Received 16 January 2024, accepted 31 January 2024, date of publication 5 February 2024, date of current version 14 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3362353

RESEARCH ARTICLE

SSDWSN: A Scalable Software-Defined Wireless Sensor Networks

MOHAMMED ALSAEEDI[1], **MOHD MURTADHA MOHAMAD**[1], **AND ANAS AL-ROUBAIEY**[2]
[1]Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia
[2]Computer Engineering Department, College of Computing and Mathematics, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia
Corresponding author: Mohammed Alsaeedi (amamohammed22@graduate.utm.my)

ABSTRACT In multi-hop wireless sensor networks (WSNs), sensors operate autonomously and make routing decisions independently. However, these devices are often located in remote or inaccessible areas and have limited energy and memory resources. As the network scales, efficient management to conserve resources and extend its lifetime becomes increasingly challenging. Software-defined WSNs (SDWSNs) offer a solution by enabling centralized control of low-power WSNs. However, continuously updating the controller with the network state generates significant traffic, resulting in energy loss, increased overhead, and reduced scalability and network lifetime. This study proposes a scalable SDWSN framework (SSDWSN) to address these challenges. The proposed approach focuses on scheduling, balanced routing, aggregation, and reducing traffic overhead caused by periodic network state updates to the controller. This paper presents the architecture of the proposed framework, along with the Deep Reinforcement Learning (DRL) agent. It also proposes two Proximal Policy Optimization (PPO)-based learning policies, namely PPO-ATCP and PPO-NSFP. These policies are designed to efficiently utilize SDWSN network resources and accurately predict the network state by continuously monitoring the synchronized network state within the controller, taking appropriate actions, and updating the learning parameters based on reward functions. The simulation results demonstrate the effectiveness of PPO-ATCP and PPO-NSFP in reducing controller-bound traffic overhead by 57% and 85%, respectively, while improving energy efficiency by 28% and 53% in SDWSNs. Additionally, PPO-NSFP achieved a minimum accuracy of 85% in network state prediction under different network-size scenarios.

INDEX TERMS Software-defined wireless sensor networks, control traffic overhead, proximal policy optimization, deep reinforcement learning, energy efficiency.

I. INTRODUCTION

In wireless Sensor Networks (WSNs), sensors serve as autonomous entities, equipped with self-governing operations and decision-making in routing, often deployed in demanding environments for specialized sensing tasks [1]. These WSN nodes, typically preconfigured with vendor-specific settings, present challenges in terms of flexibility and adaptability, leading to increased deployment costs and reduced potential for service and hardware reuse. Their constrained spatial, energy, and computational capacities further complicate efficient network management, particularly as the network size increases. The implementation of Software-Defined Networking (SDN) architecture offers a strategic resolution, centralizing control logic to effectively manage resources, especially pertinent in Low-Rate Wireless Personal Area Networks (LR-WPANs) with limited energy and memory resources [2].

The use of SDN in WSNs separates control logic from data planes, simplifying sensors into basic forwarding units and freeing them from complex networking logic and vendor-specific restrictions. This streamlines network management, enhances flexibility, and enables diverse application support through programmability and customization. SDN controllers, with their network-wide state view, enable more effective and optimal network management solutions,

The associate editor coordinating the review of this manuscript and approving it for publication was Renato Ferrero.

including energy-efficient routing and resource utilization. This has led to the conceptualization of software-defined WSNs (SDWSNs), extending the SDN principles to traditional WSNs. In SDWSNs, the sensing layer is relieved from network control responsibilities and managed either directly through a sink or indirectly via an IoT gateway, thereby enhancing network reconfigurability, maintainability, and resilience [3], [4], [5]. SDWSNs nodes are supported by SDN protocols such as OpenFlow [6]. However, the conventional OpenFlow protocol, designed for core network switches, faces compatibility challenges with sensor nodes due to their constraints in processing capabilities and energy resources. Hence, they require a modified version of such a protocol, which is tailored to their specific needs, to ensure efficient network management and energy utilization. As exemplified in previous studies [7], [8], SD-WISE demonstrates how the OpenFlow protocol can be customized to address specific challenges posed by WSNs. However, such adaptations are not as widespread in the industry as their traditional switch counterparts are. The differences between these protocols and their applicability to various sectors underscore the need for ongoing research and clarification to guide their practical implementation.

The architectural separation of the control and sensing layers in SDWSNs has several advantages; however, it also introduces significant controller-bound traffic overhead. This challenge is particularly pronounced in multi-hop WSNs, where both control and data traffic utilize the same limited wireless communication channels. This scenario results in increased energy consumption and network congestion [9]. The overhead is exacerbated by the need for continuous communication to maintain a unified network-wide view, which is fundamental for managing and optimizing network performance. This constant need for synchronization between the network and its controller is necessary to collect comprehensive traffic statistics [6]. These statistics are vital for maintaining network efficiency and reliability and significantly affect the scalability and operational efficiency of SDWSNs [10], [11], [12], [13]. However, maintaining this level of network insight and control necessitates regular updates, resulting in an increased volume of controller-bound traffic. This increase in traffic adversely impacts the overall network performance and lifespan.

Motivated by the observations highlighted earlier, this study focuses on reducing and refining controller-bound traffic overhead in SDWSNs, particularly the traffic necessary for updating the controller regarding the network states. It aims to effectively adapt network resource use in response to fluctuations in the network traffic. This study introduces a prototype for managing controller-bound traffic in SDWSNs by leveraging the global network state view of SDN controller. We propose two optimal control policies, that implement actions as flow rules in the sensing layer, using Deep Reinforcement Learning (DRL) models based on Proximal Policy Optimization (PPO). These policies aim to efficiently reduce control traffic overhead by optimizing and predicting the periodic network state information transmitted to the controller. The results demonstrate their effectiveness in approximating temporal state value functions and stabilizing actions, thereby mitigating risks associated with sudden network changes. The key contributions of this study are as follows:

- We propose a scalable SDWSN framework (SSDWSN), aimed at enhancing resource management and extending the network's lifetime.
- We propose a DRL agent to utilize SDWSN network resources and efficiently predict the network state.
- Two on-policy gradient PPO-based learning algorithms, PPO-ATCP and PPO-NSFP, were proposed for efficient scheduling, balance routing, aggregation, and reduction of SDWSN traffic overhead resulting from periodic reporting of the network state to the controller.
- We implemented SSDWSN as a prototype for scalable SDWSNs. The source code is publicly available in [14]. The simulation results indicated that the SSDWSN exhibited robust performance and scalable adaptability with respect to network size and density.

The rest of the paper is organized as follows. Section II presents related work on optimizing control traffic overhead in SDWSN. Section III describes the control traffic overhead problem in SDWSN networks. The proposed approach for scalable and efficient control of signaling traffic to the controller in SDWSN is explained in Section IV. In Section V, the performance of the proposed approach is evaluated. Finally, Section VI summarizes the research findings and directions for future research.

II. RELATED WORKS

In SDWSNs, the optimization and reduction of control traffic transmitted from the sensing layer to the controller have been addressed using various approaches. These include techniques such as balanced routing, aggregation, clustering, control traffic duty rescheduling, and network state prediction.

The key to optimizing control traffic in WSNs is effective routing. However, employing the same routing paths in large-scale dynamic networks often results in unnecessary performance degradation and scalability issues. Unlike conventional local routing decisions in WSNs, SDWSNs offer flexibility to select routing paths based on the controller's wide-network state Quality of Service (QoS) requirements, thus optimally avoiding congestion and uneven resource utilization. Unfortunately, many previous non-learning-based SDN routing solutions for WSNs, such as Energy-Aware SDN (EASDN) [15], SD-EAR [16], EOMCSR [17], and [1], [18], [19], [20], [21], [22], tend to overlook traffic characteristics and the dimensionality challenges of large-scale networks. These routing algorithms rely on forwarding rules established based on the local state of neighboring nodes, such as residual energy, traffic load, and distance to the sink node, without considering the overall network traffic distribution [23]. Consequently, this limitation can result in premature depletion of energy in certain nodes. This gap has prompted the integration of Machine Learning (ML) technology with SDN to gain insights into the hidden characteristics and time-varying behavior of SDWSNs, thereby addressing complex Traffic Engineering (TE) problems. ML-based routing strategies have emerged that leverage historical experience and network state views within the SDWSN controller [24], [25], [26], [27].

Supervised deep learning (DL) solutions have shown promise in predicting the highly nonlinear nature of network traffic. However, they demand substantial labeled historical data, making it challenging to obtain an offline training dataset, and suffer from slow adaptation to dynamic network changes. Furthermore, supervised learning bias issues raise questions regarding the adaptability of online models to network dynamics [28]. In contrast, Reinforcement Learning (RL) has been proposed for routing optimization, which relies on past experience to ensure energy-efficient utilization [27], [29], [30].

The assessment of learning success is based on parameters such as the estimated node and path lifetime, neighbor distance, residual energy, and hop count to the sink. In low-power wireless networks, the selection of an optimized routing path is critical for efficient resource utilization and prolonged battery life. Unlike RL, DRL is better suited for addressing traffic engineering challenges in highly dynamic and large-scale network environments. DRL agents continually optimize their neural networks based on the rewards received from their decisions. For example, DGRL [12] combined graph convolution with a deterministic policy gradient through an actor-critic algorithm to intelligently control traffic and reduce the risk of network congestion in SDWSNs. The control policy trained in the controller was implemented in the sensor nodes to optimize the data-forwarding process. However, highly dynamic and time-varying network traffic conditions require each sensor node to update its policy. Therefore, maintaining the control policy in the controller and sending policy actions as flow rules to the respective sensors aligns better with the SDWSN architecture and limited energy and memory resources of the sensor nodes [12]. Our proposed approach differs from these studies in that we employed PPO [31]. It introduces reward functions that encompass various optimization actions, including balanced routing, traffic duty rescheduling, and network state prediction, while also considering network scalability and resource wastage stemming from synchronizing the network state with the controller.

Clustering has emerged as another efficient resource utilization technique to address the rapid battery consumption in WSNs stemming from excessive data transmission. By leveraging global network information, the clustering technique can efficiently reduce the control traffic overhead in SDWSNs [11], [32], [33], [34], [35], [36], [37], [38], [39]. However, in multi-hop SDWSNs, clustering techniques face several challenges. Key issues include scalability problems owing to large network sizes, overload and rapid energy depletion of cluster heads, uneven energy consumption across the network, and increased routing overhead. Additionally, these networks often experience congestion and increased latency, particularly in densely deployed areas. The complexity of forming and maintaining efficient clusters in dynamic environments and ensuring reliable data transmission over multiple hops further complicate the implementation of clustering in SDWSNs [40].

Traffic aggregation and caching are alternative approaches for minimizing the control traffic overhead transmitted to the controller in SDWSNs [1], [4], [20], [27]. Our proposed aggregation policy differs in that it aggregates and removes redundancy in the packets transmitted to report the state of the sensing layer. The proposed policy employs DRL to optimally select aggregating nodes, guaranteeing an extension of the overall network lifetime. In addition, we propose predicting the continuous transmission of local information from the sensor nodes to the controller to minimize excessive energy loss and traffic overhead. Although the Markov chain and sensor node state have been used in some studies to predict node residual energy levels [10], [13], [41], our approach distinguishes itself by using DRL to handle complexity and uncertainty within SDWSN. This enables our approach to learn directly from high-dimensional network traffic data without explicitly requiring knowledge of underlying dynamics. Moreover, DRL algorithms utilize neural networks to approximate value functions or policies, enabling them to effectively handle complex and continuous state and action spaces. This is particularly advantageous when addressing the challenges of complex and unstructured environments encountered in the context of SDWSNs.

Assuming that the controller possesses sufficient computational capacity to handle heavy data traffic from the data plane, it can satisfy the computational requirements of the DRL-based models in SDWSNs. In this review, we explore several key studies in this field, focusing on their main objectives, reward functions, contributions, performance metrics, and comparisons with our proposed research. These studies collectively contribute to the ongoing efforts to optimize the lifetime of SDWSNs using RL-based routing.

Younus et al. [29] proposed a Q-learning framework to optimize routing in WSNs, focusing on enhancing network lifetime and reducing energy consumption; however, their approach did not explore the scalability of SDWSN. Our study fills this gap by applying the DRL-PPO model to overcome the scalability and stability limitations of Q-learning. Unlike [29], where nodes discard packets unless they are designated as forwarders, our method ensures that packets are not transmitted to neighboring nodes unless a specific forwarding rule is established, thereby reducing unnecessary traffic and conserving energy. Building on their earlier work, Younus et al. [30] enhanced their RL-SDWSN framework by incorporating metrics such as distance to the sink, hops to destination, node residual energy, and packet success ratio into their reward function, with the aim of optimizing routing paths and extending the network lifetime. In contrast to [29] and [30], our methodology not only aims for energy-efficient routing, but also minimizes control traffic from SDN through actions such as aggregation, duty rescheduling, and network state prediction. We utilized the network energy consumption metric and its variance to balance energy consumption across nodes, thereby preventing premature energy depletion and monitoring control overhead to assess the impact of our actions. Furthermore, [30] had each node share status data with its neighbors, increasing control-bound traffic. Our method streamlines this by having nodes send status data directly to the sink and then to the controller via a spanning tree greedy route, thereby reducing unnecessary broadcasts to neighboring nodes. However, essential data for the discovery phase are still shared with neighbors encapsulated in hello messages. Unlike [29] and [30] on small-scale real-testbed experiments with 802.11ac, our simulations, driven by a larger node count, focus on assessing network scalability with IEEE 802.15.4 LR-WPANs transceivers.

