Research Article

Stochastic Adaptive Forwarding Strategy Based on Deep Reinforcement Learning for Secure Mobile Video Communications in NDN

Bowei Hao ¹, **Guoyong Wang** ², **Mingchuan Zhang** ¹, **Junlong Zhu** ¹, **Ling Xing** ¹,
and **Qingtao Wu** ¹

¹School of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China

²School of Computer and Information Engineering, Luoyang Institute of Science and Technology, Luoyang 471023, China

Correspondence should be addressed to Mingchuan Zhang; zhang_mch@haust.edu.cn

Received 3 January 2021; Revised 16 March 2021; Accepted 31 March 2021; Published 26 April 2021

Academic Editor: Yuanlong Cao

Copyright © 2021 Bowei Hao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Named Data Networking (NDN) can effectively deal with the rapid development of mobile video services. For NDN, selecting a suitable forwarding interface according to the current network status can improve the efficiency of mobile video communication and can also avoid attacks to improve communication security. For this reason, we propose a stochastic adaptive forwarding strategy based on deep reinforcement learning (SAF-DRL) for secure mobile video communications in NDN. For each available forwarding interface, we introduce the twin delayed deep deterministic policy gradient algorithm to obtain a more robust forwarding strategy. Moreover, we conduct various numerical experiments to validate the performance of SAF-DRL. Compared with BR, RFA, SAF, and AFSndn forwarding strategies, the results show that SAF-DRL can reduce the delivery time and the average number of lost packets to improve the performance of NDN.

1. Introduction

Recently, with the development of technology and the increase of mobile devices, the proportion of mobile video services in network communications has increased rapidly. At the same time, users pay more attention to the acquired video content information and no longer pay attention to its storage location. This brings about huge challenges to the current host-based TCP/IP network architecture [1–3]. Although there are many research studies to improve current network performance [4–6], they have not been reformed in essence. To deal with these challenges, Named Data Networking (NDN) [7] is proposed as one of the most potential candidate network architectures in the future.

Because the routing nodes in NDN have the crucial feature of being cached in network, the efficiency of video communication can be effectively improved in NDN [8]. In addition, by decoupling information content and

location, NDN can well support the mobility of video communications [9]. Delay is one of the main factors affecting the efficiency of video transmission services and Quality of Experience (QoE) of video services. To reduce the delay of the video transmission service, a high-performance adaptive forwarding strategy is necessary for NDN [10]. Although the NDN architecture provides a certain degree of security for network communications [7], an effective adaptive forwarding strategy can prevent video communication from being attacked and find trusted video content [11]. Thus, adaptive forwarding is a key issue for secure mobile video communications. When users retrieve video content, they will send the Interest packet to network. After the router receives the Interest packet, it may find many available forwarding interfaces after querying the Forwarding Information Base (FIB). Therefore, according to the current interface network conditions, adaptively selecting a suitable interface for forwarding can greatly improve the efficiency of video

communications. In addition, Yi et al. [12, 13] proved that the forwarding plane in NDN is stateful, where routing only provides a guiding role for adaptive forwarding. Therefore, designing an effective adaptive forwarding strategy for NDN can greatly improve efficiency of secure mobile video communications.

Adaptive forwarding in NDN is a dynamic and complex process. When the router receives the Interest packet, it can select one or more certain available interfaces for forwarding [14–17]. Due to the selection of certain available interface forwarding, these strategies lack the exploration of unknown links and cannot find better links in time, which may cause network load imbalance. Thus, some researchers propose probability-based adaptive forwarding strategies [18–21]. In these strategies, the Interest packet will stochastically select the interface for forwarding according to the forwarding probability of the available interface. However, these strategies are less robust to emergencies in the network, such as network congestion or link failure. Therefore, assigning a suitable and robust forwarding probability to each available forwarding interface is the main research content of designing stochastic adaptive forwarding strategy in NDN.

In recent years, the rise and development of reinforcement learning theory has brought about new ideas for the design of NDN adaptive forwarding strategies [22]. Reinforcement learning aims to obtain the optimal strategy through independent learning through interaction with the environment [23]. In this paper, we use the advantages of reinforcement learning to design a more suitable and robust stochastic adaptive forwarding strategy in NDN. When a user requests mobile video data in NDN, the user cannot obtain all the mobile video data by sending one Interest packet to the server. It is necessary to continuously send multiple Interest packets with a common prefix to request all the mobile video data. We introduce the twin delayed deep deterministic policy gradient (TD3) [24] algorithm to solve this continuous control problem. Then, we take the throughput, delay, and error rate of each interface as the state of the algorithm, the total utility of the node as the reward function of the algorithm, and the forwarding probability of each interface as the action of the algorithm. Through continuous iterative training, an adaptive forwarding strategy with maximum network utility can be finally obtained.

We summarize our main contributions as follows:

We propose an adaptive forwarding framework based on deep reinforcement learning in NDN. This framework can assign a suitable and robust forwarding probability for each available interface.

We propose a stochastic adaptive forwarding strategy for secure mobile video communications based on deep reinforcement learning (SAF-DRL) which introduces the TD3 algorithm into NDN.

We conduct numerical experiments on the SAF-DRL algorithm under different topologies. By comparison with four other adaptive forwarding strategies, the experimental results show that the SAF-DRL can achieve higher network performance.

The rest of the paper is arranged as follows: Section 2 introduces related work on adaptive forwarding strictly in NDN. Section 3 discusses the system model of adaptive forwarding in NDN. Section 4 presents and describes the SAF-DRL algorithm. Section 5 evaluates the performance of the SAF-DRL. Section 6 summarizes this paper and proposes future work.

2. Related Work

In recent years, NDN has received more attention and there are a large number of researchers studying this field. Many of them are making great efforts to study adaptive forwarding strategies. In this section, we review some typical works.

Yi et al. [15] proposed a framework for adaptive forwarding in NDN networks and also proposed an adaptive forwarding strategy based on the color (GREEN, YELLOW, and RED) of the forwarding interface. After this, adaptive forwarding in NDN is mainly divided into two categories. One is the early use of mathematical optimization methods, the most important of which is the adaptive forwarding strategy based on probability, and the other is the use of reinforcement learning method in recent years through continuous iterative training to get the optimal adaptive forwarding strategy.

