



Adaptive and Energy-Efficient IoT Routing via DOS-RL with Bayesian Hyperparameter Tuning in SDN-Driven Architectures

Mrs. M. Senthamil Selvi, Dr. L. Sudha

Research Scholar, Department of Computer Science, VET Institute of Arts and Science College,

Erode, TamilNadu, India

Associate Professor, Department of Computer Science, VET Institute of Arts and Science College,

Erode, TamilNadu, India

Abstract: The Internet of things (IoT) rapidly grew, and numerous services, programs, sensor-equipped electronics, and related protocols were created and are still being worked on today by enabling physical objects to communicate with one another and share helpful data while making decisions and performing or working on their significant tasks. Wireless sensor networks (WSNs) significantly advance the IoT by serving as a permanent layer. The majority of IoT applications use WSNs as their foundation. IoT-based sensor networks must solve serious general and specialized risks and technical obstacles to ensure adoption and dissemination. In modern IoT deployments, maintaining energy-efficient, reliable, and adaptive network performance is essential but challenging due to heterogeneous devices and rapidly changing conditions. This paper presents a closed-loop, Software-Defined Network (SDN) augmented by Reinforcement Learning (RL) algorithms with Dynamic Objective Selection (DOS-RL) agent and Bayesian optimization (BO) for hyperparameter tuning. The IoT data plane continuously collects and transmits data, while real-time feedback on network conditions and QoS metrics is sent to the control plane. The DOS-RL agent, guided by adaptive weighting mechanisms, leverages Bayesian optimization to fine-tune hyperparameters dynamically, ensuring an optimal balance among correlated objectives such as energy consumption, latency, and delivery ratio. The SDN controller applies these refined routing policies back into the IoT network, preserving both efficiency and responsiveness. Through iterative adaptation and intelligent parameter optimization, the system can swiftly respond to sudden network changes, extending battery life, reducing latency, and improving overall network robustness and scalability.

Keywords: SDWSN-IoT; multi-objective routing; Bayesian optimization (BO); Hyperparameter tuning; reinforcement learning; energy-efficient routing

I. INTRODUCTION

Internet of Things (IoT) constitutes a network of heterogeneous devices communicating and exchanging data amongst themselves to provide smarter services to users [1]. The field of IoT has been witnessing increased research and development in several application areas. Smart home appliances and infrastructure, smart security and surveillance, smart road traffic management and medical emergency response systems are a few examples of IoT network's use cases. IoT networks are enormous in scale and complexity, and comprise objects like Radio Frequency Identification (RFID) tags, mobile phones and sensing devices to obtain data from the environment. Such devices, also referred to as sensor nodes, have low compute capability and limited battery life. The existing routing protocols for Wireless Sensor Networks (WSNs) [2] are complex in nature and demand a considerable use of processing power and memory which are scarce resources in the devices comprising an IoT network. There is hence, a need for simpler protocols that are able to efficaciously conserve the energy of the devices. For an energy efficient network, and especially one that involves cooperative devices, it becomes important to distribute the processing load of routing evenly amongst all devices in the network. This



ensures that a larger number of devices will remain operational for extended periods of time. On the contrary, if this processing load attributed to routing, is subjected to only a subset of the devices in the network, then this subset of devices would consume their energies at a faster rate and hence run out of battery sooner than the rest of the network. The proposed routing protocol in this paper aims to solve this problem by distributing the routing effort which in turn guarantees increased operational time of the network. The working of the protocol is detailed in later sections. Moreover, it must be noted that the term node and device both refer to the sensor nodes in the IoT network and have been interchangeably used in the text [3].

Apart from the sensor nodes, IoT networks also utilize base stations which are centers for data processing and storage. Base stations are more powerful as compared to the sensor nodes and hence come at a higher price. These are used in order to access network data and analyze it. There can be several models of IoT networks depending on the number of base stations used. For simulation of the proposed protocol, only a single base station is used as an end destination for the network data. The aim of the proposed protocol thus translates into finding an optimal energy-efficient path from the sensor nodes to the base station. Moreover, the protocol has to effectively minimize the energy expenditure of devices in the network and increase the network's operational lifetime.

To meet the application-specific requirements of the IoT in real-time, the energy cost for transmission poses a challenge in IoT applications and should be thoroughly considered [4]. IoT-based wireless sensors are deployed in computationally demanding and energy-constrained environments, necessitating the exploration of solutions that enable the prolonged operation of wireless sensor nodes without requiring battery replacements or location changes [5]. Among the various tasks performed by such nodes, data transmission emerges as the most energy-intensive, followed by others such as route computation, idle listening for potential incoming traffic, and data processing [6]. As a result, the development of effective routing protocols that identify optimal routes for a successful data transfer with acceptable energy consumption is critical for enhancing the overall performance of IoT-oriented WSNs. Despite the significant number of existing routing protocols in the literature, it is worth noting that current solutions are relatively limited in their ability to address the energy efficiency challenges of IoT-oriented WSNs. However, the integration of emerging networking technologies, such as Software-Defined Networks (SDN), into wireless sensor networks has shown promising results in the development of effective energy-efficient schemes. By removing energy-consuming tasks such as routing, data processing, and network management from the physical wireless nodes, the SDN has transformed them into data forwarding entities, significantly reducing their energy consumption [7]. The SDN architecture, with its centralized management approach that separates control logic from network devices, eliminates distributed operations and provides a holistic view of the network, better reflecting the actual network conditions. Leveraging its ability to collect comprehensive network information and create a global network view, the SDN architecture has facilitated the introduction of new optimization techniques, such as artificial intelligence and machine learning (ML) algorithms capable of solving complex problems [8].

The core challenge in applying Reinforcement Learning (RL) algorithms with Dynamic Objective Selection (DOS-RL) to IoT-based SDWSN environments lies in effectively learning policies that balance multiple, often conflicting objectives under dynamic and resource-constrained conditions. In practice, IoT networks face fluctuating topologies, limited energy supplies, and diverse Quality of Service (QoS) requirements, all of which must be addressed simultaneously. Traditional RL methods often optimize a single or static set of objectives, making them ill-suited for environments where priorities shift over time (e.g., from energy efficiency to latency reduction as network conditions change). DOS-RL attempts to resolve this by dynamically prioritizing objectives—such as energy minimization, network reliability, and throughput—but this multi-objective learning process introduces complexity. The RL agent must discern the interplay among correlated metrics, adapt its objectives in response to sudden network changes, and learn stable policies despite the frequent recalibration of goals. This complexity can slow convergence, increase computational overhead, and pose difficulties in ensuring the optimal long-term performance and reliability of IoT networks. Introduce adaptive weighting schemes that automatically adjust the importance of objectives based on real-time feedback from the network. For example, employing meta-learning or Bayesian optimization methods could help the RL agent more quickly



identify the most critical objectives at any given time and update their relative weights accordingly, ensuring that learning remains focused on the most impactful goals. The contribution of the research work is described as follows

1. The IoT network (data plane) continuously operates, collecting data and transmitting it through the network.
2. Real-time feedback (network conditions, QoS metrics) is sent upward to the control plane.
3. The DOS-RL agent integrated with Adaptive Weighting Mechanisms (Bayesian optimization) processes this feedback, dynamically selects objectives, and updates its routing policy to better meet current network demands.
4. The SDN controller then enforces these updated routing decisions on the IoT devices, ensuring energy-efficient and high-performing network operation.
5. This closed-loop process repeats, allowing for continuous adaptation to sudden network changes and long-term performance optimization.

