throughput. The goal of this study is to minimize the delay of the link, so the reward set is the average delay of the link. The reward value is expressed as shown in Equation 3:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

$$R = -\frac{\sum_{j=1}^{m} D_j}{m} \tag{3}$$

where m represents the number of flows in the network, D_j represents the link delay of the jth flow, and R is the average link delay. Because the agent needs to find the maximum reward value during training, the reward value is set as a negative value.

This study introduces deep reinforcement learning (DDQN) to compute routing decisions.The main network, Q-network, and the target network, target Q-network, are both four-layer networks, consisting of an input layer, two hidden layers, and an output layer. The values output by the output layer are mapped into the action space, corresponding one-to-one with the designed candidate actions. In the training process, the intelligent agent first interacts with the environment, takes actions A_t based on the state S_t, and receives a reward value r_t. After the agent takes action, the environment enters the next state, and these four parameters are stored in the experience pool. The network parameters are updated when the experience pool is full. Firstly, both S_t and S_{t+1} in memory $(S_t, A_t, r_t, S_{t+12})$ are input into the main network, Q-network, obtaining the value function $Q_{main}(A_t)$ and optimal Q-value $MAXQ_{main}(A_{t+1})$, which can be calculated as shown in Equation 4:

$$A_{t+1} = argmaxQ(S_{t+1}, A_{t+1}, w) \tag{4}$$

Next , $MAXQ_{main}(A_{t+1})$ is input into the target network and $Q_{targte}(A_{t+1}) = MaxQ_{main}(A_{t+1})$ is obtained through $Q_{targte}(A_{t+1})$. Then, $Q_{targte}(A_{t+1})$, $Q_{main}(A_t)$, and r_t are incorporated into the loss function to calculate the loss, which is illustrated by Equation 5:

$$loss = E[(r_t + \gamma Q_{target}(S_{t+1}, A_{t+1}, w_{target}) - Q_{main}(S_t, A_t, w_{main}))^2] \tag{5}$$

Then, the main network parameters are updated according to the loss function by being copied to the target network at regular intervals. According to the Bellman formula, the objective value function can be calculated from the estimated values $Q_{target}(S_{t+1}, A_{t+1}, w_{target})$ of r_t and a_{t+1}, as shown in Equation 6:

$$q_t = r_t + \gamma Q_{target}(S_{t+1}, argmax_{a_j} Q_{main}(S_{t+1}, A_{t+1}, w_{main}), w_{target}) \tag{6}$$

The network parameters of the original network are then updated by Equation 7:

$$w'_{main} = w_{main} + \alpha(Q_{main}(S_t, A_t, w_{main}) - q_t)\nabla Q_{main}(S_t, A_t, w_{main}) \tag{7}$$

During the training process of the primary Q-network, the experience replay technique is used to store a series of states, actions, rewards, and next states obtained by the intelligent agent interacting with the environment in an experience replay pool. During training, fixed batches of data can be randomly selected from the experience pool to increase the training speed of the intelligent agent. However, since each datum stored after the interaction between the intelligent agent and the environment contains the next moment state, there is a certain correlation between the samples. To reduce the correlation between data samples and prevent the intelligent agent from falling into the local optimum, a random strategy is adopted when selecting data sets. In addition, because each tuple has different contributions to training, some scholars have used the priority-based experience replay technique. This study adopts the method of directly selecting fixed batches of data from the total experience memory for training using the experience replay technique. When the memory is full, the principle of first-in-first-out (FIFO) is used to replace old data with new data.

The algorithm in this paper is mainly divided into four stages: local update, parameter upload, parameter aggregation, and parameter delivery. These stages are defined as follows: (1) Local update:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

The local update process adopts the routing algorithm based on DDQN to train the local model and update the parameters. (2) Parameter upload: The agent in each domain controller completes training via the local environment, and multiple domain controllers upload their own model parameters $w_i(\varepsilon)$ to the server. Here, i denotes the ID of the domain controller, while denotes the number of local iterations. (3) Parameter aggregation: D represents the delay of all links, as shown in Equation 8. The formula for parameter aggregation is presented in Equation 9, in which ρ_i represents the ratio of the parameter data volume of the i-th domain controller to the total data volume, and n represents the number of domain controllers.The calculation expression for ρ_i is shown in Equation 10. The server trains the global model through the aggregated parameters to generate global model parameters. The model convergence judgment is performed before the global model parameters are issued. If the model converges, it means that the global model has been learned, and the federated reinforcement learning routing algorithm ends. Otherwise, the algorithm enters the parameter delivery stage. (4) Parameter delivery. In the parameter delivery stage, global parameters are delivered to each domain controller. The domain controllers assign the parameters to the local model and use local data to update the model training parameters. This four-stage process is executed cyclically.

The global loss function $F(w)$ is the weighted average of the loss functions of each domain controller, as shown in Equation 11. The goal of the federated reinforcement learning routing algorithm is to minimize the loss function, as shown in Equation 12.

$$D = \sum_{i=1}^{n} D_i \tag{8}$$

$$w^{t+1}(\rho) = \frac{\sum_{i=1}^{n} D_i w_i^t(\varepsilon)}{D} = \sum_{i=1}^{n} \rho_i w_i^{t+1}(\varepsilon) \tag{9}$$

$$\rho_i = \frac{D_i}{D} \tag{10}$$

$$F(w) = \frac{\sum_{i=1}^{n} D_i F_i(w_i)}{D} = \sum_{i=1}^{n} \rho_i F_i(w_i) \tag{11}$$

$$w_{\text{find}}(\varepsilon) = \operatorname{argmin} \frac{\sum_{i=1}^{n} D_i F_i\left(w_{\text{global}}(\varepsilon)\right)}{D} = \operatorname{argmin} \sum_{i=1}^{n} \rho_i F_i\left(w_{\text{global}}(\varepsilon)\right) \tag{12}$$

The formula for proxy experience memory is shown in Equation 13:

$$M = \{(s_i^p, \pi^p(a_i|s_i^p))\}_{i=0}^{N^p} \tag{13}$$

where s_i^p indicates the proxy state in each domain controller, i represents the ID of the domain controller, and π^p represents the average proxy strategy. The proxy state set $C_i \in C$ is a further division of the state space, and the aggregation of the proxy state set C_i is the entire state space C. N indicates the number of participating domains.

