Journal of Information Systems Engineering and Management
2025, 10(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Dynamic and Adaptive Framework for Efficient Configuration and Management in Mobile Ad Hoc Networks

*[1]K. Purnima, [2]Dr.M.N. Giriprasad

[1]Research Scholar, Department of ECE, Jawaharlal Nehru Technological University Anantapur, Anantapur, AP, India.

Email: purnima.kuderu@gmail.com

[2]Professor, Department of ECE, JNTU Anantapur, Anantapur, AP, India. Email: mahendragiri1960@gmail.com

ARTICLE INFO

Received: 29 Sept 2024

Revised: 30 Nov 2024

Accepted: 12 Dec 2024

ABSTRACT

Mobile Ad Hoc Networks (MANETs) often face challenges such as frequent topology changes, limited resources, and security vulnerabilities, which affect their stability and performance. This study introduces DynaMANET, a framework designed to address these challenges by integrating routing, resource management, and address configuration. The framework employs a multi-layered approach: the Data Collection Layer gathers network metrics, the Learning and Decision-Making Layer uses Deep Q-Networks to adjust configurations, and the Context Awareness Layer applies Graph Neural Networks to detect network changes. The Feedback and Evaluation Layer utilizes Bayesian optimization to refine decisions, while the Execution and Control Layer ensures seamless implementation of configurations. Simulations were performed in a 100-node network environment modeled using the NS-3 simulator. Node mobility and varying traffic loads were considered to evaluate throughput, latency, packet delivery ratio, energy consumption, and intrusion detection rates. The framework demonstrated better throughput, reduced delays, and higher packet delivery ratios compared to E-OLSR and MARL models. Detection rates for security threats such as black hole and DDoS attacks were slightly higher. Scalability tests showed minimal throughput drop as the network size increased. In failure scenarios, the framework maintained stable communication and data delivery. These results suggest that DynaMANET addresses interconnected challenges in MANETs by combining adaptive decision-making with efficient resource management. The methods proposed provide a structured approach for improving MANET reliability under dynamic conditions.

Keywords: MANET, adaptive routing, resource optimization, Deep Q-Networks, Graph Neural Networks, Bayesian optimization, scalability, security.

INTRODUCTION

Mobile Ad Hoc Networks (MANETs) are self-organizing wireless networks that operate without fixed infrastructure. They are used in applications such as disaster recovery, military communication, and remote sensing. Their dynamic nature and lack of centralized control, however, introduce several challenges in maintaining performance, managing resources, and ensuring seamless operation under varying conditions.

MANETs face significant challenges due to their infrastructure-less and dynamic nature. Frequent topology changes caused by node mobility create the need for adaptive routing protocols to maintain connectivity. Protocols like AODV and STORM address this by offering loop-free routes and mechanisms for efficient recovery when links fail. However, such protocols often increase message overhead and delays during rapid topology changes, reducing overall network efficiency [1], [2].

Resource constraints are another pressing issue. Limited energy and bandwidth in MANETs demand efficient management to prevent node failures and network degradation. Without optimization strategies, excessive energy consumption or bandwidth congestion can significantly impact network performance. Mechanisms that enhance energy and bandwidth utilization remain critical for improving the reliability of MANETs [3].

Address configuration also presents difficulties in MANETs. In the absence of centralized control, assigning unique addresses to nodes becomes a complex task. Dynamic and hierarchical approaches aim to resolve conflicts and prevent collisions, but these methods often face scalability issues as network size increases. The need for a more scalable and efficient address configuration method is evident [4].

Current solutions often focus on specific challenges but lack integration. Centralized approaches, for instance, provide a clear and comprehensive view of the network, but their reliance on single points of control makes them vulnerable to failures. Fully distributed systems, while resilient to individual node failures, face difficulties in managing resources efficiently and scaling effectively in large networks [4]. There is a lack of unified frameworks that integrate routing, resource optimization, and address configuration to address the interconnected challenges of MANETs.

The use of multi-agent systems has enhanced self-organization and auto-configuration in MANETs, allowing nodes to collaborate and adapt to changing conditions. However, the combination of multi-agent systems with resource-constrained environments, routing, and address configuration has been minimally explored. A hybrid approach combining centralized and distributed methods offers potential but requires further research [5].

This study focuses on designing a dynamic framework for MANETs that integrates routing, resource management, and address configuration in a cohesive manner. The primary objective is to address the interconnected challenges of maintaining connectivity, optimizing resource usage, and reducing address conflicts. The framework aims to provide mechanisms that adapt to topology changes, minimize delays, and balance centralized and distributed methods to enhance scalability and performance.

The proposed framework incorporates adaptive routing protocols, efficient cluster management mechanisms, and multi-agent systems to address key issues in MANETs. It combines centralized and distributed approaches to reduce the risks associated with single points of failure while improving scalability and resource efficiency. The hybrid address configuration method aims to minimize address conflicts and ensure seamless operation even in larger networks [5], [6].

The framework contributes by improving performance metrics such as energy consumption, packet delivery ratio, and delay while offering practical solutions for dynamic and resource-constrained environments. It also evaluates trade-offs between centralized and distributed techniques, providing insights into their advantages and limitations for different network conditions.

The paper begins by reviewing related work, focusing on existing approaches and their limitations. The framework's design and components are then described in detail. This is followed by simulation results and a discussion of the findings. The paper concludes with a summary of contributions and potential directions for future research. This structure provides a logical flow for understanding the motivations, methods, and outcomes of the study.

RELATED WORK

Ivoghlian et al. [7] proposed a multi-agent deep learning framework to manage wireless networks under changing conditions. The system adjusted configurations based on parameters such as bandwidth, energy, and connectivity, which helped improve latency and data throughput. However, the framework’s applicability in large or diverse networks was not extensively explored. Adriaensen et al. [8] introduced a method for dynamic algorithm configuration, enabling self-tuning mechanisms to reduce manual intervention. The study improved performance metrics like latency and resource usage, but trade-offs between computational costs and long-term energy savings were not discussed. Lopatka et al. [9] developed a radio environment model to enhance spectrum efficiency and reduce interference. The model provided detailed insights through simulations, but its practical application faced limitations, such as hardware constraints.

Khare et al. [10] worked on a peer-to-peer resource allocation framework for mobile ad hoc networks (MANETs). The system used time and location-aware algorithms to improve resource availability and reduce message overhead. Challenges arose when accurate location data was unavailable, impacting its reliability. Zhang et al. [11] introduced RoNet, a neural-assisted framework designed to predict connectivity changes in mobile networks. While RoNet reduced disruptions in high-mobility environments, its computational demands limited its use in energy-constrained settings. Ramya et al. [12] proposed a dynamic partitioning algorithm for secure data routing. The approach reduced routing overhead and enhanced data security but lacked clarity on computational overhead introduced by security mechanisms.