Probability-based adaptive forwarding has a large number of documents in the early stage. Qian et al. [18] proposed an adaptive forwarding strategy based on probability. This strategy mainly assigns forwarding probability for each available interface and minimizes the delay of Interest packet transmission on each interface as an optimization goal. The objective function is optimized through the ant colony optimization algorithm to finally achieve the optimal adaptive forwarding strategy. Lei et al. [19] proposed a maximizing deviation based probabilistic forwarding strategy. This forwarding strategy comprehensively considers multiple related attributes of the node and by using the maximizing deviation method assigns the optimal weight to each attribute. On this basis, the comprehensive score of each available forwarding interface can be obtained, and this is taken as the forwarding probability of the forwarding interface. Lei et al. [20] proposed an entropy-based probabilistic forwarding strategy. This forwarding strategy uses the entropy weight theory to assign weights among multiple attributes. The node combines its own performance and the assigned weight to calculate the availability of each available interface and then uses this availability as the forwarding probability of the available interface. Posch et al. [21] proposed stochastic adaptive forwarding (SAF). SAF imitates the water pipe system in reality. It can guide and

distribute Interest packets intelligently in network nodes to avoid link congestion. SAF adopts the overpressure valve, takes the throughput of the link as an important measure, divides the Interest packets into satisfied and unsatisfied, and allocates the forwarding probability of each interface, so that the congested nodes can reduce the pressure independently.

As the advantages of reinforcement learning gradually manifest, some researchers use reinforcement learning to find the optimal adaptive forwarding strategy. Yao et al. [25] proposed an adaptive forwarding strategy called SMDPF. This strategy regards the request forwarding in the network as a Semi-Markov Decision Problem (SMDP). Then, based on SMDP theory and considering the randomness of network requests, an optimal adaptive forwarding strategy is designed to deal with the request forwarding by combining Q-learning with artificial neural network. Akinwande [26] proposed an adaptive forwarding strategy based on reinforcement learning and random neural network. Based on the dynamic self-awareness strategy layer, the strategy can reply to the request content quickly through local Content Store (CS). At the same time, it uses *probe* Interest packet and combines it with local routing information to actively seek new available delivery path under controllable degree. Zhang et al. [27] proposed an intelligent forwarding strategy using reinforcement learning. The strategy does not rely on the model programmed in advance but trains a neural network model to select the interface by collecting information from nodes. At the same time, it only learns new decisions by observing the results of past decisions. Zhang et al. [28] proposed an adaptive forwarding strategy using improved Q-learning. The strategy is mainly divided into two phases, exploration and exploitation. In the exploration phase, the information in the network is collected, and then the information is used as the basis to guide forwarding of Interest packets in the exploitation phase.

Probability-based adaptive forwarding can greatly reduce the resource waste caused by deterministically selecting one or more interfaces in mobile video communications. At the same time, it can avoid attacks due to the selection of certain interfaces, thereby improving security. However, adaptive forwarding based on reinforcement learning has higher robustness, especially in the case of link failures. We use the advantages of both and use reinforcement learning to train the forwarding probability assigned to each available forwarding interface, finally obtaining the adaptive forwarding strategy with high robustness in NDN.

3. System Model

In this section, we introduce the system model in detail. We summarize the major variables and expressions, which are depicted in Table 1.

We use a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ to depict the network model. The directed graph consists of two parts, the set of nodes $\mathcal{V}$ and the set of links $\mathcal{E}$, where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. For each node $i \in \mathcal{V}$, it may maintain several physical interfaces $F_{(i,j)}$, where j is the neighbor node of node i and the tuple $(i, j) \in \mathcal{E}$. We define $\mathcal{F}_i$ as the set of all physical interfaces with node i , $\mathcal{F}_i = \cup_{(i,j) \in \mathcal{E}} F_{(i,j)}$, and N_i is the number of the

interfaces for node i , $N_i = |\mathcal{F}_i|$. For node i , we define $F_{i+} \in \mathcal{F}_i$ as the set of in-interfaces which receive the Interest packet and $F_{i-} \in \mathcal{F}_i \setminus F_{i+}$ as the set of out-interfaces which return the requested data packet or forward the Interest packet to next hop node searched in the FIB. $F_L \notin \mathcal{F}_i$ is the lose-interface, where the Interest packet needs to be lost. As above, we want to learn an adaptive forwarding (AF) strategy for each node. Since the algorithm proposed in this paper only needs local information to train the AF strategy, the algorithm is installed on each node, and no communication is required between the nodes. We only focus on a single node, so we will omit the subscript for the node in the next discussion.

The content catalogue of this system can be labeled as a set $\mathcal{K}$. For $k \in \mathcal{K}$ represents the content with the common prefix requested by the user, we define $p_k(F_f)$ as the forwarding probability for the interface F_f with the common prefix k and $p_k(F_L)$ as the packet loss rate with prefix k when network congestion or link fails. In this paper, we mainly focus on a common prefix, so we want to omit subscripts for prefix in the remainder of the paper, and we will consider different prefixes in our future work. We show the system model in Figure 1. There are many interfaces for Interest packet to AF for a node in this system. For example, the mobile device (User1 and/or User2) wants to obtain the video /pre1/pre2/n1.mp4, stored in server (Server1 or/and Server2). The mobile device sends Interest packet to router R1. The router R1 has two interfaces (F_1 and F_2) to forward the Interest packet by probability of 2/3 for F_1 and 1/3 for F_2 (confirm forwarding). Then forwarding continues until the Interest packet encounters the requested content. Finally, the data packet with video is returned to the mobile device via the reverse path.

