MULTI-AGENT REINFORCEMENT LEARNING FOR NETWORK PROTOCOL SYNTHESIS

By

Hrishikesh Dutta

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical and Computer Engineering – Doctor of Philosophy

2024

ABSTRACT

The proliferation of Internet-of-Things (IoTs) and Wireless Sensor Networks (WSNs) has led to the widespread deployment of devices and sensors across various domains like wearables, smart cities, agriculture, and health monitoring. These networks usually comprise of resource-constrained nodes with ultra-thin energy budget. As a result, it is important to design network protocols that can judiciously utilize the available networking resources while minimizing energy consumption and maintaining network performance. The standardized protocols often underperform under general conditions because of their inability to adapt to changing networking conditions, including topological and traffic heterogeneities and various other dynamics. In this thesis, we develop a novel paradigm of learning-enabled network protocol synthesis to address these shortcomings.

The key concept here is that each node, equipped with a Reinforcement Learning (RL) engine, learns to find situation-specific protocol logic for network performance improvement. The nodes' behavior in different heterogeneous and dynamic network conditions are formulated as a Markov Decision Process (MDP), which is then solved using RL and its variants. The paradigm is implemented in a decentralized setting, where each node learns its policies independently without centralized arbitration. To handle the challenges of limited information visibility in partially connected mesh networks in such decentralized settings, different design techniques including confidence-informed parameter computation and localized information driven updates, have been employed. We specifically focus on developing frameworks for synthesizing access control protocols that deal with network performance improvement from multiple perspectives, viz., network throughput, access delay, energy efficiency and wireless bandwidth usage.

A multitude of learning innovations has been adopted to explore the protocol synthesis concept in a diverse set of MAC arrangements. First, the framework is developed for random access MAC setting, where the learning-driven logic is shown to be able to minimize collisions with a fair share of wireless bandwidth in the network. A hysteresis-learning enabled design is exploited for handling the trade-off between convergence time and performance in a distributed setting. Next, the ability of the learning-driven protocols is explored in TDMA-based MAC arrangement for enabling decentralized slot scheduling and transmit-sleep-listen decision making. We demonstrate how the proposed approach, using a multi-tier learning module and context-specific decision making, enables the nodes to make judicious transmission/sleep decisions on-the-fly to reduce energy expenditure while maintaining network performance. The multi-tier learning framework, comprising of cooperative Multi-Armed Bandits (MAB) and RL agents, solve a multidimensional network performance optimization problem. This system is then improved from scalability and adaptability perspective by employing a Contextual Deep Reinforcement Learning (CDRL) framework. The energy management framework is then extended for energy-harvesting networks with spatiotemporal energy profiles. A learning confidence parameter-guided update rule is developed to make the framework robust to unreliability of RL observables. Finally, the thesis investigates protocol robustness against malicious agents, thus demonstrating versatility and adaptability of learning-driven protocol synthesis in hostile networking environments.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Subir Biswas for his constant support and guidance during my PhD. He has helped and directed me on how to approach scientific research. I would also like to thank my committee members Dr. Vaibhav Srivastava, Dr. Sandeep Kulkarni and Dr. Jian Ren for their time and meticulous review of my thesis.

I would also take this opportunity to thank my colleague, co-author and friend Amit Kumar Bhuyan who has helped me with his constructive and valuable feedback on my research, besides being a companion of unforgettable fun memories. My gratitude also extends to my lab mates, Avirup Roy, Kang Gao, Daniel Mcdermott and Yida Yang, for the meaningful discussions and entertaining activities.

I am thankful to Michigan State University and Electrical and Computer Engineering Department for providing me with the opportunity to pursue my PhD. My sincere acknowledgement goes to Dr. Katy Colbry, Dr. Tim Hogan, and Dr. Nelson Sepulveda, and all the members of the MSU ECE department office, especially Lisa, Michelle and Laurene, for helping me to graduate in time.

I acknowledge the support that I received from the National Science Foundation (NSF) grants with award numbers 2040257 and 018063-00001.

My sincere gratitude goes to all who have taught me how to contribute to science and research. I am particularly indebted to Dr. Rohan Kumar Das, who first introduced me to this domain. I am sincerely thankful to Dr. Lalita Udpa and Dr. Krishnan Balasubramaniam, who have been my mentors at different stages of my career.

I am thankful to all my friends for their constant love and encouragement. I owe my gratefulness to my childhood friend Yugal Barman who has always supported me throughout the years. I acknowledge the company of all my friends, including Kangkan Baishya, Gunjan Das, Shahinoor Rahman, Debottam Dutta, Dipankar Baruah, Sarvesh Kumar, Ankush Jha, Roshani Shrestha, Santosh Balija, Debakshi Dey, Dipjyoti Bisharad, Soumyadeep Dev and many others, with whom I share memorable moments.

Finally, I would like to thank my family for their continuous support throughout my life. Words will fall short for expressing gratitude and appreciation for my Maa, Purabi Dutta, and my Baba, Satish Chandra Dutta, whose constant belief in me has made this thesis possible. I am also thankful to my little sister, Namrata Dutta, for her unconditional love and support. My appreciation also goes to my extended family, including my uncles, aunts, cousins, nephews, and nieces, for all their love and inspiration.

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Motivation

With the recent developments in Internet-of-Things (IoTs) and Wireless Sensor Networks (WSNs), a wide range of devices and sensors have been massively deployed in a variety of applications [1]. Such applications include wearables, smart city, smart home, agriculture and crop monitoring, health monitoring, etc. (Fig. 1.1) These devices and sensors are usually designed with low-cost, low-complex and energy constrained hardware [2] [3]. Popular examples of such low end IoT devices include OpenMote, IoT-LAB M3, TelosB motes, Econotag, Zolertia Z1, to name a few [3].

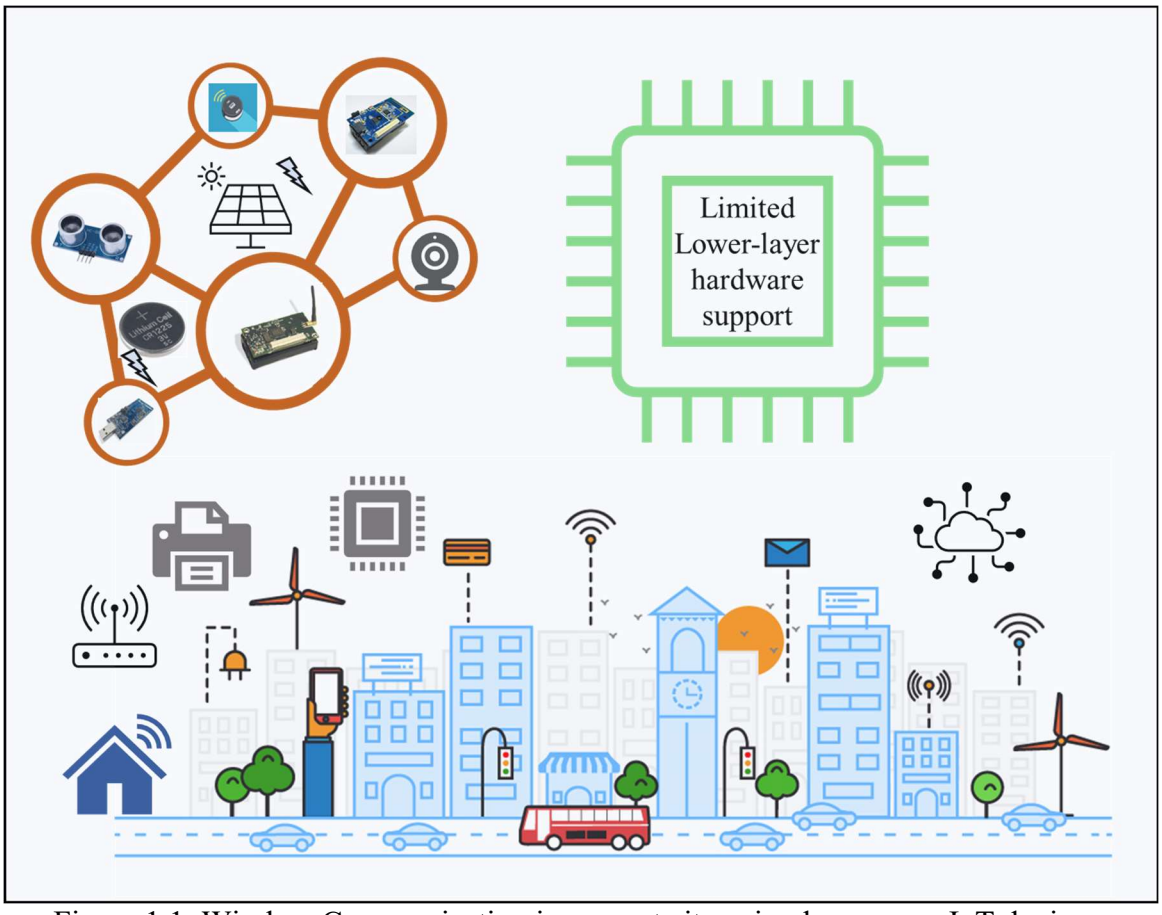

Figure 1.1: Wireless Communication in a smart city using low-power IoT devices.

In these resource constrained IoT and Wireless Sensor Networks, it is necessary to design network protocols that can judiciously utilize the available networking resources as well as minimize energy consumption. Multiple standard protocols have been developed at different layers for the IoT networking stack. Traditionally, these wireless network protocols are designed based on standards, heuristics and past experience of human designers. As for example, most of the well-known Medium Access Control (MAC) layer protocols such as ALOHA [4], CSMA [5], and their derivatives including Bluetooth [6], Zigbee [7], WiFi [8] are products of such design process. The same is true for the used routing layer protocols, such as Epidemic Routing [9], A^4LP Routing [10], PRoPHET routing [11] etc. The choice of a network protocol is often steered by the availability of lower layer transceiver hardware support for carrier sensing, collision detection, and communication energy budgets.