Guo et al. [42] developed RLBR, a Q-learning-based framework for optimizing routing in WSNs, focusing on link distance, residual energy, and hop count. Despite advancements in energy efficiency, scalability, and packet delivery, RLBR's decentralized nature, where each node functions as an independent agent, leads to challenges, such as suboptimal routing owing to the lack of global network visibility and the need for extensive memory and computational resources. Our study addresses these issues by proposing a centralized DRL-PPO model, which is a more suitable RL algorithm for dynamic and continuous-action environments.

Q-learning, as utilized in the aforementioned approaches, stands out for its simplicity and ease of implementation and is particularly suited for scenarios with discrete, limited state-action spaces where extensive training data are not required. It is less computationally intensive but struggles in large state spaces and may become trapped in local optima, particularly because it is designed for discrete and low-dimensional action spaces. In large-state scenarios, the Q-table becomes unwieldy, consuming excessive memory for search and storage. By contrast, DRL manages high-dimensional state spaces and continuous action domains. DRL excels in learning intricate policies and effectively generalizing to new states. However, the demand for substantial data and computational resources, coupled with its complexity, may result in challenges with interpretability and stability during training [43]. In the context of SDWSNs, this may not pose a significant concern because the controllers in these networks are typically equipped with considerable computational capabilities.

Rahimifar et al. [10] proposed E^2-SDWSN, a predictive model that utilizes MapReduce for energy-efficient data processing in SDWSNs. This method, while aiming to reduce energy consumption, encounters limitations owing to the high complexity and resource demands of MapReduce. Such complexities can induce latency in multi-hop WSNs and increase both communication and computational overhead, thereby impacting energy efficiency, particularly in rapidly varying WSN environments. In contrast, our approach employs a lightweight OpenFlow adaptation, SD-WISE [7], [8], which mitigates these issues by reducing the need for data exchange between nodes. Moreover, our proposed PPO-NFSP policy not only predicts battery levels but also other statistical parameters necessary to enable the controller to build fine-grained knowledge of the network state.

Huang et al. [12] proposed DGRL, a deep graph RL approach aimed at enhancing traffic routing in SDWSNs, reducing congestion, and minimizing packet delay. Our research diverges from their approach by employing on-policy gradient methods tailored for dynamic network environments, facilitating more effective exploration and adaptation under rapidly changing conditions. Although DGRL's deterministic policy gradient DRL method requires less data for convergence, it may have limitations in highly dynamic WSN traffic scenarios, owing to its exploration capabilities. In contrast, we leverage the centralized network view inherent in SDWSNs, enabling more efficient adaptation to fluctuating traffic conditions through on-policy gradient techniques. Additionally, we continuously optimize routing policies in real-time using flow-mod messages to ensure adaptive responses to network changes.

In summary, the reviewed studies made substantial contributions to the field of RL-based SDWSN routing optimization. However, gaps exist in terms of exploring alternative RL algorithms, addressing scalability issues, owing to controller-bound traffic overhead in SDWSNs, and predicting a broader range of network state parameters. Our proposed approach aims to address these gaps and extend the existing knowledge in the field by introducing a DRL-PPO model, minimizing control traffic overhead, and predicting comprehensive network state parameters.

III. PROBLEM FORMULATION

The periodic transmission of network state information to the controller in an SDWSN usually has a low duty cycle and high time-domain correlation, resulting in redundancy in transmitting the same reporting information to guarantee a consistent global network state knowledge of the stochastic nature of the sensing layer at the controller. Excessive signaling traffic to the controller adds traffic overhead and congestion to the sensing layer, degrading the network performance and scalability [10], [44]. To represent the effect of control traffic on network performance, we modeled SDWSN as a directed graph $G = (V, L)$, where V and L denote sets of nodes and links, respectively. The rules allocated for a given graph G can be represented as a two-dimensional matrix $R = (F, V)$, where F is the number of possible flows from any node to all other nodes, and V is the number of given forwarding nodes. Assuming that the wireless network is a fixed-position mesh network, all nodes act as sensing and intermediate nodes simultaneously, and therefore, can operate in the multiple-input and multiple-output (MIMO) mode.

Let $V \triangleq \{v\}_{i=0}^{n}$, where $n+1 = |V|$ has number of n sensing nodes and v_0 is the sink node. Each $v_i \in V$ has a set neigbors $N \triangleq \{j : j \neq i, d_i < d_{max}+1, d_j < d_{max}+1, i \in V, j \in V\}_1^m$, where d_i denotes the distance of v_i to the sink, and d_{max} denotes the maximum distance to the sink. Furthermore, let e_i^t denote the residual energy at time t. Assuming that all nodes work under the same physical devices and the transmission range of each node is equal, including the sink node, the main energy consumption results from packet transmission. In an SDWSN, the dynamic packet arrival rate of node v_i is denoted by λ_i and follows a Poisson distribution. In addition, packet transmission on a link for time duration τ_{ij} follows an exponential distribution with parameters $\tau_{ij} \sim E(\mu_L v_{i,j})$, where μ_L is the length of the transmitted packet [39]. Each node v_i has a limited packet buffer queue $q_i \in Q$ and the energy consumption of node v_i can be represented by the first-order energy dissipation model [45] as:

$$e_i(\tau) = P_i^{Tx}\tau + \epsilon_{cx}\lambda_{i,j}\tau \tag{1}$$

where τ is the transmission time, P_i^{Tx} is the radio transmission power, and $\lambda_{i,j}$ is the transmission rate of node v_i on link $l_{i,j}$. Coefficient ϵ_{cx} represents the circuit energy consumed during data transmission, which varies depending on whether the data is sent or received. For received packets, $\epsilon_{cx} = \epsilon_{11} + \epsilon_{12} = \epsilon_1$, reflecting energy used when receiving 1-bit data. For the transmitted packets, $\epsilon_{cx} = \epsilon_{11}$, denoting the energy spent when transmitting 1-bit data. According to [39], the weight of link (i, j) selected by node v_i to transmit a packet at time t can be estimated as:

$$w_{i,j}(t) = \eta_{i,j} \times \varepsilon_j(t) \tag{2}$$

where $\varepsilon_j(t) = e_j(t)/e_j(0)$ is the ratio of the residual energy of node j to its initial energy at time t, and $\eta_{i,j}$ is given by:

$$\eta_{i,j} = \frac{\lambda_{i,j}}{P_i^{Tx} + \epsilon_{cx}\lambda_{i,j}} \tag{3}$$

Hence, selecting an energy-efficient next-hop forwarding route in node v_i toward sink v_0 in time-varying SDWSN traffic conditions can be obtained by determining an optimal control policy at time t such that the sum of the link weights in the network is the maximum:

$$max\left(\sum_j \sum_i w_{i,j}(t)\right), \quad \forall j, i \in V \tag{4}$$

However, selecting an optimal next-hop forwarding route in node v_i toward the nearest sink that maximizes the overall network energy efficiency under time-varying and random traffic conditions of SDWSN is conceived as nondeterministic and involves solving an NP-hard optimization problem [46]. Furthermore, considering the duty cycle of the node and the rescheduling of the transmitting control traffic, the problem becomes even more complex.

Accordingly, we propose a deep reinforcement learning policy that can adaptively optimize its parameters to dynamic changes in the network to optimally perform load balance routing, redundancy removal, and duty rescheduling of the control traffic to maximize the overall energy efficiency of the network. Furthermore, another policy is proposed to predict the network state information periodically transmitted to the controller to minimize the control traffic overhead and prolong network lifetime. The following metrics were used to evaluate the performance of the proposed approach:

- Network throughput (T) is defined as the average number of successful packets delivered per unit of time from the source nodes to the nearest sink.
- Network latency (D) is defined as the average time required to deliver packets to the nearest sink. It is calculated as the average propagation, transmission, queuing, and processing time required to deliver packets to a nearby sink. The processing time of the node is deficient and is therefore ignored in this study.
- Network energy consumption (EC) is defined as the average amount of energy consumed per second in all observed nodes n. This metric considers four energy consumption states: node processing, node sleeping, and packet transmission and reception. The Network Energy Consumption EC_t at time step t is calculated as in (6). The lower the energy consumption, the longer the network lifetime.

$$EC_{i,t} = \frac{E_{i,t} - E_{i,t'}}{t - t'} \tag{5}$$

$$EC_t = \frac{\sum_{i=1}^{n}(E_{i,t} - E_{i,t'})}{n(t - t')} \tag{6}$$

where $E_{i,t}$ is the residual energy of node i at time step t, $E_{i,t'}$ is the residual energy of node i at the previous time step t', and n is the total number of nodes in the network.
- Network Energy Consumption Variance ($Var(EC)$) is defined as the variance of energy consumption per second in all observed nodes N at time step t as shown in (7). The lower the variance, the more balanced the network energy consumption.

$$Var(EC_t) = \frac{1}{N}\sum_{i=1}^{N}(EC_{i,t} - \overline{EC_t})^2 \tag{7}$$

- Network Lifetime (LT) is defined as the time span from network deployment until it can no longer function as intended. This typically occurs when the last operational node in the network, excluding the sink or base station, runs out of energy, rendering the sink unreachable. To compute the network lifetime LT_t at time step t, we determine the shortest estimated lifetime among all nodes at that time. Network lifetime is essentially determined by the node that depletes its energy first, as the network can no longer function effectively once

TABLE 1. Summary of notations.

Notation	Description
V	Set of forwarding nodes
L	Set of edges (links)
S	All possible network states
A	All possible actions
R	All possible rewards
TS	Observation Timestamp
E	Residual energy
T	Link throughput to the sink
EC	Energy consumption
LT	Network Lifetime
D	End-to-end delay
DE	Nodes' density
DI	Distance (number of hops) to the sink
Ptx	Number of transmitted packets
Btx	Number of transmitted bytes
Prx	Number of received packets
Brx	Number of received bytes
SPE	Samples per epoch
RT	Report transmission time
PD	Number of dropped packets
γ	Discounted factor
α	Learning rate
τ	Action waiting time
ϵ	Clip ratio

any single node fails. Given that EC is measured at discrete time steps, the estimated remaining lifetime of a node i at time step t, denoted as $LT_{i,t}$, is calculated as shown in (8). Consequently, LT_t was derived as shown in (9).

$$LT_{i,t} = \frac{E_{i,t}}{EC_{i,t}} \tag{8}$$

$$LT_t = min(LT_{1,t}, LT_{2,t}, \ldots, LT_{N,t}) \tag{9}$$

- Control overhead refers to the number of packets transmitted to the controller. This metric represents the overhead control traffic from updating the centralized network state view of the controller.

The objective of the proposed optimization model is to maximize the objective function defined by the above performance metrics as follows:

$$max\left(\delta\frac{T_t}{T_{max}} - \eta\frac{D_t}{D_{max}} - \varphi\frac{EC_t}{EC_{max}} - \rho Var\left(\frac{EC_t}{EC_{max}}\right)\right) \tag{10}$$

Factors δ, η, φ and ρ represent the degrees of importance of the T, D, EC and $Var(EC)$ metrics, respectively, such that $\delta + \eta + \varphi + \rho = 1$. The objective function was normalized by dividing each variable by its maximum value. The maximum values are the QoS requirements of the network operator, and are subject to the physical characteristics of the network devices. The key parameters and notation used in this study are listed in Table 1.

IV. THE PROPOSED SSDWSN

This section introduces the proposed scalable SSDWSN framework. First, we explain the proposed architecture. Subsequently, the two proposed PPO-based control policies are described.

A. SSDWSN ARCHITECTURE

The SSDWSN framework is proposed to efficiently schedule, aggregate, and minimize SDWSN traffic overhead caused by the periodic reporting of network statistics to the controller. This minimizes control traffic signaling and congestion, thereby enhancing network scalability and extending network lifetime.

As presented in Fig. 1, the functional architecture of SSDWSN includes Link Discovery (LD), Duty-cycling Schedular (DS), Topology Constructor (TC), Flow Engine (FE), and DRL-Agent. Upon running the network, each sink node initiates a registration request (RegPacket) to its designated controller via a TCP socket and awaits an acknowledgment from the controller. The controller registers only authenticated sinks and adds them to the list of registered sinks.