The interaction process of proxy experience memory between the server and domain controllers in the FedRDR algorithm is as follows:

1. First, each domain trains the model based on local data and stores tuples (S_t, A_t, r_t, S_{t+1}). The states in the tuple are then aggregated to form a proxy state set C_j. The policy in the proxy state set is the average of the multiple state policies $\pi_{\theta_i}(a|s)$ that make up the proxy state. The average strategy calculation process is as shown in Equation 14:

$$\pi_{\theta_i}^p(a_k|s_k^p) = \sum_{k=i}^{j} \frac{\pi_{\theta_i}(a_k|s_k)}{j-i} \tag{14}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

where θ_i represents the local model, s_k^p represents a series of state sets,$(s_i...s_k...s_j) \in s_k^p \in C_j$. All proxy states are combined to form a set.

2. After all the agents have filled the experience memory, the average proxy strategy is calculated; the proxy experience memory, the proxy state, and the corresponding average strategy are constructed; and these are uploaded to the server. The proxy experience memory of the ith domain controller is formulated as shown in Equation 15:

$$M_i^p = \{(s_k^p, \pi_{\theta_i}^p(a_k|s_k^p))\}_{k=0}^{N_i^p} \tag{15}$$

where N_i^p represents the local agent experience memory size and p refers to policy.

3. When the local proxy experience memory is uploaded to the server, it is then aggregated. The same proxy states of multiple domains are combined into one proxy state, and the corresponding policies are then, once again, averaged. The global average proxy experience memory formula is shown in Equation 16:

$$M^p = \{(s_k^p, \pi^p(a_k|s_k^p))\}_{k=0}^{N^p} \tag{16}$$

4. The global average proxy experience memory is then delivered from the server to the domain controller.
5. After the domain controller receives the global average proxy experience memory, it calculates the KL dispersion loss of the local mode and the global average proxy experience memory so that the agent can learn knowledge and speed up the training process of the local agent. The KL dispersion loss function is shown in Equation 17:

$$L_i^p(M^p, \theta_i) = -\sum_{k=1}^{N^p} \pi^p(a_k|s_k^p) log(\pi_{\theta_i}(a_k|s_k^p)) \tag{17}$$

The proxy state in the proxy experience memory in each domain is not the original state, and the actual policy corresponding to the proxy state is averaged over time. Therefore, sharing local agent experience memory not only protects data privacy in the domain, but also reduces the amount of communication data. The FedRDR algorithm model is shown in Figure 2:

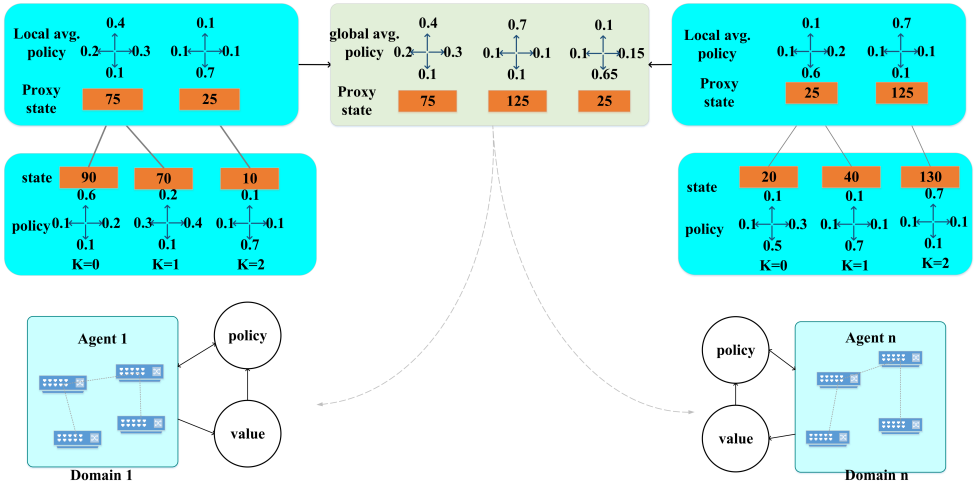

Figure 2. The FedRDR algorithm model.

Firstly, the DDQN-based intra-domain routing algorithm is introduced, as shown in Algorithm 1. Secondly, the FedRDR algorithm is introduced, as shown in Algorithm 2.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

Algorithm 1 Federated reinforcement learning model training

Require:

The short-term switch traffic F, the network topology T, and the target network update frequency ζ

Ensure:

The routing path

Initialize the hyperparameters of the DDQN agent;
Initialize the virtual simulation environment env based on Mininet, initialize the current state S, the next state S', the action A, and the reward value R;
Initialize switch flow table F, link delay, network topology T;
$T <-$ get network topology;
$P <-$ Find all candidate nodes from the source node to the destination node according to T;
$F <-$ Obtain the traffic information from switches $j \in (1, 2, ..., m)$;
for $t = 1, 2, 3, ..., T$ **do**
    Agent input state $< -S_t$:
    $A_t <-$ Agent output action;
    According to the action A_t, the domain controller selects the next hop from the candidate node P, changes the corresponding switch flow table, and forwards the data to the next hop node;
    r_t,$TMP <-$ The domain controller obtains the network link delay and the switch flow meter;
    $S_{t+1} < -(TMP - F)$, $F < -TMP$;
    $S_t < -S_{t+1}$;
    Agent puts (s_t, A_t, r_t, S_{t+1}) into the experience playback pool;
    if $Memory == M$ **then**
        Store the newly obtained result (s_t, A_t, r_t, S_{t+1}) and overwrite the old result using the FIFO principle;
        Select a fixed batch of data from $Memory$, and update the main network parameters $w_{Main}(t)$ by minimizing the loss function;
    end if
    if $t\%\zeta == 0$ **then**
        Update the target parameters $w_{target}(t) = w_{main}(t)$;
    end if
    if $t\%\varphi == 0$ **then**
        The local proxy experience memory is constructed, the proxy experience memory of each domain controller is uploaded to the server, and the global proxy experience memory M^p is obtained;
        After successfully building the global proxy experience memory, it will be distributed;
        All agents in the domain controller update the local model;
        if M^i_{global} constant **then**
            Federal reinforcement distillation ends;
        end if
    end if
end for
return M^i_{global} ;

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

Algorithm 2 FedRDR: Federated reinforcement distillation-based routing algorithm

Require:

The main network parameters w_{main} and target network parameters w_{target} in DDQN, the network parameter update frequency ζ, the federated reinforcement distillation model update frequency φ, the number of domain controllers n, and the local model for federated reinforcement distillation M^i_{global}.