A notable drawback of network design using standardized protocols is that the resulting logic often underperforms in application-specific scenarios caused by topology and load heterogeneity, and other factors. For example, with ALOHA MAC, the throughput of a network starts falling at higher loads due to collisions, thus wasting precious energy resources in an IoT/sensor network. The situation is compounded in the presence of various forms of heterogeneities. In a network with arbitrary mesh topology, a few nodes may be in more disadvantageous positions as compared to others in terms of the collisions experienced. Thus, those nodes receive a less share of the available wireless bandwidth. In other words, the network performance is heavily dependent on the network operating conditions, such as traffic load, topology etc. Thus, for maximizing performance, network parameter tuning, and optimization is required for different operating conditions. This problem aggravates in scenarios of dynamic network conditions, such as time-varying traffic and/ or topologies, where these standard protocols cannot adapt to the

changing network dynamics. The root cause of these limitations is the fact that the network nodes are statically programmed with standardized protocol logic that lacks flexibility and abilities to learn optimal behavior in specific topology and load scenarios.

In order to overcome these limitations of the standard protocols, it is shown in this thesis how Reinforcement Learning (RL) and its variants can be leveraged for network protocol synthesis. The idea is to model the network behavior as a Markov Decision Process (MDP) and solve it using RL as a temporal difference solution. The main motivation of using Reinforcement Learning for network protocol synthesis is that RL provides opportunities for the nodes to learn situation-specific policies on the fly. This allows the nodes to learn policies that can maximize network performance for different network operating conditions. Also, owing to the real-time adaptability of RL, the nodes can adjust their learnt logic according to the temporal variations of network conditions.

While the RL-driven network protocols solve the shortcomings of the traditional protocols as mentioned above, there are certain challenges and concerns that need to be taken care of while developing the frameworks for network protocol synthesis. One of these challenges is the requirement of additional memory, computation and energy resources in the wireless nodes for successful implementation of learning algorithms. In addition, the convergence of these RL-based protocols need to be fast enough to adjust to dynamic network conditions and also to minimize wastage of precious networking resources due to non-optimal solutions. Another challenge in designing these protocols is the sharing of information and learnt policies among the nodes, especially in sparse networks with limited connectivity and in networks with wireless bandwidth constraints. These aspects of learning-based protocol synthesis are considered while

developing the framework for this thesis, that can solve the limitations of the existing protocol design approach and make it suitable for use in practical and real-world applications.

1.2 Introduction to Reinforcement Learning

1.2.1 Markov Decision Process

A Markov Decision Process (MDP) [12] is a stochastic sequential decision process, in which the decision maker or the agent decides the best possible action to take in a certain scenario (denoted by state) based on rewards it received for such actions in the past. A distinctive property of an MDP is that it follows Markovian property, that is, the transition from one system state to another depends only on the current state and the current action. Formally, an MDP can be defined by the tuple $(\mathcal{S}, \mathcal{A}, T, \mathcal{R})$, where $\mathcal{S}$ is the set of all possible system states; $\mathcal{A}$ is the action space; T is the transition matrix specifying the probability of transition from one state to another as a result of an action, and finally $\mathcal{R}$ is a reward function. At each decision epoch, the agent observes the current state in which the system is in. Based on this observation, the agent selects a particular action from the action space $\mathcal{A} = \{A_1, A_2, \dots, A_M\}$. The selection of a specific action from the action space results in a change of the state of the environment where the next state is decided based on the transition probability. The system transitions take place among a set of possible states $\mathcal{S} = \{S_1, S_2, \dots, S_K\}$, depending on the action taken. The feedback of the agent's actions is received in terms of a numerical entity called as reward. The solution of the MDP is a set of actions that maximizes the long-term expected reward of the system.

1.2.2 Reinforcement Learning

Reinforcement Learning (RL) is a class of algorithms that can be used for solving an MDP [12]. These algorithms can be broadly classified into model-based and model free approaches. Model-

based RL algorithms, such as dynamic programming, learn a model of the environment (for example prediction of the next state and reward) and find the optimal action based on such predictions. The term model-free means that unlike the classical dynamic programming methods, these algorithms do not require an exact mathematical model of the system. Instead, the learning agent uses samples of state transitions and rewards by interacting with the environment to estimate the state-action value functions.

Q-Learning [13] is a temporal difference method for Reinforcement Learning algorithms. In Q-Learning, the agent interacts with the environment repeatedly to learn the best possible set of actions to maximize the long-term reward. In Q-Learning, the agent maintains a Q table where the entries correspond to the Q values for all state-action pairs. The agent prefers the action with the highest Q value for a specific state. The generalized Q value update Equation is given by Eqn. (1.1). $Q(s,a)$ represents the Q value for the current state-action pair (s,a); r is the reward received; s' is the next state the system transitions to because of action a. The hyperparameters α and γ represent learning rate and discount factor, respectively.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r(s,a) + \gamma \times \max_{\forall a' \in A} Q(s',a') - Q(s,a)\right] \tag{1.1}$$

In Multi-agent Reinforcement Learning [14], as opposed to single agent scenario discussed so far, the environment is shared by all the agents involved in learning the optimal policy. As a result, the changes in the environment states are dependent on the independent actions taken by all the agents. Moreover, the rewards received by an agent depend not only on its own action, but also on the actions of all other agents. An agent may be penalized even for a good action when other agents' actions are not optimal. Hence, while learning an optimal policy, the agent has to take care of the dependency of the reward and state transition on all other agents' actions.

The two broad approaches for learning in a multi-agent environment are 1) Centralized Learning and 2) Distributed Learning. In centralized learning approach [15] [16], a centralized entity learns the optimal policies for all the agents in the system. However, this is not a scalable mechanism, as the learning agent has to maintain Q-values for all the agents and the Q-table will grow in size with an increase in number of agents, making it computationally inefficient as well. More details on the drawbacks of using such a centralized approach from the perspective of network protocol synthesis are provided in the subsequent chapters. To overcome these limitations, distributed or decentralized learning [17] [18] approach makes the nodes learn the optimal policy independently. Each agent maintains its own Q table and does not explicitly communicate with other agents. Thus, an agent is unaware of the actions taken by the other agents. However, in this distributed learning scenario, it becomes more challenging to handle the dependency of rewards on others' actions because of the limited information about the other agents. There are approaches and algorithms that are detailed in this thesis to overcome these challenges of learning in a distributed setting and limited information visibility. Note that in this thesis, the terms distributed learning and decentralized learning are used interchangeably.

1.2.3 Multi-Armed Bandits (MAB)

Multi-Armed Bandits (MAB) is a special class of Reinforcement Learning in a non-associative setting [12]. It is applicable in situations which do not involve learning in different states of a system. In other words, there is no concept of state as in traditional reinforcement learning [12]. A much-explored variant of MAB is the 'k-armed bandit' problem, where the learning agent (bandit) has k possible arms or possible actions to choose from. Each of the k actions has an associated stochastic reward, the distribution of which is not known to the learning agent. After an arm/action is chosen, the agent gets a sample of that reward following the unknown

distribution. In other words, it gets feedback in terms of a numerical reward on how good or bad a selected action is. The agent's goal is to maximize the total accumulated reward over infinite time horizon by learning to estimate the reward distribution of the possible actions.

Formally stated, the value for an action a is denoted as: $q_*(a) \leftarrow E[R_t \mid A_t = a]$. At each timestep t, the value of each action a is estimated iteratively as $Q_t\ (a)$. The model is said to converge when the estimate $Q_t\ (a)$ becomes close to the true value for action a, that is, $q_*(a)$. The generalized MAB update rule is given by:

$$New\ estimate = Old\ estimate + Step\ size \times [Reward - Old\ estimate] \qquad (1.2)$$

For a stationary problem, meaning where the reward distributions for the actions are not time-varying, the step size decreases per step in order to provide equal weightage to the rewards in each step. However, for a nonstationary situation, it is useful to provide more weightage to the recent rewards. This is achieved by making the step size constant, that is, $\alpha_n = \alpha$.

1.3 Reinforcement Learning-based Network Protocol Synthesis

The core idea of this thesis is to introduce the concept of leaning-enabled network protocol synthesis for resource-constrained wireless networks. The online learning characteristics of RL and MAB are leveraged for developing multiple protocol-synthesis frameworks with multidimensional performance objectives. With the generalized long-term goal of using RL for protocol design at different networking layers, this work primarily focuses on Medium Access Control (MAC) layer protocol synthesis. Throughout the thesis, we develop protocols that deal with medium access performance improvement from multiple perspectives, viz., network throughput, access delay, energy efficiency and wireless bandwidth usage. Towards the later part of the thesis, we also investigate the efficiency of these learning-driven protocols in the presence

of malicious nodes. In this section, we go over the specific access control problems that have been dealt with in this thesis.