According to the α -fairness [29, 30] model widely used for Network Utility Maximization (NUM), the utility function is defined as

$$U_\alpha(x) = \begin{cases} \log x, & \text{if } \alpha = 1, \\ \frac{x^{1-\alpha}}{1-\alpha}, & \text{otherwise,} \end{cases} \quad (1)$$

where α is positive numbers and fairness tuning parameter. For $\alpha > 0$, the function is strictly nondecreasing. If $\alpha = 1$, the function is proportional fairness and is widely used in NUM. Similar to [30], we define the utility function of interface $f \in \mathcal{F}$ as

$$U(x_f, y_f, z_f) = U_{\alpha_1}(x_f) - \beta U_{\alpha_2}(y_f) - \gamma U_{\alpha_3}(z_f), \quad (2)$$

where x_f , y_f , and z_f are represented as the throughput, delay, and error rate of the f th interface, respectively; β and γ represent the relative importance of the throughput versus delay versus error rate and $\beta, \gamma \in [0, 1]$. Especially if Interest packet has to be discarded, the corresponding utility is defined as 0; that is, $U(x_L, y_L, z_L) = 0$, where x_L, y_L, z_L , respectively, represent the throughput, delay, and error rate of the lose-interface F_L . For the utility function proportional fairness, we set $\alpha_1 = \alpha_2 = \alpha_3 = 1$. The utility function becomes

TABLE 1: Variables and expressions.

Variable	Expression
N	Number of physical interfaces.
F_+, F_-	The set of in-interfaces and out-interfaces.
F_L	The lose-interface, where the Interest packet needs to be lost.
$p(F_f)$	Probability of forwarding for interface F_f .
p_L	Interest packet loss rate.
x_f, y_f, z_f	Throughput, delay, and error rate of the interface $f \in \mathcal{F}$.
$U(x_f, y_f, z_f)$	The network utility of interface $f \in \mathcal{F}$.
x_L, y_L, z_L	Throughput, delay, and error rate of the lose-interface F_L .
$U(x_L, y_L, z_L)$	The network utility of lose-interface F_L .
U_{all}	The total utility of each node.

FIGURE 1: The system model.

$$U(x_f, y_f, z_f) = \log x_f - \beta \log y_f - \gamma \log z_f. \quad (3)$$

According to [31], the total utility of each node in the network can be expressed as

$$U_{\text{all}} = \sum_{f=1}^N p(F_f) U(x_f, y_f, z_f) + p(F_L) U(x_L, y_L, z_L). \quad (4)$$

Therefore, the objective of optimizing the AF strategy is to maximize the total network utility U_{all} .

4. Stochastic Adaptive Forwarding Based on Deep Reinforcement Learning

In this section, we first introduce the background knowledge about the TD3 algorithm and then propose the SAF-DRL algorithm and describe the algorithm in detail.

4.1. Background of TD3. In this subsection, in order to better understand the SAF-DRL algorithm, we will introduce some necessary background knowledge about the TD3 algorithm.

For a standard reinforcement learning (RL), the training process is to interact with the environment during each decision epoch and finally obtain the optimal strategy gradually. The specific process is that, at epoch t , the agent observes a state $\mathbf{s}_t$ and selects an action $\mathbf{a}_t$ according to this state. After execution, the environment state will convert from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$, and, at the same time, the single-epoch reward $r(\mathbf{s}_t, \mathbf{a}_t)$ given by the environment will be obtained. The goal of reinforcement learning is to find an optimal policy π^* to maximize the discounted future reward $R_t = \sum_{k=t}^T \gamma^{k-t} r(\mathbf{s}_k, \mathbf{a}_k)$, where $\gamma \in [0, 1]$ is a discounted factor. At epoch k , for RL, the goal is to require an optimal policy π_ω under parameterization ω to maximize the expected reward,

$$J(\omega) = \mathbb{E}_{\mathbf{s}_k \sim p_{\pi, \mathbf{a}_k} \sim \pi_\omega} (R_t), \quad (5)$$

where p_π is the sampling space of the state.

In order to learn the problem of continuous decision-making, Sutton et al. [32] proposed the policy gradient (PG) method. This method uses a probability distribution function $\mathbb{P}(\mathbf{a}|\mathbf{s}; \omega)$ to represent the optimal policy for each epoch and performs action sampling according to the policy distribution at each epoch to obtain the best action value: $\pi_\omega(\mathbf{a}|\mathbf{s}) = \mathbb{P}(\mathbf{a}|\mathbf{s}; \omega)$. We can use the gradient of reward $\nabla J(\omega)$ to update the parameterized strategy π_ω . Therefore, we can get

$$\nabla_\omega J = \mathbb{E}_{\mathbf{s} \sim p_\pi, \mathbf{a} \sim \pi_\omega} [\nabla_\omega \log \pi_\omega(\mathbf{a}|\mathbf{s}) Q^\pi(\mathbf{s}, \mathbf{a})], \quad (6)$$

where, according to the expected return given by equation (5), $Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}_k \sim p_\pi, \mathbf{a}_k \sim \pi_\omega} (R_t | \mathbf{s}, \mathbf{a})$. Since the process of generating actions is a stochastic process, the last strategy learned is also a stochastic policy. However, since the action space is usually a high-dimensional vector, frequent sampling in the high-dimensional action space is an extremely computationally consuming task. Therefore, Silver et al. [33] proposed deterministic policy gradient (DPG). For each epoch of the action, the determined value is directly obtained through the function $\mu, \mathbf{a} = \mu_\omega(\mathbf{s})$. Therefore, we can get

$$\nabla_\omega J = \mathbb{E}_{\mathbf{s} \sim p_\pi} [\nabla_\omega \mu_\omega(\mathbf{s}) \nabla Q^\mu(\mathbf{s}, \mathbf{a}) |_{\mathbf{a}=\mu_\omega(\mathbf{s})}]. \quad (7)$$