Ensure:

Local decision model

Initialize local model parameters, $w^i_{Main}(0)$=$w^i_{Target}(0)$ $i \in \{1, 2, ..., i, ..., n\}$ M^i_{global};
Set the $Memory = M$ used to store the optimal results;
Set the hyperparameters of DDQN according to the Algorithm 1 description, with time frame T;
for $t = 1, 2, 3, ..., T$ **do**
    Perform the following operations on n domain controllers at the same time (take domain controller i as an example):
    Initialize switch flow table F, link delay, and network topology T;
    $T <-$ Obtain network topology;
    $P <-$ Find all candidate nodes from source to destination according to T ;
    $F <-$ Obtain the traffic information from switches $j \in (1, 2, ..., m)$;
    $S = F$
    Agent input state$< -S_t$;
    $A_t <-$ Agent output action;
    According to the action A_t, the controller selects the next hop from the candidate node P, changes the corresponding switch flow table, and forwards the data to the next hop node;
    $r_t, TMP <-$ Controller obtains the network link delay and switch flow;
    $S_{t+1} < -(TMP - F), F < -TMP$;
    $S_t < -S_{t+1}$;
    Agent puts (s_t, A_t, r_t, s_{t+1}) into the experience playback pool;
    if $Memory == M$ **then**
        Store the newly obtained result (s_t, A_t, r_t, s_{t+1}) and overwrite the old result using the FIFO principle;
        Select a fixed batch of data from $Memory$, and update the main network parameters $w_{Main}(t)$ by minimizing the loss function;
    end if
    if $t\%\zeta == 0$ **then**
        Update target network parameters $w_{target}(t) = w_{Main}(t)$;
    end if
    if $t\%\varphi == 0$ **then**
        Update target network parameters $w_{target}(t) = w_{Main}(t)$;
    end if
end for

5. Evaluation and Discussion of Results

5.1. Experimental Environment

The simulation system described in this paper is implemented using a Mininet network simulator and a Ryu controller. The algorithm itself is implemented in Python, with pytorch serving as the overall framework for the algorithm. The experimental environment used in this study is presented in Table 1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

Table 1. Experimental environment.

Type	Description
Hardware	Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz3.19 GHz
Random-access memory	8.00GB
System type	Ubuntu 20.0.4
Development language	Python
Development tool	PyCharm 2019.1.1 (Professional Edition)
Virtual environment	Mininet + Ryu

5.2. Simulation System Design

This system is implemented by the design of four modules, as follows:

- Network status monitoring module: The main function of the network status monitoring module is to obtain the network status information through echo and LLDP messages in the controller. The two types of messages interact between the SDN controller and the switch, obtaining the required states, rewards, and actions for our DDQN-based intelligent agent—more specifically, the short-term traffic information of switches, the network topology, and the link latency based on the topology structure.
- Smart routing decision module: The DDQN-based intra-domain routing algorithm is the core of the decision module. The main function of this module is to make routing decisions based on the network status. This module can be divided into three parts: environment perception, self-learning of intelligent agents, and handling of experience memory.
- Routing decision execution module: This module is mainly used to guide the switches to forward traffic according to the routing decisions made by the controller. After the routing decision module calculates the routing paths, the controller installs the flow table to the switch. The switch updates its flow table and forwards traffic based on the new flow table.
- Server processing module: The main task of the server processing module is to connect to the domain controller and receive the parameters transmitted by the domain controller. The Update_params function aggregates all the parameters uploaded by the domain controller and transmits the global parameters to each domain controller to accelerate the intra-domain training process, and iterate this process until the model converges.

5.3. Performance Evaluation

Setting appropriate hyperparameters—such as the learning rate, batch size, and memory size for experience replay—during the process of training an intelligent agent can make the model converge faster. Therefore, we conducted experiments on the hyperparameters of the intelligent agent to select the optimal parameters.

From Figure 3, it can be seen that when the learning rate is set to 0.1 and 0.01, the intelligent agent shows rapid gradient descent in the early stages, but the convergence effect of the algorithm is poor in later stages. Compared with learning rates of 0.0001 and 0.001, the intelligent agent has already converged after 200 iterations with a learning rate of 0.001. Therefore, we set the parameter learning rate to 0.001.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

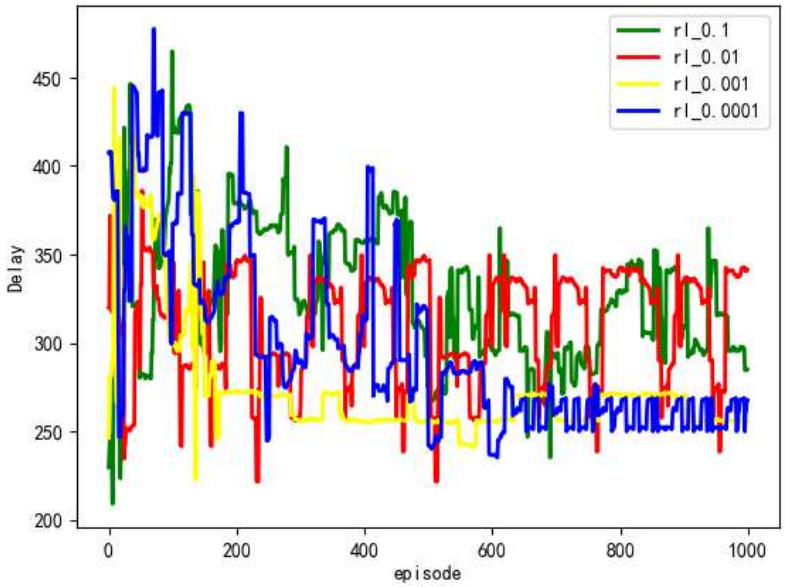

Figure 3. The effect of the learning rate on the algorithm.

We set the batch sizes to 16, 32, and 64, and the experimental results are shown in Figure 4. From the graph, it can be seen that when the batch size is set to 16, the algorithm cannot converge. Compared with a batch size of 64, the model has better convergence when the batch size is set to 32. Therefore, we set the batch size to 32.