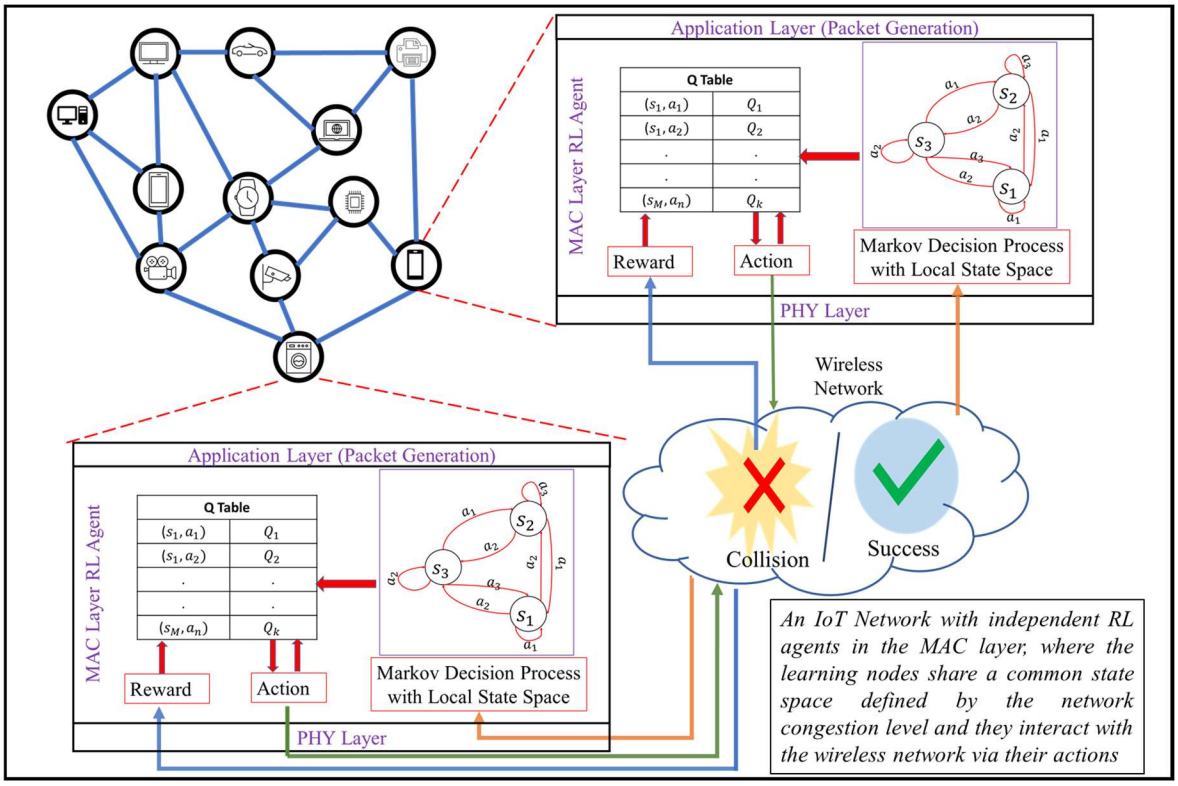

Figure 1.2: Modeling MAC layer protocol logic as Markov Decision Process (MDP).

1.3.1 Random Access MAC Protocol Synthesis

We first introduce the concept of protocol synthesis in Random Access MAC arrangement. The best practice for programming random-access MAC logic is to use known standardized protocols, including, ALOHA, CSMA, CSMA-CA, and their higher derivatives, such as Bluetooth, Zigbee, etc. The specific protocol is chosen for an application based on the lower layer hardware support for carrier sensing, collision detection, and the corresponding energy constraints. A notable drawback of network design using standardized protocols is that the resulting logic often underperforms in application-specific scenarios caused by topology- and load-heterogeneity, and other factors. For example, with baseline ALOHA MAC, the throughput of a network starts falling at higher loads due to collisions, thus wasting precious energy resources in an IoT

network. The situation is compounded in the presence of various forms of heterogeneities. The root cause of these limitations is the fact that the network nodes are statically programmed with standardized protocol logic that lacks flexibility and abilities to learn optimal behavior in specific topology and load scenarios. The proposed learning paradigm sets out to address that.

The key concept is to model protocol logic as a Multi Agent Markov Decision Process (MAMDP) [19] [20], and solve it dynamically using Distributed Reinforcement Learning as a temporal difference solution [12] under varying traffic, topology, and other network conditions. An MDP solution in this context is a set of transmission actions taken by individual network nodes, each of which acts as a separate RL agent (Fig. 1.2). In this setting, the wireless network acts as the RL environment. Based on the RL actions taken (such as different transmission strategies), the nodes receive feedback (such as success or packet collisions) which can be used as reward for updating learning parameters. The idea here is that over time, the nodes would learn policies such that maximum network performance can be achieved for different network conditions. The proposed framework supports heterogenous traffic load, network topology, and node-specific access priorities while striking a desired balance between node and network level performance. Learning adjustments to temporal variations of such heterogeneities and access priorities are also supported by leveraging the inherent real-time adaptability of Reinforcement Learning. It is shown in this thesis that the nodes can learn to self-regulate collisions in order to attain theoretically maximum MAC level performance at optimal loading conditions. For higher load, the nodes learn to adjust their transmit probabilities in order to reduce the effective load and maintain low levels of collisions, thus maintaining the maximum MAC layer performance.

1.3.2 TDMA-based MAC Protocol Synthesis

The concept of protocol synthesis is then extended to networks with time-slotted capacity. In this setting, the learning-driven access protocol is mainly developed for networks that do not have time synchronization capabilities. The motivation here is that accurate time synchronization among wireless networks nodes can be expensive to realize, especially in low-cost nodes with limited processing and communication resources. Moreover, the MAC layer performance in such networks can be very sensitive to even slight perturbations in the quality of time synchronization [21]. Hence, this learning-enabled framework aims to develop a time asynchronous TDMA protocol useful for low-cost transceivers in Wireless Sensor Networks (WSNs) and Internet-of-Things (IoTs).

The MAC slot allocation has been modelled as a distributed Multi-Agent Multi-Armed Bandits (MAB) problem, where each node acts as a 'k-armed bandit' agent, where an arm represents a MAC transmission slot that the agent can choose. Each node maintains its own notion of a fixed duration transmission frame with k-slots. Since time is not synchronized, the frames for the nodes are also not synchronized. Each node can independently select a slot, which is an arm in MAB. The learning framework allows the nodes to learn arm selection policies, together that constitutes a transmission slot selection protocol, in a distributed manner. After learning convergence, the system ensures that there are no overlapped transmissions (i.e., collisions) in the system (Fig. 1.3). In this thesis, the working of the learning-synthesized protocol is demonstrated for networks with both fully connected and partially connected arbitrary mesh topologies. The nodes in a partially connected topology learn to reuse bandwidth spatially by choosing fully to partially overlapping transmission slots when they are outside their mutual realm of influences.

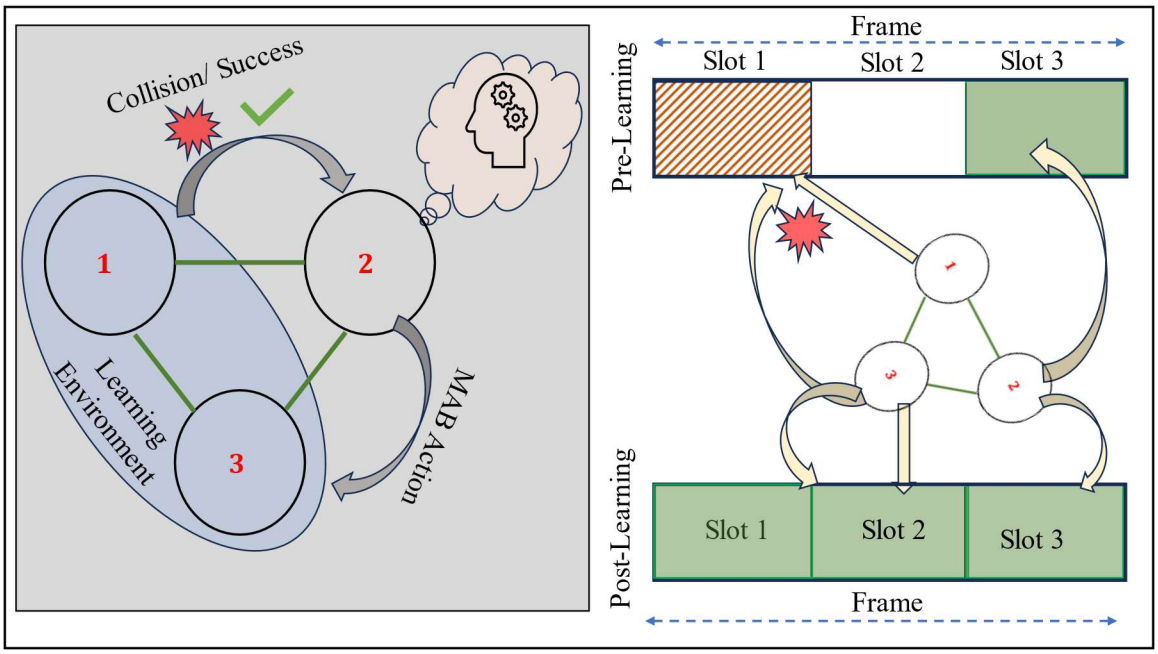

Figure 1.3: Multi-Armed Bandit for TDMA slot allocation.

1.3.3 Data Flow and Energy Management using RL

Building on the Multi-Armed Bandit-based slot allocation mechanism mentioned above, we extend the study to develop a Reinforcement Learning based framework for flow and energy management in energy-constrained wireless networks. Efficient scheduling for turning wireless transceivers on and off is an effective energy management mechanism in embedded networks of resource-constrained sensors and IoTs. In traditional approaches [22] [23], such schedules are typically pre-programmed in nodes based on manual settings and heuristics that are designed based on specific target network and traffic scenarios. As a result, such schedules cannot adapt well to network and traffic dynamics and time varying heterogeneity. Shortcomings are usually manifested in terms of not being able to maintain the desired balance between network performance (i.e., throughput and delay) and energy efficiency. In this thesis, a learning framework is proposed with the goal of overcoming these shortcomings by making the wireless nodes learn appropriate sleep-listen-transmit policies and adapt them in response to changing and

heterogeneous network and traffic conditions. This research develops an online reinforcement learning framework for sleep-awake scheduling in flow-based wireless networks, with and without time synchronization.