The LD module operates within the data plane, enabling each node to establish its initial forwarding rule toward the nearest sink. The registered sink node starts advertising itself to its directly connected neighbors by sending beacon packets (BeaconPacket). Each node remains idle from the sending beacons until it receives a beacon packet from an adjacent node at a shorter distance from the sink node. Accordingly, as proposed by SD-WISE [7], [8], a sensor node can set up an initial forwarding rule to maintain a route to the next hop of the greedy non-optimal shortest distance (number of hops) toward the nearest sink. This flow-rule entry is added to the flow table of the node during the discovery phase of the sensing layer without controller intervention to construct the base communication topology, as described in Algorithm 1. Once a sensor node has a forwarding rule to the sink, it reports its status and neighbors to the controller by sending a report packet (ReportPacket) at every configured report transmission time interval (RT). Any packet received or generated undergoes a flow-matching pipeline process to determine a wildcard-matched forwarding rule for routing to the nearest sink. The sink node forwards each received report packet directly to the controller. Consequently, the controller builds centralized fine-grained global topological and statistical knowledge of the network.

The DS module controls traffic redundancy overhead and congestion by executing scheduling flow entries installed in the flow table by the controller. These entries can schedule either the node duty cycle (active/sleep) of nodes within the same transmission range or the transmission duty cycle for reporting status of the node to avoid traffic congestion. In a dense SDWSN with a low-duty cycle, nodes can overhear redundant data by updating the global network-state view of the controller. Transmitting redundant data leads to wastage of network resources and rapidly unbalanced battery consumption, particularly in nodes that are closer to the sink. Hence, reporting the network state to the controller in SDWSN adds traffic overhead and congestion to the

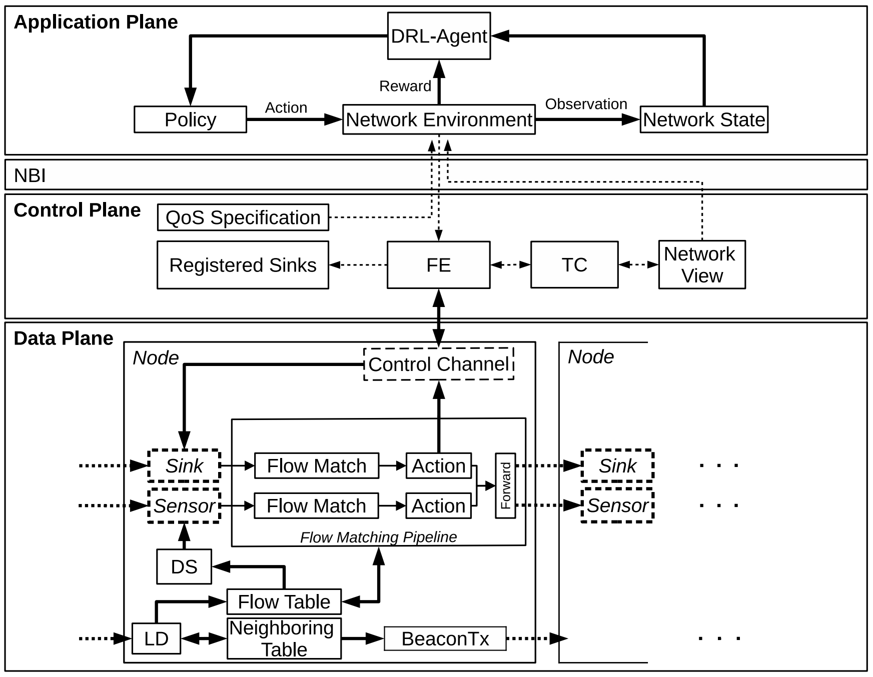

FIGURE 1. **Functional architecture of SSDWSN.**

sensing layer, thereby degrading the network's performance and scalability. In this study, we propose aggregating and rescheduling report packets within a configured RT based on a decision from the proposed DRL agent to control SDWSN traffic congestion.

The OpenFlow adaptation and flow table structure were implemented based on the design proposed by SD-WISE [7], [8]. The controller receives the flow statistics and network state information from the sensing layer. One or more DRL agents in the application layer utilize the collected data to define the network policies that perform actions (e.g., routing, duty cycling, load balancing, and forecasting) to be implemented in the sensing layer.

TC is responsible for gathering and preprocessing the network view and the flow of statistical data received from the data plane. When a controller receives a report packet, it extracts topological and statistical information (e.g., node ID, battery level, location, neighbor ID, Received Signal Strength Indicator (RSSI), and transmitted/received packets/bytes) and updates the network state view. The output of the network state view consists of a topology matrix and its respective flow feature matrix, which represent the network state view at a given timestamp. The controller forms an abstracted directed graph $G = (V, L, S)$, where V represents the nodes, L represents the edges (links) between nodes, and S represents the flow features and characteristics of each node $v_i \in V$.

The FE module runs in the controller and translates networking policies into forwarding rules. It encodes a policy action received from the DRL-Agent in the application layer into flow-rule entries encapsulated in a flow-modification packet (FlowModPacket), to be set in the corresponding node. Each flow-rule consists of match fields (headers) that specify the characteristics of the packets to which it applies, such as the source and destination IP addresses, transport layer ports, and packet type. For simplicity, in this study, we define the flow-rule header as having three primary matching elements: source, destination, and packet type. For example, a header such as $r_{v_i}, *, rp$ matches report packets from source v_i to any destination, as indicated by wildcard $*$. If a packet does not match any specific rule, it follows the base rule with wildcard header $*$, defaulting to the shortest path toward the nearest sink. This setup ensures efficient forwarding across all scenarios even without specific matching conditions. In addition, the flow engine updates the global network state view using the topological and statistical information collected from the reporting packets received by the controller.

The DRL-Agent runs in the application plane, is designed to interact with the network environment, and trains a policy to perform actions that satisfy the scalability and performance requirements of SDWSN. It exploits the network state-experience trajectory observed in the network environment (control/data planes) to learn an optimal policy that maximizes the expected long-term performance rewards. The output actions of the DRL-Agent are transmitted to the control plane via the Northbound Interface (NBI) in the form of flow-rule entries installed in the flow tables of the corresponding node. These entries have instructions to dynamically select the optimal aggregating nodes, perform report duty-cycle scheduling and load-balance routing, and forecast the network state to improve the scalability and

Algorithm 1 Link discovery algorithm.

```
Require: Let $b, p$ are respectively beacon and report packets,
    $BT$ is the beacon transmition time interval, $dst_r$ is
    the destination of executed forwarding rule, $N$ is the
    neighbors set, $d$ is the distance to the sink node $v_{sk}$.
 1: $d_{i,sk} \leftarrow d_{max}+1$ {Distance of node $v_i$ to the nearest sink}
 2: if $v_i$ is a sink then
 3:     $d_{i,sk} \leftarrow 0$
 4:     $v_i \leftarrow$ active
 5: end if
 6: procedure txBeacon {Beacons sending worker}
 7:     while True do
 8:         if $v_i$ is active & $d_{i,sk} < d_{max} + 1$ then
 9:             for $v_j \in N_i$ do
10:                 if $d_{j,sk} > d_{i,sk} + 1$ then
11:                     SEND($b_{i,j}$) {Send beacon packet to a
    neighbor node}
12:                 end if
13:             end for
14:         end if
15:         Wait a $BT$ time
16:     end while
17: end procedure
18: procedure RXBEACON {Beacons receiving worker}
19:     upon event receiving $b_{j,i}$ do
20: if $d_{i,sk} > d_{j,sk} + 1$ then
21:     $v_{i,sk} \leftarrow dst(b_{j,i})$
22:     $d_{i,sk} \leftarrow d_{j,sk} + 1$
23:     $v_i \leftarrow$ active
24:     $r_* \leftarrow$ FORWARD_RULE($*, v_j$) {Forwarding rule to
    the sink}
25:     ADD_RULE($r_*$)
26: end if
27: if $d_{i,sk} \neq d_{j,sk} + 1$ and $v_j = dst_{r_*}$ then
28:     $d_{i,sk} \leftarrow d_{j,sk} + 1$
29: end if
30: Update $N_i$
31: end procedure=0
```

prolong the lifetime of the network. We propose a DRL agent to achieve this objective, while considering the performance metrics mentioned in Section III. The critical components of the proposed agent are shown in Fig. 2. The normalized observation trajectory is passed to the policy multihead attention layer to learn the dependencies (attentions) between the state features of a node and those of other node inputs. The attention weights were added to the primary inputs, normalized, and subsequently passed to a feedforward layer to study the relationship between the independent variables (inputs) and the dependent variables (outputs). The outputs of the feedforward layer are added to the inputs, normalized, and subsequently passed to the mean and standard deviation of the learner layers. The action was sampled from a normalized distribution constructed from the output mean and standard deviation. The normalized observation trajectory was also passed to the value (critic) layer to obtain a linear value function of the trajectory state input.

TABLE 2. Summary of network state features.

Feature	Description
TS	Observation Timestamp
D	End-to-end delay
E	Residual energy
T	Link throughput to the sink
DE	Nodes' density
DI	Distance (hops) to the sink
EC	Energy consumption
RT	Report transmission time
Ptx	Number of transmitted packets
Btx	Number of transmitted bytes
Prx	Number of received packets
Brx	Number of received bytes
PD	Number of dropped packets

The link discovery phase establishes a base greedy shortest-path topology as a forwarding rule, directing each node to the next hop with the shortest distance (measured in the number of hops) toward the sink. The nodes periodically report updated topological and statistical information to the controller, thereby maintaining a consistent view of the global network state. This view of the network state is then made available to the DRL-Agent in the application layer. As represented in Algorithm 2, the agent initializes neural network (NN) models for policy (actor), value (critic), cloned value-target, and cloned policy-target. The agent interacts with the network environment at time step $t = 1$ and finishes at time step T, which equals the number of epochs multiplied by the number of samples per episode (SPE). At each time step t, the state of the network is observed by querying the controller's global view. This state is represented by the matrix $S_t = [S_1, S_2, \ldots, S_n]^T$, where n is the total number of forwarding nodes excluding sinks. Each vector S_i within this matrix corresponds to the state features of the forwarding node v_i, as detailed in Table 2. The vector for each node, S_i, includes the following state features:

$$S_i = [TS_i, E_i, D_i, T_i, EC_i, DI_i, DE_i, Ptx_i, Btx_i, Prx_i, Brx_i, PD_i, RT_i] \tag{11}$$

Thus, S_t forms a matrix where each row corresponds to the state features of a different node in the network. This matrix structure provides a comprehensive and multidimensional representation of the network's state at any given time step.

The agent begins observing the initial real-time network state S_t at time step $t = 1$ by inquiring about the global network state view of the controller. The policy network's first random action A_t is translated into network flow rules and then installed in the corresponding forwarding nodes. The agent waits for sufficient time, denoted as τ to observe the effect of the policy action in the network. The value of τ is configured based on the size of the data plane domain. The agent then observes the next network state S_{t+1} at time step $t + 1$ and calculates the action reward

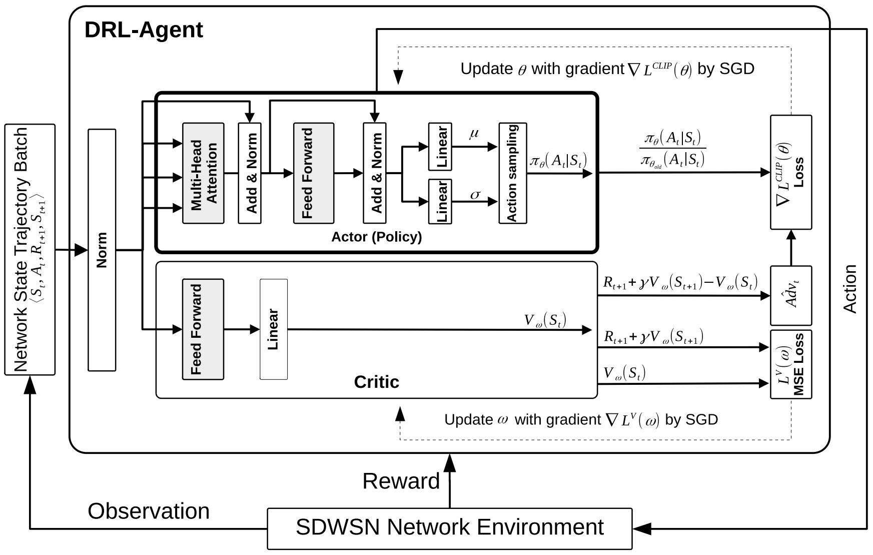
DRL-Agent
Update θ with gradient $\nabla L^{CLIP}(\theta)$ by SGD
Network State Trajectory Batch $\langle S_t, A_t, R_{t+1}, S_{t+1}\rangle$
Norm
Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Linear
Linear
μ
σ
Action sampling
$\pi_\theta(A_t|S_t)$
$\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)}$
$\nabla L^{CLIP}(\theta)$ Loss
Actor (Policy)
$R_{t+1}+\gamma V_\omega(S_{t+1})-V_\omega(S_t)$
$\hat{Adv}_t$
Feed Forward
Linear
$V_\omega(S_t)$
$R_{t+1}+\gamma V_\omega(S_{t+1})$
$V_\omega(S_t)$
$L^V(\omega)$ MSE Loss
Critic
Update ω with gradient $\nabla L^V(\omega)$ by SGD
Action
Reward
Observation
SDWSN Network Environment

FIGURE 2. DRL agent of SSDWSN.