In order to deal with high-dimensional input problems, DeepMind introduced deep learning into Q-learning and proposed Deep Q-Network (DQN) [34]. At the epoch k , DQN uses a greedy strategy $\pi(\mathbf{s}_k) = \text{argmax}_{\mathbf{a}_k} Q(\mathbf{s}_k, \mathbf{a}_k)$ and then trains by minimizing loss,

$$\text{Loss}(\omega^Q) = \mathbb{E} [\text{Tar}_k - Q(\mathbf{s}_k, \mathbf{a}_k | \omega^Q)], \quad (8)$$

where Tar_k is the target Q value, which can be expressed as

$$\text{Tar}_k = r(\mathbf{s}_k, \mathbf{a}_k) + \gamma Q(\mathbf{s}_{k+1}, \pi(\mathbf{s}_{k+1} | \omega^\pi) | \omega^Q). \quad (9)$$

For continuous control problems, the actor-critic method [35] is often used to solve the problem. This method usually uses the policy gradient method to find the optimal policy. DeepMind combined DQN and DPG and proposed a new actor-critic method, deep deterministic policy gradient (DDPG) [36]. The training of critic network in DDPG is based on the DQN method, and the training of actor network is based on the DRL method. The specific update formula is

$$\nabla_{\omega^\mu} J = \mathbb{E} \left[\nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a} | \omega^Q) \Big|_{\mathbf{s}=\mathbf{s}_k, \mathbf{a}=\mu(\mathbf{s}_k)} \cdot \nabla_{\omega^\mu} \mu(\mathbf{s} | \omega^\mu) \Big|_{\mathbf{s}=\mathbf{s}_k} \right]. \quad (10)$$

Although DDPG has achieved certain success, there are still problems such as overestimation of Q value and excessive variance. Therefore, Fujimoto et al. [24] proposed an improved DDPG algorithm TD3. The first point of improvement is to eliminate the problem of overfitting. TD3 introduces the idea of Double DQN [37] and uses two critic networks. In the practice of actor-critic algorithm, due to the high similarity between the current network and the target network, independent evaluation cannot be made. In order to solve this problem, Double DQN uses this update method:

$$\begin{aligned} y_1 &= r + \eta \hat{Q}(s', \pi(s' | \omega^{\phi_1}) | \hat{\omega}_1^q), \\ y_2 &= r + \eta \hat{Q}(s', \pi(s' | \omega^{\phi_2}) | \hat{\omega}_2^q), \end{aligned} \quad (11)$$

where $\pi(\cdot | \omega^{\phi_1})$ and $\pi(\cdot | \omega^{\phi_2})$ are optimized with $\hat{Q}(\cdot | \hat{\omega}_1^q)$ and $\hat{Q}(\cdot | \hat{\omega}_2^q)$, respectively. After practice, it is found that the final effects of the two actor networks are relatively similar, so one actor network is selected. The update goals of both critic networks are the same, $y_2 = y_1$. Because actor network will choose high evaluations, the overestimated ones will gradually accumulate, so when choosing a smaller evaluation, the final goal can be expressed as

$$y_1 = r + \eta \min_{i=1,2} \hat{Q}(s', \pi(s' | \omega^{\phi_i}) | \hat{\omega}_i^q). \quad (12)$$

In order to solve the problem of excessive variance, TD3 introduced the idea of delaying strategy updates. This is to adjust the update frequency of critic network to be a bit higher than that of actor network. Solve the problem that the DQN is constantly updated, which may cause blind iteration of actor network. When calculating the Q function, a certain action is randomly selected within a certain range to achieve smooth strategy, so as to get rid of the influence of false peaks. In this way, the problem of incorrect strategy caused by inaccurate Q function in DDPG can be solved.

4.2. SAF-DRL Algorithm. In this subsection, we present the stochastic adaptive forwarding strategy based on DRL (SAF-DRL) for secure mobile video communications in NDN. For all DRL approaches, the state space, the action space, and the reward function are important components:

STATE: the state space mainly consists of three parts: throughput, delay, and error rate of each interface in the network. Therefore, the state at epoch t is $\mathbf{s}_t = [(x_t^1, y_t^1, z_t^1), (x_t^2, y_t^2, z_t^2), \dots, (x_t^N, y_t^N, z_t^N)]$.

ACTION: the action is defined as the adaptive forwarding probability of each interface and the Interest packet loss rate p_L for specific content. Therefore, the action at epoch t is $\mathbf{a}_t = [(p_t(F_1)), (p_t(F_2)), \dots, (p_t(F_N)), (p_t(F_L))]$, where $p_t \in [0, 1]$ and $\sum_{j=1}^N p_t(F_j) + p_t(F_L) = 1$.

REWARD: the reward function is the objective of AF strategy, which is the total utility of each node in the network. Formally, $r_t = \sum_{f=1}^N p_t(F_f) U_t(x_f, y_f, z_f) + p_t(F_L) U_t(x_L, y_L, z_L)$.

Please note that the design of the state space, action space, and reward function will seriously affect the performance of the SAF-DRL algorithm. Our design is based on the full consideration of mobile video communications and can be closer to the real situation. Moreover, the probability selection interface can improve the security of mobile video communications, especially when encountering link failures or congestion; and reinforcement learning shows higher robustness than mathematical calculations, thus ensuring the performance of mobile video communications.

Since each node receives a large number of Interest packets to request mobile video content, we assume that the

Interest packets received are continuous. Nodes need to continuously make forwarding decisions for these Interest packets. At the same time, in order to make the forwarding probability assigned to each available interface in mobile video communication have high robustness, we use the TD3 algorithm in the actor-critic algorithm. In addition, as the TD3 algorithm is also a deep reinforcement learning [34] algorithm, it can deal with the high-dimensional problems caused by a large number of entries in FIB in reality.

The situation faced by all interfaces of the router constitutes the training environment of DRL Agent. We collect the information of throughput, delay, and error rate for each interface, which constitutes the Agent's state space. This information is mainly composed of local storage in the forwarding process and does not communicate with other nodes in the network. The Interest packet forwarding probability of each interface in the node constitutes the action space of the Agent. Then actions are performed to update the forwarding probability of the interface, and finally a more suitable and robust forwarding probability is got by training. The suitable and robust forwarding probability on the interface can effectively improve the forwarding efficiency, thereby improving the overall performance of the node. As for the reward, it has been discussed in Section 3. We show the detailed framework of SAF-DRL in Figure 2. For the available forwarding interface of the node, we collect the local information of throughput, delay, and loss rate as the states. Then the Agent uses the TD3 method to train the collected information. Through training, we get an AF strategy that maximizes rewards.

