Cost-efficient Federated Reinforcement Learning-Based Network Routing for Wireless Networks

Zakaria Abou El Houda[1], Diala Nabousli[1], and Georges Kaddoum[1]

[1]Resilient Machine Learning Institute (ReMI), École de Technologie Supérieure (ÉTS), Montréal, Canada
zakaria.abou-el-houda.1@ens.etsmtl.ca; Diala.Nabousli@etsmtl.ca; georges.kaddoum@etsmtl.ca.

***Abstract*—Advances in Artificial Intelligence (AI) provide new capabilities to handle network routing problems. However, the lack of up-to-date training data, slow convergence, and low robustness due to the dynamic change of the network topology, makes these AI-based routing systems inefficient. To address this problem, Reinforcement Learning (RL) has been introduced to design more flexible and robust network routing protocols. However, the amount of data ($i.e.$, state-action space) shared between agents, in a Multi-Agent Reinforcement Learning (MARL) setup, can consume network bandwidth and may slow down the process of training. Moreover, the curse of dimensionality of RL encompasses the exponential growth of the discrete state-action space, thus limiting its potential benefit. In this paper, we present a novel approach combining Federated Learning (FL) with Deep Reinforcement Learning (DRL) in order to ensure an effective network routing in wireless environment. First, we formalize the problem of network routing as a problem of RL, where multiple agents that are geographically distributed train the policy model in a fully distributed manner. Thus, each agent can quickly obtain the optimal policy that maximizes the cumulative expected reward, while preserving the privacy of each agent's data. Experiments results show that our proposed Federated Reinforcement Learning (FRL) approach is robust and effective.**

***Index Terms*—Network Routing; Federated Learning; Reinforcement Learning.**

I. Introduction

The Internet of Things (IoT) paradigm has emerged as a distributive technology that is giving rise to a plethora of new services and applications. IoT connects multiple devices, called objects, that perform and automate our daily life tasks. The IoT development has gained considerable momentum over the past few decades. The need for routing optimization in such environment has become prominent, especially with the increase in the network traffic volume, Quality of Service (QoS) demand, and security requirements [1], [2]. The current network routing schemes such as Open Shortest Path First (OSPF), are facing obstacles to satisfy the increasing demand of today's network in terms of high speed and low latency.

To address this issue, several state-of-the-art network-based routing schemes have been proposed by integrating Artificial Intelligence (AI) techniques. These techniques have gained considerable momentum over the past few decades, they become efficient to solve complex problems, including network routing. Mao et al. [3] proposed a novel routing strategy that uses DL architecture ($i.e.$, Deep Belief Architectures (DBA)) to ensure an effective network routing. The proposed DBA algorithm consists of multiple Restricted Boltzmann Machines (RBM) layers, including an input layer that takes as input the network data traffic observed at each router and one hidden layer. First, the authors used a Greedy Layer-Wise scheme to train the model. Then, they updated the values of weights and biases using the gradient descent optimization method. The authors evaluated their proposed routing strategy based on the delay, throughput, and signaling overhead. Sharma et al. [4] designed a novel ML-based approaches for network routing, called MLProph. MLProph is an enhanced version of a previous work, called PROPHET+ [5], a probabilistic routing scheme for opportunistic networks. MLProph aims to maximize the data delivery rate while minimizing the transmission delay. For this aim, the authors used two ML-based approaches, namely Neural Networks (NN) and Decision Trees (DT); these ML-based approaches took as input a set of parameters $e.g.$, buffer capacity, number of successful deliveries, and node popularity; while the output value indicates whether a successful delivery is likely if transmitted on this link. The authors evaluated their proposed routing strategy based on delivery chance and average latency, outperforming PROPHET+ in both metrics. However, the lack of up-to-date training data, slow convergence, and low robustness due to the dynamic change of the network topology and its complexity, makes these AI-based routing systems [3]–[6] inefficient. To address this problem, Reinforcement Learning (RL) has been introduced to design more flexible and robust network routing protocols.