R_{t+1}. For each time step t, the agent stores its experience as a tuple $<S_t, A_t, R_{t+1}, S_{t+1}>$ in its trajectory memory. The size of the trajectory memory is limited to the number of samples per epoch SPE such that tuples are inserted and removed according to the first-in, first-out (FIFO) rule. For every trajectory batch in a training step t in each epoch, the trainer updates the value of the critic network parameters ω by performing a gradient descent step that minimizes the mean squared difference between the target state value $R_t + \gamma V_\omega(S_{t+1})$ and the current state value $V_\omega(S_t)$. It also updates the policy network parameters θ by performing a gradient ascent step that maximizes the clipped surrogate objective function of the minimum clipped or unclipped probability ratio of the advantage function. At the end of each epoch, the agent updates both the cloned target state value parameters ω_{targ} and cloned policy target parameters θ_{target} using the currently updated parameters ω and θ, resets the dataset buffer, and goes through the steps above again. The dataset replay buffer is configured to have a size of one epoch dataset to avoid learning the same features from previous network state experiences in time-varying traffic network conditions, as in SDWSN. The agent can efficiently approximate and learn the advantage of the state-value function of the learning policy to perform an action on the data plane considering the dynamic conditions of the network. Because of the dynamic nature of the network, the PPO is used to improve the agent's training stability by avoiding destructively large weight updates by clipping the ratio of the difference between the current and old policy to a specific range, as in the PPO's surrogate objective function 12. Therefore, the final objective is the lower bound (pessimistic bound) of the unclipped ratio of the probability of taking action A_t at network state S_t in the current policy, over the same probability for the old policy multiplied by the advantage $\hat{Adv}_t$ of the expected (target) network state value to the current network state value.

$$L_\theta^{CLIP} = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{Adv}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{Adv}_t)\right] \tag{12}$$

$$r_t(\theta) = \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)} \tag{13}$$

B. THE PROPOSED DRL-PPO BASED ADAPTIVE TRAFFIC CONTROL POLICY (PPO-ATCP)

The proposed PPO-ATCP policy aims to optimize the traffic overhead and congestion resulting from periodic transmission of the network state to the controller. The policy learns from the online experience of interacting with the network environment to efficiently perform three main actions. The first action is to select the optimal nodes for reporting packet aggregation and redundancy removal. An aggregation node is designated to collect non-redundant report information received from neighboring nodes, and then forward it to the next hop toward the sink. This action selects nodes that are identified as the most energy-efficient and congestion-minimizing. It aims to eliminate redundancy in updating the controller with the network state, thereby reducing unnecessary controller-bound traffic overhead and overall energy consumption. The second action is to apply load-balance routing by adaptively distributing the load of the reported traffic to the controller. This action adds a new forwarding rule in a node to forward report packets generated from the same node using a different next-hop route, whereas the other packets are forwarded using the default established greedy shortest-path forwarding rule toward the nearest sink. The third action is to reschedule the transmission of the report packets within the configured RT. The last two actions are designed to prevent traffic congestion and evenly distribute the traffic load arising from the periodic reporting of the network state to the controller. The following action and

Algorithm 2 SSDWSN DRL Agent Algorithm

```
Require: Samples per epoch $SPE$, batch size $z$, $\epsilon = 0.2$, discount factor $\gamma$ and learning rate $\alpha$, weights $\delta, \eta, \varphi$ and $\rho$, and action waiting time $\tau$
1:  Initialize environment Env
2:  Initialize policy parameters $\theta$
3:  Initialize value parameters $\omega$ and target parameters $\omega_{targ}$
4:  Initialize repeat buffer
5:  Run procedure PLAY_ENV
6:  procedure PLAY_ENV
7:  Observe initial state $S_1$
8:  for sample $t \in [1, SPE]$ do {Observation step}
9:      Select $A_t \sim \pi(S_t|\theta)$
10:     Execute $A_t$ in the network environment $E$
11:     Wait a $\tau$ time, enough to observe the effect of $A_t$ on Env
12:     Observe $S_{t+1}$
13:     Calculate $R_{t+1}$
14:     Insert transition $G_t \leftarrow (S_t, A_t, R_{t+1}, S_{t+1})$ into the buffer
15:     $S_t \leftarrow S_{t+1}$
16: end for
17: end procedure
18: for epoch $\in [1, \infty)$ do
19:     for step $t \in [0, T-1]$ do {Training step}
20:         $S_t, A_t, R_{t+1}, S_{t+1} \leftarrow batch$
21:         Update the value network with SGD:
22:         $L_\omega \leftarrow \frac{1}{|S_t|}[R_{t+1} + \gamma V(S_{t+1}|\omega_{targ}) - V(S_t|\omega)]^2$
23:         $\omega \leftarrow \omega - \alpha \nabla L(\omega)$
24:         Update the policy network with SGA:
25:         $\hat{Adv}_t \leftarrow R_{t+1} + \gamma V(S_{t+1}|\omega_{targ}) - V(S_t|\omega)$
26:         $r_t \leftarrow \frac{\pi_\theta(A_t,S_t)}{\pi_{\theta_{old}}(A_t,S_t)}$
27:         $L_\theta^{CLIP} \leftarrow \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{Adv}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{Adv}_t]$
28:         $\theta \leftarrow \theta + \alpha \nabla L_\theta^{CLIP}$
29:     end for
30:     $\omega_{targ} \leftarrow \omega$
31:     Reset buffer
32:     Run procedure PLAY_ENV
33: end for= 0
```

reward are proposed to learn the optimal dynamic traffic control policy.

1) ACTION

The output of learning policy $\pi_\theta = \mathbb{P}[A|S;\theta], \mathbb{P} \sim \mathcal{N}(A|\mu_\theta, \sigma_\theta)$ is a normal probability distribution over actions A for a given state S. Action A at a time step t is a multivariate normal distribution of n-dimensional random vector $A_t = [A_i, A_{i+1}, \cdots, A_n]^T$, where $A \sim \mathcal{N}(\mu, \Sigma)$ and $\forall i \in [1, n], n = |V|$ number of forwarding nodes excluding sinks. Each action A_i is defined as a vector probability of aggregation nodes (AN), load-balancing next-hop (NH), and transmission time slot of reporting topological and statistical information to the sink (RS), and can be written as follows:

$$A_i = [AN_i, NH_i, RS_i] \tag{14}$$

Given a probability distribution over actions at time step t of mean vector μ_t and standard deviation vector σ_t, an action A_t is sampled as a matrix of action probabilities of an observed network state S_t.

2) REWARD

The main goal of RL is to learn the optimal policy π^* that maximizes the expected cumulative reward R. Thus, we define the reward function R at the next time step $t+1$ as the proposed normalized objective function defined in 10 and can be written as follows:

$$R_{t+1} = \delta \frac{T_{t+1}}{T_{max}} - \eta \frac{D_{t+1}}{D_{max}} - \varphi \frac{EC_{t+1}}{EC_{max}} - \rho Var\left(\frac{EC_{t+1}}{EC_{max}}\right) \tag{15}$$

To perform dynamic aggregation and routing to reduce the communication cost efficiently, utilize network resources, and prolong the network lifetime, the agent continues to learn a policy to perform actions that select aggregation nodes, next-hop forwarding rules, and report-to-controller dute-cycles adaptively to temporal dynamic network changes that maximize the reward.

As presented in Algorithm 3, for each sampled state S_t, an action A_t is generated by policy. The sampled state matrix S_t is equal to the number of active nodes, excluding sinks, multiplied by the number of network state features. Each node's action A_i consists of three random probabilities $A_i = [AN_i, NH_i, RS_i]$, where $AN_i, NH_i, RS_i \in [0, 1]$ and $AN_i + NH_i + RS_i \neq 1$. The probability value of AN_i determines whether a node is designated as an aggregation node. A node is set as an aggregation node if the probability value is greater than the threshold probability $AN_i > \lambda$ as follows:

$$v_{i,an} = \begin{cases} 1, & AN_i > \lambda, \text{ where } \lambda = 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

The probability value of NH_i is transformed into the gradient angle $NH_i = NH_i\pi$. Assuming that the controller has the information of every node's position (x, y) in a graph, the angle θ between v_i and its neighbor v_j is calculated as $\theta = atan2((v_{j,y} - v_{i,y}) - (v_{j,x} - v_{i,x}))$. The next hop for forwarding the generated report packets from node v_i is calculated as follows:

$$val = \begin{cases} NH_i - \theta, & \theta > 0 \\ NH_i - \theta + 2\pi, & \text{otherwise} \end{cases}$$
$$v_{i,nh} = \begin{cases} v_j, & val < \pi \ \& \ d_{i,sk} > d_{j,sk} \\ Null, & \text{otherwise} \end{cases} \tag{17}$$

Algorithm 3 The PPO-ATCP policy-action algorithm.

```
Require: Network topology $G = (V, E)$, state $S_t$ and action
    $A_t$ at a time step $t$, and aggregation selection probability
    threshold $\lambda = 0.5$
0:  procedure ACTION$G, S_t, A_t, \lambda$
1:  for $i \in [1, |S_t|]$ do
2:      $AN_i, NH_i, RS_i \leftarrow A_i$
3:      $N_i \leftarrow$ Neighbors set of node $v_i$
4:      $P_i \leftarrow$ Position of node $v_i$
5:      $NH_i \leftarrow NH_i\pi$
6:      $v_{i,nh} \leftarrow$ Null {Selected next-hop node}
7:      $r_{i,nh} \leftarrow$ Null {Forwarding rule to next-hop node}
8:      $v_{i,an} \leftarrow$ False {Selected aggregation node}
9:      $v_{i,rt} \leftarrow RT_{min}$ {Reporting to controller timeout}
10:     for $j \in [0, |N_i| - 1]$ do
11:         $P_j \leftarrow$ Position of neighbor node $v_j$
12:         $val \leftarrow \pi$
13:         if $atan2(P_j - P_i) > 0$ then
14:             $val \leftarrow \left|NH_i - atan2(P_j - P_i)\right|$
15:         else
16:             $val \leftarrow \left|NH_i - (atan2(P_j - P_i) + 2\pi)\right|$
17:         end if
18:         if $val < \pi \;\&\; d_i > d_j$ then
19:             $v_{i,nh} \leftarrow v_j$
20:         end if
21:     end for
22:     if $AN_i > \lambda$ then
23:         $v_{i,an} \leftarrow$ True
24:     end if
25:     $v_{i,rs} \leftarrow RS_i(RT_{max} - RT_{min}) + RT_{min}$
26:     FLOW_MOD ($v_{i,an}, v_{i,nh}, v_{i,rs}$)
27: end for
28: end procedure
29: procedure RECV_ACTION (FLOW_MOD)
30: $v_{i,an}, v_{i,nh}, v_{i,rs} \leftarrow$ FLOW_MOD
31: $v_{i,an} \leftarrow v_{i,an}$
32: if $v_{i,nh} \neq Null$ then
33:     $r_{i,nh} \leftarrow$ FORWARD_RULE ($v_i, v_{nh}$)
34: end if
35: $v_{i,rs} \leftarrow v_{i,rs}$
36: end procedure $= 0$
```

After this action is applied, node v_i may have two forwarding rules for routing packets to the sink. The first forwarding rule is the base shortest path forwarding rule r_*, which is configured during the link discovery phase. It forwards any packet generated or received at v_i toward the sink v_{sk}. The second forwarding rule is the re-routing rule $r_{v_i,*,rp}$ which forwards only report-type packets (rp) generated in v_i to the sink. Thus, it minimizes overhead on the heavy-loaded routing path, balances energy consumption, and enhances the utilization of link resources. Finally, the normalized probability value of RS_i is transformed into a time slot in the interval of a preconfigured RT as follows:

$$v_{i,rs} = RS_i(RT_{max} - RT_{min}) + RT_{min} \tag{18}$$

The controller agent transmits the three action values $(v_{i,an}, v_{i,nh}, v_{i,rs})$ to the corresponding node and waits for a time τ sufficient to observe the impact of the policy action in the network.

C. THE PROPOSED DRL-PPO BASED NETWORK STATE FORECASTING POLICY (PPO-NSFP)

The proposed PPO-NSFP policy aims to minimize the traffic overhead from periodic transmission of the reporting network state to the controller. The policy learns to forecast network statistical information periodically reported to the controller. During the prediction period, the trained action policy performs a forecasting action to predict the network state. The agent sends instructions to the sensing layer as flow entries to drop report packets. The timeout of the installed dropping flow entries is equal to the prediction time estimated by the agent. The application layer updates the global network state view of the controller based on the actions taken by the trained policy. Once the timeout of the dropping flow rules expires, the sensing layer restarts, signaling the network state to the controller. The following action and reward are proposed to learn the optimal network state forecasting policy.