II. RELATED WORKS

In [9], for instance, the authors proposed a block chain-SDN-based allocated design for NFV-enabled smart cities. In addition, the authors presented an energy optimized group leader determination method that can efficiently select a group leader. As an added bonus, the SDN controller manages and monitors how the IoT components function. In this study, we use block chain technology to locate and counteract cyber-attacks on Internet of Things (IoT) infrastructure. In regards to throughput, time, fuel usage, and communication overhead, the test results show that the suggested architecture performs better than the existing structure. Distribution-based routing techniques, which make use of the network's topology to route packets, are notoriously inefficient. Since the overhead of centralized algorithms is low, and the chance of route failure is also low, they are advantageous.

In [10], a smart routing system for IoT-enabled WSNs is proposed using Deep Reinforcement Learning (DRL), which helps to drastically cut down on delay while simultaneously increasing the network's lifetime. The proposed method divides the network into uneven clusters according to the data transfer in each sensor, thus significantly extends the network's lifespan. The experimental outcomes show that the proposed strategy is effective in terms of network throughput, energy efficiency, delay in communication, and the percentage of living nodes. A multilayer SDN-based system [11] is proposed to speed up data monitoring & load balancing across IoT devices in a local region and across network clusters. The proposed architecture safeguards against the controller becoming a bottleneck and it facilitates the use of various management & load balancing mechanisms in a hierarchy and multi-step setting. Experimental findings showed that their technique enhanced processing performance by decreasing average turnover and waiting times. The proposed method also optimizes the utilization of network resources by distributing jobs uniformly across the system. The software-defined sensor network was created when SDN & WSN were fused to create a more resilient system, and then in [12] a fuzz route discovery protocol was proposed to solve the problem of wasted energy in a WSN setup. (SDWSN). The FTDP is used in the SDWSN architecture to determine which new hop is optimal given the residual power (RP), node price (NP), Amount of nearest neighbors (NN), & queue Extent (QE). The sink in algorithm is fed details about the network's nodes and the deciding factors. After that, the data is sent to the controller, who then makes the next move based on the data and a fuzzy system.

In [13], the authors zeroed in on a service called SDN Based On load Balancing (SBLB) that takes into account both the response time and resource utilization of cloud server customers. The components of SBLB are an application module, a software-defined-networking (SDN) controller and server farms talking to one another through switches and routers with SDN functionality. There are other sub-modules within the program itself, including active load balancing, observing, & service categorization. All incoming communications are processed instantly, and the controller also oversees the pool of available hosts. The algorithm cuts down on typical response and reply times, as well.



As described in [14], Hybrid Cloud Trying to offload is set up, in which the jobs connected by advanced tasks are delegated to the servers, & the answers are then returned to related elements. This is an innovative approach for complex IoT applications, as it allows the IoT node to offload work to the most suitable fog node or cloud, depending on application needs and local fog node availability. Nodes' ability to offload work to each other or the cloud computing helps to distribute the workload and improves the state of affairs. A Markov Decision Processes model is used to describe the issue (MDP). In addition, a Q-learning dependent approach is introduced for resolving the system and picking the best offloading strategy. The proposed method has been shown to perform better than existing approaches in terms of time, number of tasks executed, and load balancing in numerical simulations. Network congestion can be reduced through careful management of traffic flows, as suggested in [15], which proposes an Admission Method Of control (Opt-ACM) based on optimum load balancing. An MILP-based optimization problem is developed, and the effectiveness of Opt-ACM is verified by running it via the popular mathematical optimization solver, Gurobi. In [16], we see the proposal of a revolutionary Economical SDN-Based Wireless Sensor Infrastructure (ESD-WSN) that utilizes SDWSN and the Internet of Things (IoT) using proxy. Proxy nodes with a lot of processing power are picked to handle control traffic & data aggregation. Multiple rounds are used to select proxies, with the resulting node taking on some of the controller's duties. For Internet of Things (IoT) applications across wireless mesh networks, a new routing mechanism based on clustering has been presented in [16]. Message exchange rates are kept low by using the cluster head nodes as well as the relay node in this approach. The relay can facilitate conversations across different clusters. In order to cut back on power usage, the authors of [17] suggest cutting down on the size and number of elements in the flow table. As part of this technique, only highly probable flow entries are kept in the table, while the rest are discarded. The Hidden Markov Models will purge the flow table of entries if there are none that correspond during a given time period (HMM).

In [18], a novel architecture for efficient routing called Quality-of-Service-based Routing Protocol with Software-Defined Features of the input less Entity (QSDNWISE) is presented. Because of the high power requirements of individual nodes, this design makes extensive use of clusters or cluster - head arranged in a dense cluster around the sink. The method executes separate route for each type of data by dividing them apart.

Furthermore, several studies have explored the application of machine learning algorithms in finding optimal data transfer routes within SDN-based network architectures for the SDWSN. For instance, a study [19] introduced an intelligent architecture for selflearning control strategies in software-defined networks by proposing a routing scheme based on deep reinforcement learning (DRL). This scheme, known as NetworkAI, utilizes deep reinforcement learning techniques and leverages network monitoring technologies like inband network telemetry to dynamically generate control policies, leading to nearoptimal decision-making. Similarly, in [20], the authors presented a scheme to optimize the routing path in the SDWSN using the Reinforcement Learning (RL) algorithm. Their approach incorporates energy efficiency and network Quality-of-Service (QoS) parameters into the reward function. The proposed routing scheme compares various SDN-based techniques, including the traditional SDN, energy-aware SDN (EASDN), QR-SDN, TIDE, as well as non-SDN-based techniques such as Q-learning and RL-based routing (RLBR). The results indicate that the RL-based SDWSN outperforms other approaches in terms of the network lifetime and packet delivery ratio. In this research work introduces a routing scheme that aims to enhance the capabilities of SDWSN-IoT by integrating deep learning techniques, leveraging the inherent features of SDWSN-IoT such as network programmability and comprehensive topology monitoring. The proposed framework facilitates dynamic learning and the adaptation to network changes, enabling the proactive installation and continuous updating of routes based on rapidly changing link states, thereby ensuring that swift and efficient routes are found for data forwarding.

III. PROPOSED METHODOLOGY

This paper presents a closed-loop; SDN-driven architecture augmented by a Deep Online Scheduling Reinforcement Learning (DOS-RL) agent and Bayesian optimization for hyperparameter tuning is presented in figure 1. The proposed system dynamically adjusts routing strategies to optimize correlated objectives, including energy consumption, latency, and delivery ratio.

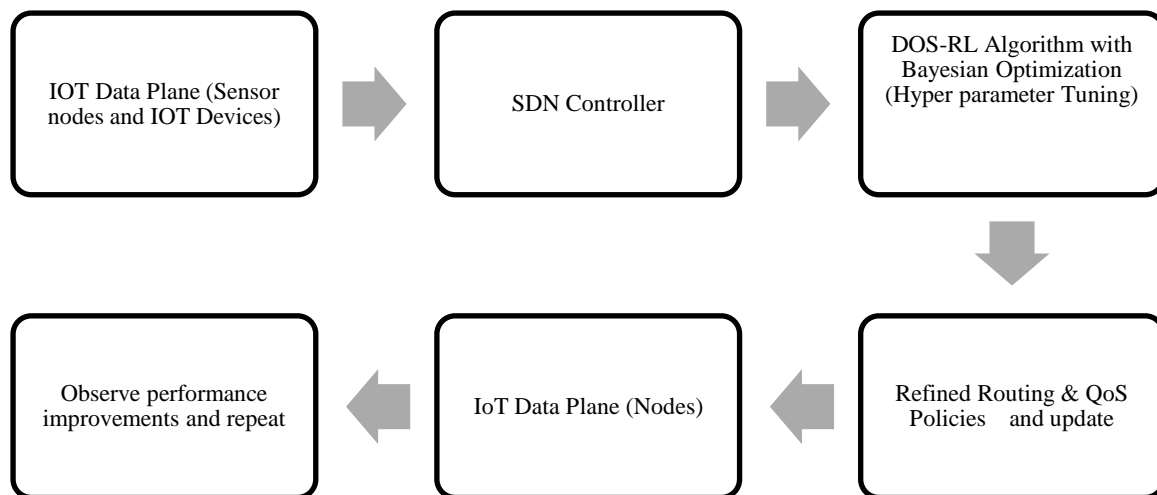


Figure 1: Proposed Methodology Flow

The system diagram consists of the following interconnected component are shown in figure 2:

- **IoT Data Plane:** Includes sensor nodes, local gateways, and monitoring systems.
- **SDN Controller (Control Plane):** Centralized decision-making and policy application.
- **DOS-RL Agent:** Decision-making module with dynamic objective selection.
- **Bayesian Optimization Module:** Hyperparameter tuning and performance feedback.
- **Feedback Loop:** Continuous data flow between the IoT network and SDN controller.