We propose the SAF-DRL algorithm as Algorithm 1. First the algorithm initializes two critic networks $Q(s, a|\omega_1^q)$ and $Q(s, a|\omega_2^q)$ and one actor network $\mu(s|\omega^u)$ (line 1), and the parameters ω_1^q , ω_2^q , and ω^u are the weights of two critic networks and the actor network, respectively. The corresponding target networks are initialized as $Q(s, a|\omega_1^q)$, $Q(s, a|\omega_2^q)$, and $\mu(s|\omega^u)$ (line 2). The parameters $\hat{\omega}_1^q$, $\hat{\omega}_2^q$, and $\hat{\omega}^u$ are the weights of the target network, and their values are defined as ω_1^q , ω_2^q , and ω^u . In order to enable the target network to be updated slowly, a delayed update method is used. On the basis of soft update, the target networks are updated with the update rate ϕ after every d critic update (line 16). We use the uniform distribution $U(0, |F_-|)$ to initialize forwarding probability of each available forwarding interface, $p(F_-) = (1/|F_-|)$, where $|F_-|$ is the number of the out-interfaces, and we initialize the Interest packet loss rate to 0, $p(F_L) = 0$ (line 4).

We use replay buffer B to store the transition samples and we initialize it in line 3. We first store the experience gained through interaction with the environment in B (lines 6–9) and then we train the actor network and critic network according to sample M random transition samples from B (lines 10–16). The function of the $\text{clip}(\mathcal{N}(0, \hat{\sigma}), -k, k)$ is that the value range of $\mathcal{N}(0, \hat{\sigma})$ is limited between $-k$ and k . When the value is less than $-k$, the value is equal to $-k$, and when the value is greater than k , the value is equal to k (line 11). Calculate the minimum Q value of the two critic networks (line 12) and compute the critic update by minimizing commonly used mean square error loss (line 13). The computation of the actor network update uses the DPG approach [33] (line 15). According to the SAF-DRL

algorithm, we can get an optimal Interest packet forwarding probability for each available interface in the node. Finally, we can get an adaptive forwarding strategy that maximizes network utility.

5. Numerical Experiment

We conducted numerical experiments on SAF-DRL method in the NDN environment. The node has certain computing capabilities and can adaptively forward requests for users. At the same time, the node can move to a certain extent, and the main content requested is the video service. Thus, we use one node in the network as an agent to study. Then, a comparison is performed with the other four adaptive forwarding strategies in terms of delivery time, the average number of lost packets, load balancing factor, and hop count.

We use the Python language to generate three topologies based on Erdős-Rényi (ER) [38] model, Barabási-Albert (BA) [39] model, and random model, as shown in Figure 3. Each topology is composed of 100 nodes. The possibility of creating links between two nodes in ER topology is set to 0.08, one edge is added between two nodes in BA topology at a time, and the distance threshold between two nodes in the random topology is set to 0.15. For each link of the three topologies, we allocate bandwidth of 1 Mbps to 5 Mbps, and the delay of each link is within [10 ms, 30 ms]. We randomly selected 10 nodes in the network as the users, 10 nodes as servers, and the rest as routers. In order to better study the adaptive forwarding strategy, we set the CS capacity on each node to 0. We set $\beta = \gamma = 1$ to balance the importance of throughput, delay, and error rate. The experiment has gone through 1500 time-slots' iterative training every cycle, and finally we take the average value through multiple experiments. We run and train SAF-DRL algorithm on Windows 10 operating system with Intel (R) Core (TM) i5 2.4 Ghz CPU with 8 GB memory.

To explore the reward convergence of different agents under different network topologies, we adopted three topologies of ER topology, BA topology, and random topology and performed experiments on five agents (1, 10, 30, 50, and 70) on each topology. In Figure 4, we can see that, under the same topology, although all agents converge at different speeds, they eventually converge to approximately the same stable value. Because there are certain differences among the three topologies (such as the degree of connectivity), the final stable convergence values obtained by the different topologies are not completely equal, but the difference among the convergence values is very small. This proves that our scheme is convergent and can be used in different topologies. Therefore, in the follow-up experiments, we only explore the comparative experiments under ER topology, and the trend is the same in other topologies.

In order to evaluate the performance of our algorithm in many ways, we compare our SAF-DRL algorithm with four other adaptive forwarding strategies:

BestRoute (BR) [14]: interest packets are forwarded through the available interface with the lowest cost (e.g., hop count).

```

1 Randomly initialize critic network  $Q(s, a|\omega_1^q)$ ,  $Q(s, a|\omega_2^q)$  and actor network  $\mu(s|\omega^u)$  with random parameters  $\omega_1^q$ ,  $\omega_2^q$  and  $\omega^u$  respectively;
2 Initialize target critic network  $\hat{Q}(s, a|\hat{\omega}_1^q)$ ,  $\hat{Q}(s, a|\hat{\omega}_2^q)$  and target actor network  $\hat{\mu}(s|\hat{\omega}^u)$  with parameters  $\hat{\omega}_1^q = \omega_1^q$ ,  $\hat{\omega}_2^q = \omega_2^q$  and  $\hat{\omega}^u = \omega^u$  respectively;
3 Initialize replay buffer  $B$ ;
4 Initialize the forwarding probability of the out-interface  $F_-$ , and  $p(F_-) = 0$ ;
5 Receive the observed initial state  $s_1$ ; /** Decision Epoch */
6 for  $t = 1$  to  $T$  do
7 Obtain the action  $a_t = \mu(s_t|\omega^u) + \mathcal{N}_t$  by the current policy  $\mu(s_t|\omega^u)$  and exploration noise  $\mathcal{N}_t \sim \mathcal{N}(0, \sigma)$ ;
8 Execute action  $a$ , receive reward  $r$  and observe next state  $s$ ;
9 Store transition sample  $(s, a, r, s')$  in  $B$ ; /** Training Transition Sampling */
10 Sample a mini-batch of  $M$  transition  $(s, a, r, s')$  from  $B$ ;
11 Execute action  $\hat{a} = \hat{\mu}(s|\hat{\omega}^u) + \mathcal{N}$ , where the  $\mathcal{N} \sim \text{clip}(\mathcal{N}(0, \hat{\sigma}), -k, k)$ ;
12 Compute the Q value of critic network:  $y = \begin{cases} r, & \text{if } j = M \\ r + \eta \min_{i=1,2} \hat{Q}(s', \hat{a}|\hat{\omega}_i^q), & \text{otherwise} \end{cases}$ 
13 Update the critic  $\omega_i^q$  by minimizing the loss:  $\text{Loss} = \min_{\omega_i^q} (1/M) \sum (y - Q(s, a|\omega_i^q))^2$ 
14 if  $t \bmod d$  then
15 Compute the actor update by policy gradient:
 $\nabla_{\omega^u} J = (1/M) \sum \nabla_{\omega^u} \mu(s) \cdot \nabla_a Q(s, a|\omega_i^q)|_{a=\mu(s)}$ ;
/** Target Network Update */
16 Update the parameters of the target networks:
 $\hat{\omega}_i^q \leftarrow \phi \omega_i^q + (1 - \phi) \hat{\omega}_i^q$ ;
 $\hat{\omega}^u \leftarrow \phi \omega^u + (1 - \phi) \hat{\omega}^u$ ;
17 end if
18 end for