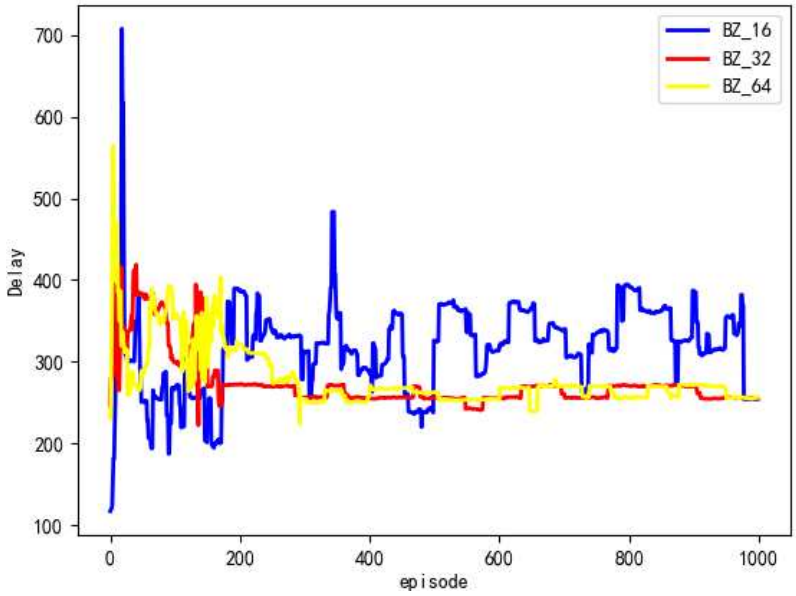

Figure 4. The impact of the data batch size on the algorithm.

We determined the other hyperparameters of the agent through experiments, and finally set the hyperparameters of the agent as shown in Table 2:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

Table 2. Agent model hyperparameters.

Hyperparameters	Value
Learning rate	0.001
Batch size	32
Memory	10000
Reward discount	0.9
MAX STEPS	1000
Network update frequency	100

To verify the performance of the domain-based routing algorithm based on deep reinforcement learning, after determining the model hyperparameters of the intelligent agent, we conducted comparative experiments. The compared algorithms were the depth first search (DFS) algorithm and the Q-learning-based routing algorithm using reinforcement learning.

By sending network traffic packets through iperf, three traffic packets were sent from h11, h12, and h13 to h41, h42, and h43, respectively. The bandwidth requirements for each stream were set as shown in Equations 18, 19, and 20:

$$F_1 : h_{11} -> h_{41} : 3Mbit/s \tag{18}$$

$$F_2 : h_{12} -> h_{42} : 2Mbit/s \tag{19}$$

$$F_3 : h_{13} -> h_{43} : 2Mbit/s \tag{20}$$

Comparative analysis of the three routing algorithms was conducted under this network state, and the experimental results are shown in Figure 5. From the graph, we can see that the delay of data transmission using the routing strategy designed by the DFS algorithm is about 130ms, which is not the optimal forwarding path. However, when the traffic in the network increases, the DFS algorithm cannot implement the optimal forwarding strategy. For state-aware intelligent routing algorithms, the final link delay under the converged state of both is about 25ms. For the same device, the Q-learning algorithm requires more time to reach convergence compared to the DDQN routing algorithm.

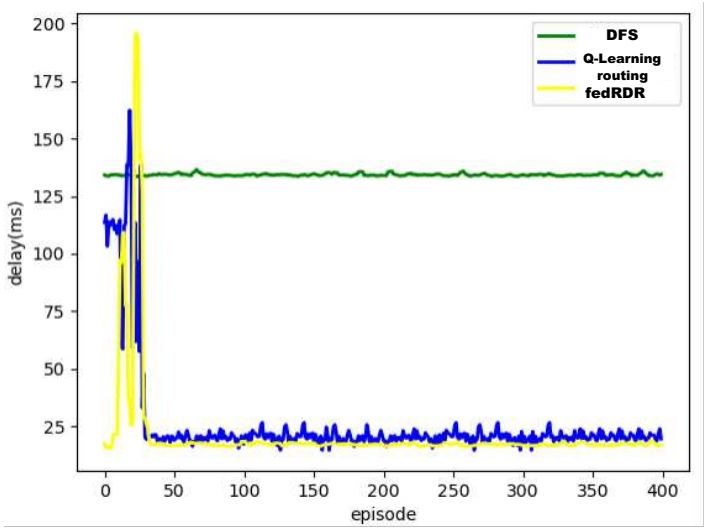

Figure 5. A performance comparison of the three algorithms based on Figure 4.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

To demonstrate the advantages of our proposed algorithm, we increased the number of switches to 8 and the number of flows to be forwarded in the network to 8, and compared the performance of the three routing algorithms. The experimental results are shown in Figure 6.

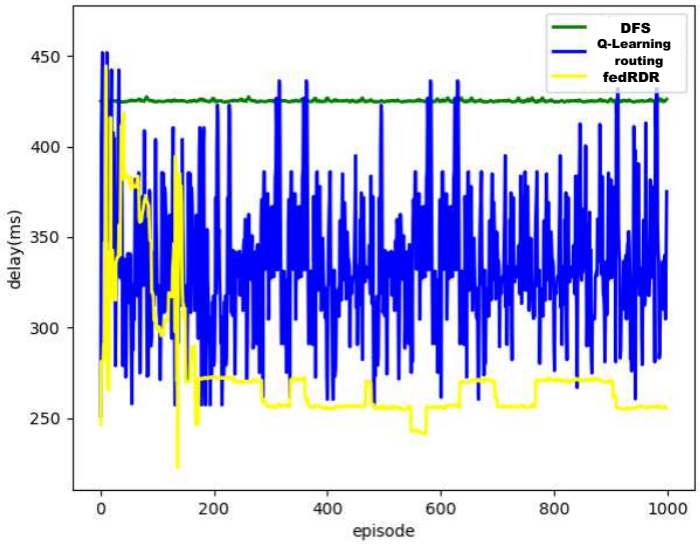

Figure 6. Performance comparison of the three algorithms based on 8 switches.

From the graph, it is evident that the routing strategy calculated by the DFS algorithm still leads to a high network delay. Furthermore, it was discovered in this study that the Q-learning-based routing algorithm failed to converge within 1000 iterations. When the number of switches and flows increases, the number of Q-table items also increases significantly, resulting in slow or even non-convergence of the algorithm. In contrast, our proposed DDQN-based routing algorithm is capable of quickly converging and guiding traffic forwarding, even in the presence of a large state space and action space.