Learning in this framework happens in three distinct tiers. In the first tier, using Multi-Armed Bandits (MAB) learning, nodes independently learn to select collision-free transmission slots. The second tier uses Reinforcement Learning (RL) in order to learn whether to transmit or to sleep in the transmission slot selected in tier-1. This learning happens in a per-flow context in which multiple data flows may exist within each network node. The objective of the learning agent in this tier is to save energy by judicious sleeping, while keeping end-to-end packet delay low. Finally, in tier-3, another Multi-Armed Bandit agent learns whether to sleep or remain awake during a slot that was decided to be a non-transmitting slot by the MAB agent in tier-1. This agent ensures that the node remains awake only on the slots it is intended to receive packets from one of its neighbors, thus minimizing the idling energy expenditure. The objective of the coordinated learning by these three agents is to maximize per flow throughput while keeping energy expenditure at check via judicious sleep-wake decisions. In order to make the proposed system adaptive to heterogeneous traffic and scalable with network topologies, a contextual deep Q-learning based approach is developed, where a neural network is used for transmission scheduling for a time-varying data rate. In addition to being adaptive to network and traffic heterogeneity and dynamics, the proposed learning module is able to handle the trade-offs between throughput and energy efficiency based on application-specific requirements. The three-dimensional performance improvement of the adaptive and scalable framework is pictorially depicted in Fig. 1.4.

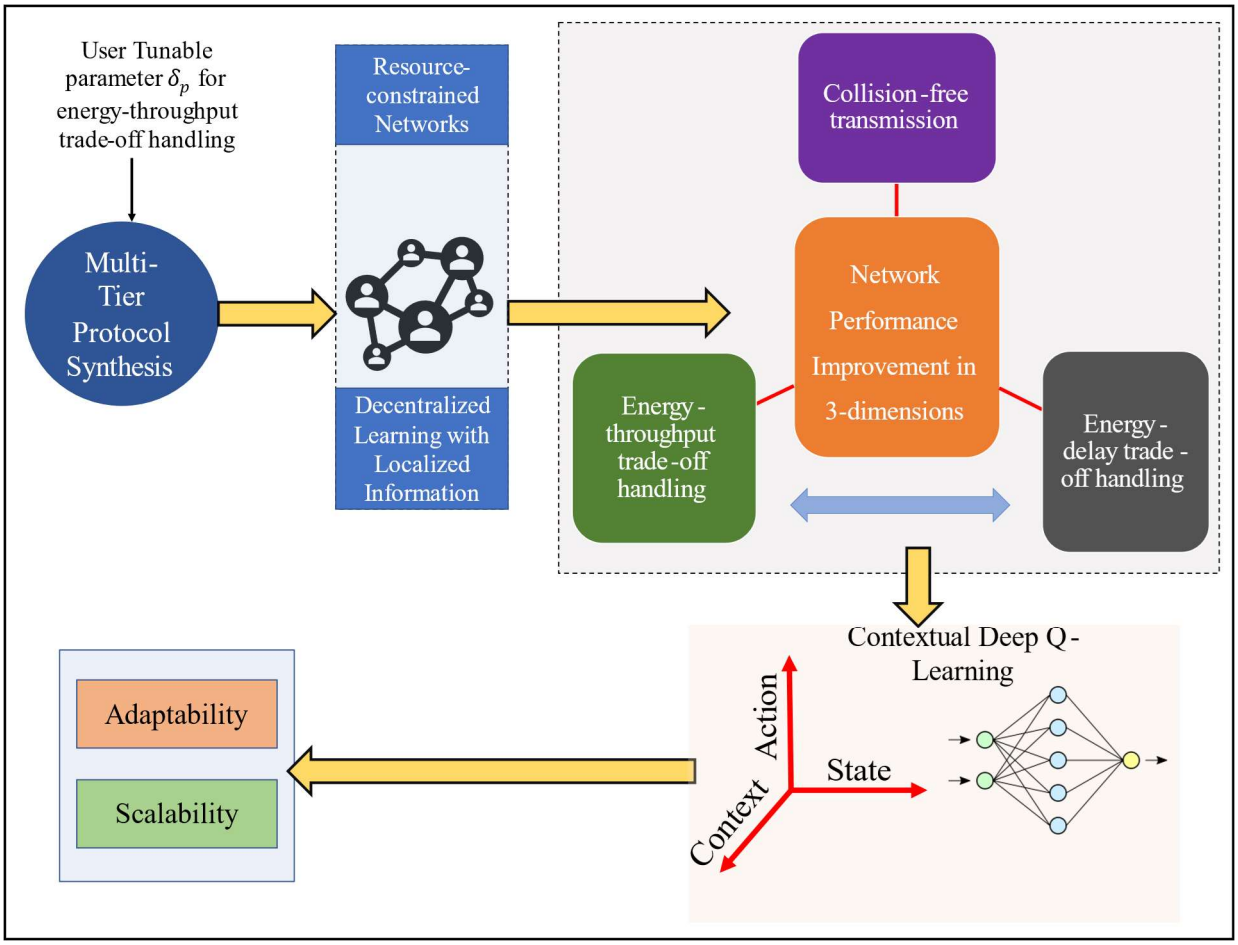

Figure 1.4: Three-dimensional Network Trade-off handling using Contextual Deep Reinforcement Learning.

1.3.4 Cooperative RL for Energy Management in Networks with Energy Harvesting

Building on the RL-based protocol synthesis for energy and data flow management formulated with the assumption of a homogeneous and temporally invariant energy profile, we now broaden its scope to encompass networks characterized by a spatiotemporal energy profile. Such spatiotemporal energy variation exists in wireless networks with nodes capable of harvesting energy. The harvested energy can be from different sources, viz., solar energy, vibration-based energy, etc. Such an energy harvesting node has a specific inflow of energy as a function of time and location. Traditional sleep scheduling protocols, where nodes are preprogrammed, do not allow nodes to react according to the spatiotemporal energy profiles and variabilities. Hence, in

this thesis, an online learning-based paradigm is proposed that allows the nodes to learn a joint transmit-sleep scheduling policy in order to overcome the above limitations. This is accomplished using a cooperative RL module, where two learning agents, deployed per node, learn cooperatively, a joint sleep-transmit schedule to achieve the network performance goals. In addition, the nodes share learning confidence parameter with their neighbors, that is used for a weighted update of learning parameters to account for any bad learning updates of neighboring nodes. These two levels of learning cooperation are denoted as inter and intra-node cooperation as shown in Fig. 1.5. This cooperative learning makes the framework suitable for multi-hops network so that learning errors are not propagated along the data flow.

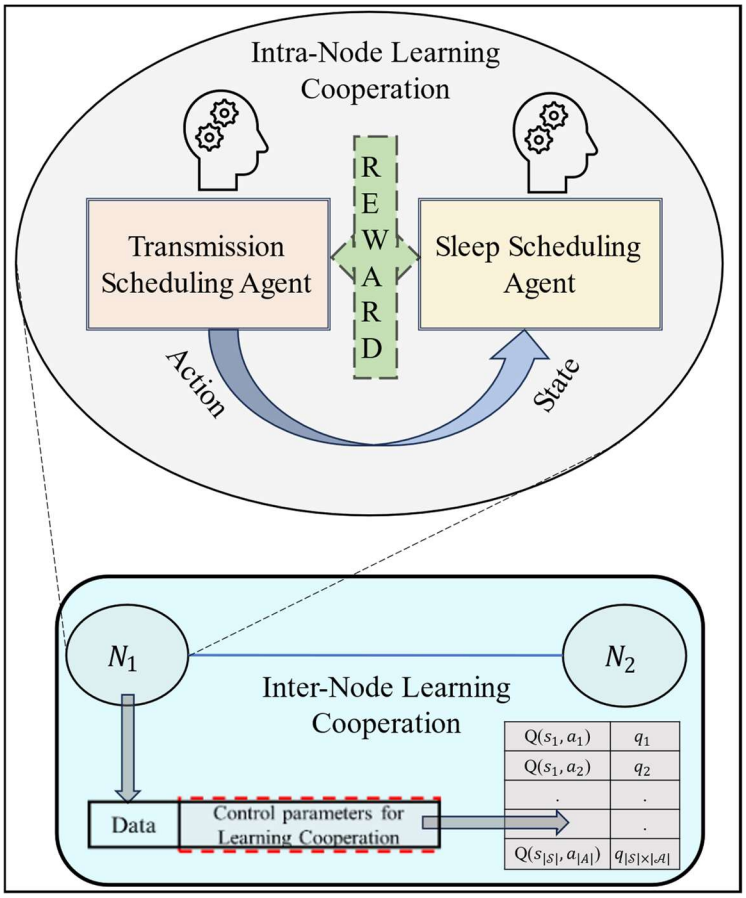

Figure 1.5: Cooperative Reinforcement Learning for flow-energy management.

1.3.5 Thwarting MAC Layer Attacks in Wireless Networks using Multi-Armed Bandits

We finally investigate the efficacy of the protocol synthesis paradigm in the presence of unreliable information because of malicious nodes in a network. To this end, a learning-driven approach for decentralized slot allocation is developed for scenarios where malicious nodes try to disrupt the TDMA schedule of other nodes. The learning policies are developed with the goal of defending against several quasi-random MAC attacks. The primary learning objective is to minimize the degradation in network performance caused by the malicious nodes. And this is done while minimizing the bandwidth share of the malicious nodes. These objectives are achieved using a Multi-Armed Bandit (MAB)-driven architecture that allows the nodes to learn transmission schedule on-the-fly and without the need for any central arbitrator. Two different scheduling policies are introduced: robust and reactive policies; and the characterization of these policies and their use in different application-specific scenarios are presented. An analytical model of the system is developed to find the benchmark throughput for different malicious policies. It is demonstrated that the proposed framework allows network nodes to learn slot scheduling with the computed benchmark throughput.