1) ACTION

A subset of the network state S_t is defined as a vector $B_t = [B_i, \ldots, B_n]^T$, where $B_t \subset S_t$. Each B_i is defined as a vector of a subset of network state features corresponding to a forwarding node v_i as follows:

$$B_i = [E_i, Ptx_i, Btx_i, Prx_i, Brx_i] \tag{19}$$

The output of learning policy $\pi_\theta = \mathbb{P}[A|S;\theta], \mathbb{P} \sim \mathcal{N}(A|\mu_\theta, \sigma_\theta)$ is a normal probability distribution over actions A for a given state S. Action A at a time step t is a multivariate normal distribution of n-dimensional random vector $A_t = [A_i, A_{i+1}, \cdots, A_n]^T$, where $A \sim \mathcal{N}(\mu, \Sigma)$ and $\forall i \in [1, n], n = |V|$ number of forwarding nodes excluding sinks. Each action A_i is defined as a vector probability of residual energy (E_i), transmitted packets (Ptx_i), transmitted bytes (Btx_i), received packets (Prx_i) and received bytes (Brx_i), and can be written as follows:

$$A_i = [E_i, Ptx_i, Btx_i, Prx_i, Brx_i] \tag{20}$$

Given a probability distribution over actions at time step t of mean vector μ_t and standard deviation vector σ_t, an action A_t is sampled as a matrix of action probabilities of an observed network state S_t.

2) REWARD

The main goal of RL is to learn the optimal policy π^* that maximizes the expected cumulative reward R. Thus, we define the reward function R at the next time step $t + 1$ as

Algorithm 4 The PPO-NSFP Policy-Action Algorithm

```
Require: Network topology $G = (V, E)$, state $S_t$, observed
    action features $B_t \subset S_t$ and forecasting action $A_t$ at a time
    step $t$, and prediction time $\tau$ in seconds.
 0: procedure ACTION $G, S_t, A_t, B_t, \tau$
 1:   $A_t \leftarrow B_t + B_tA_t$
 2:   for $i \in [1, |V|]$ do
 3:       FLOW_MOD(DROP_RULE($r_{v_i,*,rp}$)) {Drop report
          packets ($rp$) sent from $v_i$}
 4:   end for
 5:   for $t \in [1, |\tau|]$ do
 6:       UPDATE_GRAPH($A_t$)
 7:       $\tau \leftarrow \tau - 1$
 8:   end for
 9: end procedure
10: procedure RECV_ACTION (FLOW_MOD)
11:   $r_{v_i,*,rp} \leftarrow$ FLOW_MOD
12:   if $v_{i,nh} \neq Null$ then
13:       ADD_RULE($r_{v_i,*,rp}$)
14:   end if
15: end procedure = 0
```

a function of current observed action features B_t, forecasted action A_t, and the target observed action features B_{t+1}, where $\forall i \in [1, n], n = |B_t|$ & $\forall j \in [1, m], m = |B_t^T|$. The reward function is expressed as follows:

$$R_{t+1} = \sum_{j=1}^{n} \frac{((B_t + B_tA_t) - B_t)(B_{t+1} - (B_t + B_tA_t))}{(max(B_t))^2} \tag{21}$$

$$R_{t+1} = \sum_{j=1}^{n} \frac{B_tA_t(B_{t+1} - B_t - B_tA_t)}{(max(B_t))^2} \tag{22}$$

To efficiently reduce the communication cost and prolong the network lifetime, the agent continues to learn a policy that performs actions to predict the network state at a prediction time τ equal to the observation time. During the prediction time, the agent sends instructions to the data plane to stop reporting the network state to the controller and updates the controller's global network view using the action predicted by the trained policy.

As presented in Algorithm 4, the action probability A_t is transformed into action values, such that $A_t = B_t(1 + A_t)$. For each node v_i in the observed state S_t, a dropping flow rule is sent in a control packet to the node to drop any report packet generated from the same node. The controller's global state view is updated using the transformed predicted action A_t in a time step equal to prediction time τ. To provide more details on the proposed approach, a visual representation of the forecasting action of the network state is shown in Fig. 3.

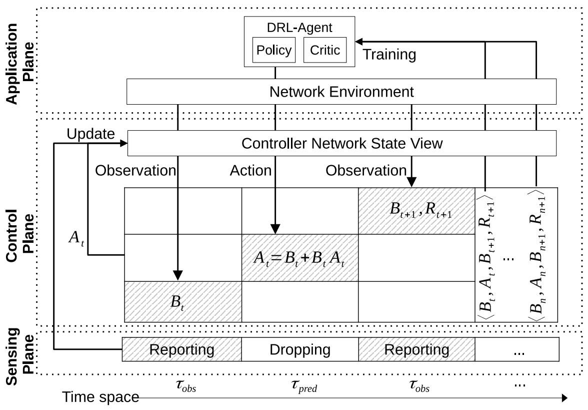

FIGURE 3. PPO-NSFP policy-action pipeline.

V. PERFORMANCE EVALUATION

In this section, we evaluate the scalability and performance of our proposed approach. We present the network topology and traffic simulation setup, and then discuss the results.

A. EXPERIMENTAL SETUP

Experimenting with the scalability of SDWSN networks requires large-scale network infrastructure. However, such experiments can be difficult and expensive to implement because of the large number of nodes, heterogeneous technologies, and complex configurations. Therefore, a simulation is a more practical and appropriate method for such experiments. Moreover, the available SDN simulators could not align accurately with the experimental specifications and requirements of this study. A simulator was implemented and built on the OpenFlow implementation in SD-WISE [7], [8], [9]. We made the source code implementation of the SSDWSN publicly available in [14] for future research and improvement.

The SSDWSN data plane module primarily consists of a spawned process called *Node* to simulate the asynchronous behavior of actual networking nodes. Every created data plane node (e.g., Sink, Mote) is a spawned process that works asynchronously and has asynchronous workers and blocked threads to handle the transmitted and received packets into and from FIFO non-blocking queues attached to the node's endpoint. Each node is equipped with an IEEE 802.15.4 transceiver. Two types of nodes can be created, namely sinks and motes. The structures of both the nodes are similar, with three main differences in their functionalities. The sink is a gateway between the controller and sensing layer and has a rechargeable battery. By contrast, the mote node indirectly communicates with the controller, is equipped with a sensing device, and has a dischargeable battery. The sink node communicates with the wireless nodes through a user datagram protocol (UDP) full-duplex socket. This channel simulates the radio transceiver in a wireless sensor node. Different antenna and radio transmission parameters can be configured to define the received RSSI, that is, the antenna height, gain, range, transmission power, and frequency. The SSDWSN control plane module mainly consists of a spawned process called a controller that simulates the asynchronous behavior of an actual SDN controller. A controller can be hosted in a sink or a remote server. As a practical solution, the controller is independent software hosted in remote server clusters directly connected to the network sinks. One or more controllers can be created, and every spawned controller has asynchronous workers to handle the transmitted and received packets into/from the FIFO non-blocking queues. A propagation model and various sensor types (e.g., temperature, humidity, pressure, optics, and sound) were implemented to increase the level of realism. An extensive explanation of these models is presented in this section. The rest of this section is organized as follows: the configuration of the network simulation, setup for traffic simulation, and scope and selected performance metrics of the simulation process. The scalability and performance metrics were selected based on previous studies [44], [47].

1) NETWORK SIMULATION SETUP

This study focuses on the application of WSNs in areas such as detecting and monitoring wildfires, particularly in forested or hard-to-reach regions. These regions, which are frequently prone to wildfires, present challenging terrain and varied vegetation, thus necessitating dense deployment of sensors. This approach ensures detailed data collection and facilitates early detection of potential risks. In remote and intricate environments, employing SDN is crucial for effective network management and control. However, this setup presents its own set of challenges. A dense network of sensors generates a high volume of traffic to keep the SDN controller updated with the network's status, leading to increased energy consumption and the risk of network congestion. To address these challenges, our approach proposes to minimize the energy consumption associated with frequent updates to the SDN controller.

In addition, our approach is relevant in scenarios that require intensive environmental monitoring. In ecological and environmental research, for instance, studying micro-climates or forest biodiversity, a high sensor density is instrumental for detecting subtle environmental variations and delivering comprehensive datasets. Similarly, in regions experiencing rapid ecological shifts due to climate change, intensive monitoring is vital for accurately tracking these changes and understanding their impact on local ecosystems. In urban settings, the deployment of a high-density sensor network serves multiple purposes, including traffic monitoring, pollution tracking, and noise-level assessment. Although deploying up to 250 nodes per 1 Km^2 may appear excessive for certain applications, it is justified for comprehensive, high-resolution urban monitoring. To model the signal propagation in these challenging forested or remote environments, the log-distance propagation model [48] was used with an exponent value of three. The model is used to predict the propagation loss for a wide range of environments and considers random shadowing effects owing to signal blockage by hills, dense foliage, varying terrain, and buildings.

At the MAC sub-layer, IEEE 802.15.4, is used to represent the LR-WPANs wireless transceiver with a data rate of 250 kbps at a channel bandwidth of 5 MHz and a frequency of 2.4 GHz. The inter-transmission time between consecutive beacons (hello packets) was set to 1 s and had a size of 30 bytes. The UDP was used in the network layer to simulate wireless communication between the nodes. The FIFO queuing strategy was used to buffer incoming and outgoing packets with a maximum capacity of 25 buffering packets. To effectively gather data features from sensor nodes, the proposed approach leverages SD-WISE [7], [8], customized to suit node constraints and prioritize crucial sensor-specific information, such as energy usage, RSSI, and node communication states. The process involves computing end-to-end delay and throughput at the sink node upon data packet arrival and reporting these findings to the controller. Additionally, metrics such as hop distance to the sink and network density are obtained from the controller's maintained global state, aligning the data collection with SDWSNs requirements, distinct from the standard OpenFlow protocols built for wired networks. These collected features play a crucial role for the DRL-Agent to continually act at the sensing layer, allowing the effective assessment of rewards and policy refinement using updated network state information. Our approach implements a 5 s reporting interval, balancing data consistency and energy efficiency, as listed in Table 3. This interval is managed by the PPO-ATCP policy to optimize transmission times within this window, reduce congestion, and enhance data precision. This dynamic adjustment of reporting times based on network conditions optimizes resource utilization, ensures timely data collection, and enables efficient network traffic management to improve overall performance.

The communication type is Constant Bit Rate (CBR), and the maximum transmission unit (MTU) is 127 bytes. According to the studies in [44] and [47], the number of deployed nodes used to evaluate the scalability of SDWSN ranges from 50 to 300. In this study, the network size varied between [25, 250] sensor nodes communicating with one sink node. All sensor nodes, excluding sinks, simultaneously work as sensing and intermediate nodes and generate sensing data randomly. The transmission range of each node was set to 100 m, and the total simulation time was 3600 s. The experimental parameters are listed in Table 3.

2) TRAFFIC SIMULATION SETUP

To generate realistic simulation traffic, the following simulation models and hardware simulators were used.

The log-distance propagation model is a propagation model simulates the random shadowing effects of blocking the signal by the surrounding barriers (e.g., buildings, trees, hills) [48]. The equation for calculating the log path loss at

TABLE 3. Experiment parameters.

Parameter	Value
Number of epochs	inf[a]
Transmission Range	100 m
Samples per epoch	1
MAC Protocol	IEEE 802.15.4
Batch size	250
MTU (Data Packet)	127 Bytes
Hidden layer size	256
Initial Energy	15×10^6 mJ
Actor learning rate	10^{-4}
Buffer Size	25 MTUs
Critic learning rate	10^{-3}
Mote module	MICAz 2.4 GHz
Discount factor	0.99
Application	Temperature monitoring
Clipping coefficient	0.3
Max delay	200 ms
Entropy coefficient	0.1
Report duty-cycle	5 s
Optimization iteration	4
Simulation Time	3600 s
Action time	10 s
Monitoring Area	1 Km^2
Observation time	10 s
number of nodes	25, 50, 75, 100, 125, 150, 175, 200, 225, and 250

[a]The Number of epochs is infinite and restricted to the simulation time or network partition occurrence.

an arbitrary distance $d > d_0$ is given below, where $P_L(d_0)$ (close-in reference distance) is the path loss at a distance d_0 m from the transmitter for a distance beyond d_f and n is the path loss exponent, which depends on the type of environment. In addition, X is a random variable of a standard normal distribution with *sigma* expressed in dB. Typically, $d_0 = 1$ to 10 m for the microcell and $d_0 = 1$ Km for a large cell.

$$[P_L(d)]dB = [P_L(d_0)]dB + 10\,n\,log_{10}\left(\frac{d}{d_0}\right) + X$$
$$for\ d_f \leq d_0 \leq d \qquad (23)$$

Depending on the simulation environment, the path loss exponent n can be one of the following values: free space = 2, obstructed in building = 4 to 6, inside a building (line-of-sight) = 1.6 to 1.8, shadowed urban cellular radio=3 to 5, urban area cellular radio = 2.7 to 3.5, and obstructed in factory = 2 to 3. In this study, we employed a path loss exponent of three to model signal propagation in environments with varied terrain and dense vegetation. This exponent value is critical for accurately simulating how signals traverse through such complex settings, thereby enhancing the reliability of our wireless sensor network under these challenging conditions.