In our approach, we designed a process for uploading the local model parameters to the server after every 100 iterations in each domain. The server then aggregates the parameters from the three domains. Based on the aggregated parameters, the server undergoes an additional 100 iterations of training before distributing the parameters back to the domain controllers. The domain controllers continue training the model based on the received parameters, and this process repeats until the model reaches a stable state. The experimental results of the federated reinforcement learning routing algorithm are depicted in Figure 7. After receiving the aggregated parameters from the server, the model of each domain controller achieved convergence after around 20 iterations of training.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

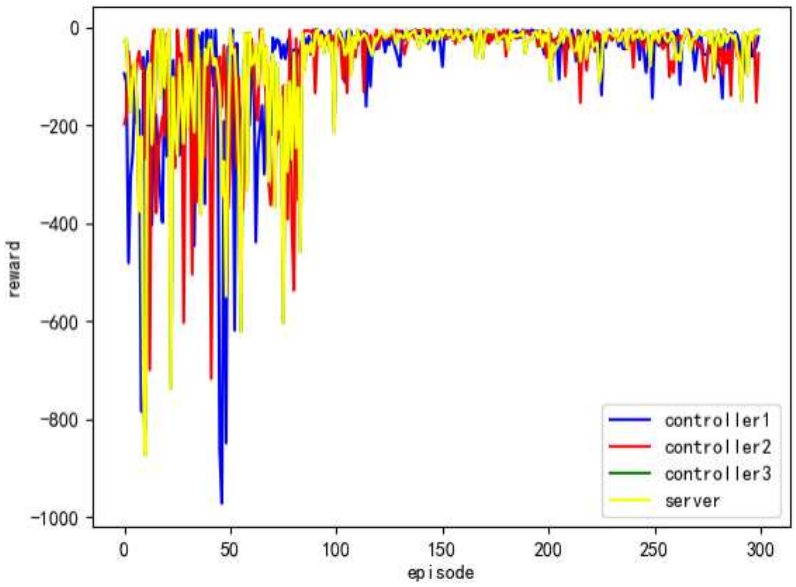

Figure 7. Three-domain federation reinforcement learning routing.

When a new task domain emerges, the model parameters that already exist in the server are sent to the new domain. The new domain does not need to train the local model from scratch, which saves time when it comes to agent training. We selected one of two domains with the same task for local training. After local model training is completed, the model parameters migrate to the other domain controller via the server to verify the impact of the model parameters within one domain on the acceleration effect of the agent in another domain. The experimental results are shown in Figure 8:

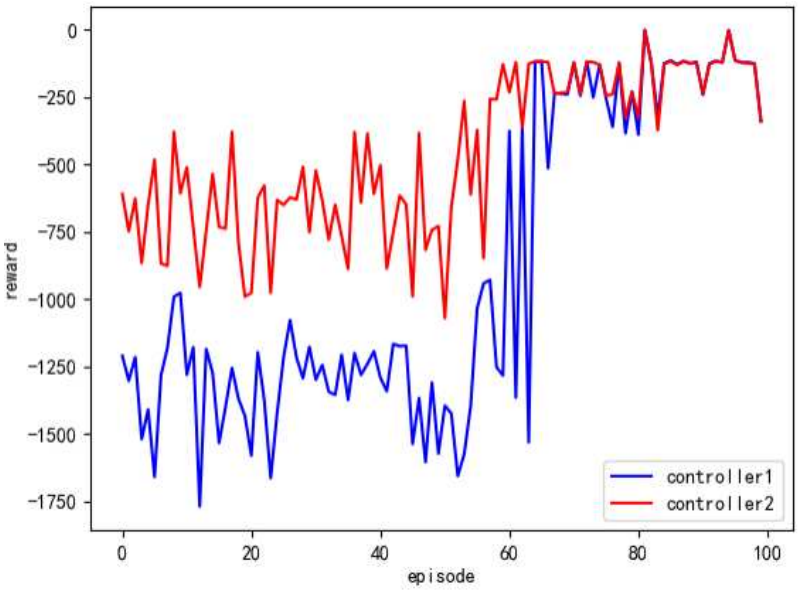

Figure 8. The accelerated effect of training based on federated reinforcement learning.

In the experiment, Controller 1 represents an intelligent agent that has not undergone any parameter assignment training, while Controller 2 represents an intelligent agent after parameter assignment training. From the graph, it is evident that the intelligent agent in Controller 1 reaches a convergence state after almost 70 iterations, whereas the intelligent agent in Controller 2 achieves convergence after approximately 60 iterations. This experimental result demonstrates that the federated reinforcement learning-based multi-domain intelligent routing algorithm can expedite the training process of intelligent agents within a single domain.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

Next, we examined the impact of the number of domains on the convergence of the global model. We conducted experiments with 3, 5, and 8 domains. The training process was consistent across all experiments, where parameters were sent from the domain controller to the server every 100 iterations. The server then performed parameter aggregation and trained for an additional 100 iterations based on the aggregated parameters before distributing the parameters back to the domain controllers. This communication process was repeated three times.

The experimental results are depicted in Figure 9. From the graph, it can be observed that when there are more domains, the model exhibits less fluctuation after nearly 100 iterations and achieves a slight advantage in terms of convergence speed. However, irrespective of whether 3, 5, or 8 domains participate in the federated reinforcement learning-based multi-domain joint process, the accuracy of the model is hardly affected. A smaller number of domains does not lead to inaccurate models in the federated reinforcement learning-based multi-domain joint process. However, as the number of domains increases, the amount of uploaded data also increases, thereby enhancing the generalizability and applicability of the trained model. The experimental results demonstrate the significant utility advantages of the federated reinforcement learning-based multi-domain intelligent routing algorithm.

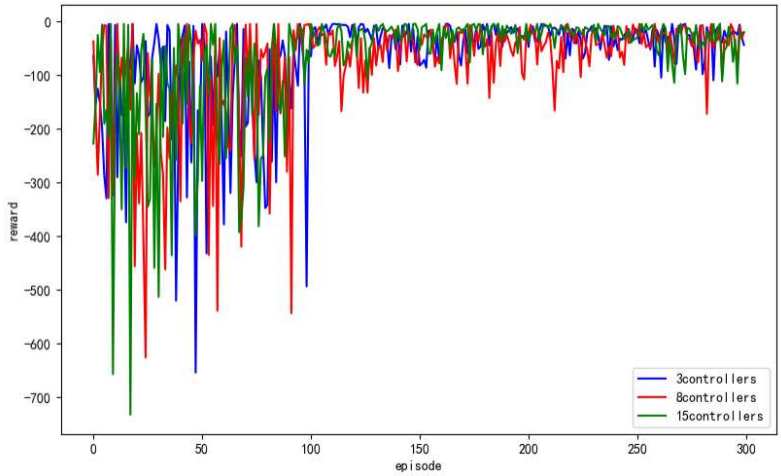

Figure 9. The effect of the number of domains on the model.