1.4 Dissertation Objectives

The main goal of this dissertation is to develop a multi agent learning framework for protocol synthesis in wireless networks. The framework is targeted for resource-constrained networks with thin energy budget. The developed approach is tuned for random-access and TDMA-based MAC arrangements, and is generalized for different mesh topologies and heterogeneous traffic patterns. The performance objective of the learning-synthesized protocols is to improve network efficiency from multiple perspectives, viz., throughput, delay, fairness, bandwidth energy usage.

Specifically, the working of the developed architecture is demonstrated to achieve the following goals.

1. The first goal is to develop a decentralized Reinforcement Learning framework for random access MAC protocol synthesis in low-cost and low-complex networks with simple wireless transceivers having limited hardware capabilities. The developed design aims at maximizing network throughput, fair sharing of wireless bandwidth among nodes and handling network traffic heterogeneity.

2. The concept of random-access MAC protocol synthesis is extended for partially connected networks with limited information visibility. The main aim here is to maximize network throughput while considering fair share of bandwidth, in the presence of heterogeneous traffic, topology and QoS requirement.

3. While the above two objectives cater to networks with random access MAC arrangement, the third goal of this dissertation is to demonstrate learning-based protocol synthesis for time-slotted MAC access. The aim here is to develop a Multi-Armed Bandit based architecture for slot allocation in resource-constrained wireless networks without network time synchronization. A novel Hysteretic MAB and slot-defragmentation-based design is proposed for managing trade-off between learning convergence time and bandwidth utilization efficiency.

4. The concept of RL-based MAC protocol synthesis is extended to develop a multi-tier learning framework for data flow and energy management in networks with limited energy resources, which is the fourth objective of this thesis. The multi-tier system aims at managing trade-off between throughput and energy efficiency by making judicious decisions on sleep-listen-transmit scheduling. The developed protocol synthesis framework is refined to

demonstrate its effectiveness in adapting to time-varying flow rate and scalable with network size with the use of a Contextual Deep Q-Learning framework.

5. With the goal of broadening the applicability of the learning-synthesized protocol to networks with spatiotemporal energy profile, a learning model is developed for joint transmit-sleep scheduling in energy-harvesting nodes. The proposed system uses a cooperative RL approach, with a localized learning confidence parameter sharing strategy, where two learning agents, deployed per node, jointly learn transmit and sleep scheduling strategies to manage network energy budgets.

6. The final objective of the thesis is to analyze the ability of the learning framework to deal with unreliable information in networks with coexisting malicious nodes. Specifically, we consider the problem of TDMA slot allocation in presence of malicious nodes trying to disrupt the schedule of other nodes. The developed MAB-model aims at minimizing the throughput reduction caused by the malicious nodes and reducing the effective bandwidth share of the malicious nodes themselves.

1.5 Scope of Dissertation Thesis

In this thesis, we address all the objectives of the dissertation as listed in the last section, including published findings related to these objectives. We have tried to develop a holistic approach for learning-enabled protocol synthesis for solving a wide range of access control problems. The target mechanisms developed in this thesis are intended primarily for applications in Internet of Things (IoT) and Wireless Sensor Networks (WSN) with low-cost wireless transceivers having limited hardware capabilities. A pictorial summary of the investigated issues in this thesis is shown in Fig. 1.6. The overall flow of the thesis is structured as follows.

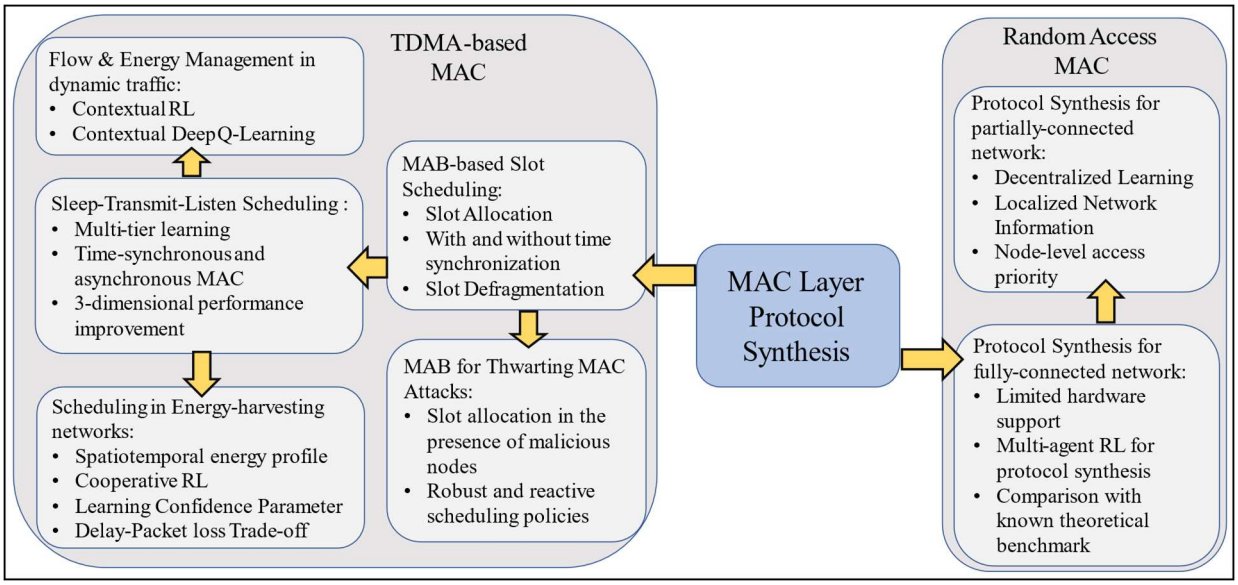

Figure 1.6: Pictorial representation of the investigated issues in this thesis.

Chapter 2 presents a literature review of RL and MAB-based approaches used for performance improvement in wireless networks. In this chapter, these existing RL and MAB based approaches are introduced and summarized, and it is analyzed which type of MAC allocations problems they solve in different network and loading conditions. We also identify the limitations of these existing mechanisms and show how the approach proposed in this thesis overcomes these limitations.

In 3, we propose a multi-agent reinforcement learning based medium access framework for resource-constrained wireless networks. It is shown that by learning to adjust MAC layer transmission probabilities, the protocol is not only able to attain theoretical maximum throughput at an optimal load, but unlike classical approaches, it can also retain that maximum throughput at higher loading conditions. The findings of this chapter have been reported in [24].

Chapter 4 extends the concept of protocol synthesis developed in chapter 3 for partially connected networks with limited information availability. A distributed and multi-Agent RL framework is

used as the basis for protocol synthesis. Distributed behavior makes the nodes independently learn optimal transmission strategies without having to rely on full network level information and direct knowledge of the other nodes' behavior. We have reported the study related to chapter 4 in [25] [26].

Chapter 5 deals with transmission scheduling in time-slotted access arrangement using Multi-Armed Bandits. MAC slot allocation is formulated as an MAB problem where the nodes act as independent learning agents and learn transmission policies that ensure collision free transmissions. The findings of chapter 5 have been documented in [27] [28].

Chapter 6 proposes distributed defragmented backshift operation for managing the bandwidth redundancy-convergence time trade-off in networks without time synchronization capability. In order to reduce the bandwidth overhead while maintaining a desired MAB learning speed, a novel slot defragmentation mechanism is introduced. It is shown that using the proposed mechanism, performance close to time-synchronous network can be achieved. The results presented in this chapter are published in [27] [29].

In chapter 7, a learning-based framework for MAC sleep-listen-transmit scheduling in wireless networks is proposed. It shows how the framework allows wireless nodes to learn policies that can support throughput-sustainable flows while minimizing node energy expenditure and sleep-induced packet drops and delays. We have presented the findings of this chapter in [30].

Chapter 8 addresses the limitations of the RL-based sleep scheduler, viz., scalability with network size and adaptability to dynamic load. This is achieved using a Contextual Deep Q-Learning (CDQL) and Multi-Armed Bandit-based framework, that makes the system adaptive to dynamic and heterogeneous network traffic conditions. The trade-off between energy efficiency and

learning convergence behavior is analyzed and it is shown how Deep Q-Learning can manage this trade-off. We have documented the study pertaining to this chapter in [31].

In chapter 9, we extend the learning-driven sleep scheduling framework for energy harvesting network with spatiotemporal energy profile which is achieved through the cooperation of multiple learning agents for joint transmit-sleep scheduling in such networks. The findings of chapter 9 are presented in [32], [33].

Chapter 10 aims at exploring the ability of the learning paradigm for scheduling in the presence of malicious nodes trying to disrupt TDMA schedule of other nodes. An MAB-based architecture is proposed for thwarting different quasi-random attacks of the malicious nodes. In this chapter, the malicious nodes are assumed to work independently without collusions. This study has been documented in [34].

Chapter 11 builds on the scenario developed in chapter 10 where the assumption of information sharing among malicious nodes are relaxed. Here the efficacy of the synthesized logic is demonstrated in scenarios where colluding malicious nodes attempt to disrupt slot scheduling collectively.

Finally, the thesis is summarized, and a list of future research directions is compiled in chapter 12.

Chapter 2: Related Work

Reinforcement Learning (RL) has been used in the literature for designing MAC protocols for wireless networks. In general, the main objective of using RL in wireless networks is to improve network performance in terms of throughput, energy, and latency. In this chapter, we investigate the existing works that explore RL and its variants as solutions to different access control problems.

2.1 Reinforcement Learning for Resource Allocation

There are works that deal with RL-driven access control mechanisms for random access MAC arrangement. In [35], the authors propose a MAC protocol designed specifically for wireless sensor networks with an objective of minimizing the energy requirement and maximizing the throughput of the network. The system is implemented in slotted time, where each node learns, using stateless Q-learning, to find a unique slot to transmit, so that the collisions are reduced. Besides assuming a homogeneous network, this work does not show how the system would behave for a very high network traffic.