kim et al. [7] proposed a novel deep reinforcement learning (DRL)-based routing scheme suitable for SDN environment. In the proposed solution, the DRL agent selects the optimal route ($i.e.$, set of link weights) that minimizes the packet losses and the end-to-end delay of the network. To address the problem of long learning process, the authors developed an M/M/1/K queue-based network model and perform an offline training of DRL. The authors evaluated their proposed routing strategy in terms of the packet losses and the end-to-end delay of the network; outperforming conventional hop-count routing protocols in all these metrics. Younus et al. [8] proposed a novel deep reinforcement learning (DRL)-based routing scheme that includes energy efficiency-based metrics as well as Quality-of-Service (QoS). The authors evaluated their proposed routing strategy in terms of lifetime and packet delivery ratio (PDR); outperforming energy-aware

SDN-based routing schemes in terms of lifetime (from 8% to 33%) and packet delivery ratio (PDR) (from 2% to 24%). Kato et al. [6] proposed a novel routing scheme that uses supervised deep neural network to ensure an effective routing in heterogeneous networks. Their proposed scheme uses the history of the heterogeneous network traffic to route packets in an adaptive manner. More specifically, the scheme takes as input, the values of the number of packets transmitted by each node, and gives as an output the whole path to final destination ($i.e.$, edge router). The proposed routing scheme includes three phases. First, it uses the OSPF protocol to gather network training data. Then, it trains using these data based on greedy layer-wise and back-propagation approaches. Finally, the inference model was used to select the whole routing path. The authors evaluated their proposed routing strategy in terms of signaling throughput, overhead, and average delay per hop, outperforming the OSPF protocol in all these metrics.

Sun et al. [9] have studied the problem of Network Routing in a Software Defined Networking (SDN) environment using deep reinforcement learning (DRL) techniques. They proposed ScaleDeep, a scalable DRL-based routing scheme for SDN that allow for a flexible and robust network routing. The proposed DRL-based algorithm adapts routing policies by automatically adjusting the link weights of the driving nodes. Their experiments results showed that the proposed scheme reduces the flow time completion 36% with better robustness to minor topology changes. Lin et al. [10] have proposed a novel QoS-aware adaptive routing (QAR) scheme based on a multi-layer hierarchical SDNs. The objective of their proposed hierarchical control plane is to minimize signaling delay in large SDNs via three-levels design of controllers. Also, the authors have proposed a novel QAR algorithm that uses reinforcement learning and QoS-aware reward function to achieve an adaptive and time-efficient network routing. The authors evaluated the performance of their proposed scheme through extensive simulation using a Python-based in-house simulator.

Although the implementation of DRL improves the efficiency/flexibility in traditional/AI-based routing schemes, deploying DRL directly in a large-scale network requires a large amount of policy space to exert control over all nodes or routing flows [1], [11], [12]. This makes it difficult for the DRL agent to converge to a stable state and achieve good performance. The existing DRL-based routing schemes ($e.g.$, [7]–[10]) are fully dependent on the network topology. If the topology of a network undergoes a minor change ($e.g.$, the addition or deletion of a link or node), the well-trained DRL agent cannot use its acquired knowledge to serve the new topology. Moreover, the amount of data ($i.e.$, state-action space) shared between agents can consume network bandwidth and may slow down the process of training. In this paper, we design a novel approach that combines Federated Learning (FL) with Deep Reinforcement Learning (DRL) in order to ensure an effective network routing in wireless environments. First, we formalize the problem of network routing as a problem of RL, where multiple agents that are geographically distributed train the policy model in a fully distributed manner. Thus, each agent can quickly obtain the optimal policy that maximizes the cumulative its expected reward, while preserving the privacy of each agent's data. Experiments results confirm that our proposed Federated Reinforcement Learning (FRL) approach is robust and effective under different network conditions.