Prashanth et al. [13] implemented a reinforcement learning-based channel assignment system to optimize spectrum usage in wireless networks. Results showed reduced delays and better channel reliability, though real-time training challenges were not addressed. Janani et al. [14] designed a distributed secured broadcast strategy to improve communication reliability in MANETs. Cryptographic measures enhanced data integrity, but detailed simulations were missing to validate its performance in dense networks. Yan et al. [15] employed Bayesian optimization to refine network configurations, improving response time and energy efficiency. However, its dependence on prior knowledge

limited adaptability in unfamiliar conditions. He et al. [16] presented PAD-Net, a neural network-based dynamic adaptation framework. It enhanced resource use and performance but required high computational power, making it unsuitable for resource-limited setups.

Zhang et al. [17] created a hybrid routing framework integrating proactive and reactive protocols to optimize resource usage in MANETs. While it improved energy efficiency and connectivity, its complexity introduced scalability challenges in large networks. Nikolaidis et al. [18] reviewed advancements in MANET management, highlighting the importance of adaptive protocols but offering no experimental validation or specific methodologies. Ahmed et al. [19] explored blockchain for trust management in MANETs, improving trust accuracy and reducing vulnerabilities. Energy and computational costs associated with blockchain remained key challenges. Anakath et al. [20] proposed a topology management strategy that reduced reconfiguration delays using predictive models. However, its reliance on accurate mobility predictions posed limitations in unpredictable environments.

Wang et al. [21] introduced AgileCtrl, a self-adaptive configuration framework that adjusted to workload changes and improved resource utilization. Computational complexity, however, posed challenges in constrained environments. Saudi et al. [22] presented a protocol for managing routing, resource allocation, and security in MANETs. It enhanced throughput and reduced delays but introduced trade-offs in communication overhead due to its modular design. Each article addressed specific issues, yet scalability, computational demands, and practical deployments were common concerns across the reviewed works.

The literature review discusses different approaches to solving challenges in mobile ad hoc networks (MANETs), focusing on dynamic management, resource optimization, security, and adaptive configurations. Each study offers valuable insights, but certain recurring challenges and limitations are evident. Solutions for managing dynamic topologies and resources vary in focus and method. Ivoghlian et al. [7] and Zhang et al. [17] introduced frameworks that combine learning-based mechanisms and hybrid routing protocols. These approaches demonstrated improvements in resource usage and connectivity during changing network conditions. However, scalability remains a concern, particularly in large and diverse environments. Anakath et al. [20] suggested predictive topology adjustments to maintain stability and reduce delays. While it addressed reconfiguration efficiency, the reliance on precise mobility predictions makes its use less reliable in unpredictable setups.

Security-focused studies highlighted methods to ensure data integrity and protect against attacks. Ahmed et al. [19] employed blockchain to manage trust relationships among network nodes, combining cryptographic techniques with distributed validation. While this approach strengthened trust accuracy, its energy-intensive requirements posed challenges for resource-constrained settings. Janani et al. [14] proposed a broadcast strategy that used cryptographic measures to improve reliability. Although this system reduced dependency on centralized nodes, detailed simulations were lacking, leaving its scalability in dense networks uncertain. Adaptive frameworks for real-time network adjustments featured prominently. Adriaensen et al. [8] presented self-tuning mechanisms to enhance resource efficiency and reduce latency. Similarly, Wang et al. [21] offered AgileCtrl, which dynamically adjusted configurations based on workload. While these systems improved operational efficiency, their computational demands raised concerns about applicability in low-power environments. Neural-assisted methods, such as RoNet by Zhang et al. [11] and PAD-Net by He et al. [16], introduced predictive features for better connectivity. Despite showing promise in managing high mobility, their reliance on resource-heavy neural networks limited their practicality in constrained networks. Studies addressing spectrum optimization introduced unique approaches but faced challenges in adaptability. Lopatka et al. [9] used simulation-based spectrum modeling to improve efficiency and reduce interference. However, the absence of real-world implementation limited its broader relevance. Yan et al. [15] applied Bayesian optimization to refine network configurations. While this method reduced response times and energy usage, its dependency on prior knowledge restricted its use in environments with limited data.

Across the literature, recurring gaps are evident. Many studies relied on controlled simulations and theoretical models without addressing real-world deployment challenges. The balance between computational efficiency and resource constraints remains a significant obstacle. While individual studies focused on specific issues, few explored how to integrate solutions into a single cohesive framework. Addressing scalability, adaptability, and practical deployment challenges in a unified way is essential for advancing solutions in MANETs. The review suggests that future research should prioritize methods that align theoretical models with real-world conditions, focusing on the combined challenges of dynamic environments and limited resources.

METHODS AND MATERIALS

This section explains the framework's structure and the techniques used to manage Mobile Ad Hoc Networks (MANETs). It organizes the framework into layers, each with a specific role, such as gathering network data, selecting configurations, and applying adjustments. The layers work together to address challenges like managing dynamic topologies, optimizing resource usage, and detecting anomalies. Each part of the framework is tailored to ensure efficient operation under changing conditions while maintaining scalability and adaptability. The following sections describe how these components function and interact.

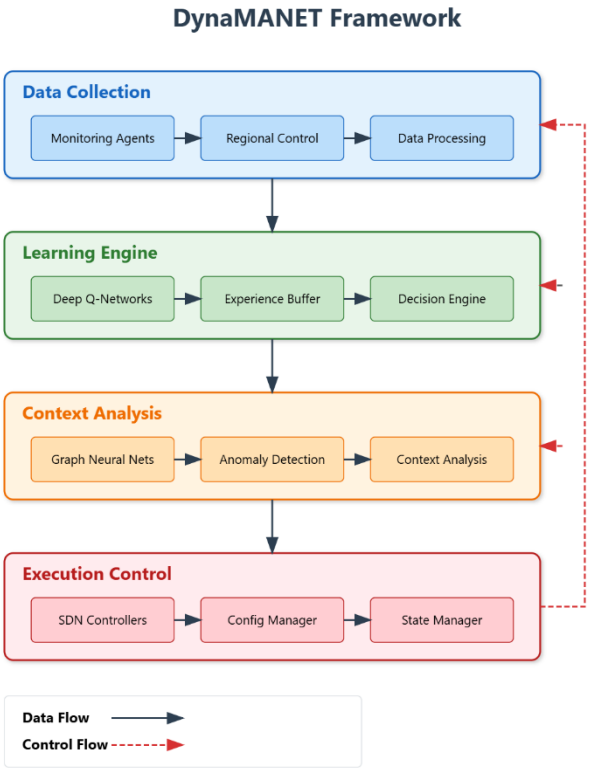

Figure 1: System Architecture of DynaMANET Framework

The diagram shows figure 1 the layered structure of the DynaMANET framework, designed for configuring and managing Mobile Ad Hoc Networks (MANETs). It includes five layers working together to address network challenges. The Data Collection Layer gathers real-time information, such as energy usage and connectivity, through Distributed Monitoring Agents (DMAs) and regional controllers. The Learning and Decision-Making Layer uses Deep Q-Networks (DQN) to analyze this data and adjust configurations like routing and energy settings based on network conditions. The Context Awareness Layer applies Graph Neural Networks (GNNs) to study node relationships, detect anomalies, and highlight unusual patterns such as packet drops. The Feedback and Evaluation Layer refines decisions using Bayesian Optimization by evaluating metrics such as throughput, latency, and packet delivery rates. The Execution and Control Layer enforces these decisions across the network using Software-Defined Networking (SDN) controllers while handling node or controller failures to maintain functionality. Arrows in the diagram show the flow of data, decisions, and feedback among the layers, emphasizing their interdependence and adaptability under dynamic conditions.