Maximizing resource allocation, and hence throughput, for secondary users in a network is explored using a learning-based MAC protocol in [36]. In this time-slotted system, under the assumption of a homogeneous secondary network, Q-learning is used to solve a partially observable Markov process (POMDP) to reduce the interference between the primary and secondary users.

In [37] and [38], the authors have developed protocols, where a learning algorithm allows nodes to find an efficient radio schedule based on the node packet traffic and the traffic load of its neighbors so that performance of the network is improved. They show that the designed protocols

perform better, in terms of throughput and delay, compared to the standard MAC protocols, in dense, homogeneous networks.

A multi-agent reinforcement learning-based protocol has been proposed in [39] for optimizing resource constraints, such as, energy and bandwidth, in a dense wireless sensor network, where the nodes can adapt to the changing network traffic conditions. The nodes learn to self-configure in the network by adaptively activating the neighbor nodes, so as to improve the overall network performance.

Two Q learning-based research allocation protocols are proposed in [40], [41] aiming for maximizing the throughput in a dense network. The nodes without prior information about the network size, can learn to adjust to a dynamic network with increase the number of nodes in the network. The nodes with the help of carrier-sensing, learn to transmit/ wait [40] or to increase/ decrease the contention window [41] so that collisions are reduced.

The authors in [42], [43] have proposed MAB-based learning as a tool for slot selection in frame-based ALOHA. The learning-based protocols are shown to outperform standard MAC protocols, viz, slotted ALOHA.

RL based protocols are designed for an underwater network without the carrier sensing capacity of the nodes in [44] and [45]. This protocol provides efficient channel access to the nodes using Q learning. Because of the adaptive nature of the protocol, it is suitable for a dynamic environment. However, it is designed specifically for large, distributed networks.

In [46], the authors have proposed a deep reinforcement learning-based MAC protocol to maximize network throughput. The protocol was implemented in a slotted system, where the nodes learn when to transmit so that the network throughput is maximized. The learning node is

deployed in a network with other nodes using TDMA/ALOHA protocols. In this work, the results are only shown for a particular case, when the load is higher than optimal.

An RL-based MAC protocol is proposed in [47], which increases the network throughput by reducing the collisions. In a slotted-time system, the nodes learn using stateless Q learning to select an action strategy to select a slot efficiently so that collisions are reduced.

Reinforcement Learning and Multi-Armed Bandits have been used in the literature as mechanisms for TDMA slot selection in wireless networks. The work in [48] use Reinforcement Learning to develop a self-learning technique for allocating TDMA MAC slots. This topology-dependent contention-based mechanism is shown to achieve higher throughput than the existing standard MAC protocols. One common assumption used in these papers is the availability of time synchronization and nodes' collision detection abilities.

In [49], the authors proposed a distributed contention-based scheduling mechanism in wireless mesh networks. The work deals with assigning fair bandwidth allocation scheme for multi-radio multichannel WMNs. The authors in [50] propose a distributed algorithm for TDMA slot allocation in networks using two-hops neighbor information.

Reinforcement Learning-based joint time-slot and channel allocation in a Time Slotted Channel Hopping (TSCH) network is proposed in [51]. The paper also optimizes transmission power using RL to reduce energy usage. The design is based on the adaptability requirement of Industrial Internet-of-Things networks. A similar learning-based slot allocation scheme is developed in [52] for optimizing energy and packet delay in large networks with high traffic loading. Another RL-based congestion control scheme for satellite IoT networks is proposed in

[53], where the aim is to allocate channels efficiently in a TSCH network, with the central arbitration of a satellite.

2.2 Reinforcement Learning for Energy Management

Energy management in wireless networks using Reinforcement Learning has been an area of interest for researchers. In this context, this problem has been dealt with mainly from two perspectives: (a) radio sleep-awake scheduling ([54], [55], [56], [57]) and (b) transmission power control ([58], [59], [60]).

The use of Reinforcement Learning for sleep-awake duty cycle control is explored in [37], [61], [49], [62]. These mechanisms address a number of drawbacks in traditional non-RL-based access protocols including SMAC [63] and TMAC [64], viz, performance degradation in diverse topological conditions and in low network traffic scenarios. However, one notable shortcoming of these approaches is that they drop packets due to inappropriate sleeps during intended listen periods in certain situations. Furthermore, the functioning and performance of the system proposed in [37], [61] heavily depends on algorithmic hyperparameters that are chosen manually for different network and traffic conditions. Additionally, the mechanisms along with the one in [62] rely on an explicit control channel that the nodes use to exchange learning related information.

The approach in [54] uses RL to learn an optimal transmission duty cycle that minimizes energy expenditures while maintaining high throughputs and low access latency. It uses a contention-based slotted system with carrier-sensing abilities. Although the mechanism is shown to achieve better performance than comparable existing works, learning here depends on centralized arbitration of certain gateway nodes. Additionally, the assumption of complete availability of

instantaneous network state information at each node restricts the mechanism's practical feasibility. The framework presented in [65] uses Multi-Armed Bandits (MAB) to learn an optimal back-off period in a contention-based time-slotted underwater network. The objective is to simultaneously minimize collisions and energy usage. The main limitation here is that the approach works only for multi-point-to-point single hop networks with centralized coordination. The authors in [66], [67] use RL to learn and adaptive transmission duty cycles in wireless networks where a centralized gateway or a coordinator acts as the learning agent.

In [61], the authors have developed a MAC protocol named QL-MAC with the aim to preserve energy in order to extend the network lifetime. This protocol is claimed to be useful for high-density communications in wireless sensor networks to meet stringent energy requirements. Learning allows the nodes to infer each other's behaviors so that they can adopt a good sleep/active scheduling policy in a wireless sensor network. This limits the number of time slots when the radio is turned on. In this work, the authors have addressed the problem of energy optimization, without considering the throughput of the network.

A Reinforcement Learning-driven sleep scheduling mechanism for cooperative computing method is developed in the paper [68] for energy efficiency improvement. The paper develops a computing node selection algorithm for enhanced battery lifetime. A similar energy management scheme using deep RL for sleep scheduling and computation and communication resource allocation is proposed in [69]. The developed architecture is shown to be applicable for 5G Mobile Edge Computing Networks for Industrial IoT. In [70], the authors propose an RL-enabled sleep-scheduling algorithm integrated with compressive data gathering technique for decreasing the energy expenditure in WSNs. It uses Q-learning for maintaining a balance between data reconstruction and energy expenditure. The authors in [71] propose an RL-based solution that

helps resource-constrained nodes enhance their performance by saving battery power and maintaining the quality of transmitted data. However, these papers do not consider optimizing the transmission scheduling decisions, which play a significant role in efficient energy management.

The work in [59] proposes a Reinforcement Learning algorithm for controlling the access time and transmit power of the sensor nodes in WSN. The paper demonstrated that the proposed mechanism improves the Quality-of-Service, achieves high energy efficiency and transmission reliability. Similarly, the mechanisms in [58], [60] use centralized RL for transmission power control and relay station selection in order to achieve energy efficiency. The work in [72] developed a resource-scheduling mechanism using node-level deep Reinforcement learning in order to improve transmission reliability.

2.3 Research Gaps

In this section we present an overview of the limitations of the literature reviewed in this chapter. Specifically, the following research gaps can be pointed out.

1. Requirement of complex lower-layer hardware support: Many of the approaches for learning-driven access layer protocols rely on complex lower-layer hardware support, such as time-slotting, time synchronization, carrier sensing etc. However, such sophistications may be absent in many low-cost sensor/IoT nodes. In addition, these complexities often result in high energy expenditures.
2. Assumption of fully reliable learning observables: The current literature on learning-enabled network performance management assumes that the observed environmental feedback are reliable for learning updates. However, in many practical networks, these

assumptions are untenable, because of the presence of noise, malicious agents, etc.

3. Lack of a unified framework for handling multidimensional trade-off of throughput, delay and energy efficiency: The present approaches deal with efficient resource utilization for improving throughput while reducing access delay and energy expenditure. Nevertheless, these performance metrics are interdependent, and cannot be treated separately. There lacks a unified architecture that deals with the trade-off handling along these multiple dimensions.
4. Scalability Issues: The current widely accepted learning approaches for wireless network protocols depend on a centralized arbitrator in which a learning hub maintains and updates network-level information, and learns the optimal transmission policies (i.e., protocol) for all network nodes. Apart from the fact that centralized learning puts heavy computation burden on the central server, such an approach requires additional energy and bandwidth overhead for communicating the policies and learning observables frequently upload network information and download transmission policies to and from the centralized learning entity. Another issue from implementability perspective is that for networks with multiple number of hops, learning errors get accumulated at each hop, raising questions on scalability.

This thesis addresses the above gaps in the literature by accommodating decentralized learning, and removal of crucial assumptions of complex hardware capabilities, such as, carrier sensing, collision detection, and network time synchronization. The resulting framework is aimed for protocol synthesis in networks of low-cost and resource constrained sensor and IoT devices for which those assumptions are not expected to be valid. In addition, a multi-tier architecture is developed to consider a multi-dimensional trade-off handling between throughput, delay and

energy expenditure. With the goal of making the learning framework scalable and adaptive to changing networks, a contextual deep learning model is developed. A co-operative learning model ensures that learning errors do not accumulate over multiple hops in the network. In order to study the effect of unreliable learning observables, a setup is considered where certain malicious nodes attempt to degrade the performance of other nodes. Strategies are proposed for the non-malicious in order to defend against such attacks.