IEEE 802.15.4 and 6LoWPAN in Linux is a library that simulates the IEEE 802.15.4 MAC protocol and resides inside the main Linux kernel network stack [49]. It allows the use of all 6LowWPAN capabilities and features and can be configured using the wpan-tools distribution. The configuration interface for the ieee802154 subsystem is exposed over the nl802154 netlink interface and allows the configuration of various phy and MAC layer parameters and properties. In this study, wpan-hsim was used to configure the hardware simulator driver.

Data sampling and sensor modeling are different types of sensors that are implemented to simulate the behavior of the proposed solutions under different QoS requirements. In this simulation, all the sensor nodes were configured to transmit temperature data. A mathematical representation of these sensors is presented in Table 4. During the simulation process, temperature sensing data were randomly generated to simulate time-varying dynamic traffic conditions, as listed in Table 4. The simulation scope, assumptions, and limitations are as follows:

- The nodes used in the simulation were the wireless sensors, sinks, and SDN controllers.
- Network topology is unchanged over the entire transmission frame.
- All sensing layer nodes operate as sensor and intermediate nodes simultaneously.
- For simplicity, and to test the proposed approach under high-energy consumption environments, the nodes remained active during the simulation time.
- The entire simulation process was executed in one workstation with the specification of 12 cores CPU Intel (R) Core (TM) i5-10600K 4.10 GHz unlocked, GPU-NVIDIA 12288MiB, and 16 GB RDD4 RAM.
- There is always a line-of-sight between each node and its one-hop neighbors. However, the Log-Distance propagation model [48] has been used to evaluate the network scalability and performance under realistic channel propagation conditions.

B. SIMULATION RESULTS AND DISCUSSION

This section presents the simulation results for each proposed policy, based on the performance and learning metrics discussed in Section III. We considered various network sizes to measure the scalability of applying the proposed policies while increasing the number of nodes. In each topology, a sink node is located centrally, and the worst-case scenario is performed in which all nodes transmit sensing data, and the transmission range is equal to 100 m, including the sink node. As a result, the majority of nodes must traverse multiple hops to reach the sink.

Fig. 4 shows the average end-to-end delay required to deliver packets to the nearest sink. It can be observed that the delay and packet loss started to increase rapidly as soon as the network reached the size of 175 nodes due to the highly congested number of packets. The implementation of the proposed PPO-ATCP policy led to a reduction in the number of dropped packets. This reduction is achieved by dynamically balancing traffic loads, rescheduling reporting packets, and aggregating non-redundant traffic, as shown in Fig. 5. Compared to SDWSNs without the proposed control policies, the action performed by PPO-ATCP was able to maintain a slow increase in delay and packet loss as the size of the network increased. Furthermore, most dropped

TABLE 4. Numerical representation of the simulated sensors.

Type	Value	Sensing data interval
Temperature	$X \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu \sim \mathcal{U}(a,b)$, $a \leq x \leq b$, and $a = 5\,\text{s}$ and $b = 15\,\text{s}$, and $\sigma = 10C^o$	$X \sim \mathcal{U}(a,b)$, where $a \leq x \leq b$, and $a = -30C^o$ and $50C^o$
Switch-On/Off	$(0, 1)$	$X \sim \mathcal{U}(a,b)$, where $a \leq x \leq b$, and $a = 0.1\,\text{s}$ and $b = 15\,\text{s}$
Ambient Sound	Real noncompressed sound	100 samples/second
Camera	MPEG2 video of equal frames size and bit rate $X \sim \mathcal{U}(a,b)$, where $a = 50\,\text{kbps}$ to $b = 200\,\text{kbps}$	$X \sim \mathcal{U}(a,b)$, where period of motion value $a \leq x \leq b$ and $a = 1\,\text{s}$ and $b = 5\,\text{s}$

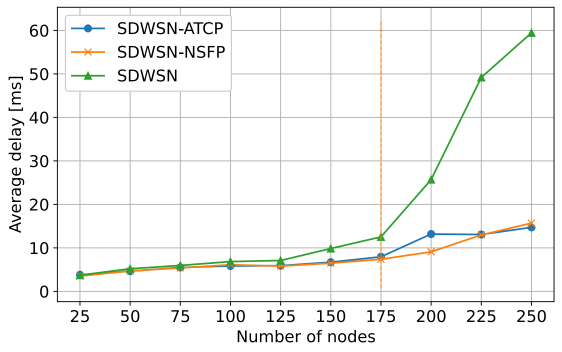

FIGURE 4. SSDWSN PPO-ATCP and NSFP performance evaluation - delay.

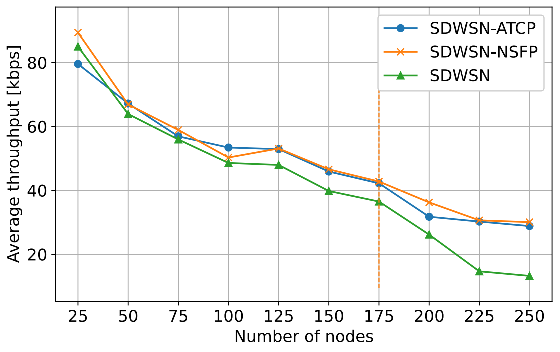

FIGURE 6. SSDWSN PPO-ATCP and NSFP performance evaluation - throughput.

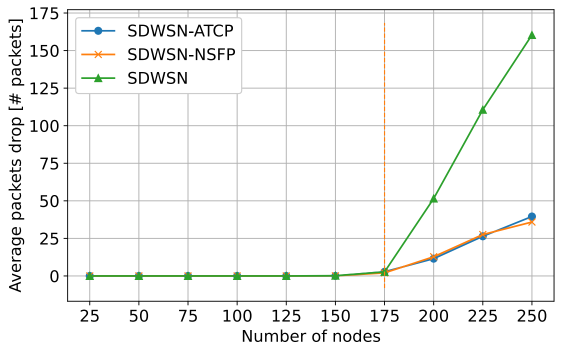

FIGURE 5. SSDWSN PPO-ATCP and NSFP performance evaluation - packets drop.

packets occurred before or during the initial observation and application of the first random policy action. The PPO-NSFP policy also exhibited the same performance, with a slightly better delay and packet loss owing to the decrease in the number of packets transmitted to the controller. However, PPO-NSFP reduces the number of report packets without performing any other optimization on the control traffic. It also reduces delay and packet loss, as in PPO-ATCP. These results prove that the source of traffic overhead is primarily caused by the transmission of local information to the controller to maintain global network state knowledge.

As shown in Fig. 6, the Average Number of packets delivered per unit time from the source nodes to the nearest sink decreases rapidly as the network size increases, because of the massive amount of traffic transmitted to the controller. PPO-ATCP and PPO-NSFP exhibited robust and similar packet delivery rates. However, PPO-ATCP outperforms PPO-NSFP in terms of delay and throughput for large-scale networks because it reduces the amount of traffic without efficiently utilizing network resources.

In the presented analysis of energy consumption rate variance in Fig. 7, a distinct correlation was observed between the increase in network size, ranging from 25 to 250 nodes, and the increase in energy consumption variance. This trend suggests a challenge in maintaining energy efficiency in larger networks, possibly owing to amplified network traffic and consequent collision or retransmission occurrences. Notably, the accelerated energy depletion of nodes proximal to the sink, a critical issue in network management, underscores the necessity of adaptive clustering mechanisms in dense networks [33]. However, in sparse multi-hop SDWSN architectures, this approach alone does not sufficiently address the imbalance in the energy distribution near the sink, leading to a reduction in the overall network lifetime. In contrast to SDWSNs without the proposed control policies, PPO-NSFP and PPO-ATCP exhibited significantly lower variances in energy consumption, indicating superior energy management. This energy utilization efficiency suggests that PPO-NSFP and PPO-ATCP contribute to prolonged network lifetimes and enhanced reliability. The results of this study emphasize the importance of integrating advanced network management protocols in large-scale or highly dense networks to optimize energy consumption and enhance network sustainability.

Fig. 8 shows the average remaining energy for each simulation scenario. The results showed a rapid decay in the overall energy as the network size increased because of the increased frequency of communication with the

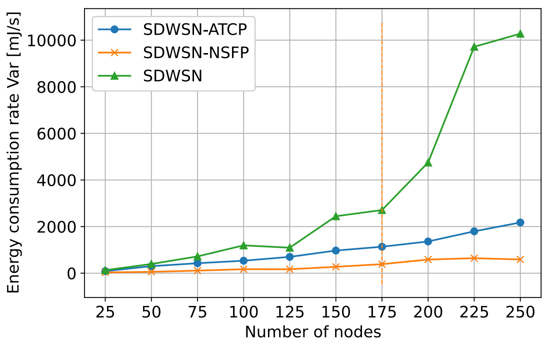

FIGURE 7. SSDWSN PPO-ATCP and NSFP performance evaluation - energy consumption variance.

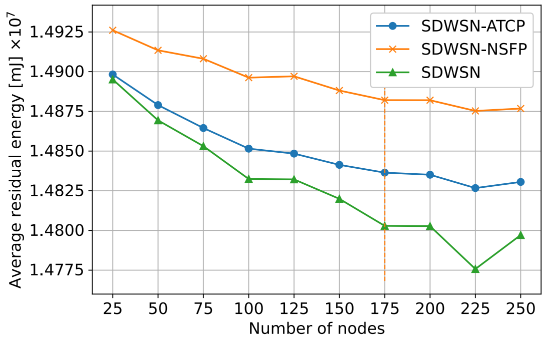

FIGURE 8. SSDWSN PPO-ATCP and NSFP performance evaluation - residual energy.

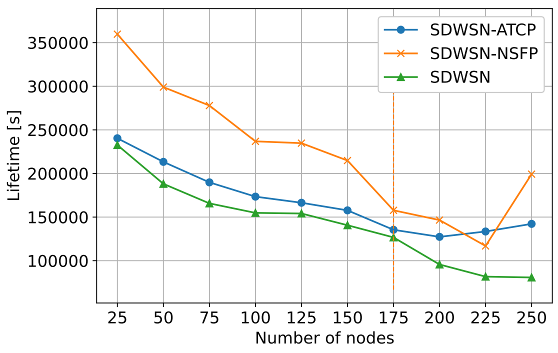

FIGURE 9. SSDWSN PPO-ATCP and NSFP performance evaluation - lifetime.

controller to report the topological and statistical information. By removing the redundancy in the traffic transmitted to the Controller, PPO-ATCP saved 28% of the energy consumed in the same network size without applying the policy, as explained later in this discussion. In contrast, PPO-NSFP saves approximately 53% of the energy consumed by the same network size without applying the policy, thereby significantly prolonging the network lifetime.

Fig. 9 illustrates a comparative study of network lifetime under varying node densities, emphasizing the advancements in SDWSN when employing PPO-ATCP and PPO-NSFP policies over traditional SDWSN setups. The results reveal that, as the node count increases, traditional SDWSN networks exhibit a pronounced decline in lifetime, which reflects the escalating energy demands and management complexities in denser configurations. In contrast, the integration of PPO-ATCP and PPO-NSFP demonstrated a more resilient network performance with notably slower rates of lifetime reduction. The PPO-NSFP, leveraging a predictive algorithm, effectively extends network lifetime, underscoring the utility of anticipatory energy management strategies in sustaining network operations. Remarkably, PPO-ATCP demonstrated an enhancement in network lifetime at higher node densities, notably outperforming PPO-NSFP once the network size exceeded 175 nodes. This turning point suggests that in highly dense networks, the dynamic traffic management capabilities of PPO-ATCP become increasingly critical, potentially mitigating the energy-intensive impact of high node interactivity and communication overhead.

The results further revealed the intricate balance required in network protocol design, in which both predictive state management and adaptive traffic control play pivotal roles. This suggests that, while PPO-NSFP excels in scenarios with moderate node densities through efficient energy distribution, PPO-ATCP's strength lies in its ability to manage the compounded network traffic challenges presented in large-scale deployments. These insights are particularly relevant for the design and optimization of networks in scenarios such as smart cities and IoT systems, where network density and reliability are critical factors. Overall, the findings shown in Fig. 9 demonstrate that the adoption of DRL-based policy-driven approaches in SDWSNs can significantly enhance network lifetime and energy efficiency, particularly when addressing scalability issues.

The average number of packets transmitted from the sinks to the controller is shown in Fig. 10. Because the simulation used only one sink node, this number is equal to the maximum number of packets transmitted from the sink node to the controller. It can be observed that the controller overhead is directly proportional to the network size and increases rapidly. Synchronizing the network state with the controller to maintain centralized fine-grained global topological and statistical network knowledge burdens the network traffic and degrades network performance and scalability. However, the results show that the traffic overhead in SDWSN decreases as the network size reaches 175 nodes owing to the rapid packet drop from network congestion. PPO-ATCP reduces control traffic overhead by adaptively removing redundancy in reporting packets while maintaining a centralized fine-grained network view. The results show that by applying the PPO-ATCP control policy, the control traffic slowly increases as the network size increases, which saves network resources and prolongs network lifetime. PPO-NSFP, on the other hand, significantly reduces the control traffic by predicting the network state and letting the DRL agent update the controller instead of receiving the traffic from the sensing plane for a configurable amount of prediction time. Depending on the accuracy of the prediction model, which we will discuss later, PPO-NSFP can save network energy and guarantee a long network lifetime.