The remainder of this paper is organized as follows. Section II presents the design and specification of our proposed FRL architecture for wireless networks. Section III presents the performance evaluation of our proposed FRL. At last, section IV concludes the paper.

II. FRL-Enabled Network routing for Wireless Networks

In this section, we introduce the problem statement; then, we detail our proposed solution and the mathematical formulation.

A. Problem Statement

We formalize the problem of network routing as a problem of RL, where multiple agents actively interacting with an environment ($i.e.$, the wireless network) by executing a set of actions ($e.g.$, forward a packet). We assume an architecture as shown in Fig. 1; it shows the system architecture of our considered hierarchical wireless topology network. Our proposed architecture consists of three tiers. In the lower tier ($i.e.$, the first level of the proposed architecture), we consider a set of wireless nodes that are distributed in a particular geographical area. The second level of the proposed architecture is represented by multiple Base Stations (BSs: A, B, C and D) ($i.e.$, Cluster Head (CH)) that conduct local training, where each CH covers a particular geographical area, in which a set of wireless devices are already deployed. Each node needs to make a similar decision task in its observed environment ($i.e.$, select the optimal route) with little interaction with each others. A CH node is also responsible for establishing the communication link between the first and third levels. The upper tier ($i.e.$, the third level of the proposed architecture) is represented by an SDN controller, which is responsible for aggregating local model updates. Our proposed FRL-based algorithm includes two stages. First, the initial route discovery; this stage corresponds to the initial training of RL agents that is inspired by a reactive routing scheme to find the initial route tables during the path discovery process based on Route Request (RR) and Route Reply (RP) messages. Then, the packet routing; this stage corresponds to the normal operation of the routing algorithms, and corresponds to the value estimation function based on RL, as well as the feedback reward from the selected neighboring node.

B. Proposed Solution

In this paper, we aim to combine Federated Learning (FL) with Reinforcement Learning (RL) in order to ensure an effective network routing in wireless environment. There are two common routing protocol techniques for wireless sensor networks (WSNs). The first technique, called reactive, is an