```

ALGORITHM 1: SAF-DRL for secure mobile video communications in NDN.

FIGURE 2: The framework of SAF-DRL.

Stochastic Adaptive Forwarding (SAF) [21]: interest packets are forwarded through the interface with the largest throughput-based measurement.

Adaptive Forwarding Strategy in NDN (AFSndn) [28]: interest packets are forwarded through the interface with the lowest delay.

FIGURE 3: The three different network topologies. (a) The ER topology. (b) The BA topology. (c) The random topology.

FIGURE 4: Continued.

FIGURE 4: Main theoretical results under different topologies. (a) The local reward of different agents (1, 10, 30, 50, and 70) under the ER topology. (b) The local reward of different agents under the BA topology. (c) The local reward of different agents under the random topology.

Request Forwarding Algorithm (RFA) [40]: interest packets are forwarded through the interface with the least count of pending Interest packets.

We mainly compare our algorithm with other algorithms in four aspects.

5.1. Delivery Time. The delivery time is mainly the average time it takes for the Interest packet to find the specific content it requests. The delivery time can be specifically defined as

$$D = \frac{\sum_{i=1}^K (\text{get}_i - \text{send}_i)}{K}. \quad (13)$$

Here, send_i represents the moment when the i -th Interest packet is sent, get_i represents the moment when the target node receives the i -th Interest packet, and K represents the total number of Interest packets requested.

5.2. The Average Number of Lost Packets. The average number of lost packets indicates the average number of Interest packets lost due to other reasons (e.g., not finding the target node or network congestion) during all episodes. The average number of lost packets can be specifically defined as

$$\text{Lost} = \frac{\sum_{i=1}^T \text{lost}_i}{T}. \quad (14)$$

Here, lost_i represents the number of Interest packets lost in the i -th episode and T represents the number of episodes.

5.3. Load Balancing Factor. The load balancing factor represents the degree of dispersion of the number of Interest packets forwarded by each node in the network. We use

coefficient of variation for calculation, so the load balancing factor can be specifically defined as

$$Cv(\text{NI}) = \frac{\text{stdev}[\text{NI}(v)]}{\mathbb{E}[\text{NI}(v)]}. \quad (15)$$

Here, $\text{NI}(v)$ represents the number of Interest packets forwarded by the v node and $\text{stdev}[\text{NI}(v)]$ represents the standard deviation of $\text{NI}(v)$.

5.4. Hop Count. The hop count represents the average number of hops experienced by all Interest packets when they find their target node. The hop count can be specifically defined as

$$H_c = \frac{\sum_{i=1}^K h_i}{K}. \quad (16)$$

Here, h_i represents the number of hops taken by the i -th Interest packet before finding the target node.

We describe in detail the performance of each aspect as follows.

5.4.1. Delivery Time. From Figure 5, we can see the delivery time of the five algorithms under different bandwidth (1 Mbps, 3 Mbps, and 5 Mbps). The delivery time of SAF-DRL is lower than the delivery times of the other four algorithms. This is because the SAF-DRL algorithm uses delay as one type of the link status information, and delay is also one of the indicators of the reward function, which can minimize the delay. Therefore, the delay of SAF-DRL is lower than those of SAF, RFA, and BR. Although the AFSndn algorithm also considers the delay information as the indicator of the reward function, the AFSndn algorithm needs to spend a certain amount of time in the early stage of forwarding for exploration. At the same time, because the

FIGURE 5: Comparison of delivery time with four other algorithms in different bandwidths (1 Mbps, 3 Mbps, and 5 Mbps).

FIGURE 6: Comparison of the average number of lost packets with four other algorithms in different LFR (10%, 30%, 50%, and 70%).

AFSndn algorithm uses Q-learning in reinforcement learning and Q-learning has certain limitations in processing high-dimensional data, it will increase the delivery time to be higher than that of our SAF-DRL algorithm as the number of entries in the FIB increases. Because the BR algorithm selects a link for forwarding, causing network congestion to exceed other algorithms, its delivery time becomes the longest. The RFA algorithm can avoid link congestion through load balancing, which can reduce the delivery time to a certain extent.