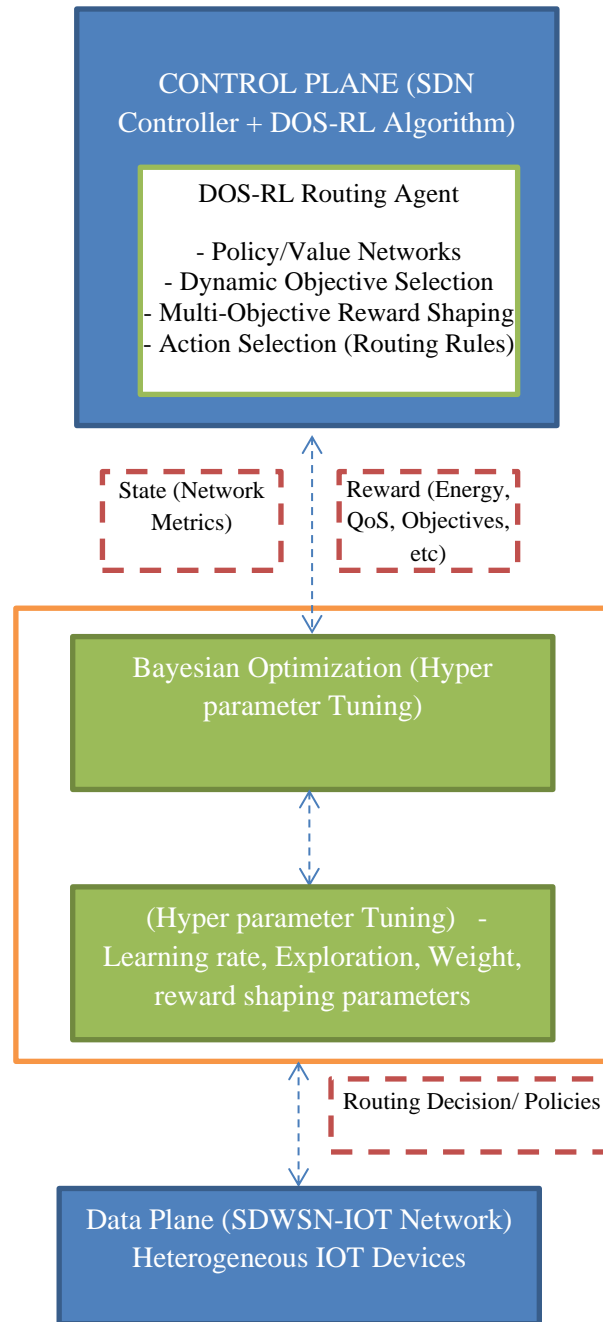


Figure 2: Proposed Interconnected Diagram

To design a network architecture that integrates the **AIoT data plane** (with sensor nodes, local gateways, and monitoring systems), the **SDN controller** (control plane) for centralized decision-making, and the **DOS-RL routing scheme**, we need to consider the roles and interaction of each component in a cohesive framework. This setup allows the network to dynamically adapt to real-time conditions using reinforcement learning and intelligent control through SDN.

A. vIoT Data Plane: Includes sensor nodes, local gateways, and monitoring systems.

The **AIoT data plane** is responsible for the physical network infrastructure, consisting of sensor nodes, local gateways, and monitoring systems. These components are responsible for collecting data, transmitting it, and potentially acting on certain routing decisions before the data reaches the SDN controller.



Components of the AIoT Data Plane:

Sensor Nodes:

1. These are the edge devices that collect real-time data from the environment (e.g., temperature, humidity, energy consumption, traffic, etc.).
2. Sensor nodes are typically energy-constrained and need to minimize energy consumption while maximizing data collection and transmission efficiency.

Local Gateways:

1. Gateways aggregate data from multiple sensor nodes, process it locally, and forward it to the SDN controller.
2. They may handle some routing tasks to reduce the load on the controller by performing local decisions based on high-level policies.

Monitoring Systems:

1. These are responsible for monitoring network conditions and the performance of IoT nodes and gateways (e.g., packet loss, delay, energy consumption).
2. They collect feedback from the data plane and report it back to the SDN controller for further policy adjustments.

AIoT Data Plane Workflow:

- Sensor nodes continuously collect data and report it to local gateways.
- Local gateways may preprocess the data and send summaries or aggregated data to the SDN controller.
- Monitoring systems track the network conditions (e.g., congestion, latency, energy use) and relay this information back to the SDN controller to provide real-time status updates.

B. SDN Controller (Control Plane): Centralized decision-making and policy application.

The **SDN controller** serves as the brain of the network, responsible for centralized decision-making, policy generation, and application. The SDN controller manages the **control plane** and communicates with the **data plane** (sensor nodes, gateways, and monitoring systems).

Functions of the SDN Controller:

Centralized Decision Making:

- The SDN controller receives real-time metrics from the data plane (IoT nodes, gateways, and monitoring systems).
- It processes these metrics to make high-level decisions about network behavior, such as optimizing routing, managing congestion, or deciding when to reroute traffic.

Policy Application:

- The SDN controller generates routing policies based on the **DOS-RL agent's** decisions (using reinforcement learning).
- It pushes these policies to the data plane for execution. The policies can include routing tables, QoS (Quality of Service) parameters, and energy management instructions.



Dynamic Policy Adjustment:

- The controller continuously adjusts policies based on feedback from the monitoring systems. If performance metrics like energy, latency, or packet loss deteriorate, the controller may trigger policy updates to improve the network's performance.

Monitoring and Feedback Loop:

- The controller continuously monitors the feedback from the AIoT data plane and uses **Bayesian Optimization (BO)** to fine-tune the **DOS-RL agent's hyperparameters** for better routing decisions.

SDN Controller Workflow:

- Collect network data from the AIoT data plane (sensor nodes, gateways, and monitoring systems).
- Use the **DOS-RL agent** to evaluate the network state and decide on optimal routing actions.
- Apply these routing decisions as network policies and push them to the data plane.
- Continuously monitor performance and adjust policies based on real-time feedback, using BO to optimize the DOS-RL agent.

C. The DOS-RL Routing Scheme

The realistic conditions of an SDWSN-IoT network are subjected to dynamic changes in resources such as battery capacity, CPU capacity, memory, and bandwidth, as well as link quality changes during network operation. RL-based routing schemes have demonstrated considerable advantages in designing network operation policies that can handle such changes. However, such routing protocols often fail to respond quickly enough to such changes. To address this issue, we propose the DOS-RL routing scheme which allows the RL agent to learn from multiple correlated objectives simultaneously and to adaptively choose the objective it believes is the most promising from the current state. Therefore, we have turned the problem of energy efficiency into an RL process by modeling it into a four-tuple(S, A, P, R), defining the states (S), actions (A), policy (P), and reward functions (R) of the DOS-RL scheme.