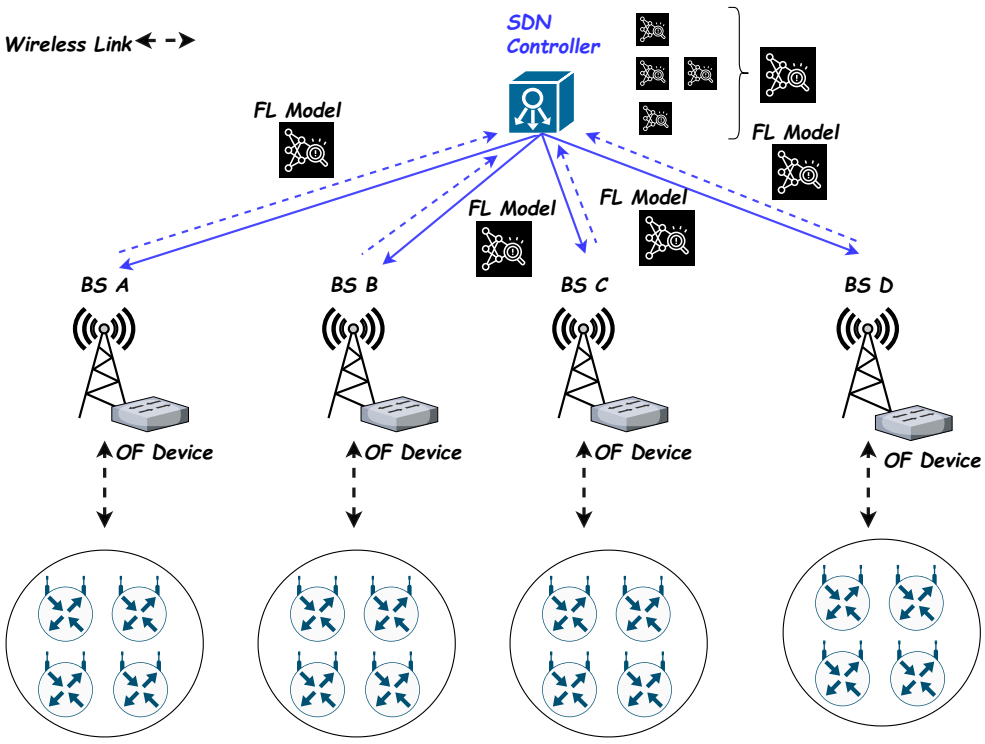


Fig. 1. The considered hierarchical wireless topology network

on-demand strategy, where a network route/path is created upon an initial request from a source node to transmit a data to a destination node. To this aim, a route discovery process ($i.e.$, based on Route Request (RR) and Route Reply (RP) messages) is initiated each time there is a data packet to be transmitted. Once done, the established route became inactive; such routing technique is suitable for WSNs, however it may introduce an additional delay, since the path must be explored for each packet transmission. The second technique, called proactive, is a table-based strategy, where each node in the network has information about all possible routes in its routing table. Each node continuously updates such information by sending control packets (route update). This technique can ensure the flexibility and reduce transmission time delays, in comparison with to the reactive one. However, as the overall size of the network increases, the number of the route update messages and the size of routing tables will grow exponentially, reducing the overall performances ($e.g.$, throughput) of the network. In our proposed Federated Reinforcement Learning-based strategy, we use a hybrid network routing technique, combining reactive and proactive ones to allow for an efficient and flexible network routing for WSNs, this will allow to take benefit of the two techniques. The routing strategy is based on the estimated reward value; in this work, we consider three types of rewards. The first one is a negative reward strategy that is attributed to a particular route upon an unsuccessful transmission. The second one is a low reward strategy; a low reward value is attributed to a particular route via a particular node when this latter sends the ACK and successfully transmits the packet to the destination, but the transmission time is high ($i.e.$, bad route). The third one is a high reward strategy, which means that the selected node/neighbor, had generated the best path towards the destination. In this work, we consider the end-to-end delay (round trip time from the transmitter to the receiver).

C. Mathematical Formulation

In this section, we first present the mathematical formulation of our studied problem. Then, we present our Federated Reinforcement Learning- based Network routing strategy.

1) Markov Decision Process Formulation: A Markov decision process (MDP) is a stochastic discrete process that models decisions on the basis of a mathematical system, characterized by the triplet (S_f, A_f, R_f) where S_f, A_f, and R_f indicate state, action, and the reward function of an agent f, respectively. They are defined as follows: (1) state space: can either be a packet that is successfully delivered to a destination, or the loss of a packet during transmission; (2) action space: An action can be described as the decision ($e.g.$, select next-hop, drop the packet) reached by the agent after taking into account the state of the environment; and (3) reward function: A reward is defined as an incentive mechanism that rewards/punishes an agent based on its performed action. In our work, we have considered three types of rewards. The first one is a negative reward strategy that is attributed to a particular route upon an unsuccessful transmission. More

specifically, when a node sends a packet to the next hop and does not receive an ACK message, we assign a negative reward to the selected route via that node. The second one is a low reward strategy; a low reward value is attributed to a particular route via a particular node when this latter sends the ACK and successfully transmits the packet to the destination, but the transmission time is high ($i.e.$, bad route). The low reward strategy reduces the probability of choosing bad nodes/neighbors. The third one is a high reward strategy, which means that the selected node/neighbor, had generated the best path towards the destination. The transition probability from state s_t to s_{t+1} is expressed as follows:

$$P(s_{t+1}|s_1,\ldots,s_t) = P(s_{t+1}=j|s_t=i) = p_{i,j}, \quad (1)$$

where $\sum_{j=1}^{n} p_{i,j} = 1 \, \forall i = 1,\ldots,n$, where n is the number of states.