5.4.2. The Average Number of Lost Packets. Figure 6 shows the average number of lost packets of the five algorithms under different Link Failure Rate (LFR) (10%, 30%, 50%, and 70%). As can be seen from the figure, with the gradual increase of LFR, the average number of lost packets of the five algorithms is increasing. But SAF-DRL algorithm has always had a lower average number of lost packets, among which the BR algorithm has the largest average number of lost packets. This is because the BR algorithm uses the least hop count as the forwarding basis, which may cause network congestion and eventually may discard a large number of Interest packets. The RFA algorithm only uses the count of pending Interest packets as the basis for forwarding probability. Although network congestion can be avoided as much as possible, interfaces with a small number of pending Interest packets may have poor link status, so the number of lost Interest packets is only lower than that in the BR algorithm. The SAF algorithm considers information such as link throughput and can select an effective interface for forwarding, thereby reducing the number of lost packets. The AFSndn algorithm is more robust than the SAF algorithm through reinforcement learning training. However, due to the large number of Interest packets sent in the previous exploration phase, only a few are effective, and a large number of unused Interest packets are discarded. The SAF-DRL algorithm has relatively the lowest average number of lost packets, because it considers multiple different types of link state information, and training through reinforcement learning has high robustness. At the same time, each interface can be assigned a higher efficiency forwarding probability, which reduces the average number of lost Interest packets.

5.4.3. Load Balancing Factor. Figure 7 shows the results of the load balancing factor under different LFR. It can be seen from the figure that, in the case of four link failures, the SAF-DRL algorithm has a lower load balancing factor. When the user retrieves the content, the BR algorithm only considers the shortest path for forwarding. With the increase of LFR, the congestion on this link intensifies, and then the resources on other links are idle, making the network load unbalanced and the load balancing ability is poor. The SAF algorithm selects links for forwarding. After the Interest packet cannot be satisfied, the SAF algorithm will distribute the traffic on the failed link to other links according to throughput-based measure, which can appropriately improve the load capacity of the network. AFSndn is based on the information in the early exploration phase, and when guiding the forwarding of Interest packets, it tries to avoid network congestion, ensuring the load capacity of the network. Compared with the SAF algorithm and AFSndn algorithm, the SAF-DRL algorithm is more robust due to the reinforcement learning training, which makes the forwarding probability distribution allocated on each available forwarding interface more robust, especially when the link fails. The RFA algorithm has the lowest load balancing factor, because RFA algorithm uses the count of pending Interest packets as the reference basis for forwarding. The count of pending Interest packets can

FIGURE 7: Comparison of load balancing factor with four other algorithms in different LFR (10%, 30%, 50%, and 70%).

FIGURE 8: Comparison of hop count with four other algorithms in different LFR (10%, 30%, 50%, and 70%).

reflect the load status in a period of time in the future, so RFA algorithm can effectively balance the load with the lowest load balancing factor.

5.4.4. Hop Count. Figure 8 shows the results of the average hop count. It shows that the average hop count of the BR algorithm is the lowest, and the average hop count of the RFA algorithm is the highest. The average hop counts of the SAF algorithm, AFSndn algorithm, and SAF-DRL algorithm

are between the two, and the average hop count of AFSndn algorithm is higher than those of SAF algorithm and SAF-DRL algorithm. This is mainly because the BR algorithm mainly considers forwarding through the least hop count and does not consider other indicators, so the average hop count of the BR algorithm is the lowest. However, the RFA algorithm only considers the count of pending Interest packets and does not care about delaying this information, so its hop count is the highest. As for the AFSndn algorithm, in the early exploration phase, Interest packets will be

forwarded through all available interfaces, and there are a certain number of links with very long paths, which leads to higher average hop count. Since the SAF algorithm and the SAF-DRL algorithm are adaptive forwarding strategies based on probability, the forwarding probability of selecting a link with better performance is greater, but the link with better performance is not necessarily the shortest. The SAF algorithm only uses throughput as the measure. However, the SAF-DRL algorithm takes delay and other information into account, which is equivalent to considering the length of the link to a certain extent, so that it can find the target node faster with fewer hops.

6. Conclusion and Future Work

In this paper, we have proposed stochastic adaptive forwarding strategy based on deep reinforcement learning (SAF-DRL), a novel adaptive forwarding strategy for secure mobile video communications in NDN. SAF-DRL can forward each Interest packet with a common prefix according to the forwarding probability. To obtain a more robust forwarding probability on each available interface, we have also introduced the twin delayed deep deterministic policy gradient to NDN for adaptive forwarding. Through numerical experiments, the results have shown that SAF-DRL algorithm can achieve final stability under ER topology, BA topology, and random topology. Compared with BR, RFA, SAF, and AFSdn, SAF-DRL has obvious advantages in delivery time and the average number of lost packets. Since we only considered the same video prefix in this paper, in future work, we will consider the priority between different video content prefixes requested by mobile devices; and different applications require different weights for different status information. For example, the transmission of live broadcast service requires lower delay. We will combine the content priority and the weight of the interface status to improve the security and efficiency of mobile video communications.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant nos. 61971458 and 61976243, in part by the Leading Talents of Science and Technology in the Central Plain of China under Grant no. 214200510012, in part by the basic research projects in the University of Henan Province under Grant no. 19zx010, and by the Key Project of the Education Department of Henan Province under Grant no. 20A520011.