Next, we verified the effect of the federated reinforcement learning-based multi-domain intelligent routing algorithm on reducing the training energy consumption within a single domain. The results are shown in Figure 10:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

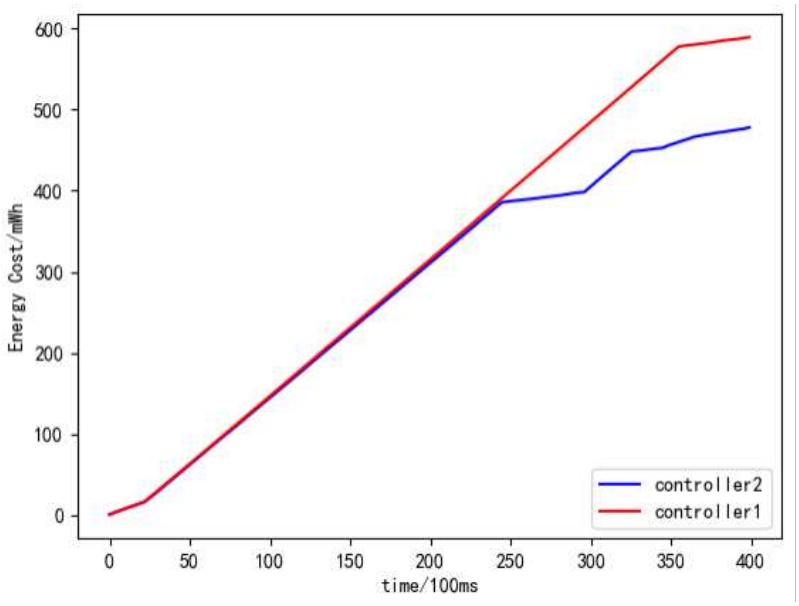

Figure 10. The impact of energy consumption based on federated reinforcement learning.

The horizontal axis represents time, with each recording interval being 100ms. The vertical axis represents energy consumption, measured in milliwatt-hours (mWh). From the graph, it is evident that the energy consumption of the agents in both controllers is nearly identical until the 250th recording. However, since we assigned the model parameters from Controller 1 to Controller 2, the training speed in the second domain controller is accelerated, leading to faster model convergence. Consequently, the training process can be terminated once the model converges. As a result, the second controller concludes training earlier, reducing the energy consumption of the UAV acting as the domain controller due to reduced computation requirements.

The federated reinforcement learning-based multi-domain intelligent routing algorithm necessitates interaction and communication between domain controllers and servers to update the model parameters of intelligent agents. During the communication between the server and domain controllers, we collected data on the amount of exchanged data in model parameter interactions between domain controllers and servers, as illustrated in Figure 11.

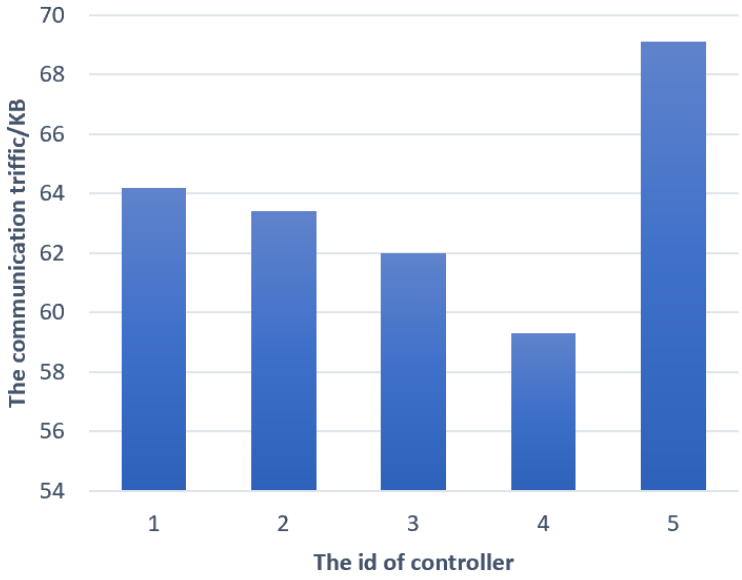

Figure 11. The amount of data communicated between the server and the domain controller.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

From the figure, we can observe the specific size of the data volume for each communication between the domain controllers and the server. The average communication data volume is approximately 63.6KB.

In summary, the federated reinforcement learning routing algorithm can expedite the training process of intelligent agents within domains, while reducing the computational energy consumption of domain controllers. The experimental results provide evidence of the significant utility advantages of the federated reinforcement learning-based multi-domain intelligent routing algorithm.

Initially, the experiment validated the effectiveness of the FedRDR algorithm in accelerating the training of intelligent agents within domains. In federated reinforcement distillation, the agent's experience memory stored in domain controllers is uploaded to the server, and intelligent agents from other domains learn from this experience memory. The comparison results are depicted in Figure 12.

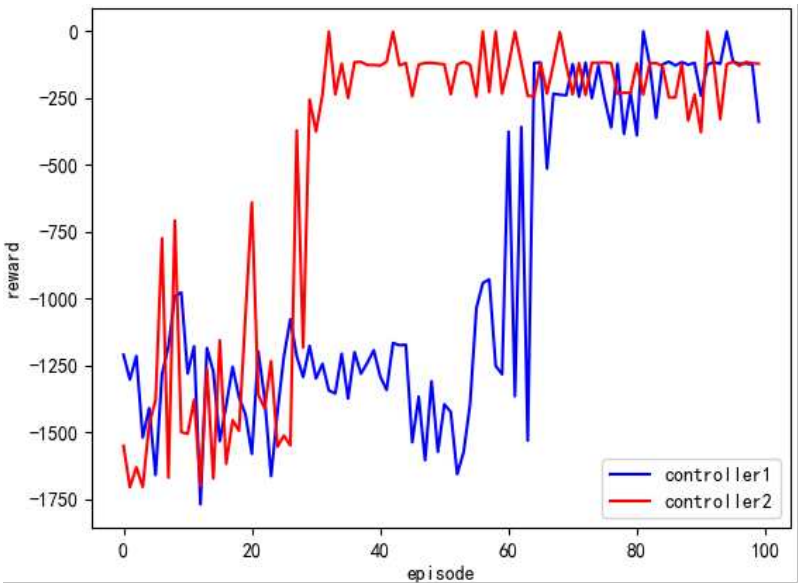

Figure 12. The effects of federated reinforcement distillation on accelerating model training.

The intelligent agent of domain Controller 1 undergoes local environment training to generate a local experience memory. Once the experience memory storage is full, it is sent to domain Controller 2 to observe its effect on accelerating the training of intelligent agents within the domain. From Figure 12, it can be observed that the model in domain Controller 1 reaches convergence after approximately 70 iterations of training, while the intelligent agent in domain Controller 2 achieves convergence after around 40 iterations of training. This indicates that knowledge transfer can enhance the learning speed of intelligent agents. During the training process in domain Controller 2, the training data are not obtained through the interaction between the local intelligent agent and the environment; instead, they are provided by domain Controller 1. This reduction in the interaction process between the intelligent agent and the environment contributes to the acceleration of training. Therefore, the proposed federated reinforcement distillation-based multi-domain intelligent routing algorithm can expedite the training process of intelligent agents within domains.