We define the policy function as follows:

$$\pi(s|v) = P[A_t = a|S_t = s], \quad (2)$$

where a is the action taken by an agent given a state s.

The agent can evaluate the action v using a state value function, defined as follows:

$$\nu_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1}|S=s], \forall s \in S, \quad (3)$$

where r_{t+1} is the reward value at the time $t+1$ and $0 < \gamma < 1$ is a discount factor.

The agent tries to maximize the Cumulative Reward (CR) defined as follows:

$$CR = \max_\pi \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r|S=s], \quad (4)$$

The Q-learning can be used to estimate the quality of a given state-action pair, as follows:

$$\mathbb{Q}_\pi(s,a) = \sum_{s',r} P[s',r|s,a]\big(r + \gamma\nu(s')\big), \quad (5)$$

where $P[s',r|s,a]$ and r represent the transition probability from state s to state s' and the obtained reward, receptively. The Q-learning algorithm updates the function $\mathbb{Q}$ as follows:

$$\mathbb{Q}(s,a) \leftarrow \mathbb{Q}(s,a) + \alpha\big(r + \gamma \max_a \mathbb{Q}(s',a') - \mathbb{Q}(s,a)\big) \quad (6)$$

where α, γ, s', and a' indicate the learning rate, the discount factor, the next time step state and action taken, respectively.

Q-learning is a model-free algorithm that does not need to know the environment model but estimates the Q-value of each action performed in each state that the user encounters while interacting with the environment. Q-learning uses a Q-table to store expected rewards for a given state, which can be a problem when a large number of states and actions of a single agent or where multiple agents may be involved. DQN can solve this issue; by using multiple neural layers. The agents calculate the Q-vector (Q') as follows:

$$Q = R + \gamma Q'; \quad (7)$$

where R, γ, and (Q) are the agent reward vector, a discount factor, and the targeted Q value, respectively.

The loss function to optimize is defined as:

$$\mathcal{L} = \frac{1}{n}\sum_{i=0}^{n}(Q(s,a) - r + \gamma Q(s',a'))^2, \quad (8)$$

where Q and Q' are the approximated Q-value and the targeted Q-value, respectively.

2) Federated Deep Reinforcement Learning: First, we formalize the problem of network routing as a problem of RL, where multiple agents that are geographically distributed train the policy model in a fully distributed manner. Each agent independently executes routing actions (see Fig.1) based on the current state of their environment and obtains positive (successful transmission) or negative (unsuccessful transmission) rewards for evaluation. In contrast to RL, FRL keeps the data locally, and learns a global model through the shared parameters sent by the agents. This allows each agent to quickly obtain the optimal policy that maximizes its cumulative expected reward, while protecting each agent's privacy. In the FRL problem, we assume that N agents $\{F\}_{i=1}^{N}$ can observe their environment $\{E\}_{i=1}^{N}$ with the objective to ensure an efficient network routing. Let G be the collection of all environments. Each environment $\{E\}_i$ of the i^{th} agent has a similar model $i.e.$, state/action space and reward function. The objective is to find the optimal global parameters of the model as follows:

$$\min_{w_r \in R^d} F(w_r) \quad \text{where} \quad F(w_r) = \frac{1}{N}\sum_{i=1}^{N} F_i(w_r), \quad (9)$$

where N is the number of agents and $F_i(w_r)$ is the local objective function for the i^{th} collaborator at each round r.