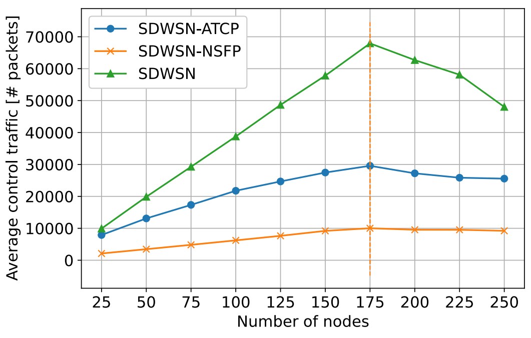
FIGURE 10. **SSDWSN PPO-ATCP and NSFP performance evaluation - controller overhead.**

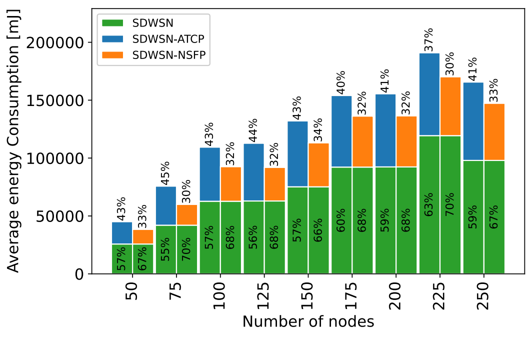
FIGURE 12. **Percentage of energy consumption of SDWSN-ATCP/NSFP vs. SDWSN.**

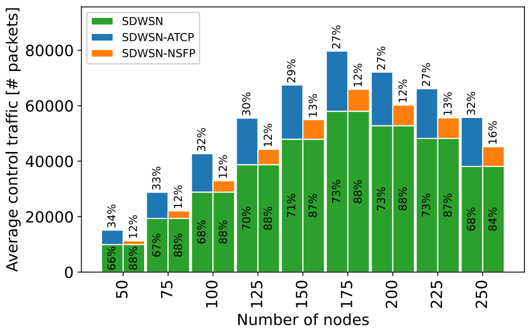
FIGURE 11. **Percentage of control traffic overhead of SDWSN-ATCP/NSFP vs. SDWSN.**

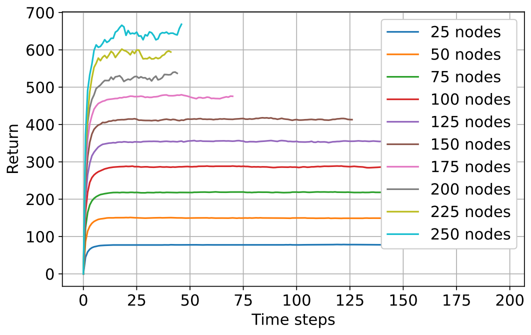
FIGURE 13. **SSDWSN PPO-ATCP return.**

The portion of control packets transmitted to the controller compared to the total traffic transmitted before and after applying the proposed policy was calculated for different network sizes to evaluate the scalability of the proposed approach, as shown in Figs. 11 and 12. The Percentage of improvement that PPO-ATCP and PPO-NSFP contribute to reducing the overall control traffic overhead and energy consumption is calculated as follows:

$$\zeta = \frac{1}{n}\sum_{i=1}^{n}\frac{Ptx_i - Ptx_0 - Ptx'_i + Ptx'_0}{Ptx_i - Ptx_0} \times 100 \tag{24}$$

$$\vartheta = \frac{1}{n}\sum_{i=1}^{n}\frac{E_0 - E_i - E'_0 + E'_i}{E_0 - E_i} \times 100 \tag{25}$$

where Ptx_0 and E_0 are the average number of packets transmitted to the controller and the residual energy, respectively, resulting from the smallest network size simulation, $\forall i \in [1, n]$, n is the number of experimental scenarios excluding the experiment with a small network size. Ptx' and E' are the average number of transmitted packets to the controller and the residual energy, respectively, after applying the proposed policy. Accordingly, it can be observed that PPO-ATCP reduces the overall controller traffic by $\zeta^{ATCP} = 57\%$ less than the exact network size without applying the policy. In addition, it reduces the overall energy consumption by $\vartheta^{ATCP} = 28\%$ more than that of the same network size without applying the policy. However, PPO-NSFP reduces the overall controller traffic by $\zeta^{NSFP} = 85\%$ less than that of the exact network sizes without applying the policy, and reduces the overall energy consumption by $\vartheta^{NSFP} = 53\%$ more than that of the same network size without applying the policy.

The proposed DRL agent is trained using the observation trajectory collected through interactions with the network environment. It continuously observes the synchronized network state view in the controller and updates the learning parameters of the policy and critic models based on the estimated advantage of the expected (target) network state value over the current network state value. As shown in Figs. 13 and 14, the cumulative reward increased and stabilized after a few epoch time steps. It can be observed that large-scale networks require more time steps to reach their peak accumulative reward and converge to the optimal policy compared with lower-scale networks. However, owing to the dynamic and stochastic nature of network traffic, the cumulative reward can temporarily fluctuate, which can be observed from the fluctuation of the policy (actor) loss, as shown in Figs. 15 and 16.

The mean squared error losses of the target and current critic network values of PPO-ATCP and PPO-NSFP are shown in Figs. 17 and 18, respectively. The PPO-ATCP policy exhibits a higher critic loss than the PPO-NSFP because of its more complex role in actively optimizing network traffic, which involves evaluating changing conditions and adjusting the policy parameters accordingly. In contrast, PPO-NSFP, which focuses primarily on predicting network states, deals with a less complex task involving more consistent patterns, allowing it to achieve faster convergence. Therefore, the complexity of tasks directly influences the

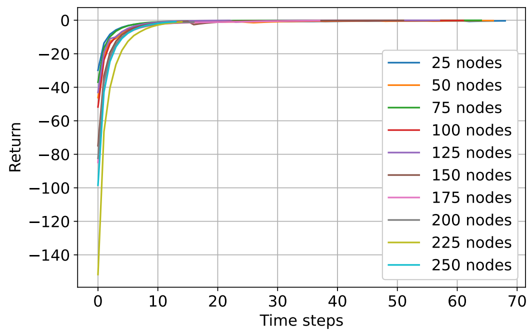

FIGURE 14. SSDWSN PPO-NSFP return.

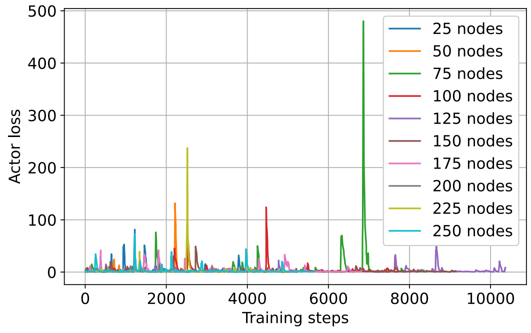

FIGURE 15. SSDWSN PPO-ATCP actor loss.

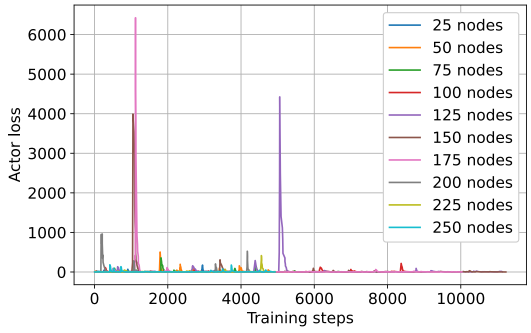

FIGURE 16. SSDWSN PPO-NSFP actor loss.

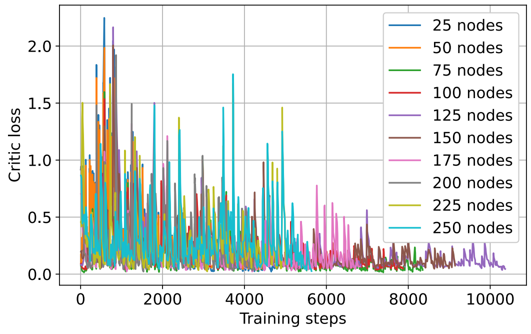

FIGURE 17. SSDWSN PPO-ATCP critic loss.

learning process, with PPO-ATCP's intricate optimization leading to higher initial losses before convergence. The critic loss gradually decreases as the agent learns the optimal policy. Because the PPO-NFSP is a forecasting policy, the actual observation of the network state during the prediction

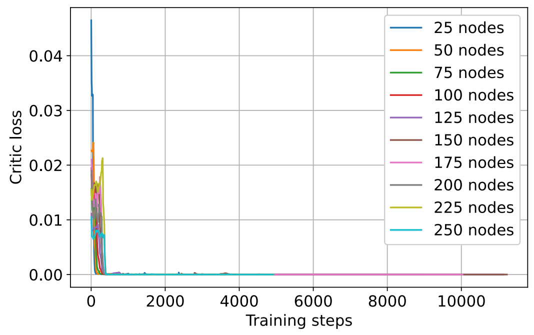

FIGURE 18. SSDWSN PPO-NSFP critic loss.

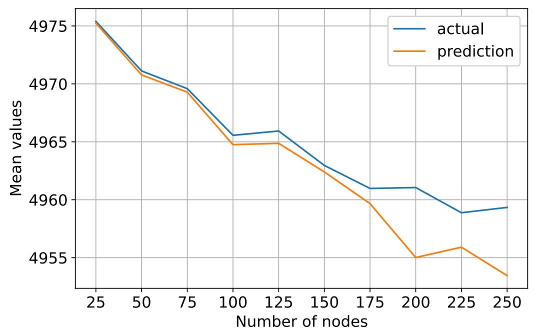

FIGURE 19. SSDWSN PPO-NSFP mean actual vs. prediction-testing.

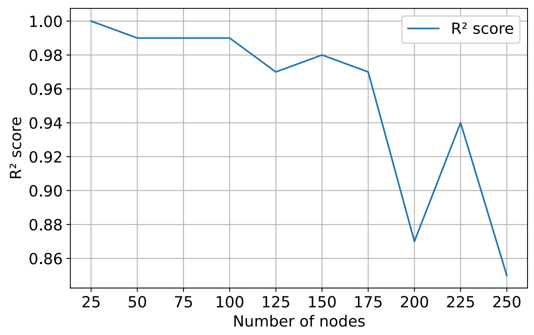

FIGURE 20. SSDWSN PPO-NSFP R2 score - testing.

time is not synchronized with the controller because the reporting packets are intentionally dropped to save network resources. To evaluate the forecasting policy, a copy of the dropped observed network state is saved during the prediction time to test the accuracy of the proposed policy. Fig. 19 shows the difference between the mean actual and predicted values for each simulation scenario. The results show that smaller networks converged faster than larger ones, as shown in Fig. 20, where the R^2 score gradually decreased as the network size increased. The R^2 score shows that the PPO-NSFP policy can predict the network state (e.g., battery level and transmitted/received packets in each node) with approximately 85% accuracy when 250 nodes are reported to one sink node.

Although the aforementioned results are promising, the implementation of DRL in real-time SDWSNs encounters significant challenges owing to potential delays in state observations, consequently impacting the promptness of decision-making actions. The inherent high variability and rapid changes, characteristic of highly time-varying WSNs, add further complexity to the application of DRL in such environments. This challenge primarily stems from the consistency issue within SDN, which is especially prominent in multi-hop, highly time-varying WSN setups. The observations made by the DRL agent may not align consistently with the actual network status, potentially leading to outdated actions. Overcoming this issue requires dedicated research efforts aimed at enhancing the consistency of observations in SDWSNs, particularly in multi-hop high-time-varying network scenarios. Achieving a balance between maintaining high consistency and optimizing the performance of the DRL agent involves dynamically adapting reporting intervals based on prevailing network conditions. Furthermore, adopting predictive methodologies, as proposed in the PPO-NSFP approach, holds promise in mitigating this challenge by reducing the necessary traffic to update the controller with network state information during prediction intervals, while ensuring sustained high consistency during observation intervals.

VI. CONCLUSION

This study introduces an SDWSN framework (SSDWSN) to improve network performance and scalability. Two on-policy PPO-based learning models, PPO-ATCP and PPO-NSFP, were proposed to efficiently utilize SDWSN network resources and predict the network state (e.g., battery level and transmitted/received packets in each node). The results showed that PPO-ATCP avoids traffic congestion by adapting actions on the sensing layer for duty rescheduling, redundancy removal, and load balance routing of the reporting traffic to the controller. The simulation results confirm that the proposed PPO-ATCP and PPO-NSFP policies effectively reduce controller-bound traffic overhead by 57% and 85%, and improve energy efficiency by 28% and 53%, respectively, in SDWSNs. Additionally, PPO-NSFP achieved a minimum accuracy of 85% in network state prediction under similar network size scenarios. The results prove that the proposed approach significantly reduces network energy consumption and prolongs network lifetime. Furthermore, the accuracy of the proposed learning model can be improved by dynamically tuning the prediction time and other network hyperparameters to fit network conditions and complexity.