References

- [1] M. Zhang, Y. Zhou, W. Quan, J. Zhu, R. Zheng, and Q. Wu, "Online learning for iot optimization: a frank-wolfe adam-based algorithm," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8228–8237, 2020.
- [2] F. Song, Y.-T. Zhou, L. Chang, and H.-K. Zhang, "Modeling space-terrestrial integrated networks with smart collaborative theory," *IEEE Network*, vol. 33, no. 1, pp. 51–57, 2019.
- [3] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, V. C. M. Leung, and MASM, "MASM: a multiple-algorithm service model for energy-delay optimization in edge artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4216–4224, 2019.
- [4] F. Song, Z. Ai, Y. Zhou, I. You, K.-K. R. Choo, and H. Zhang, "Smart collaborative automation for receive buffer control in multipath industrial networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1385–1394, 2020.
- [5] C. Xu, J. Zhu, and D. O. Wu, "Decentralized online learning methods based on weight-balancing over time-varying digraphs," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–13, 2018.
- [6] F. Song, Y.-T. Zhou, Y. Wang, T.-M. Zhao, I. You, and H.-K. Zhang, "Smart collaborative distribution for privacy enhancement in moving target defense," *Information Sciences*, vol. 479, pp. 593–606, 2019.
- [7] L. Zhang, A. Afanasyev, J. Burke et al., "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.
- [8] M. Zhang, B. Hao, F. Song, M. Yang, J. Zhu, and Q. Wu, "Smart collaborative video caching for energy efficiency in cognitive content centric networks," *Journal of Network and Computer Applications*, vol. 158, p. 102587, 2020.
- [9] M. Meddeb, A. Dhraief, A. Belghith et al., "AFIRM: adaptive forwarding based link recovery for mobility support in NDN/IoT networks," *Future Generation Computer Systems*, vol. 87, pp. 351–363, 2018.
- [10] Y. Ye, B. A. Lee, R. Flynn, N. Murray, and Y. Qiao, "HLAF: heterogeneous-latency adaptive forwarding strategy for peer-assisted video streaming in NDN," in *Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC)*, pp. 657–662, Heraklion, Greece, July 2017.
- [11] Z. Rezaeifar, J. Wang, H. Oh, S.-B. Lee, and J. Hur, "A reliable adaptive forwarding approach in named data networking," *Future Generation Computer Systems*, vol. 96, pp. 538–551, 2019.
- [12] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.
- [13] C. Yi, J. P. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in named data networking," in *Proceeding of the 1st International Conference on Information-Centric Networking (ICN)*, pp. 27–36, Paris, France, August 2014.
- [14] A. Afanasyev, J. Shi, B. Zhang et al., "NFD developer's guide," <https://named-data.net/publications/techreports/nfd-developer-guide/> Technical Report NDN-0021.
- [15] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 3, pp. 62–67, 2012.
- [16] Y. Ren, Z. Li, J. Li et al., "A dynamic multi-path forwarding strategy for information centric networks," in *Proceedings of the 21st IEEE International Conference on High Performance*

- Computing and Communications; (HPCC/SmartCity/DSS)*, pp. 2495–2501, Sydney, Australia, September 2019.
- [17] B. Abdelkader, M. R. Senouci, and B. Merabti, “Parallel multipath forwarding strategy for named data networking,” in *Proceedings of the 13th International Joint Conference on E-Business and Telecommunications (ICETE)*, pp. 36–46, SciTePress, Lisbon, Portugal, July 2016.
 - [18] H. Qian, R. Ravindran, G. Wang, and D. Medhi, “Probability-based adaptive forwarding strategy in named data networking,” in *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1094–1101, Ghent, Belgium, May 2013.
 - [19] K. Lei, J. Yuan, and J. Wang, “MDPF: an NDN probabilistic forwarding strategy based on maximizing deviation method,” in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, San Francisco, CA, USA, November 2015.
 - [20] K. Lei, J. Wang, and J. Yuan, “An entropy-based probabilistic forwarding strategy in named data networking,” in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*, pp. 5665–5671, London, UK, June 2015.
 - [21] D. Posch, B. Rainer, and H. Hellwagner, “SAF: stochastic adaptive forwarding in named data networking,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1089–1102, 2017.
 - [22] N. C. Luong, D. T. Hoang, S. Gong et al., “Applications of deep reinforcement learning in communications and networking: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.
 - [23] S. Çalisir and M. K. Pehlivanoglu, “Model-free reinforcement learning algorithms: A survey,” in *Proceedings of the 27th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, Trabzon, Turkey, April 2019.
 - [24] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 1582–1591, Stockholm, Sweden, July 2018.
 - [25] J. Yao, B. Yin, and X. Tan, “A SMDP-based forwarding scheme in named data networking,” *Neurocomputing*, vol. 306, pp. 213–225, 2018.
 - [26] O. Akinwande, “Interest forwarding in named data networking using reinforcement learning,” *Sensors*, vol. 18, no. 10, pp. 3354–3373, 2018.
 - [27] Y. Zhang, B. Bai, K. Xu, and K. Lei, “IFS-RL: An intelligent forwarding strategy based on reinforcement learning in named-data networking,” in *Proceedings of the 2018 Workshop on Network Meets AI & ML*, pp. 54–59, NetAI@SIGCOMM, Budapest, Hungary, August 2018.
 - [28] M. Zhang, X. Wang, T. Liu, J. Zhu, and Q. Wu, “AFSndn: a novel adaptive forwarding strategy in named data networking based on Q-learning,” *Peer-to-Peer Networking and Applications*, vol. 13, no. 4, pp. 1176–1184, 2020.
 - [29] J. Mo and J. Walrand, “Fair end-to-end window-based congestion control,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.
 - [30] K. Winstein and H. Balakrishnan, “TCP ex machina: computer-generated congestion control,” in *Proceedings of the ACM SIGCOMM 2013 Conference (SIGCOMM)*, pp. 123–134, Hong Kong, China, August 2013.
 - [31] S. Ramakrishnan and V. Ramaiyan, “Completely uncoupled algorithms for network utility maximization,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 607–620, 2019.
 - [32] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in Neural Information Processing Systems*, vol. 12, no. NIPS, pp. 1057–1063, 1999.
 - [33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, “Deterministic policy gradient algorithms,” in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pp. 387–395, Beijing, China, June 2014.
 - [34] V. Mnih, K. Kavukcuoglu, D. Silver et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
 - [35] V. R. Konda and J. N. Tsitsiklis, “Actor-Critic algorithms,” in *Advances in Neural Information Processing Systems*, pp. 1008–1014, MIT Press, Cambridge, MA, USA, 1999.
 - [36] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, PR, USA, May 2016.
 - [37] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2094–2100, New York, NY, USA, February 2016.
 - [38] P. Erdős and A. Rényi, “On random graphs I,” *Publicationes Mathematicae*, vol. 4, pp. 3286–3291, 1959.
 - [39] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 509–512, 1999.
 - [40] G. Carofiglio, M. Gallo, and L. Muscariello, “Optimal multipath congestion control and request forwarding in information-centric networks: protocol design and experimentation,” *Computer Networks*, vol. 110, pp. 104–117, 2016.