Both the federated reinforcement learning routing algorithm and the FedRDR algorithm demonstrate the capability of accelerating the training of intelligent agents within domains. We compare their effects on accelerating the training of intelligent agents, and the experimental results are illustrated in Figure 13.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2023 doi:10.20944/preprints202310.1118.v1

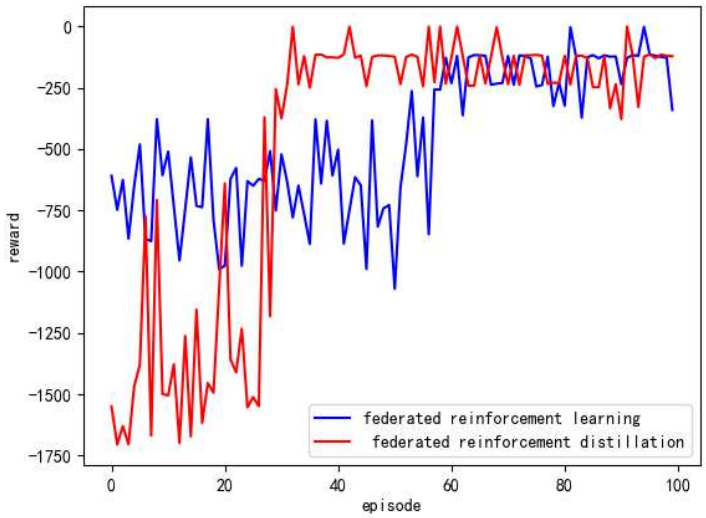

Figure 13. A comparison of the acceleration effects of federated reinforcement learning and federated reinforcement distillation.

From the graph, it is evident that the federated reinforcement learning model achieves convergence after approximately 60 iterations of training. On the other hand, the federated reinforcement distillation model enters the convergence state after about 40 iterations of training. This indicates that knowledge transfer through federated reinforcement distillation has a better acceleration effect and reduces the training time for single-domain intelligent agents compared to parameter transfer through federated reinforcement learning.

In the federated reinforcement learning routing algorithm, we also examined the size of communications between the server and domain controllers during their interactions, which averaged around 63.6KB. Additionally, we collected the amount of data transferred between the five domain controllers and the server during each communication, as depicted in Figure 14.

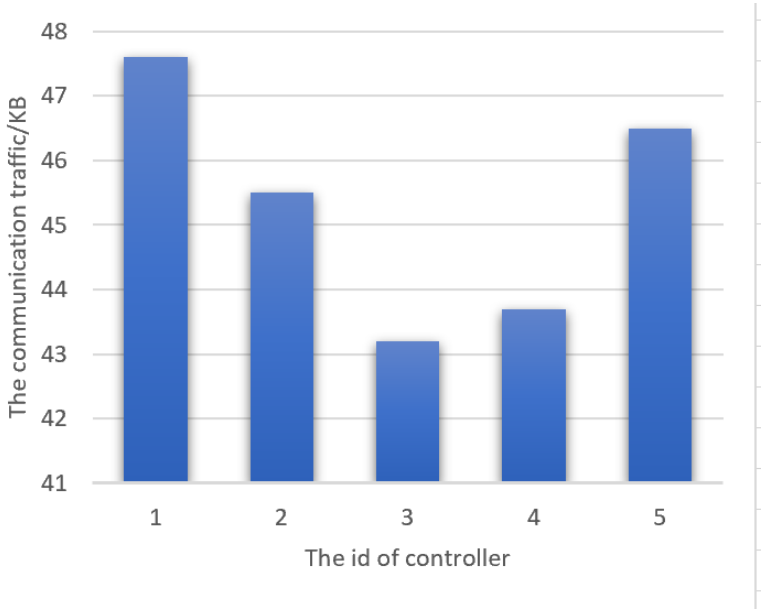

Figure 14. The traffic size based on federated reinforcement distillation.

 doi:10.20944/preprints202310.1118.v1

The maximum amount of data transferred between each domain controller and the server during communication was 47.6KB, whereas in the federated reinforcement learning routing algorithm, the lowest amount of communication data was 59.3KB. Federated reinforcement distillation demonstrates better communication efficiency compared to federated reinforcement learning. By proxying the state and aggregating the policy of the states, federated reinforcement distillation reduces the communication overheads between the server and domain controllers. In the multi-domain framework utilizing federated reinforcement learning, the average amount of communication between each domain and the server per iteration is approximately 63.6KB, while in the multi-domain framework based on federated reinforcement distillation, it is around 45.3KB. Compared to parameter transmission through the federated reinforcement learning algorithm, FedRDR can reduce the amount of transmitted parameters by approximately 29%. The average communication amounts are presented in Table 3:

Table 3. The traffic based on federated reinforcement learning framework and federated reinforcement distillation framework.

Algorithm	Amount of communication data per round
Federated reinforcement learning	63.6KB
Federated reinforcement distillation	45.3KB

As presented in the above table, the FedRDR algorithm offers a significant reduction in the transmitted data volume, thereby enhancing the training speed of single-domain intelligent agents. Additionally, when it comes to federated reinforcement distillation, the transmitted data only comprise the training data, irrespective of the intricacy of the models within each domain.

In a nutshell, the FedRDR algorithm not only expedites the training process of intelligent agents in individual domains, but also diminishes the communication data between domain controllers and servers. Our empirical findings demonstrate that the efficacy of the FedRDR algorithm becomes more pronounced when confronted with intricate environments and tasks.

6. Discussion

In order to achieve the rapid deployment of communication networks in disaster-stricken areas and transportation hotspots in the Internet of Things (IoT), and to provide link services for ground nodes, this paper present a temporary network with drones as communication nodes. In addition, we propose a hierarchical, multi-domain data transmission architecture based on SDN. We further divide the control layer of SDN, enhancing the network control capability and simplifying network management. By utilizing the computing and communication capabilities of drones, communication devices in the network can communicate more rapidly. Moreover, when high-performance drones serve as domain controllers, deploying efficient routing algorithms on them can make the entire IoT system more flexible and efficient. Therefore, this study focuses on the design of the FedRDR algorithm and its deployment on high-performance drones. Compared to the federated reinforcement learning routing algorithm, the FedRDR algorithm reduces approximately 29% of the transmission parameter quantity and further accelerates the convergence speed of the model. This paper verifies the proposed routing algorithm's excellent performance in drone-assisted IoT systems based on the simulation system. It demonstrates the significance of the proposed architecture and algorithm for temporary networking supported by drones, thereby resolving the communication transmission obstacles in traditional IoT systems used in disaster-stricken areas and transportation hotspots.