2.4 Summary

In this section, we reviewed the existing works that deal with network performance improvement using Reinforcement Learning and its variants. We summarize the specific problems that these papers aim at solving and a high-level approach that they follow to solve them. The key limitations of these works and the existing literature gaps are then encapsulated. Finally, we explain how this thesis fits in this research landscape and how the concepts developed here fills in the identified research gaps.

Chapter 3: Random Access MAC Protocol Synthesis for Fully Connected Topology

In this chapter, we introduce the concept of network protocol synthesis using Reinforcement Learning. Towards the long-term goal of developing a general-purpose learning framework, we start with a basic scenario in which wireless nodes run a rudimentary MAC logic without relying on carrier sensing and other complex features from the underlying hardware. In other words, the developed mechanisms are suitable for very simple transceivers that are found in low-cost wireless sensors, Internet of Things (IoTs), and other embedded devices. The access problem is formulated as a Markov Decision Process (MDP) and solved using Multi-Agent Reinforcement Learning (RL) with every node acting as a distributed learning agent. The solution components are developed step by step, starting from a single-node access scenario in which a node agent incrementally learns to control MAC layer packet loads for reining in self-collisions. The strategy is then scaled up for multi-nodes fully connected scenarios by using more elaborate RL reward structures. It is shown that by learning to adjust MAC layer transmission probabilities, the protocol is not only able to attain the benchmark throughput, but unlike classical approaches, it can also retain that maximum throughput at higher loading conditions. Additionally, the mechanism is agnostic to heterogeneous loading while preserving that feature. It is also shown that access priorities of the protocol across nodes can be parametrically adjusted. Finally, it is also shown that the online learning feature of reinforcement learning is able to make the protocol adapt to time-varying loading conditions.

3.1 Motivation

The current best practice for programming MAC logic in an embedded wireless node is to implement known protocols such as ALOHA, Slotted-ALOHA, CSMA, and CSMA-CA (i.e.,

WiFi, BT, etc.) depending on the available lower layer hardware support. The choice of such protocols is often driven by the heuristics and past experience of network designers. While such methods provide a standard method for network and protocol deployment, they do not necessarily maximize the MAC layer performance in a fair manner, especially in the presence of network and data heterogeneity. An example is when nodes without carrier sensing abilities run ALOHA family of protocols, their performance start degrading due to collisions when the application traffic load in the network exceeds an optimal level. Such problems are further compounded in the presence of various forms of heterogeneity in terms of traffic load, topology, and node-specific access priorities. The key reason for such performance gap is that the nodes are statically programmed with a protocol logic that is not cognizant of time-varying load situations and such heterogeneities. The proposed framework allows wireless nodes to learn how to detect such conditions and change transmission strategies in real-time for maximizing the network performance, even under node-specific access prioritization.

The key concept in this work is to model the MAC layer logic as a Markov Decision Process (MDP) [73] and solve it dynamically using Reinforcement Learning (RL) as a temporal difference solution [12] under varying traffic and network conditions. An MDP solution is the correct set of transmission actions taken by the network nodes, which act as the MDP agents. RL provides opportunities for the nodes to learn on the fly without the need for any prior training data. The developed mechanism allows provisions for heterogenous traffic load, network topology, and node-specific priority while striking the right balance between node level and network level performance. Learning adjustments to temporal variations of such heterogeneities and access priorities are also supported by leveraging the inherent real-time adaptability of Reinforcement Learning.

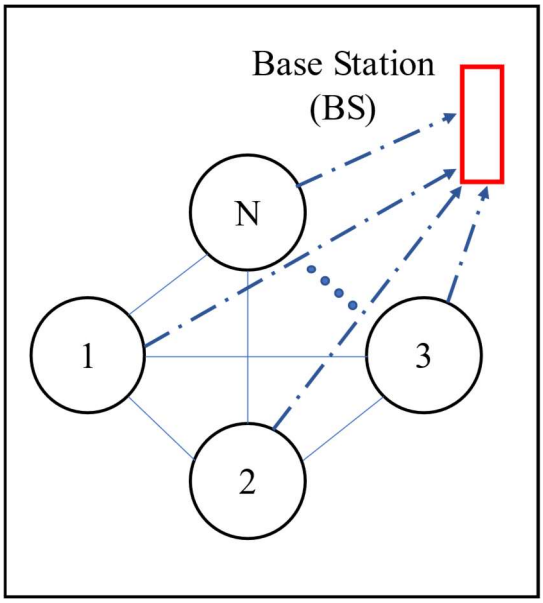

Figure 3.1: Network and data upload models for fully connected nodes.

3.2 Network and Traffic Model

The network model considered in this chapter is a multi-point to point sparse network. As shown in Fig. 3.1, N fully connected sensor nodes send data to a wirelessly connected base station using fixed size packets. Performance, which is node- and network-level throughputs, is affected by collisions at the base station caused by overlapping transmissions from multiple nodes. The primary objective of a learning-based MAC protocol is to learn a transmission strategy that can minimize collisions at the base station. It is assumed that nodes do not have carrier sensing abilities, and each of them is able to receive transmissions from all other nodes in the network.

3.3 Modeling Protocol Synthesis as Markov Decision Process

3.3.1 Markov Decision Process (MDP)

An MDP is a Markov process [12] in which a system transitions stochastically within a state space $\{S_1, S_2, S_3, \ldots\ldots, S_N\}$ as a result of actions taken by an agent in each state. When the agent is in state S_k and takes an action a_k, a set of transition probabilities determine the next system

state S_{k+1}. A reward, indicating a physical benefit, is associated with each such transition. Formally, an MDP can be represented by the tuple $(\mathcal{S}, \mathcal{A}, T, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of all possible actions, T is the state transition probabilities, and $\mathcal{R}$ is the reward function. For any system, whose dynamic behavior can be represented by an MDP, there exists an optimal set of actions for each state such that the long-term expected reward can be maximized [74]. Such optimal action sequence is referred to as a solution to the MDP.

3.3.2 Reinforcement Learning (RL)

RL is a class of algorithms for solving an MDP [75] [76] without necessarily requiring an exact mathematical model for the system, as needed by the classical dynamic programming methods. As shown in Fig. 3.2, an agent interacts with its environment by taking an action, which causes a state-change. Each action results in a reward that the agent receives from the environment. Q-Learning [13], a model-free and value-based reinforcement learning technique, is used in this work. Using Q-Learning, by taking each possible action in all the states repeatedly, an agent learns to take the best set of actions that represents the optimal MDP solution, which maximizes the expected long-term reward. Each agent maintains a Q-table with entries $Q(s, a)$, which represents the Q-value for action a when taken in state s. After an action, the Q-value is updated using the Bellman's equation given by Eqn. (3.1), where r is the reward received, α is a learning rate, γ is a discount factor, and s' is the next state caused by action a:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r(s,a) + \gamma \times \max_{\forall a' \in A} Q(s',a') - Q(s,a) \right] \tag{3.1}$$

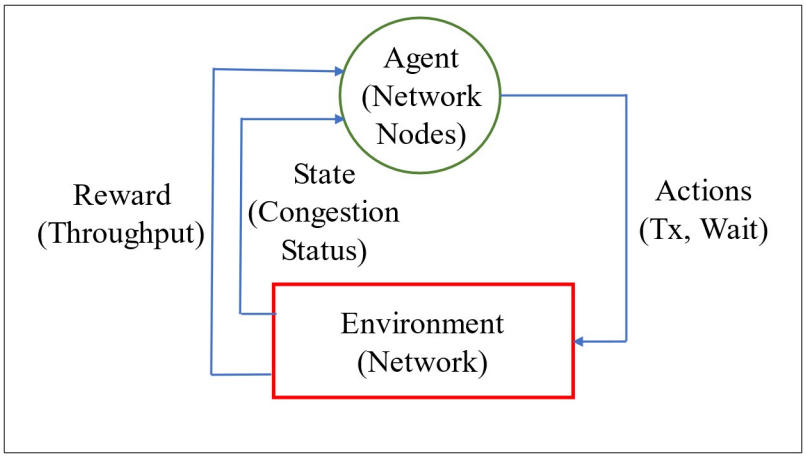

Figure 3.2: Reinforcement Learning (RL) for network protocol MDP.

3.3.3 Modeling Network Protocols as MDP

We represent the MAC layer protocol logic using an MDP, where the network nodes behave as the learning agents. The environment is the network itself within which the nodes/agents interact via their actions. The actions in this case are to transmit with various probabilities and to wait, which the agents need to learn in a network state-specific manner so that the expected reward, which is throughput, can be maximized. A solution to this MDP problem would represent a desirable MAC layer protocol.

State Space: The MDP state space for an agent/node is defined as the network congestion level as perceived by the agent. Congestion is coded with two elements, namely, *inter-collision probability* and *self-collision probability*. An inter-collision takes place when the collided packets at a receiver are from two different nodes. A self-collision occurs at a transmitter when a node's application layer generates a packet in the middle of one of its own ongoing transmissions. It will be shown later as to how using RL, a node learns to deal with both *self-collisions* and *inter-collisions*, thus maximizing the reward or throughput.

Inter-collision and self-collision probabilities are defined by Eqns. (3.2) and (3.3) respectively. To keep the state space discrete, these probabilities are discretized in multiple distinct ranges.

$$P_{SC} = \frac{Number\ of\ self\ collisions}{Number\ of\ transmitted\ packets} \tag{3.2}$$

$$P_{IC} = \frac{Number\ of\ inter\ collisions}{Number\ of\ transmitted\ packets} \tag{3.3}$$

Action Space: As for the agents' actions, two different formulations are explored, namely, *fixed actions* and *incremental actions*. With the first one, the actions are defined as the probabilities of packet transmissions by an agent/node. For the latter, the actions are defined as the change of the current transmission probability. Details about these two action space formulations and their performance are presented in Sections 3.5 and 3.6.