- **The Reward Function (R):** In reinforcement learning, the agent evaluates the effectiveness of its actions and improves its policy by relying on rewards collected from the environment. The rewards obtained are typically dependent on the actions taken, with varying actions resulting in differing rewards. To implement the proposed DOS-RL scheme, we define three reward functions, each corresponding to one of the correlated objectives: the selection of routing paths with sufficient energy for data forwarding (o_1), load balancing (o_2), and the selection of paths with a good link quality for a reliable data transfer (o_3). Route selection based on any of these objectives is expected to improve the energy efficiency of the IoT network system. Details on each objective of the DOS-RL scheme and its related reward functions are provided below:
 - Energy-consumption:** To achieve objective o_1 , we consider the energy-consumption parameter, which is a critical factor in determining the overall energy efficiency of an SDWSN: for instance, in a scenario where node i forwards a packet to node j . The reward received by node i for selecting node j as its relay node in terms of energy consumption is estimated by the reward function $RE_{i,j}$ for the state-action pair, (s_i, s_j) as:

$$RE_{i,j} = \text{Remaining Energy}(s_j) - \text{Remaining Energy}(s_i)$$

where $\text{Remaining Energy}(s_j)$ represents the remaining energy of node i in state s_i as a percentage; meanwhile, $\text{Remaining Energy}(s_i)$ represents the remaining energy of node j in state s_j as a percentage. The



formula calculates the difference in remaining energy between node i and j after the data transmission. A higher value of $RE_{i,j}$ indicates that node i has consumed less energy in forwarding the packet to node j , which is desirable to achieve the energy efficiency objective (o_1).

- ii. **Load balance:** To optimize energy efficiency and network performance in the SDWSN, the careful selection of relay nodes and balanced workload distribution are crucial. Otherwise, some nodes along overused paths may become overloaded, leading to bottlenecks and delays, which can result in a degraded network performance. For objective o_2 , we utilize the parameter-available buffer length to estimate the degree of queue congestion in relay nodes. The reward $R_{Q_{i,j}}$ for load balancing as observed by node i when selecting node j is computed as follows:

$$R_{Q_{i,j}} = \frac{\text{Available_Buffer_Length}(s_j)}{\text{Max_Buffer_Length}} - \frac{\text{Load}(s_i)}{\text{Max_Load}}$$

where $\text{Available_Buffer_Length}(s_j)$ represents the available buffer length of node j in state s_j , Max_Buffer_Length represents the maximum buffer length of a node, $\text{Load}(s_i)$ represents the current load of node i in state s_i , and Max_Load represents the maximum load capacity of a node. This formula considers both the available buffer length of node j and the current load of node i . The second term subtracts the load of node i from the load balancing reward value, allowing for a more comprehensive estimation that takes into account both node i and node j in the load balancing process. A higher value of $R_{Q_{i,j}}$ indicates a better load balancing situation for the given state-action pair (s_i, s_j) .

- iii. **Link quality:** A wireless link can be measured by retrieving useful information from either the sender or receiver side. To achieve o_3 , we use simple measurements to estimate the link quality based on the parameter packet reception ratio (PRR), measured as the ratio of the total number of packets successfully received to the total number of packets transmitted through a specific wireless link between two nodes. Unlike other sophisticated techniques, this approach involves a low computation and communication overhead. Instead of using an instant value of the PRR, we calculate an average-over-time using an Adaptive Weighted Moving Average (AWMA) filter. Suppose node i is forwarding data packets to node j , the reward $R_{LQ_{i,j}}$ received by node j based on PRR using AWMA is estimated as follows:

$$R_{LQ_{i,j}} = \text{PRR}_{\text{est}_{i,j}(k-1)} \cdot \alpha_{\text{AWMA}} + \text{PRR}_{\text{sample}_{i,j}}(k) \cdot (1 - \alpha_{\text{AWMA}})$$

whereby $\text{PRR}_{\text{est}_{i,j}(k-1)}$ is the previously estimated average, $\text{PRR}_{\text{sample}_{i,j}}(k)$ is the most recent measured value of the packet reception ratio calculated, and α_{AWMA} is the filter parameter.

- **The State Space (S):** We define the state space as a graph corresponding to the global topology created by the RNs on the data plane, as seen by the intelligent RL controller. Each state in the state space corresponds to an RN, and a state transition refers to a link connecting two RNs. The intelligent RL controller uses the partial maps created by the topology discovery module to create a global topology. Therefore, the cardinality of the set of states depends on the number of nodes that can actively participate in routing.
- **The Action Space (A):** The action space, denoted as A , includes all possible actions that an agent can undertake from a given state of the RL environment. It defines the choices available to the agent at each time step, presenting the range of options to the agent. In our specific problem, the discrete action space comprises a finite number of actions that the RL agent can select when in a particular state $s_i \in S$. The cardinality of A at state i is determined by the number of nodes eligible to participate in the routing process from that specific state.



- The Optimal Policy (P): The policy determines how the learning agent should behave when it is at a given state with the purpose of maximizing the reward value in the learning process. Our proposed scheme estimates the Q-function of every objective o simultaneously and decides, before every action selection decision, which objective estimate o_{best} an agent will consider in its decision-making process. We use the concept of confidence on computed Q-values by representing each action as a distribution and using a normal distribution of Q-values and the mean values to keep track of the variance as shown in Equation (2). The agent approximates the optimal Q-function by visiting all pairs of action-states and stores the updated Q-values in the Q-table. In our proposed scheme, the approximated Q-value $Q(s_t, a_t)$ represents the expected cumulative reward when the RL agent is in the state s_t and takes action a_t , transitioning to a new state s_{t+1} while maximizing the cumulative rewards for an objective o_n , where $n \in N$ and N denote the set of all objectives. The Q-learning equation to update Q-values is designed as shown below in Equation (7):

$$Q^n(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q^n(s_t, a_t) + \alpha \cdot [R_{t+1}^n + \gamma \cdot \max_{\forall a \in A} \{Q^n(s_{t+1}, a)\}].$$

To steer the exploration behavior of the learning agent by incorporating some heuristic knowledge on the problem domain, we introduce an extra reward F_{t+1} onto the reward received from the environment R_{t+1} . The newly added shaping reward function F is included when updating the Q-learning rule as follows:

$$Q^n(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q^n(s_t, a_t) + \alpha \cdot [R_{t+1}^n + F_{t+1} + \gamma \cdot \max_{\forall a \in A} Q^n(s_{t+1}, a)].$$

To avoid changes on the optimal policy, F is implemented as the difference of some potential value, $F_{t+1} = \gamma\varphi(t + 1) - \varphi(t)$ over the state space, where φ is a potential function that provides some hints on states. In our study, we define φ as the Normalized Euclidean Distance between the current state(s) and the goal state(G) expressed as:

$$\varphi = 1 - \frac{\text{distance}(S_t, G)}{\max_{\forall x \in S} (\text{distance}(x, G))'}$$

where $\text{distance}(S_t, G)$ is the Euclidean distance between the current state S_t and the goal state G , and $\max_{\forall x \in S} (\text{distance}(x, G))'$ represents the maximum distance between any state x in the state space S and the goal state G . The potential function guides the agent towards the goal state by giving bigger rewards to states that are closer to the goal and smaller rewards for states that are farther away

- **In DOS Q-routing:** To find the best action-value of the Q-function, the learning agents use an action selection mechanism to trade-off between the exploitation and exploration of available action space. To achieve this, our proposed scheme uses the e-greedy exploration and exploitation method, whereby $e \in [0, 1]$, allowing the agent to exploit with probability $pr = e$ and explore with probability $pr = 1 - e$. The agent action selection is determined by a randomly generated number $x \in [0, 1]$, of which if $x < e$, the agent exploits it by taking an action that returns the most expected optimal value; otherwise, it explores it by selecting a random action based on the most confident objective o_{best} as observed from the current state by the learning agent as shown below:

$$\text{actionSelection} = \begin{cases} \max_{a \in A} \{Q(s, a, o_{best})\}, & \text{if } x < e \\ \text{randomAction}(Q(s, a, o_{best}), \dots, Q(s, a_n, o_{best})), & \text{otherwise} \end{cases}$$



D. Bayesian Optimization Module: Hyperparameter tuning and performance feedback.

Bayesian Optimization (BO) is a probabilistic model-based optimization technique designed to optimize objective functions that are expensive to evaluate. In the context of optimizing the DOS-RL agent within an SDN environment, BO dynamically fine-tunes hyperparameters of the reinforcement learning model to enhance its performance.