Data Collection Layer

The data collection layer gathers important network information from all nodes. It monitors changes in topology, tracks energy usage, and observes network performance. These metrics help in making adjustments to improve how the network operates under different conditions.

Purpose: This layer observes and collects data continuously. It tracks metrics such as node connectivity, signal strength, and the delivery of packets between devices. These metrics are essential for analyzing network behavior and detecting potential issues. The information serves as input for decision-making processes in the system.

Distributed Monitoring Agents: Distributed Monitoring Agents (DMAs) are lightweight programs installed on network nodes. They work by collecting data periodically and sending it to a regional controller. Each DMA collects multiple metrics at regular intervals, denoted by $D_i(t)$, where: Eq 1

$$D_i(t) = \{M_1(t), M_2(t), \dots, M_k(t)\} \quad \text{....(Eq 1)}$$

In this equation, $M_k(t)$ represents a specific metric such as bandwidth or energy level measured at time t. To avoid overwhelming the network with frequent data transmissions, DMAs follow a staggered schedule: Eq 2

$$T_i = T \cdot \left(1 + \frac{NodeID_i}{N}\right) \quad \text{....(Eq 2)}$$

Here, T is the base interval, $NodeID_i$ is the unique identifier for the node, and N is the total number of nodes in the network. This schedule ensures that data from different nodes does not overlap, reducing congestion.

Robustness: When a node cannot send data directly to the controller due to connectivity problems, neighboring nodes help forward the information. If i is the node needing assistance and j is a neighboring node, the relay condition can be expressed as: Eq 3

$$R_{ij}(t) = \begin{cases} 1, & if the connection between node i and the controller is unavailable \\ 0, & otherwise \end{cases} \quad \text{....(Eq 3)}$$

This approach ensures that data continues to flow even if certain connections are temporarily unavailable. Relay mechanisms provide stability to the network's monitoring system.

Scalability: The system distributes monitoring responsibilities across multiple regions, each managed by a regional controller. This design prevents any single controller from handling too much data. For a region r with nodes $\mathcal{N}_r$, the load on its controller, $L(C_r)$, is calculated as: Eq 4

$$L(C_r) = \sum_{i \in \mathcal{N}_r} |D_i(t)| \quad \text{....(Eq 4)}$$

The system ensures that this load does not exceed a defined limit, L_{max}. This balance allows the network to scale efficiently as more nodes are added, avoiding performance bottlenecks.

The data collection layer uses Distributed Monitoring Agents to gather real-time metrics. These agents transmit data using a structured schedule to minimize network congestion. Relay mechanisms ensure uninterrupted monitoring, even during connectivity issues. The distributed design of this layer enables it to scale without overwhelming individual components. These features make it suitable for managing dynamic networks.

Learning and Decision-Making Layer

The learning and decision-making layer identifies the best configurations for the network based on current conditions. It adapts to changes in node behavior, resource availability, and security concerns using reinforcement learning. By analyzing various metrics, it selects actions that align with predefined goals.

This layer uses reinforcement learning to adjust network settings in real time. It analyzes the current state of the network, evaluates potential actions, and selects configurations to improve performance and address security issues.

Deep Q-Networks (DQN): Deep Q-Networks (DQN) connect the state of the network to possible actions, estimating the impact of each decision. The value of taking an action a in a state s, represented as $Q(s, a)$, is calculated based on the immediate reward and future outcomes: Eq 5

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a') \quad \text{....(Eq 5)}$$

Here, r is the reward for the action, γ is a discount factor that prioritizes current rewards over distant ones, and $Q(s', a')$ is the value of the next state and action. A neural network approximates $Q(s, a)$, enabling the model to handle complex relationships between states and actions.

State Space: The state space captures the network's current conditions. It includes metrics such as throughput (T), latency (L), mobility and density of nodes (N), and security events (S). These variables define a state s as: Eq 6

$$s = \{T, L, N, S\} \quad \text{....(Eq 6)}$$

Each metric provides specific information, such as how fast data is moving, delays in communication, changes in node positions, and recent security alerts.

Action Space: The action space consists of all possible changes to the network's configuration. Each action a involves one or more adjustments, such as updating routing tables (R), switching communication protocols (P), or modifying energy-saving settings (E): Eq 7

$$a = \{R, P, E\} \text{ ....(Eq 7)}$$

The DQN evaluates all available actions and picks the one that provides the most benefit based on current conditions.

Reward Function: The reward function assigns a value to each action based on its results. It balances performance goals, like improving throughput or reducing latency, with security needs, such as preventing attacks. The reward r for an action a in state s is defined as: Eq 8

$$r = w_p \cdot P(s, a) + w_s \cdot S(s, a) \text{ ....(Eq 8)}$$

Here, P(s, a) measures performance outcomes, S(s, a) reflects security improvements, and w_p and w_s are weights that adjust the priority between performance and security.

Robustness: The DQN handles a wide range of conditions by learning from past experiences stored in a memory buffer. It uses techniques like replaying these experiences to refine decision-making and updating its estimates gradually to avoid sudden changes.

Scalability: The system divides the network into regions, each managed by a local DQN agent. These regional agents learn configurations for their specific areas. A global coordinator combines their outputs to maintain consistency across the entire network. The regional setup keeps the computational load manageable and ensures that decisions can be made quickly.

The learning and decision-making layer uses Deep Q-Networks to evaluate network conditions and select configurations. It processes metrics like throughput and node mobility, considers multiple actions, and balances rewards to align with network goals. A distributed design ensures scalability, allowing the system to handle larger networks while maintaining efficiency.

Context Awareness Layer

The context awareness layer observes the network to understand its condition and recognize any unusual patterns or threats. It studies relationships between devices and their connections to detect changes in the network and predict how it might behave under different circumstances.

The layer continuously analyzes the network to gather insights about its structure and behavior. It identifies disruptions in communication or abnormal activities, such as a node dropping all packets. By doing so, it helps adjust the network's configuration to prevent potential problems.

Graph Neural Networks (GNNs): Graph Neural Networks (GNNs) are used to process the network as a graph. Each node in the graph represents a device, and the edges between nodes represent communication links. A graph $G = (V, E)$ is defined, where V is the set of nodes, and E is the set of edges. Each node $v_i \in V$ has features x_i, such as energy level or traffic data, while each edge $e_{ij} \in E$ may include attributes like transmission delay or signal strength.

The GNN updates the representation of each node by combining its own features with information from its neighbors. The updated node representation $h_i^{(t)}$ at step t is calculated as: Eq 9

$$h_i^{(t)} = \sigma\left(W^{(t)} \cdot AGGREGATE\left(\{h_j^{(t-1)}, w_{ij} | j \in \mathcal{N}(i)\}\right) + b^{(t)}\right) \text{ ....(Eq 9)}$$

Here:

- $\mathcal{N}(i)$ refers to the neighbors of node i,
- $W^{(t)}$ and $b^{(t)}$ are learnable parameters for this step,
- $\sigma(\cdot)$ is an activation function, such as ReLU,
- AGGREGATE(·) combines information from neighboring nodes, often using simple operations like averaging or summing.