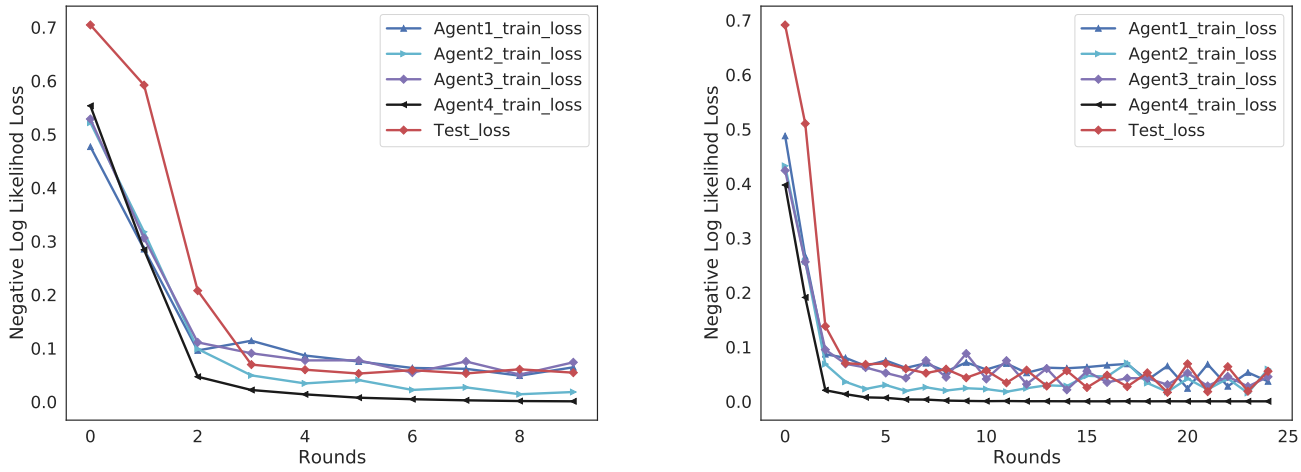
Note that each environment $\{E\}_i$ is independent from the others, in such a way that state/action space and reward function of $\{E\}_i$ do not depend on state/action space and reward function of other environments. Thus, the conditions of FRL is presented as:

$$S_i = S_j, A_i = A_j, E_i \neq E_j \forall i,j \in \{1,2,...,N\}; \quad (10)$$

where S_i and S_j are the same state space experienced by the i^{th} and j^{th} agents, respectively; while A_i and A_j are the same action space experienced by the i^{th} and j^{th} agents, respectively, and E_i and E_j are the different independents environments of the i^{th} and j^{th} agents, respectively.

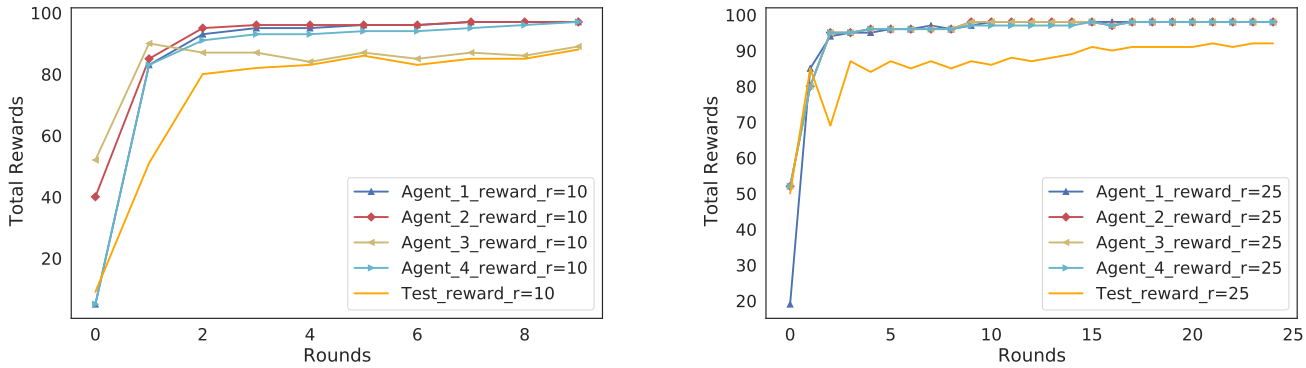
III. Performance Evaluation

The implementation of our proposed FRL is done using Pysyft [13], a generic library for privacy-aware deep learning, built on top of PyTorch. Our test environment consists of multiple agents (see BSs: A, B, C and D in Fig. 1) ($i.e.$, Cluster Head (CH)) that conduct local training, where each CH covers a particular domain, in which a set of wireless devices are already deployed. Each node needs to make a similar decision task in its observed environment ($i.e.$, select



(a) (b)

Fig. 2. Model loss for (a) 10 rounds; and (b) 25 rounds.