The main Objective is to Optimize the performance of the DOS-RL agent in SDN-based IoT networks by dynamically adjusting its hyperparameters, such as:

- Learning Rate (α): Determines the step size for updating the Q-value.
- Discount Factor (γ): Balances the importance of immediate vs. future rewards.
- Exploration Rate (ϵ): Controls the trade-off between exploration and exploitation.

The advantages of BO is efficiently explores the hyperparameter space with fewer evaluations and it handles noisy and expensive objective functions (e.g., evaluating RL performance metrics like energy efficiency and latency). The BO models the objective function $f(x)$ as a probabilistic function using a surrogate model (typically a Gaussian Process, GP). It predicts the expected performance for a given hyperparameter configuration and refines the search using an acquisition function.

i. BO Workflow in Optimizing DOS-RL

The problem is modeled as a minimization or maximization problem:

$$\text{Maximize } f(\Theta) \text{ where } \Theta = \{\alpha, \gamma, \epsilon\}$$

Where:

- $f(\Theta)$: Objective function representing the performance of DOS-RL (e.g., reward or combined metrics like energy efficiency and latency).
- Θ : Hyperparameter set to optimize.

The workflow of BO is

1. **Search Space:** Define the bounds for each hyperparameter:

$$\alpha \in [0.01, 0.1], \gamma \in [0.8, 1.0], \epsilon \in [0.1, 0.3]$$

2. **Surrogate Model:** Use a Gaussian Process (GP) to model the objective function:

$$P(f | \theta) \sim N(\mu(\theta), \sigma^2(\theta))$$

- $\mu(\theta)$: Mean prediction of the objective function at θ .
 - $\sigma^2(\theta)$: Variance representing uncertainty.
3. **Acquisition Function:** Guides the selection of the next hyperparameter configuration by balancing exploration (uncertain regions) and exploitation (regions with high performance).
 - Common acquisition functions:
 - **Expected Improvement (EI):**

$$EI(\theta) = E[\max(0, f(\theta^*) - f(\theta))]$$



Where θ^* is the best observed configuration.

4. Integration of BO in SDN with DOS-RL

4.1 Workflow in SDN Context

1. **IoT Data Collection:**
 - IoT nodes transmit real-time network metrics (e.g., energy levels, traffic conditions) to the SDN controller.
2. **Policy Decision:**
 - The DOS-RL agent selects routing actions based on the current policy.
3. **Performance Evaluation:**
 - The SDN controller monitors performance metrics, such as energy consumption, latency, and delivery ratio.
4. **BO-Based Optimization:**
 - The BO module:
 - Collects feedback on DOS-RL performance.
 - Optimizes hyperparameters to improve future policy decisions.
5. **Policy Deployment:**
 - The SDN controller updates routing policies in the data plane.

4.2 Metrics for BO Optimization

1. **Reward Function:** Combines energy, latency, and delivery ratio:

$$R_t = w_1 \cdot \left(1 - \frac{E_i}{E_{max}}\right) + w_2 \cdot \left(1 - \frac{L_{ij}}{L_{max}}\right) + w_3 \cdot \frac{P_{ij}}{P_{max}}$$

2. **Objective Function:** Combines rewards over multiple episodes:

$$f(\Theta) = \frac{1}{T} \sum_{t=1}^T R_t$$

Where T is the number of iterations in an RL episode.

Algorithm 2: DOS-RL with Hyperparameter Tuning Bayesian Optimization:

Input: Learning rate α ; Exploration-Exploitation parameter e ; Number of learning episodes n ; Potential function for each state φ ; All link pairs (src, dst): V Links pair; Link states, set of learning objectives: N ;

Bayesian Optimization Parameters: Surrogate model-Gaussian process (GP); Acquisition function-Expected Improvement (EI); Hyperparameter search space: ($\alpha \in [0.01, 0.1]$; $\gamma \in [0.8, 1.0]$; $\epsilon \in [0.1, 0.3]$)

Output:



- Optimizaed Q values $Q(S,A)$
- Set up of computed path stored in route repository

Procedure:

Procedure:

1. Initialization:

- Initialize $Q(S, A) = 0, \forall S \in S, A \in A$.
- Initialize Bayesian Optimization with a Gaussian Process surrogate model.
- Define the shaped reward function:


$$R'_t = R_t + F_{t+1}$$

2. Hyperparameter Tuning Using Bayesian Optimization:

- Define the objective function for BO:

$$f(\Theta) = \frac{1}{n} \sum_{i=1}^n R_i$$

where $\Theta = \{\alpha, \gamma, \epsilon\}$.

- Optimize the hyperparameters:
 1. Fit the Gaussian Process $P(f|\Theta)$ to the observed rewards.
 2. Use the Expected Improvement (EI) acquisition function to select the next hyperparameter set: 



hyperparameter set:

$$EI(\Theta) = \mathbb{E}[\max(0, f(\Theta^*) - f(\Theta))]$$

3. Update $\Theta = \{\alpha, \gamma, \epsilon\}$ with the optimal values from BO.

3. Compute Loop-Free Paths:

- $Sj(G) \leftarrow STP(VLinksPair)$: Compute the set of loop-free paths from the initial state to the goal state G .

4. Q-Learning with DOS-RL:

For each objective $o \in N$, compute Q-values for all $(src, dst) \in Sj(G)$:

a. Learning Loop:

For $episode = 1$ to n :

- Start at initial state $s_t = src \in S$.
- While $s_{t+1} \neq dst$:
 1. Select action A_t for state s_t using ϵ -greedy method:

$$A_t = \begin{cases} \operatorname{argmax}_A Q(S_t, A), & \text{with probability } 1 - \epsilon, \\ \text{random action}, & \text{with probability } \epsilon. \end{cases}$$

2. Compute the shaped reward:

2. Compute the shaped reward:

$$R'_t = R_t + F_{t+1}$$

3. Update Q-values:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \cdot [R'_t + \gamma \cdot \max_{A'} Q(S_{t+1}, A') - Q(S_t, A_t)]$$

4. Transition to the new state S_{t+1} :

$$S_t \leftarrow S_{t+1}$$

b. After all episodes, store the Q-table for the objective.

5. Path Selection:

- For each (src, dst) , select the path with the maximum Q-values from the Q-table.

6. Store Results:

- Save the computed paths in the route repository.

Hence The SDN controller continuously updates its routing policies based on the optimized actions provided by the DOS-RL agent and hyperparameter tuning from BO. These updates are deployed to the data plane, directly impacting how the data packets are routed within the network. Hence after optimizing the routing policy, the controller updates the data plane with the new policy. The network traffic is routed according to the updated policies, completing the feedback loop.



I. EXPERIMENTAL RESULTS

This section presents an evaluation of the proposed DOS-RL routing scheme. The chapter is organized into multiple sections and subsections, which outline the simulation tools and frameworks used to measure the scheme's performance. Furthermore, it includes details about the test environment, performance metrics, learning parameter settings, and the observed results, which will be presented and discussed.