In future studies, we plan to enhance the control traffic optimization and network state prediction in WSNs using a DRL multi-agent approach and the two proposed control policies simultaneously. In addition, we plan to extend our approach to adaptively manage the dynamic network topologies. To advance the practicality and effectiveness of our approach, we will conduct comprehensive evaluations, including simulations and real-world deployments, while also comparing our method with other RL-based approaches. Furthermore, we will focus on adapting the OpenFlow protocol for WSNs to enhance the applicability of SDN in resource-constrained environments.

ACKNOWLEDGMENT

Mohammed Alsaeedi would like to thank the University of Technology Malaysia for providing the research environment and the facilities for supporting this research.

REFERENCES

[1] F. F. Jurado-Lasso, K. Clarke, A. N. Cadavid, and A. Nirmalathas, "Energy-aware routing for software-defined multihop wireless sensor networks," *IEEE Sensors J.*, vol. 21, no. 8, pp. 10174–10182, Apr. 2021.

[2] S. Rout, K. S. Sahoo, S. S. Patra, B. Sahoo, and D. Puthal, "Energy efficiency in software defined networking: A survey," *Social Netw. Comput. Sci.*, vol. 2, no. 4, pp. 1–15, Jul. 2021.

[3] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[4] S. Buzura, B. Iancu, V. Dadarlat, A. Peculea, and E. Cebuc, "Optimizations for energy efficiency in software-defined wireless sensor networks," *Sensors*, vol. 20, no. 17, p. 4779, Aug. 2020.

[5] A. Hawbani, X. Wang, L. Zhao, A. Al-Dubai, G. Min, and O. Busaileh, "Novel architecture and heuristic algorithms for software-defined wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2809–2822, Dec. 2020.

[6] *OpenFlow Switch Specification*, Standard ONF TS-025, 2015.

[7] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 513–521.

[8] A.-C. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SD-WISE: A software-defined wireless sensor network," *Comput. Netw.*, vol. 159, pp. 84–95, Aug. 2019.

[9] S. S. G. Shiny and K. Murugan, "TSDN-WISE: Automatic threshold-based low control-flow communication protocol for SDWSN," *IEEE Sensors J.*, vol. 21, no. 17, pp. 19560–19569, Sep. 2021.

[10] A. Rahimifar, Y. Seifi Kavian, H. Kaabi, and M. Soroosh, "An efficient Markov energy predictor for software defined wireless sensor networks," *Wireless Netw.*, vol. 28, no. 8, pp. 3391–3409, Nov. 2022.

[11] A. Shafique, M. Asad, M. Aslam, S. Shaukat, and G. Cao, "Multi-hop similarity-based-clustering framework for IoT-oriented software-defined wireless sensor networks," *IET Wireless Sensor Syst.*, vol. 12, no. 2, pp. 67–80, Apr. 2022.

[12] R. Huang, W. Guan, G. Zhai, J. He, and X. Chu, "Deep graph reinforcement learning based intelligent traffic routing control for software-defined wireless sensor networks," *Appl. Sci.*, vol. 12, no. 4, p. 1951, Feb. 2022.

[13] G. A. N. Segura and C. B. Margi, "Centralized energy prediction in wireless sensor networks leveraged by software-defined networking," *Energies*, vol. 14, no. 17, p. 5379, Aug. 2021.

[14] M. Alsaeedi. (2023). *Scalable Software-Defined Wirless Networks Simulator (SSDWSN)*. [Online]. Available: https://github.com/moalsaeedi/ssdwsn

[15] M. U. Younus, S. U. Islam, and S. W. Kim, "Proposition and real-time implementation of an energy-aware routing protocol for a software defined wireless sensor network," *Sensors*, vol. 19, no. 12, p. 2739, Jun. 2019.

[16] A. Banerjee and D. Hussain, "SD-EAR: Energy aware routing in software defined wireless sensor networks," *Appl. Sci.*, vol. 8, no. 7, p. 1013, Jun. 2018.

[17] R. Kumar, U. Venkanna, and V. Tiwari, "EOMCSR: An energy optimized multi-constrained sustainable routing model for SDWSN," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 1650–1661, Jun. 2022.

[18] B. B. Letswamotse, R. Malekian, and K. M. Modieginyane, "Adaptable QoS provisioning for efficient traffic-to-resource control in software defined wireless sensor networks," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 6, pp. 2397–2405, Jun. 2020.

[19] Z. Ding, S. Xing, F. Yan, W. Xia, and L. Shen, "An interference-aware energy-efficient routing algorithm with quality of service requirements for software-defined WSNs," *IET Commun.*, vol. 13, no. 18, pp. 3105–3116, Nov. 2019.

[20] G. Li, S. Guo, Y. Yang, and Y. Yang, "Traffic load minimization in software defined wireless sensor networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1370–1378, Jun. 2018.

[21] B. B. Letswamotse, R. Malekian, C. Chen, and K. M. Modieginyane, "Software defined wireless sensor networks and efficient congestion control," *IET Netw.*, vol. 7, no. 6, pp. 460–464, Nov. 2018.

[22] J. Wang, Y. Miao, P. Zhou, M. S. Hossain, and S. M. M. Rahman, "A software defined network routing in wireless multihop network," *J. Netw. Comput. Appl.*, vol. 85, pp. 76–83, May 2017.

[23] F. F. Jurado-Lasso, L. Marchegiani, J. F. Jurado, A. M. Abu-Mahfouz, and X. Fafoutis, "A survey on machine learning software-defined wireless sensor networks (ML-SDWSNs): Current status and major challenges," *IEEE Access*, vol. 10, pp. 23560–23592, 2022.

[24] X. Cui, X. Huang, Y. Ma, and Q. Meng, "A load balancing routing mechanism based on SDWSN in smart city," *Electronics*, vol. 8, no. 3, p. 273, Mar. 2019.

[25] B. Mao, F. Tang, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "A novel non-supervised deep-learning-based network traffic control method for software defined wireless networks," *IEEE Wireless Commun.*, vol. 25, no. 4, pp. 74–81, Aug. 2018.

[26] S. Misra, S. Bera, M. P. Achuthananda, S. K. Pal, and M. S. Obaidat, "Situation-aware protocol switching in software-defined wireless sensor network systems," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2353–2360, Sep. 2018.

[27] R. Huang, X. Chu, J. Zhang, and Y. H. Hu, "Energy-efficient monitoring in software defined wireless sensor networks using reinforcement learning: A prototype," *Int. J. Distrib. Sensor Netw.*, vol. 2015, pp. 1–12, Oct. 2015.

[28] Q. Zhang, X. Wang, J. Lv, and M. Huang, "Intelligent content-aware traffic engineering for SDN: An AI-driven approach," *IEEE Netw.*, vol. 34, no. 3, pp. 186–193, May 2020.

[29] M. U. Younus, M. K. Khan, M. R. Anjum, S. Afridi, Z. A. Arain, and A. A. Jamali, "Optimizing the lifetime of software defined wireless sensor network via reinforcement learning," *IEEE Access*, vol. 9, pp. 259–272, 2021.

[30] M. U. Younus, M. K. Khan, and A. R. Bhatti, "Improving the software-defined wireless sensor networks routing performance using reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3495–3508, Mar. 2022.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[32] S. Din, A. Paul, A. Ahmad, and J. H. Kim, "Energy efficient topology management scheme based on clustering technique for software defined wireless sensor network," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 2, pp. 348–356, 2019.

[33] 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.

[34] S. A. Asakipaam, J. J. Kponyo, J. O. Agyemang, and F. Appiah-Twum, "Design of a minimal overhead control traffic topology discovery and data forwarding protocol for software-defined wireless sensor networks," *Int. J. Commun. Netw. Inf. Secur. (IJCNIS)*, vol. 12, no. 3, pp. 450–458, Apr. 2022.

[35] Q. Liu, L. Cheng, R. Alves, T. Ozcelebi, F. Kuipers, G. Xu, J. Lukkien, and S. Chen, "Cluster-based flow control in hybrid software-defined wireless sensor networks," *Comput. Netw.*, vol. 187, Mar. 2021, Art. no. 107788.

[36] S. S. G. Shiny, S. S. Priya, and K. Murugan, "Repeated game theory-based reducer selection strategy for energy management in SDWSN," *Comput. Netw.*, vol. 193, Jul. 2021, Art. no. 108094.

[37] N. Samarji and M. Salamah, "ESRA: Energy soaring-based routing algorithm for IoT applications in software-defined wireless sensor networks," *Egyptian Informat. J.*, vol. 23, no. 2, pp. 215–224, Jul. 2022.

[38] L. Sixu, W. Muqing, and Z. Min, "Particle swarm optimization and artificial bee colony algorithm for clustering and mobile based software-defined wireless sensor networks," *Wireless Netw.*, vol. 28, no. 4, pp. 1671–1688, Mar. 2022.

[39] Z. Ding, L. Shen, H. Chen, F. Yan, and N. Ansari, "Energy-efficient topology control mechanism for IoT-oriented software-defined WSNs," *IEEE Internet Things J.*, vol. 10, no. 15, pp. 13138–13154, Aug. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10079104

[40] R. Molose, B. Isong, N. Dladlu, and A. M. Abu-Mahfouz, "Analysis of hierarchical cluster-based energy-aware routing protocols in WSNs for SDWSN application," in *Proc. Int. Conf. Electr., Comput. Energy Technol. (ICECET)*, Dec. 2021, pp. 1–8.

[41] A. Rahimifar, Y. S. Kavian, H. Kaabi, and M. Soroosh, "Predicting the energy consumption in software defined wireless sensor networks: A probabilistic Markov model approach," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 10, pp. 9053–9066, Oct. 2021.

[42] W. Guo, C. Yan, and T. Lu, "Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 2, Feb. 2019, Art. no. 155014771983354.

[43] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[44] F. F. Jurado-Lasso, K. Clarke, and A. Nirmalathas, "Performance analysis of software-defined multihop wireless sensor networks," *IEEE Syst. J.*, vol. 14, no. 4, pp. 4653–4662, Dec. 2020.

[45] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[46] M. K. Awad, M. H. H. Ahmed, A. F. Almutairi, and I. Ahmad, "Machine learning-based multipath routing for software defined networks," *J. Netw. Syst. Manage.*, vol. 29, no. 2, pp. 1–30, Apr. 2021.

[47] R. C. A. Alves, D. A. G. Oliveira, G. A. N. Segura, and C. B. Margi, "The cost of software-defining things: A scalability study of software-defined sensor networks," *IEEE Access*, vol. 7, pp. 115093–115108, 2019.

[48] S. Srinivasa and M. Haenggi, "Path loss exponent estimation in large wireless networks," in *Proc. Inf. Theory Appl. Workshop*, Feb. 2009, pp. 124–129.

[49] S. Schmidt. (2018). *Linux-WPAN*. [Online]. Available: https://linux-wpan.org/

MOHAMMED ALSAEEDI received the B.S. degree in computer science from King Abdulaziz University (KAU), in 2008, and the M.S. degree in computer networks from the King Fahd University of Petroleum and Minerals (KFUPM), in 2014. He is currently pursuing the Ph.D. degree with Universiti Teknologi Malaysia (UTM). His research interests include the Internet of Things data analytics, machine learning for smart cities, and intent-based software-defined networking.

MOHD MURTADHA MOHAMAD received the B.E. degree in computer from Universiti Teknologi Malaysia (UTM), in 2000, and the M.S. degree in embedded system engineering and the Ph.D. degree in electrical engineering from Heriot-Watt University, U.K., in 2007. He started his career as a Tutor with UTM, where he is currently an Associate Professor with the Faculty of Engineering, School of Computing. His current research interests include underwater wireless sensor networks, underwater acoustic sensor networks, autonomous underwater vehicles, application development, assembly language, programming, and indoor positioning. In 2001, he received a scholarship from the Department of Public Service (JPA) for the M.S. degree.

ANAS AL-ROUBAIEY received the B.S. degree in computer engineering from Alexandria, Egypt, in 2001, and the M.S. and Ph.D. degrees from the King Fahd University of Petroleum and Minerals, Saudi Arabia, in 2009 and 2015, respectively. His master's degree focused on computer networks, while the Ph.D. degree is in computer science and engineering. He was a Teaching Assistant with Sanaa and Taiz University, Yemen, until 2004. He currently holds a research position with the King Fahd University of Petroleum and Minerals. He has served as a reviewer for several ISI journals. His diverse research interests include intrusion detection systems, middleware, systems integration, real-time and distributed systems, software defined networking, wireless sensor and actuator networks, energy-aware protocols, and power harvesting in WSN. He boasts a numerous publications and patents in these domains.