The final representation $h_i^{(T)}$ summarizes the role of the node in the graph after T iterations. This information helps identify patterns and detect anomalies such as black hole attacks.

Robustness: The GNN can handle changes in the network, such as adding or removing nodes or links. When a part of the network changes, only that portion needs to be updated, avoiding the need to retrain the entire model. This saves time and resources while keeping the network analysis consistent.

Scalability: To analyze large networks, the GNN works with smaller pieces of the graph, called subgraphs. A subgraph $G_r = (V_r, E_r)$, where $V_r \subset V$ and $E_r \subset E$, includes only a part of the network. The GNN processes these smaller graphs separately, which reduces computational effort. Later, the results from these subgraphs are combined to form a broader view of the entire network.

The context awareness layer uses GNNs to study the network's structure and behavior by treating it as a graph. It detects anomalies like packet-dropping nodes by analyzing relationships between devices and their communication links. This layer handles changes in the network efficiently and can process large networks by dividing them into smaller parts. Its purpose is to provide timely information about the network's condition, enabling better control and adjustments.

Feedback and Evaluation Layer

The feedback and evaluation layer observes how the network performs after applying new configurations. It uses this information to guide future decisions and improve how the system handles changes in conditions. This layer focuses on identifying what works well and what needs adjustment by analyzing results from recent actions.

The goal of this layer is to continuously check the performance of the network. Metrics such as data transfer speed (throughput), packet delivery rate, communication delays, and detection of security threats are tracked. Based on these observations, the system refines its strategies to better align with the network's needs.

Bayesian Optimization-Based Feedback: Bayesian optimization is used to understand the relationship between configuration settings and their outcomes. It models the performance of the network as a function f(x), where x represents the configuration parameters. Since the exact relationship between x and f(x) is often unknown, Bayesian optimization builds an approximation $\hat{f}(x)$ based on observed data: Eq 10

$$\hat{f}(x) = SurrogateModel(x|\mathcal{D}) \quad \text{....(Eq 10)}$$

The data $\mathcal{D}$ includes pairs of past configurations and their results, $\{x_i, f(x_i)\}$. Using this surrogate model, Bayesian optimization selects the next configuration x^* by maximizing an acquisition function a(x), which predicts which configurations are worth exploring: Eq 11

$$x^* = \underset{x}{argmax}\, a(x) \quad \text{....(Eq 11)}$$

This method balances exploring new possibilities and refining known ones to gradually improve decisions. For example, if a configuration reduces delays but lowers packet delivery, Bayesian optimization adjusts to find a balance that satisfies both objectives.

Robustness: Bayesian optimization accounts for uncertainties in how network configurations perform. It uses probability to predict the impact of changes, providing confidence that the chosen actions will work across different scenarios. By learning from variations in network conditions, the method helps reduce errors and ensures stable performance over time.

Scalability: The layer is designed to handle large networks by focusing on smaller groups, or clusters, within the system. Each cluster evaluates its configurations independently, allowing feedback loops to work faster and use fewer resources. For a cluster $\mathcal{C}_r$, the feedback process updates the model $\hat{f}_r(x)$ using the cluster's performance data $\mathcal{M}_r$: Eq 12

$$\hat{f}_r(x) \leftarrow Update(\hat{f}_r(x), \mathcal{M}_r) \quad \text{....(Eq 12)}$$

This localized approach minimizes delays, as clusters do not need to wait for the entire network to report its performance. Insights from these smaller groups are then combined to ensure consistency across the system.

The feedback and evaluation layer uses Bayesian optimization to assess how configurations impact network performance. It continuously tracks key metrics, evaluates outcomes, and adjusts future actions to address changing conditions. By focusing on localized evaluations, it efficiently handles large networks while ensuring accurate updates to the learning process. This layer plays a central role in maintaining the network's adaptability and stability.

Execution and Control Layer

The execution and control layer applies the configurations decided by the system. It ensures the network operates according to the recommended changes and enforces security rules to maintain proper functioning. This layer serves as the final point where decisions are turned into actions.

The primary task of this layer is to distribute configuration changes across the network. It manages tasks such as updating node settings, adjusting communication paths, and applying security protocols. This layer also ensures that security measures are activated where needed, preventing unauthorized activity and maintaining the network's stability.

Software-Defined Networking (SDN) Controllers: Software-Defined Networking (SDN) controllers are used to manage and apply the configurations. Each SDN controller oversees a specific part of the network, ensuring that the nodes in its region follow the given instructions. The controller sends commands to nodes and monitors their responses to ensure the configurations are implemented correctly.

Consider a region r managed by an SDN controller C_r, which sends a set of actions $\mathcal{A}_r$ to the nodes $\mathcal{N}_r$ in that area: Eq 13

$$\mathcal{A}_r = \{a_i | i \in \mathcal{N}_r\} \text{ ....(Eq 13)}$$

Here, a_i represents the action applied to a specific node i, such as updating its routing protocol or adjusting its power level. After applying the action, the node sends feedback f_i back to the controller: Eq 14

$$\mathcal{F}_r = \{f_i | i \in \mathcal{N}_r\} \text{ ....(Eq 14)}$$

This feedback helps confirm that the configuration has been implemented successfully. If any issues arise, the controller can resend instructions or adjust the commands as needed.

Robustness: The layer is designed to handle disruptions in the network. If a controller cannot communicate with certain nodes due to a failure, nearby controllers take over to ensure those nodes still receive instructions. For example, if C_r fails to deliver $\mathcal{A}_r$, the neighboring controller C_s will temporarily manage the affected nodes. This approach ensures that configuration changes continue to reach all parts of the network, even during unexpected failures.

Scalability: The network is divided into regions, each managed by a dedicated SDN controller. This setup reduces the workload on individual controllers and avoids bottlenecks. Instead of a single controller handling the entire network, smaller regions are managed independently. Each region's controller operates locally, sending instructions only to nodes within its area. The total workload for each controller is proportional to the number of nodes in its region: Eq 15

$$L(C_r) = \sum_{i \in \mathcal{N}_r} |a_i| \text{ ....(Eq 15)}$$

By keeping $L(C_r)$ below a defined limit, the system ensures that no controller is overloaded. This division of responsibilities allows the network to handle more nodes without delays or resource constraints.

The execution and control layer uses SDN controllers to apply the recommended configurations and enforce security measures. It communicates directly with nodes, ensuring that changes are implemented and monitored. By dividing the network into regions and allowing controllers to manage smaller areas, this layer remains efficient and responsive even in large or dynamic networks. Through redundancy and localized management, it ensures continuous operation despite potential failures.

Workflow of the Framework

The framework operates through a series of steps to monitor, analyze, adapt, and optimize the network.

Data Collection: Distributed Monitoring Agents (DMAs) gather metrics like bandwidth, energy levels, and topology. The collected data, $D_i(t)$, is sent to regional controllers, providing a snapshot of the network state.

State Analysis: Graph Neural Networks (GNNs) represent the network as a graph $G = (V, E)$, where nodes V and edges E have features. GNNs process these features iteratively to detect anomalies or changes, outputting updated node representations $h_i^{(T)}$.