Simulation Environment

To evaluate the effectiveness of our proposed scheme, we conducted simulations using ns-3 [41], an open-source system-level network simulation tool capable of creating an environment for the easy transfer of states and actions between AI frameworks and the ns-3 simulation environment. To achieve this, we employ the ns3-ai [42] module, which enables seamless integration between ns-3 and open-source AI frameworks like TensorFlow and PyTorch by utilizing shared memory. The ns3-ai module consists of two components: the ns-3 interface, implemented in C++, and the AI interface developed in Python. These components work together to ensure the fast and efficient exchange of large data volumes from a C++ program to a Python program. By utilizing shared memory, the ns3-ai module facilitates communication between multiple processes. Unlike the ns3-gym framework [43], which relies on pipes or sockets, the use of shared memory allows for the creation of a highly efficient core module for data transfer.

To implement our proposed scheme, we have designed a system architecture that combines the ns3-ai and the SDWSN architecture, as shown in Figure 2. The proposed architecture, depicted in Figure 3, consists of two main components: the ns-3 simulator and the AI framework. The ns-3 simulator serves as the data generator by providing environments to create simulation scenarios. It generates relevant information, which is then fed into the AI framework for training the model to make real-time decisions. The AI framework processes the data received from the ns-3 simulator and trains the model to make intelligent decisions. The shared memory pool facilitates the seamless data exchange between the ns-3 and the AI framework, allowing both sides to access and manipulate the data. Control signals, managed by four modules operating at the control layer, ensure smooth communication and coordination between the ns-3 and the AI framework. This integration enables efficient decision-making and the evaluation of our proposed routing scheme.

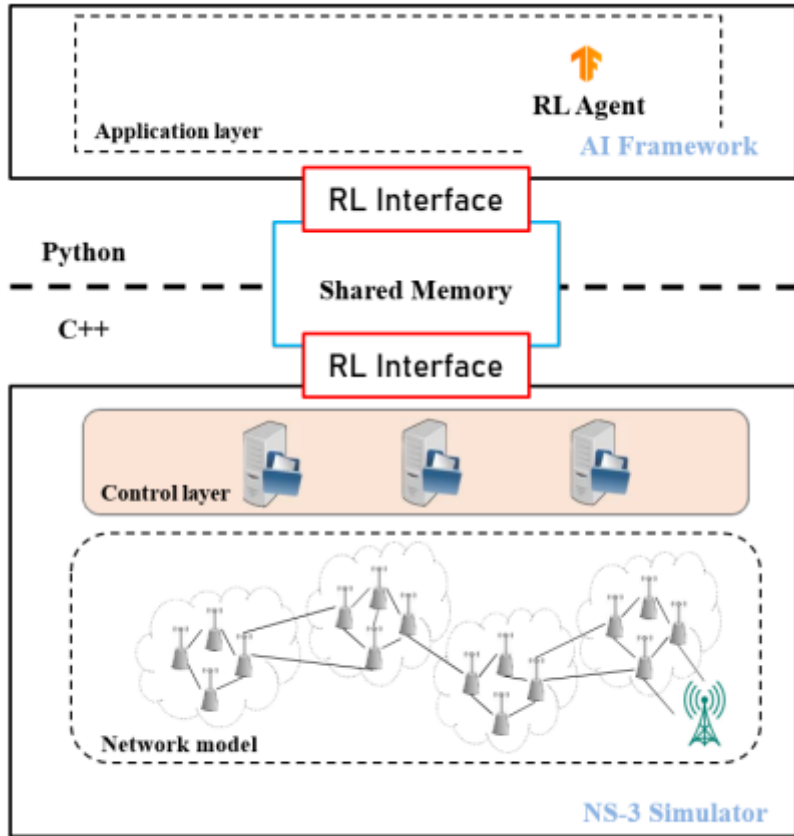


Figure 3. DOS-RL SDWSN Architecture.