Author Contributions: Conceptualization, J.L. and A.L.; methodology, J.L.and A.L; software, A.Land S.C.; validation, J.L.,A.L.and S.C.; formal analysis, A.L.and S.C.; investigation, J.L.,A.L.and S.C.; resources, A.L., W.F.and S.C.; data curation, W.F.and A.L.; writing—original draft preparation, A.L.and S.C.; writing—review and editing, A.L.,X.W.and S.C.; visualization, A.L.,X.W.and S.C.; supervision, A.L.,X.W.and S.C.; project administration, J.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key Research and Development Projects (2022YFB450 0800); the Applied Basic Research Program Project of Liaoning Province (2023JH2/101300192); the Fundamental Research Funds for the Central Universities (N2116014); the National Natural Science Foundation of China (62032013 62072094); nd the New Generation Information Technology Innovation Project of the Ministry of Education (2021ITA10011)

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: Not applicable.

References

1. Xie, Jiapin and Fu, Qiyong and Jia, Riheng and Lin, Feilong and Li, Ming and Zheng,Zhonglong."Optimal Energy and Delay Tradeoff in UAV-Enabled Wireless Sensor Networks", *Drones* **2023**, 368.
2. Wang Y, Farooq J. Resilient UAV formation for coverage and connectivity of spatially dispersed users. *ICC 2022-IEEE International Conference on Communications* **2022**, *10*, 225-230.
3. "Take a look at AT&T's 'Flying COWs' - drones that returned cell service to hurricane Ida-hit Louisiana".Available online: https://news.yahoo.com/look-ts-flying-cows-drones-113500471.html.(Sept.2021)
4. G. He, W. Bao and Y. Hui, "A UAV Emergency Network User Allocation Method for Load Balancing," in *2022 8th International Conference on Big Data and Information Analytics (BigDIA), Guiyang, China*, **2022**
5. Xu Z, Xinlu L, Lin X L. "Research on the Construction of Forestry Protection Drone Project-Take the Construction of Forest Fire Monitoring Project of Huizhou Engineering Vocational College as an Example" in *6GN for Future Wireless Networks: 4th EAI International Conference, 6GN 2021, Huizhou, China, October 30–31, 2021*,pp. 230-244.
6. Deng, Xiaoheng and Wang, Leilei and Gui, Jinsong and Jiang, Ping and Chen, Xuechen and Zeng, Feng and Wan, Shaohua."A review of 6G autonomous intelligent transportation systems: Mechanisms, applications and challenges", *Journal of Systems Architecture* **2023**, 102929.
7. Abdellatif A A, Mhaisen N, Mohamed A, et al."Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data,"*Future Generation Computer Systems*, vol.128, pp.406-419, 2022.
8. L. Liu, J. Zhang, S. H. Song and K. B. Letaief, "Communication-Efficient Federated Distillation with Active Data Sampling," *ICC 2022 - IEEE International Conference on Communications, Seoul, Korea*, pp. 201-206,2022.
9. Paguem Tchinda A."Optimisation of Wireless Disaster Telecommunication Network based on Network Functions Virtualisation under special consideration of Energy Consumption,". *University of Plymouth*, 2022.
10. Alsamhi S H, Shvetsov A V, Kumar S, et al."UAV computing-assisted search and rescue mission framework for disaster and harsh environment mitigation," *Drones*, vol.6, no.7, pp.154,2022.
11. Liu C , Feng W , Chen Y , et al. "Cell-Free Satellite-UAV Networks for 6G Wide-Area Internet of Things,"*IEEE Journal on Selected Areas in Communications*, vol.39, no.4, pp.1116-1131,2021.
12. G. B. Tarekegn, R. -T. Juang, H. -P. Lin, Y. Y. Munaye, L. -C. Wang and M. A. Bitew, "Deep-Reinforcement-Learning-Based Drone Base Station Deployment for Wireless Communication Services," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21899-21915, 1 Nov.1, 2022.
13. C. Zhang, L. Zhang, L. Zhu, T. Zhang, Z. Xiao and X. G. Xia, "3D deployment of multiple UAV-mounted base stations for UAV communications," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2473-2488, April 2021.
14. Wang, L.; Zhang, H.; Guo, S.; Yuan, D. "3D UAV Deployment in Multi-UAV Networks with Statistical User Position Information," *IEEE Commun. Lett.* vol.26, no.6, pp.1363–1367,2022.
15. Shi, W.; Li, J.; Wu, H.; Zhou, C.; Cheng, N.; Shen, X. "Drone-Cell Trajectory Planning and Resource Allocation for Highly Mobile Networks: A Hierarchical DRL Approach," *IEEE Internet Things J*, vol.8, no.12, pp.9800–9813,2020.
16. Z. Dai, Y. Zhang, W. Zhang, X. Luo and Z. He, "A Multi-Agent Collaborative Environment Learning Method for UAV Deployment and Resource Allocation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 120-130, 2022.

17. Wu, Chuhan and Wu, Fangzhao and Lyu, Lingjuan and Huang, Yongfeng and Xie, Xing. "Communication-efficient federated learning via knowledge distillation," *Nature communications*,**2022**,2032.
18. Wang H P, Stich S, He Y, et al. "Communication-efficient federated learning via knowledge distillation,"*International Conference on Machine Learning*,pp.23034-23054,2022.
19. T. -J. Yang, D. Guliani, F. Beaufays and G. Motta, "Partial Variable Training for Efficient on-Device Federated Learning," *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, Singapore*, pp. 4348-4352,2022.
20. Cha, H. and Park, J. and Kim, H. and Kim, S. L. and Bennis, M."Federated Reinforcement Distillation with Proxy Experience Memory"*arXiv preprint arXiv:1907.06536, 2019*,doi:10.36227/techrxiv.12645497.
21. Rischke, J. and Sossalla, P. and Salah, H. and Fitzek, Fhp and Reisslein, M."QR-SDN: Towards Reinforcement Learning States, Actions, and Rewards for Direct Flow Routing in Software-Defined Networks"*IEEE Access, 2020, 8:174773-174791*,doi:10.1109/ACCESS.2020.3025432.