Decision-Making: Deep Q-Networks (DQN) use the analyzed state to determine optimal actions a^*, maximizing the Q-value: Eq 16

$$a^* = \underset{a}{argmax} Q(s, a) \text{ ....(Eq 16)}$$

Actions include protocol updates, power adjustments, or security enhancements.

Configuration Execution: Software-Defined Networking (SDN) controllers apply the selected actions $\mathcal{A}_r$ to nodes in their regions. Nodes execute the configurations and send feedback $\mathcal{F}_r$ to confirm deployment.

Feedback Loop: Bayesian optimization evaluates the performance of configurations and updates the reinforcement learning model. The surrogate model $\hat{f}(x)$ predicts outcomes and selects the next configuration x^* to improve efficiency: Eq 17

$$x^* = \underset{x}{argmax} a(x) \quad \text{....(Eq 17)}$$

This workflow ensures continuous monitoring, informed decision-making, timely execution, and iterative improvement, keeping the network adaptable and efficient.

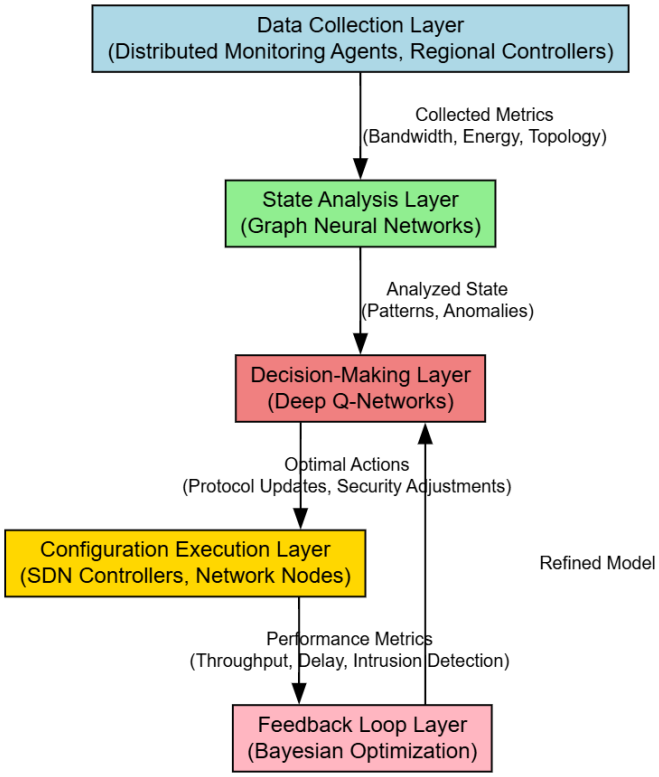

Figure 2: Workflow for Dynamic Network Management Framework

The workflow diagram shows figure 2 the structure of a system designed to manage networks dynamically through five distinct layers. The Data Collection Layer gathers real-time information like bandwidth usage, energy levels, and topology changes using Distributed Monitoring Agents, which relay this data to regional controllers. The State Analysis Layer processes this information with Graph Neural Networks, examining connections between devices to detect unusual patterns and changing conditions. Based on these findings, the Decision-Making Layer uses Deep Q-Networks to select actions, such as updating communication protocols or adjusting power levels, that are most suited to the current network state. These selected actions are implemented in the Configuration Execution Layer, where Software-Defined Networking controllers distribute commands to nodes while monitoring their responses for confirmation. Finally, the Feedback Loop Layer evaluates how these configurations perform using Bayesian optimization, refining the system's ability to make better decisions in future cycles. Each layer communicates with the next, ensuring a continuous flow of data, decisions, and updates for seamless network management.

EXPERIMENTAL STUDY

The experimental study examines how the framework handles various challenges in Mobile Ad Hoc Networks (MANETs). It focuses on understanding how well the framework adapts to changes in network conditions, ensures uninterrupted data flow, and addresses security threats in a decentralized and dynamic environment. The experiments are designed to evaluate performance by analyzing metrics like throughput, latency, and packet delivery ratio. These metrics show how efficiently the framework processes data and maintains communication. Scalability is tested by increasing the number of nodes and monitoring the framework's ability to handle more complex topologies. The study also examines robustness by introducing scenarios such as node failures, communication disruptions, and simulated attacks to measure the framework's capacity to maintain functionality. The testing environment simulates real-world MANET conditions, including dynamic node mobility and varying communication loads. Network

simulators provide the platform for implementation, while reinforcement learning libraries and graph analysis tools help manage decision-making and anomaly detection. Data collection focuses on key metrics, including energy consumption and intrusion detection rates, to provide insights into how the framework performs under different conditions. This approach ensures a detailed understanding of its behavior and reliability across multiple scenarios.

Experimental Setup

Network Environment: The experiments utilized the NS-3 simulator to model a Mobile Ad Hoc Network (MANET) with 100 nodes spread over a 1000m x 1000m area. Node mobility followed the Random Waypoint Mobility Model, with nodes moving at speeds between 1 m/s and 10 m/s, pausing for 2 seconds between movements. Three different network topologies were used: random, grid, and clustered. Communication relied on the AODV routing protocol, and traffic was generated using constant bit rate (CBR) flows, transmitting 512-byte packets at a rate of 4 packets per second. Background traffic introduced additional flows to simulate congestion and evaluate the framework's performance under varying network loads.

Evaluation Metrics: The performance and behavior of the framework were assessed using the following metrics:

1. **Throughput:** Throughput measures the total data successfully delivered to the destination per second. It is expressed in kilobits per second (kbps) and highlights how efficiently the framework handles data transmission.
2. **Latency:** Latency calculates the average time it takes for packets to travel from the source to the destination. It is measured in milliseconds and provides insights into delays introduced during communication.
3. **Packet Delivery Ratio (PDR):** PDR represents the percentage of data packets successfully delivered compared to those sent. It indicates the reliability of data transmission across the network.
4. **Energy Consumption:** Energy consumption measures the average energy used by the network nodes during the simulation. This metric is essential for evaluating how the framework manages power in resource-constrained environments.
5. **Intrusion Detection Rate (IDR):** IDR assesses the framework's ability to detect security threats, representing the percentage of identified threats out of the total threats introduced into the network. It evaluates the security mechanisms built into the framework.

Experimental Results

The experiments measured throughput, latency, and packet delivery ratio (PDR) under different network conditions, including variations in mobility and traffic load. Results showed that DynaMANET consistently handled data transmission better than enhanced optimized link state routing (E-OLSR) [19] and Multi-Agent Reinforcement Learning (MARL) [7], with minimal differences between DynaMANET and E-OLSR.

Table 1 throughput values remained higher for DynaMANET in all mobility scenarios. Low mobility conditions resulted in higher throughput for all models, while high mobility led to slight reductions. The consistency of throughput in DynaMANET indicates its capacity to adapt to changes in the network.

Table 1: Throughput (kbps)

Model	Low Mobility	Medium Mobility	High Mobility
DynaMANET	980	960	920
E-OLSR	970	950	910
MARL	940	920	880

The radar chart figure 3 visualizes throughput values, highlighting how DynaMANET maintains slightly better performance than E-OLSR across mobility scenarios. MARL shows lower throughput, especially at higher mobility levels.