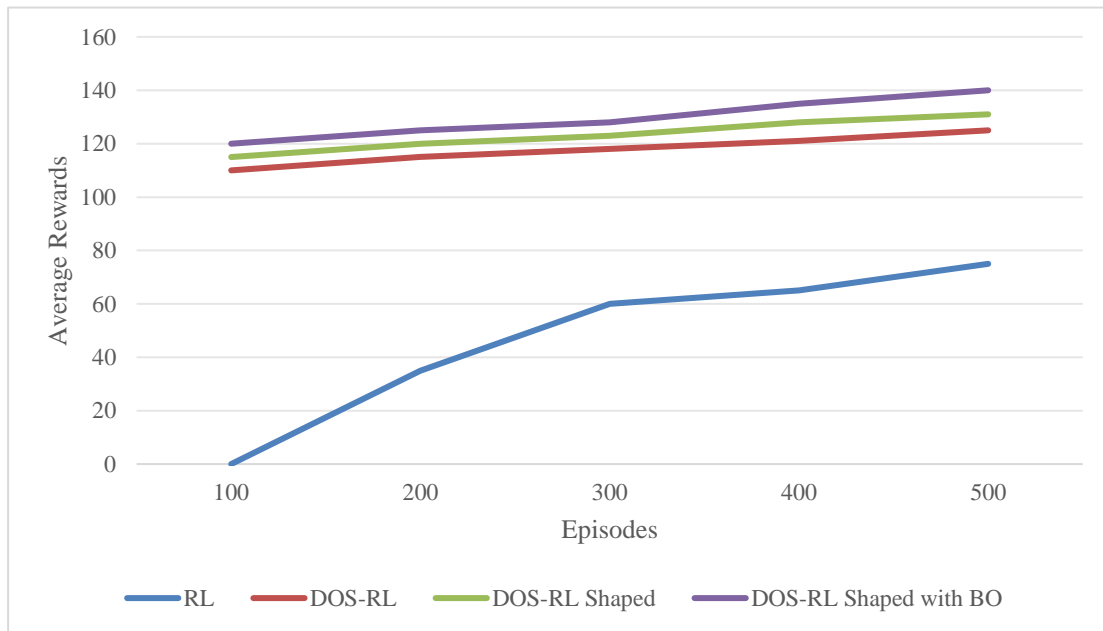


Figure 4 (a): Reward per Episodes

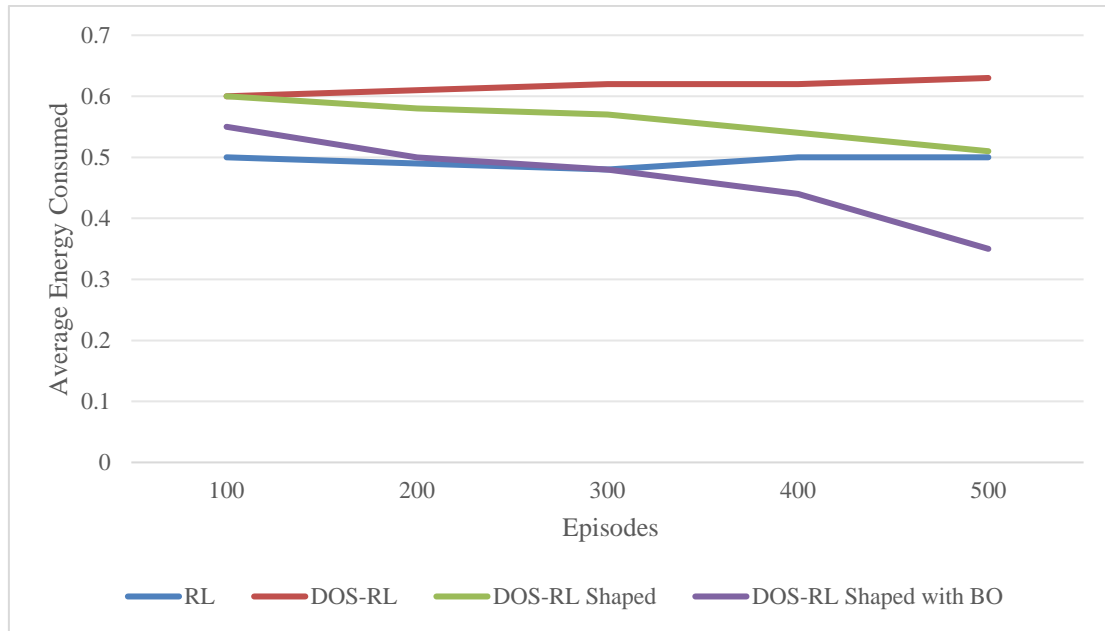


Figure 4(b): Comparison of Energy consumption

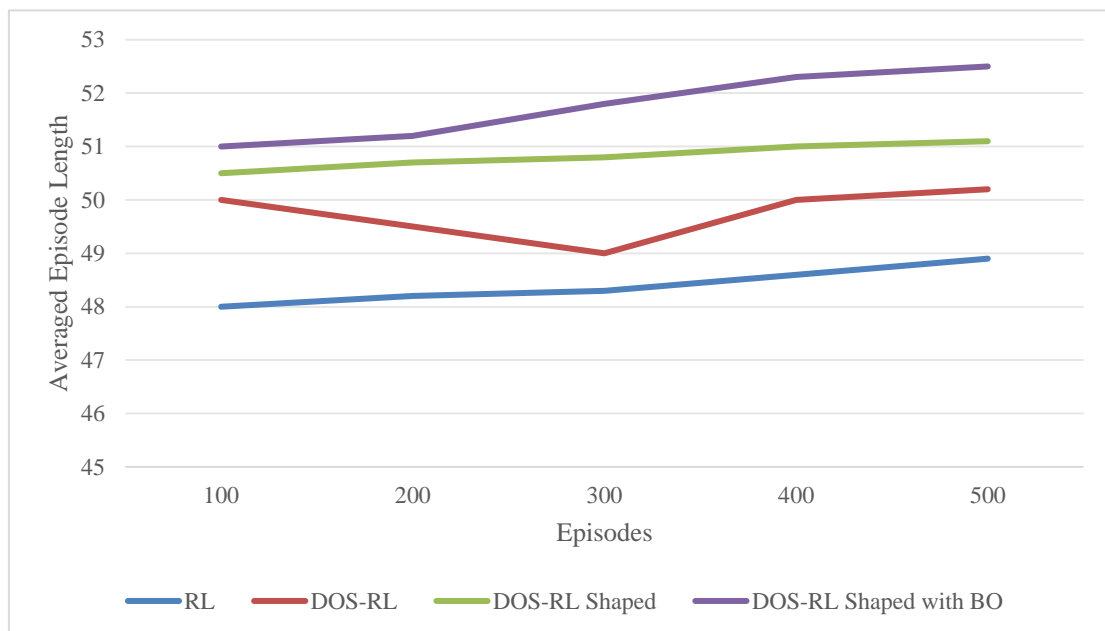


Figure 4(c): Comparison of Energy consumption

To evaluate the convergence speed of the algorithms, Figure 4a plots the learning curve for the average rewards collected and the average energy consumed per number of episodes in Figure 4b. In Figure 4c,d, we can see plots for the average episode length and the average frequency of hitting obstacles per number of episodes during the entire simulation time. In Figure 4a, we can see how the poor performance of the RL algorithm is affected by its frequency of hitting obstacles during the first 100 episodes. The DOS-RL with shaped rewards outperforms the rest of the algorithm by proving its efficiency in conserving energy (Figure 4b) by nearly 20% with the highest average collected at a cost of a slightly longer episode length compared to the RL algorithm. In summary, unlike the traditional RL, the DOS algorithms seem to perform much better by allowing the agent to



explore different strategies that cater to different objectives. This allows the algorithms to find a balance between objectives more effectively.

Packet Delivery Ratio of Routing Protocols

In our evaluation, we examine the performance of three routing protocol schemes in a network environment comprising 30 randomly distributed Relay Nodes (RNs). We investigate how the Packet Delivery Ratio (PDR) of these schemes is affected when varying the number of traffic sources and the relay node density.

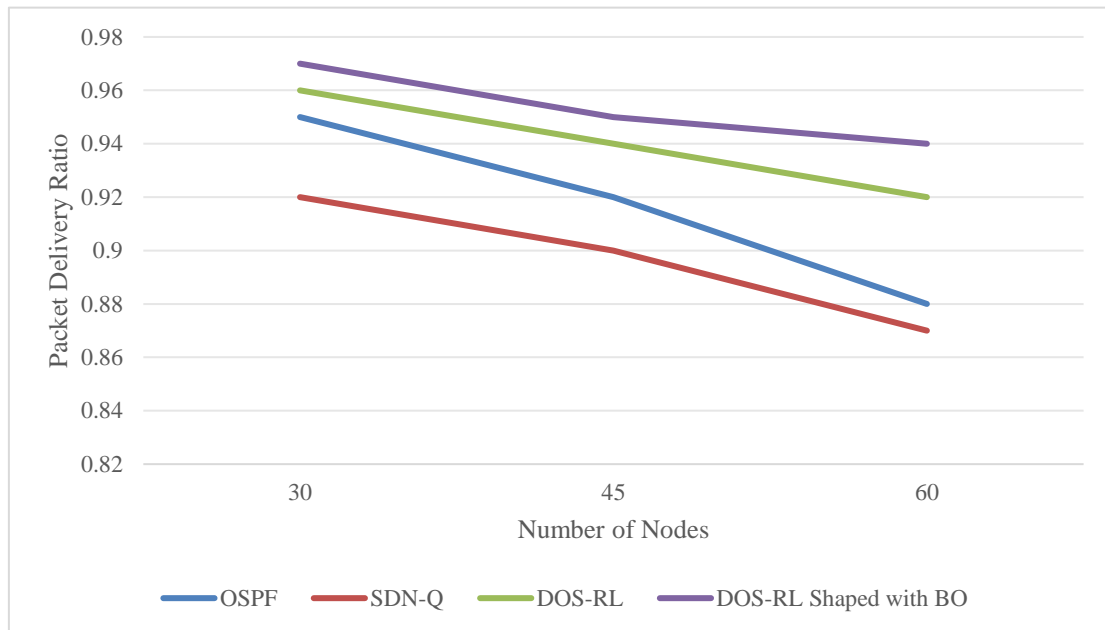


Figure 5: Packet Delivery Ratio

From figure 5, we observe that the difference in the PDR remains relatively small across the different numbers of the RNs. In Figure 5b, we notice a gradual decline in the PDR as the number of nodes increases. This decline can be attributed to factors such as increased network complexity, scalability, congestion, and increased overhead resulting from a larger number of nodes sharing the network resources. Nevertheless, our proposed scheme continues to outperform the other routing schemes by maintaining a comparatively higher PDR, even when the number of RNs is doubled.