(a) (b)

Fig. 3. Total rewards for (a) 10 rounds; and (b) 25 rounds.

the optimal route). We have varied the number of training rounds and local epochs from 10 to 25 and from 1 to 5, respectively. Figs. 2(a) and 2(b) show the training curves of the trained models over the rounds; they show the negative log likelihood loss values, respectively, during training and testing phases. We observe that the agent's loss decreases until it reaches a minimum (almost zero), which indicates that agents are capable of learning from each other while protecting each one privacy. Figs. 3(a) and 3(b) show the convergence of our proposed FRL algorithm over 10 rounds and 25 rounds, respectively. They show the cumulative average rewards per training rounds. We observe that the average reward improves as the training rounds increase, until it reaches the maximum, after only two rounds of federated training.

IV. Conclusion

In this paper, we have designed a novel approach that combines Federated Learning (FL) with Reinforcement Learning (RL) in order to ensure an effective network routing in wireless environments. Our proposed FRL allows multiple agents to collaborate in order to learn a shared policy in a fully distributed manner. Thus, each agent can quickly obtain the optimal policy that maximizes the cumulative expected reward, while preserving the privacy of each agent's data. Results showed that our proposed Federated Reinforcement Learning (FRL) approach is robust and effective while protecting each agent's privacy.

References

[1] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," *IEEE Access*, vol. 7, pp. 55 916–55 950, 2019.

[2] J. Lu, D. He, and Z. Wang, "Secure routing in multihop ad-hoc networks with srr-based reinforcement learning," *IEEE Wireless Communications Letters*, vol. 11, no. 2, pp. 362–366, 2022.
[3] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1946–1960, 2017.
[4] D. K. Sharma, S. K. Dhurandher, I. Woungang, R. K. Srivastava, A. Mohananey, and J. J. P. C. Rodrigues, "A machine learning-based protocol for efficient routing in opportunistic networks," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2207–2213, 2018.
[5] T.-K. Huang, C.-K. Lee, and L.-J. Chen, "Prophet+: An adaptive prophet-based routing protocol for opportunistic network," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 112–119.
[6] N. Kato, Z. M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," *IEEE Wireless Communications*, vol. 24, no. 3, pp. 146–153, 2017.
[7] G. Kim, Y. Kim, and H. Lim, "Deep reinforcement learning-based routing on software-defined networks," *IEEE Access*, vol. 10, pp. 18 121–18 133, 2022.
[8] M. U. Younus, M. K. Khan, and A. R. Bhatti, "Improving the software-defined wireless sensor networks routing performance using reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3495–3508, 2022.
[9] P. Sun, Z. Guo, J. Li, Y. Xu, J. Lan, and Y. Hu, "Enabling scalable routing in software-defined networks with deep reinforcement learning on critical nodes," *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 629–640, 2022.
[10] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *2016 IEEE International Conference on Services Computing (SCC)*, 2016, pp. 25–33.
[11] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: Techniques, applications, and open challenges," *CoRR*, vol. abs/2108.11887, 2021. [Online]. Available: https://arxiv.org/abs/2108.11887
[12] A. Sacco, F. Esposito, and G. Marchetto, "A federated learning approach to routing in challenged sdn-enabled edge networks," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 150–154.
[13] "Pysyft." [Online]. Available: https://github.com/OpenMined/PySyft