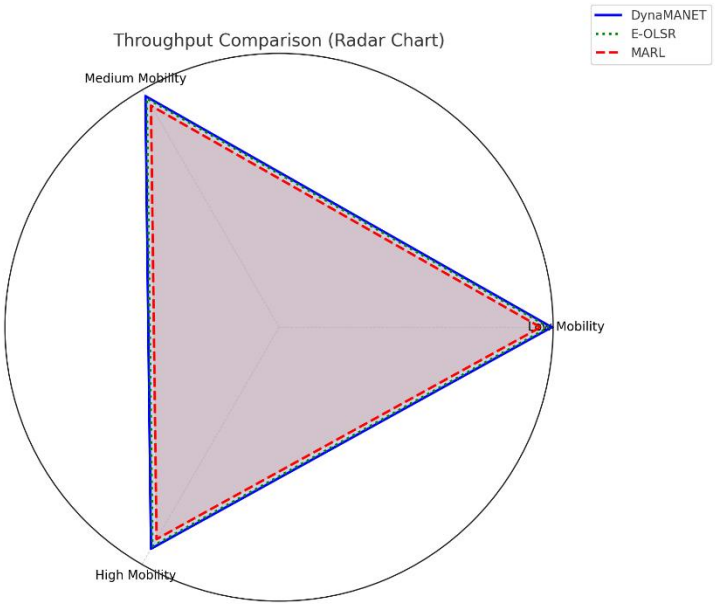

Figure 3: Throughput comparisons

The figure 3 Throughput comparisons emphasize the minor differences between DynaMANET and E-OLSR, with MARL trailing significantly.

Latency analysis showed table 2 that DynaMANET incurred slightly lower delays than E-OLSR across all traffic levels. MARL exhibited noticeably higher delays, particularly under high traffic.

Table 2: Latency (ms)

Model	Low Traffic Load	Medium Traffic Load	High Traffic Load
DynaMANET	45	60	80
E-OLSR	50	65	85
MARL	60	75	95

The figure 4 provides an overview of latency distributions, showing tighter ranges for DynaMANET compared to E-OLSR. MARL has the widest delay distribution, indicating inconsistencies in communication.

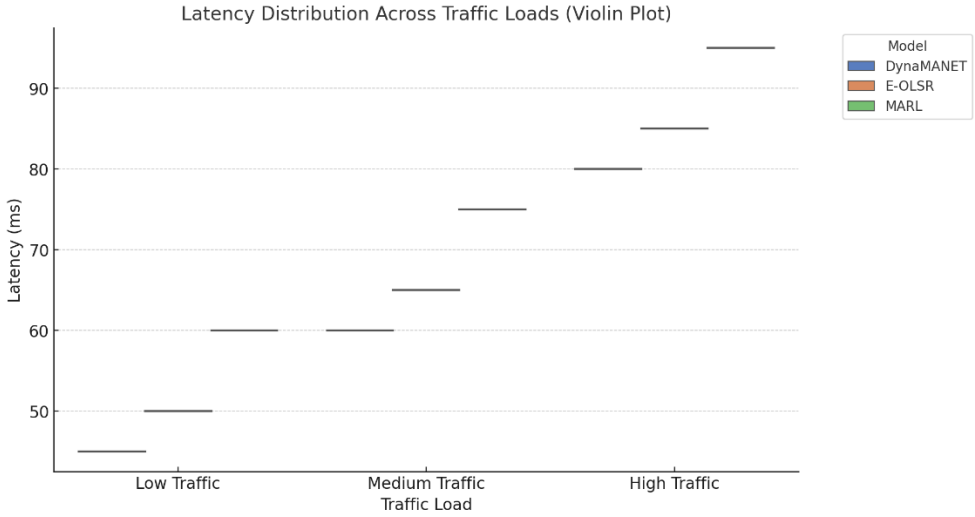

Figure 4: Latency Distributions Across Traffic Loads

Figure 4 latency distributions reflect how DynaMANET maintains steadier delays under different traffic conditions.

Table 3 Packet delivery ratio (PDR) values were slightly higher for DynaMANET compared to E-OLSR, with both models delivering significantly more packets than MARL. Increased traffic lowered the PDR for all models, but the trend remained consistent.

Table 3: PDR (%)

Model	Low Traffic	Medium Traffic	High Traffic
DynaMANET	96.5	94.2	91.8
E-OLSR	95.8	93.5	90.5
MARL	92.0	89.5	85.3

The figure 5 shows PDR trends across traffic loads, with DynaMANET maintaining consistent packet delivery reliability.

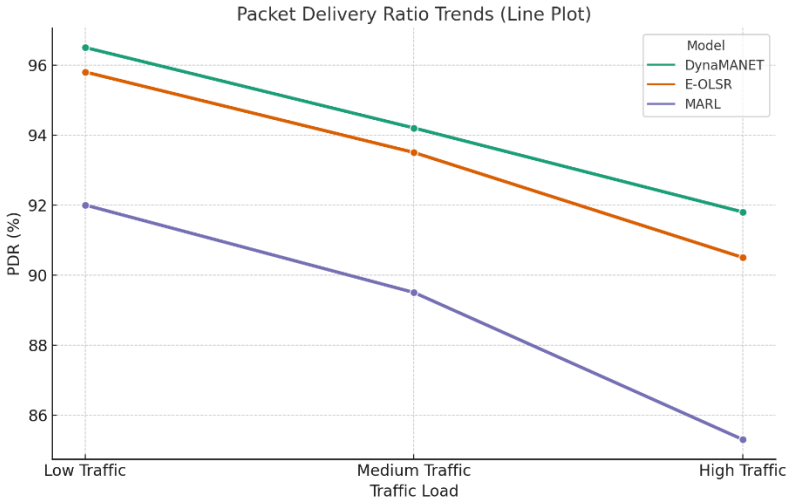

Figure 5: Packet Delivery Ratio

The figure 5 PDR values show DynaMANET delivering packets slightly more reliably than E-OLSR, with MARL falling behind at higher traffic levels.

Security Evaluation: The models were tested for their ability to detect and mitigate anomalies such as black hole and DDoS attacks shown in table 4. Detection rates were higher for DynaMANET and E-OLSR, with a small margin favoring DynaMANET. MARL performed noticeably worse in both scenarios.

Table 4: Intrusion Detection Rate (%)

Model	Black Hole Attack	DDoS Attack
DynaMANET	97.2	96.5
E-OLSR	96.8	96.0
MARL	92.0	91.0

The heatmap figure 6 visually shows detection rates, with lighter colors indicating better detection performance. DynaMANET's slightly higher detection rates are noticeable.