End-To-End Delay of Routing Protocols

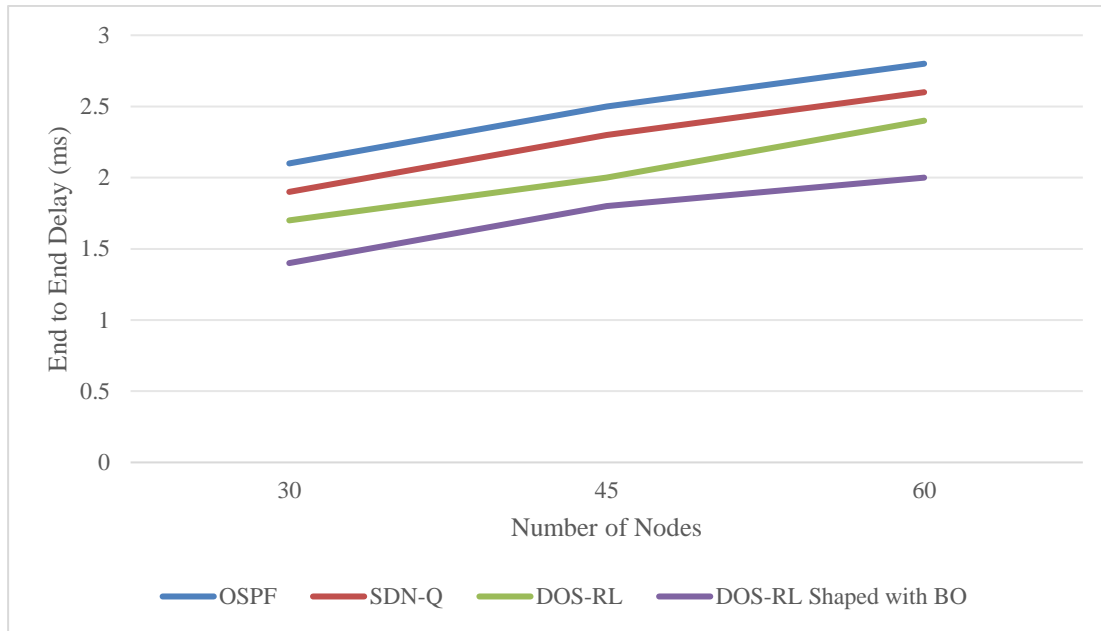


Figure 6: End to End Delay Comparison

Next, we look at the E2E simulation results of the three protocols as shown in Figure 6a. The OSPF routing algorithm typically aims for a low routing delay by selecting the shortest path which is expected to perform well under normal conditions; however, the results state otherwise. From the results, it can be seen that the OSPF has the worst performance. However, our proposed scheme adjusts well by adjusting learning objectives in real-time. The SDN-Q fails to perform well because the paths it selects last longer and hence, increase the probability for congested paths to occur.

II. CONCLUSION

In this paper, we have proposed a novel approach to optimizing routing in Internet of Things (IoT) networks by leveraging Software-Defined Networking (SDN), Reinforcement Learning (RL) with Dynamic Objective Selection (DOS-RL), and Bayesian Optimization (BO) for hyperparameter tuning. As IoT networks continue to grow and evolve, addressing challenges such as energy efficiency, latency, and adaptability is crucial for maintaining optimal network performance. The integration of SDN provides centralized control, enabling dynamic and intelligent policy adjustments in real time. The DOS-RL agent, with its ability to prioritize and optimize multiple objectives, ensures that routing decisions are adapted to the network's current state, balancing factors like energy consumption, latency, and packet delivery. The Bayesian Optimization technique refines the performance of the DOS-RL agent by dynamically adjusting its hyperparameters, ensuring that the agent learns and adapts more effectively over time. This closed-loop system enables the IoT network to respond swiftly to changing conditions, enhancing its energy efficiency, reducing latency, and improving the delivery ratio, all while maintaining scalability and robustness. Through iterative adaptation and intelligent parameter optimization, the proposed system demonstrates significant potential for improving the performance of modern IoT networks. By combining the strengths of SDN, RL, and Bayesian Optimization, this approach not only addresses the core challenges of IoT routing but also sets the stage for future advancements in adaptive, energy-efficient, and scalable IoT systems.

III. REFERENCES

1. Kasim Al-Aubidy, A.W. Al Mutairi, and Ahmad Derbas. Real-time healthcare monitoring system using wireless sensor network. *International Journal of Digital Signals and Smart Systems*, 1:26, 01 2017.



2. Jacques Bahi, Wiem Elghazel, Christophe Guyeux, Mourad Hakem, Kamal Medjaher, and Noureddine Zerhouni. Reliable diagnostics using wireless sensor networks. *Computers in Industry*, 104:103 – 115, 2019.
3. S.E. Bouzid, M. Mbarki, C. Dridi, and M. N. Omri. Smart adaptable indoor lighting system (SAILS). In *2019 IEEE International Conference on Design Test of Integrated Micro Nano-Systems (DTS)*, pages 1–6, April 2019.
4. Ngu, A.H.; Gutierrez, M.; Metsis, V.; Nepal, S.; Sheng, Q.Z. IoT middleware: A survey on issues and enabling technologies. *IEEE Internet Things J.* 2016, 4, 1–20.
5. Ijmaru, G.K.; Ang, K.L.M.; Seng, J.K. Wireless power transfer and energy harvesting in distributed sensor networks: Survey, opportunities, and challenges. *Int. J. Distrib. Sens. Netw.* 2022, 18, 15501477211067740.
6. Amini, N.; Vahdatpour, A.; Xu, W.; Gerla, M.; Sarrafzadeh, M. Cluster size optimization in sensor networks with decentralized cluster-based protocols. *Comput. Commun.* 2012, 35, 207–220.
7. Kobo, H.I.; Abu-Mahfouz, A.M.; Hancke, G.P. A survey on software-defined wireless sensor networks: Challenges and design requirements. *IEEE Access* 2017, 5, 1872–1899.
8. Zhao, Y.; Li, Y.; Zhang, X.; Geng, G.; Zhang, W.; Sun, Y. A survey of networking applications applying the software defined networking concept based on machine learning. *IEEE Access* 2019, 7, 95397–95417.
9. M. J. Islam, A. Rahman, S. Kabir et al., “Blockchain-SDN based energy-aware and distributed secure architecture for IoTs in smart cities,” *IEEE Internet of Things Journal*, 2021.
10. Sangamithra, B., Swamy, B. M., & Kumar, M. S. (2021). A comparative study on a privacy protection in personalized web search. *Materials Today: Proceedings*.
11. Z. Eghbali and M. ZolfyLighvan, “A hierarchical approach for accelerating IoT data management process based on SDN principles,” *J. Netw. Comput. Appl.*, vol. 181, no. 4, 2021, Art. no. 103027.
12. Ganesh, D., Sunil Kumar, M., & Rama Prasad, V. V. (2017). Mutual Trust Relationship Against Sybil Attack in P2P E-commerce. In *Innovations in Computer Science and Engineering* (pp. 159-166). Springer, Singapore
13. S. Kumar, K. Cengiz, S. Vimal, and A. Suresh, “Energy efficient resource migration based load balance mechanism for high traffic applications IoT,” *Wireless Pers. Commun.*, p. 5111, Feb. 2022.
14. Ganesh, D., & Kumar, M. S. (2016). Improving Network Performance in Wireless Sensor Networks: A Survey. *Int. J. Web Technol*, 5(1).
15. R. Kumar, U. Venkanna, and V. Tiwari, “Opt-ACM: An optimized load balancing based admission control mechanism for software defined hybrid wireless based IoT (SDHW-IoT) network,” *Comput. Netw.*, vol. 188, Apr. 2021, Art. no. 107888.
16. J. Li, B. N. Silva, M. Diyan, Z. Cao, and K. Han, “A clustering based routing algorithm in IoT aware wireless mesh networks,” *Sustain. Cities Soc.*, vol. 40, pp. 657–666, Jul. 2018.
17. Ramu, M.M., Shaik, N., Arulprakash, P., Jha, S.K. and Nagesh, M.P., Study on Potential AI Applications in Childhood Education. *International Journal of Early Childhood*, 14(03), p.2022.
18. X. Tan, H. Zhao, G. Han, W. Zhang, and T. Zhu, “QSDN-WISE: A new QoS-based routing protocol for software-defined wireless sensor networks,” *IEEE Access*, vol. 7, pp. 61070–61082, 2019
19. Yao, H.; Mai, T.; Xu, X.; Zhang, P.; Li, M.; Liu, Y. NetworkAI: An intelligent network architecture for self-learning control strategies in software defined networks. *IEEE Internet Things J.* 2018, 5, 4319–4327.
20. Younus, M.U.; Khan, M.K.; Bhatti, A.R. Improving the software-defined wireless sensor networks routing performance using reinforcement learning. *IEEE Internet Things J.* 2021, 9, 3495–3508.