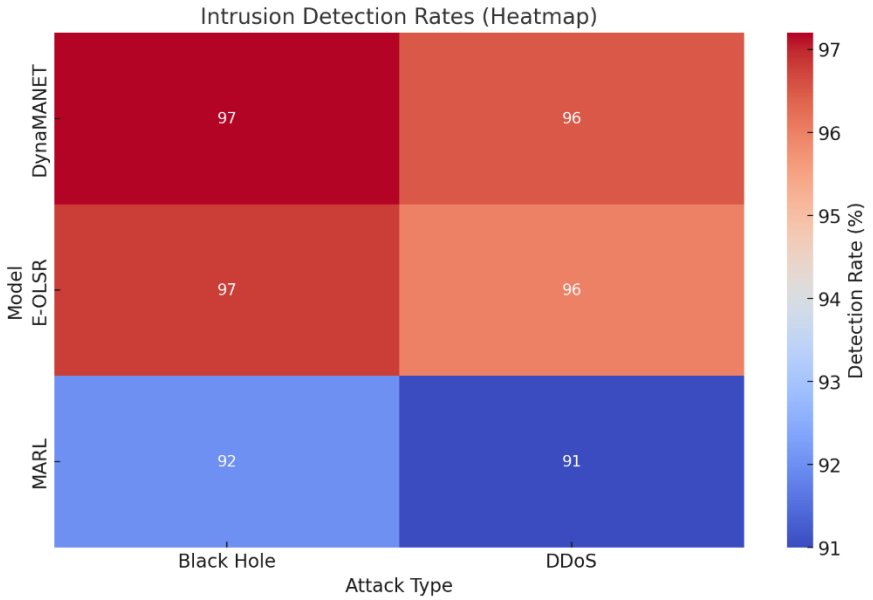

Figure 6: Intrusion Detection Rates

Figure 6 Intrusion detection rates are compared, showing DynaMANET performing marginally better than E-OLSR in detecting anomalies.

False positives were also evaluated. DynaMANET produced fewer incorrect detections compared to E-OLSR, while MARL showed the highest rate of false positives. False positive rates for all models illustrate how DynaMANET minimizes incorrect threat detections compared to the other methods.

Scalability Testing: The ability to scale with increasing network size was tested by varying the number of nodes from 50 to 200. The table 5 throughput drop remained minimal for DynaMANET, with E-OLSR close behind. MARL showed larger reductions as the network grew.

Table 5: Throughput Drop (%)

Model	50 Nodes	100 Nodes	200 Nodes
DynaMANET	2.5	3.8	5.5
E-OLSR	3.0	4.5	6.5
MARL	5.0	6.5	9.0

The figure 7 visualizes throughput drop densities, showing DynaMANET adapting well as the network scales.

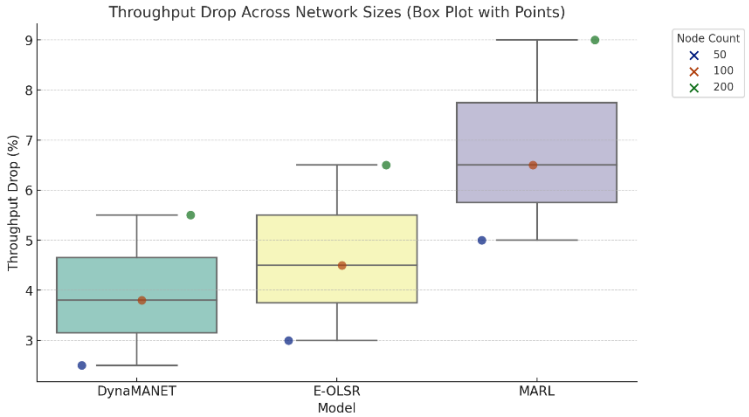

Figure 7: Throughput drop distributions Across Network Sizes

Figure 7 Throughput drop distributions reveal how DynaMANET handles scaling better than the other models.

Robustness to Failures: The table 6 models were tested for their performance under simulated node and controller failures. DynaMANET maintained packet delivery better than E-OLSR, with both models significantly outperforming MARL.

Table 6: PDR Under Node Failures (%)

Model	10% Node Failure	20% Node Failure
DynaMANET	90.2	85.5
E-OLSR	89.5	84.0
MARL	84.0	78.5

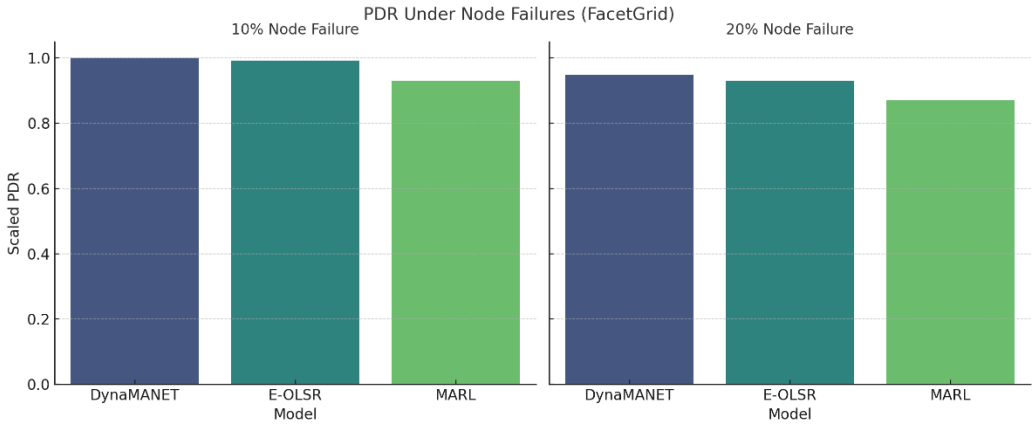

Figure 8: PDR Under Node Failures

PDR during failures shows DynaMANET performing slightly better than E-OLSR, with MARL showing figure 8 significant drops.

Packet delay variability figure 9 was evaluated under failure scenarios. DynaMANET showed smoother communication with fewer spikes in delay compared to E-OLSR and MARL.

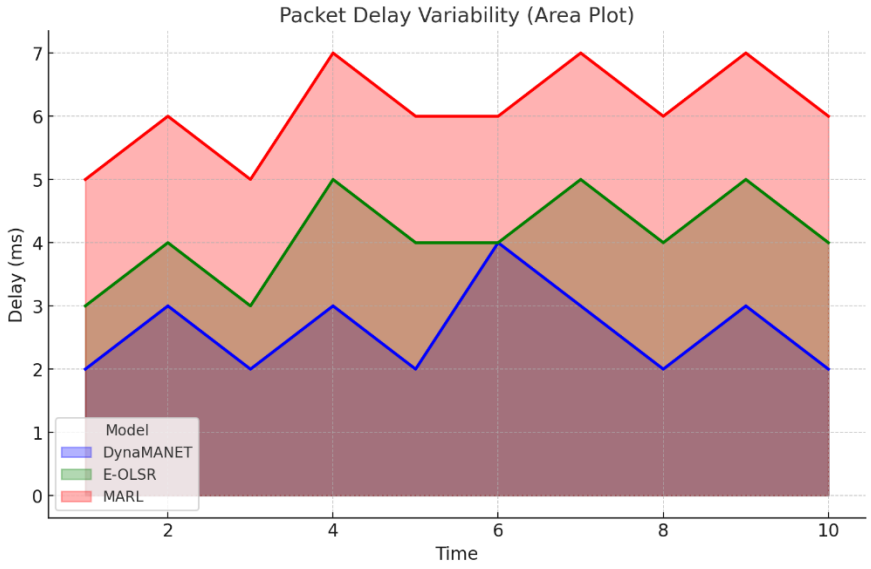

Figure 9: Packet Dealy Variability

Figure 9 performs slightly better than E-OLSR in most metrics while significantly outperforming MARL. The differences between DynaMANET and E-OLSR were small but consistent, indicating incremental improvements across performance, security, scalability, and robustness.

CONCLUSION

The study focused on addressing challenges in Mobile Ad Hoc Networks (MANETs) by introducing DynaMANET, a framework designed to integrate adaptive routing, resource management, and address configuration. The experiments demonstrated how DynaMANET maintained consistent throughput, minimized delays, and ensured higher packet delivery ratios compared to E-OLSR and MARL models. Detection rates for black hole and DDoS attacks were also slightly better, showing the framework's ability to adapt to dynamic conditions while managing resources efficiently. The findings suggest that combining centralized and distributed methods can address

interconnected challenges in MANETs. The layered approach allowed the framework to balance computational loads and maintain stable operations even during network failures. These results indicate that the proposed methods are suitable for improving reliability and scalability in resource-constrained, dynamic networks. The reliance on simulated environments is a limitation, as real-world conditions often introduce additional complexities. High computational requirements for certain techniques, like neural networks, may restrict applicability in low-power environments. Future research could explore lightweight algorithms, real-world testing, and the integration of advanced security mechanisms to enhance applicability and performance. DynaMANET provides a practical method for managing MANETs under changing conditions. The framework offers insights into combining adaptive methods to improve performance and security, making it a relevant solution for dynamic and resource-limited environments.

REFERENCES

[1] Giordano, Silvia. "Mobile ad hoc networks." Handbook of wireless networks and mobile computing (2002): 325-346.

[2] Balaji, K. "A frame work for integrated routing, scheduling and Traffic Management in MANET." International Research Journal of Engineering and Technology 2, no. 9 (2015): 2337-2344.

[3] Dr. Gomathi. U. "The Distributed Multi Constrained Channel Access Mechanism By Using Framework For Mobile Adhoc Network". International Journal for Multidisciplinary Research (IJFMR), Volume 6, Issue 3, May-June 2024.

[4] Wang, Xiaonan, and Huanyan Qian. "Dynamic and hierarchical IPv6 address configuration for a mobile ad hoc network." International Journal of Communication Systems 28, no. 1 (2015): 127-146.

[5] Korichi, Afaf, and Youcef Zafoune. "MAAC Protocol: Mobile Agents based Address Auto-Configuration Protocol for MANET." In 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT), pp. 194-199. IEEE, 2018.

[6] Dalasaniya, Krishna, and Nitul Dutta. "An efficient cluster management mechanism for manets." In 2014 IEEE International Conference on Computational Intelligence and Computing Research, pp. 1-6. IEEE, 2014.

[7] Ivoghlian, Ameer, Zoran Salcic, and Kevin I-Kai Wang. "Adaptive wireless network management with multi-agent reinforcement learning." Sensors 22, no. 3 (2022): 1019.

[8] Adriaensen, Steven, André Biedenkapp, Gresa Shala, Noor Awad, Theresa Eimer, Marius Lindauer, and Frank Hutter. "Automated dynamic algorithm configuration." Journal of Artificial Intelligence Research 75 (2022): 1633-1699.

[9] Lopatka, Jerzy, Anna Kaszuba-Checinska, and Radoslaw Checinski. "Radio Environment Model for High-Fidelity Simulator of Mobile Ad Hoc Networks with Dynamic Spectrum Management." In 2023 16th International Conference on Signal Processing and Communication System (ICSPCS), pp. 1-7. IEEE, 2023.

[10] Khare, Akhil, G. Madhu, and Pallavi Khare. "Location and Time Aware Resource Seeking Framework for Mobile P2P and Ad Hoc Networks." In 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 776-780. IEEE, 2022.

[11] Zhang, Yuru, Yongjie Xue, Qiang Liu, Nakjung Choi, and Tao Han. "RoNet: Toward Robust Neural Assisted Mobile Network Configuration." In ICC 2023-IEEE International Conference on Communications, pp. 3878-3883. IEEE, 2023.

[12] Ramya, V., N. Kousika, P. Rajasekaran, and J. JaganPradeep. "Secure and Efficient Modified Dynamic Partition Routing Algorithm for Mobile Ad Hoc Networks." In 2022 International Conference on Advanced Computing Technologies and Applications (ICACTA), pp. 1-5. IEEE, 2022.

[13] Prashanth, Suhaas Krishna, and S. Senthil. "Stochastic-reinforcement learning assisted dynamic power management model for zone-routing protocol in mobile ad hoc networks." Wireless Personal Communications 120, no. 1 (2021): 203-230.

[14] Janani, V. S., and M. Devaraju. "An efficient distributed secured broadcast stateless group key management scheme for mobile Ad Hoc networks." In 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), pp. 1-5. IEEE, 2022.

[15] Yan, Jia, Qin Lu, and Georgios B. Giannakis. "Bayesian optimization for online management in dynamic mobile edge computing." IEEE Transactions on Wireless Communications (2023).

[16] He, Shwai, Liang Ding, Daize Dong, Boan Liu, Fuqiang Yu, and Dacheng Tao. "PAD-net: An efficient framework for dynamic networks." arXiv preprint arXiv:2211.05528 (2022).

[17] Zhang, Yupeng, Shunkang Hu, and Zenghua Zhao. "A Unified Routing Framework for Resource-Constrained Mobile Ad Hoc Networks." In 2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 1358-1363. IEEE, 2024.

[18] Nikolaidis, Ioanis, and Kui Wu, eds. Ad-Hoc, Mobile and Wireless Networks: 9th International Conference, ADHOC-NOW 2010, Edmonton, AB, Canada, August 20-22, 2010, Proceedings. Vol. 6288. Springer, 2010.

[19] Ahmed, Huda A., and Hamid Ali Abed Al-Asadi. "A Blockchain-Enabled Trust Management Framework for Energy-Efficient and Secure Routing in Mobile Ad-Hoc Networks." TEM Journal 13, no. 2 (2024).

[20] Anakath, A. S., R. Kannadasan, G. Simi Margarat, and A. Pasumpon Pandian. "Dynamic Topology Management in Ad-Hoc Networks for Improved Performance." In 2024 Second International Conference on Advances in Information Technology (ICAIT), vol. 1, pp. 1-5. IEEE, 2024.

[21] Wang, Shu, Henry Hoffmann, and Shan Lu. "AgileCtrl: a self-adaptive framework for configuration tuning." In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 459-471. 2022.

[22] Saudi, Nur Amirah Mohd, Mohamad Asrol Arshad, Alya Geogiana Buja, Ahmad Firdaus Ahmad Fadzil, and Raihana Md Saidi. "Mobile ad-hoc network (MANET) routing protocols: A performance assessment." In Proceedings of the Third International Conference on Computing, Mathematics and Statistics (iCMS2017) Transcending Boundaries, Embracing Multidisciplinary Diversities, pp. 53-59. Springer Singapore, 2019.