Rewards: In a fully connected network, each RL agent/node keeps track of its own throughput and those of all other agents. Using such information, an agent-i constructs a reward function as follows.

$$R_i = \{\rho \times S + \sum_{\forall i} \mu_i \times s_i + \sigma \times (f_i)\} \tag{3.4}$$

S is the network-wide throughput (expressed in packets/packet duration, or Erlang) computed by adding measured throughputs for all nodes including node-i itself; s_i is the throughput of node-i, and $f_i = -\sum_{\forall j \neq i} |s_i - s_j|$ represents a fairness factor. A larger f_i indicates a fairer system from node-i's perspective. The coefficients ρ and σ are learning hyper-parameters. The coefficient ρ provides weightage towards maximizing network-wide throughput, and σ contributes towards making the throughput distribution fairer. The node-specific parameter μ_i determines a media access priority for node-i. In addition to the baseline reward structure in Eqn. 3.4, a learning recovery protection is provided by assigning a penalty of 0.8 to all agents if the network-wide throughput S ever goes to zero.

3.4 Decentralized Multi-agent Reinforcement Learning MAC (DMRL-MAC)

Using the state and action spaces deigned in Section 3.3, we develop an Multi-agent RL (MRL) system in which each network node acts as an independent RL agent. Decentralized agent behavior give rise to a medium access control protocol that is termed as DMRL-MAC. We used a special flavor of co-operative multi-agent RL, known as Hysteretic Q-learning [14].

Hysteretic Q-learning (HQL): HQL is used in a cooperative and decentralized multi-agent environment, where each agent is treated as an independent learner. An agent, without the knowledge of the actions taken by the other agents in the environment, learns to achieve a coherent joint behavior. The agents learn to converge to an optimal policy without the requirement of explicit communications. The Q-table update rule in Eqn. (3.1) is modified here as:

$$\delta = r + (\gamma) \times \max_{\forall a' \in A} Q(s', a') - Q(s, a)$$

$$Q(s, a) \leftarrow \begin{cases} Q(s, a) + \alpha \times \delta, & if\ \delta \geq 0 \\ Q(s, a) + \beta \times \delta, & else \end{cases} \tag{3.5}$$

In a multi-agent environment, the rewards and penalties received by an agent depend not only on its own action, but also on the set of actions taken by all other agents. Even if the action taken by an agent is optimal, still it may receive a penalty as a result of the bad actions taken by the other agents. Therefore, in hysteretic Q-learning, an agent gives less importance to a penalty received for an action that was rewarded in the past. This is taken into consideration by the use of two different learning rates α and β. This can be seen in the Q-table update rule in Eqn. 3.5, where α and β are the increase and decrease rates of the Q values. The learning rate is selected based on the sign of the parameter δ. The parameter δ is positive if the actions taken were beneficial

for attaining the desired optimum of the system and vice-versa. In order to assign less importance to the penalties, β is chosen such that it is always less than α. However, the decrease rate β is set to non-zero in order to make sure that the hysteretic agents are not totally blind to penalties. In this way, the agents make sure to avoid converging to a sub-optimal equilibrium.

Hysteretic Q-learning is decentralized in that each agent maintains its own Q table. The size of a table in a node is independent of the number of nodes in the network, and it grows linearly with the number of its own actions. It is shown in [14] that with the right set of assumptions, the convergence behavior of hysteretic Q-learning is close to that of centralized learning.

In the proposed DMRL-MAC protocol, each node in the network acts as a hysteretic agent, which is unaware of the actions taken by the other nodes/agents in the network. The actions are evaluated by the rewards assigned using Eqn. (3.4).

3.5 Single Agent Reinforcement Learning

Before delving into multi-nodes scenarios, we experiment with the key concepts of RL-based MAC behavior with a single node that executes MAC logic for sending data to a base station. We have implemented single-node pure ALOHA simulation experiments in which the MAC layer transmits a packet whenever it arrives from the application layer. Fig. 3.3 shows single-node throughput comparison for ALOHA and RL-synthesized DMRL-MAC protocol. The maximum ALOHA throughput is around 0.68 Erlangs at an application layer load of 2.4 Erlangs. In the absence of inter-collisions with other nodes, throughput loss is solely due to the self-collisions discussed above. The throughput reduces with load asymptotically (not shown) as increasing self-collisions eventually prevent any packets from being successfully transmitted.

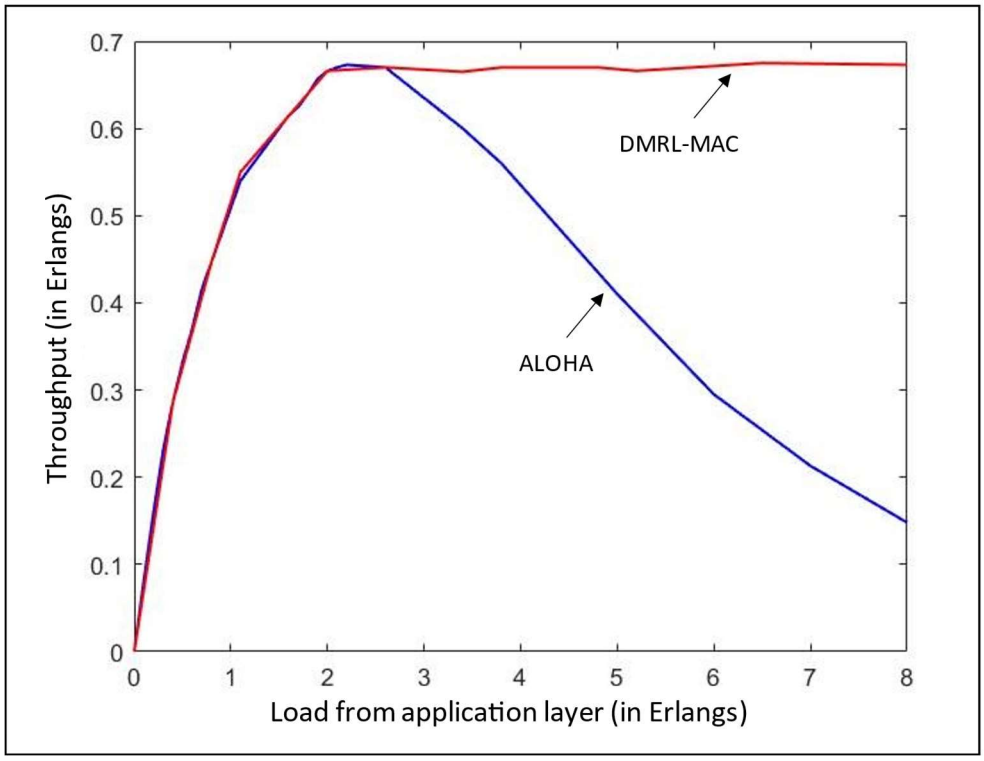

Figure 3.3: Single node throughput under self-collisions; throughput comparison between ALOHA and RL-synthesized MAC Logic.

Table 3.1: Baseline experimental parameters

No. of packets per epoch	1000
α	0.1
γ	0.95
$\in$	$0.5 \times e^{-epoch\ ID/200}$
ρ	1.0
σ	0
μ_1	0

On the other hand, if Q-Learning [13] is used for medium access following the reward formulation in Eqn. 3.4, the agent learns an appropriate action from action space specified in Section 3.3. The state space for single node RL-based MAC is obtained by discretizing the self-collision probability into 6 equal discrete levels. The learning hyperparameters and other relevant Q-Learning parameters are summarized in Table 3.1.

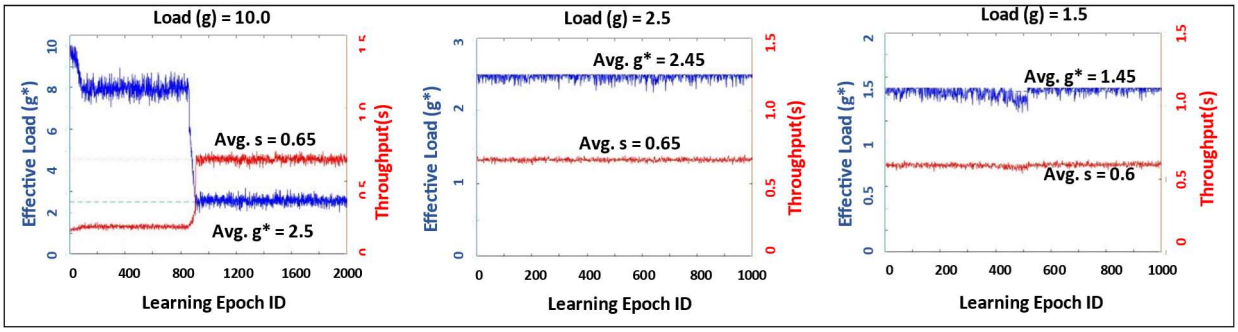

Figure 3.4: Convergence plots for RL-based MAC logic: effective load and throughput for three different initial loads using fixed action strategy.

As shown in Fig. 3.3, the RL-based MAC is able to achieve the maximum ALOHA throughput by learning to take the appropriate transmission actions from its available action space. However, unlike the regular ALOHA logic, the RL-based logic can maintain the maximum throughout even after the load exceeds 2.4 Erlangs, which is the optimal load for the ALOHA logic. This is achieved by adjusting the transmit probability so that the self-collisions are reduced. For ALOHA, if a node is in the middle of a transmission, it transmits the packets irrespective of its current transmission status. But with RL, the node can learn not to transmit when it is in the middle of an ongoing transmission. As a result, the effective load g^* to the MAC layer is lowered compared to the original load g from the application layer, thus maintaining the maximum throughout even when the application layer load is increased. Such learning provides a new direction for medium access, which will be explored further for multi-node networks later in this chapter.

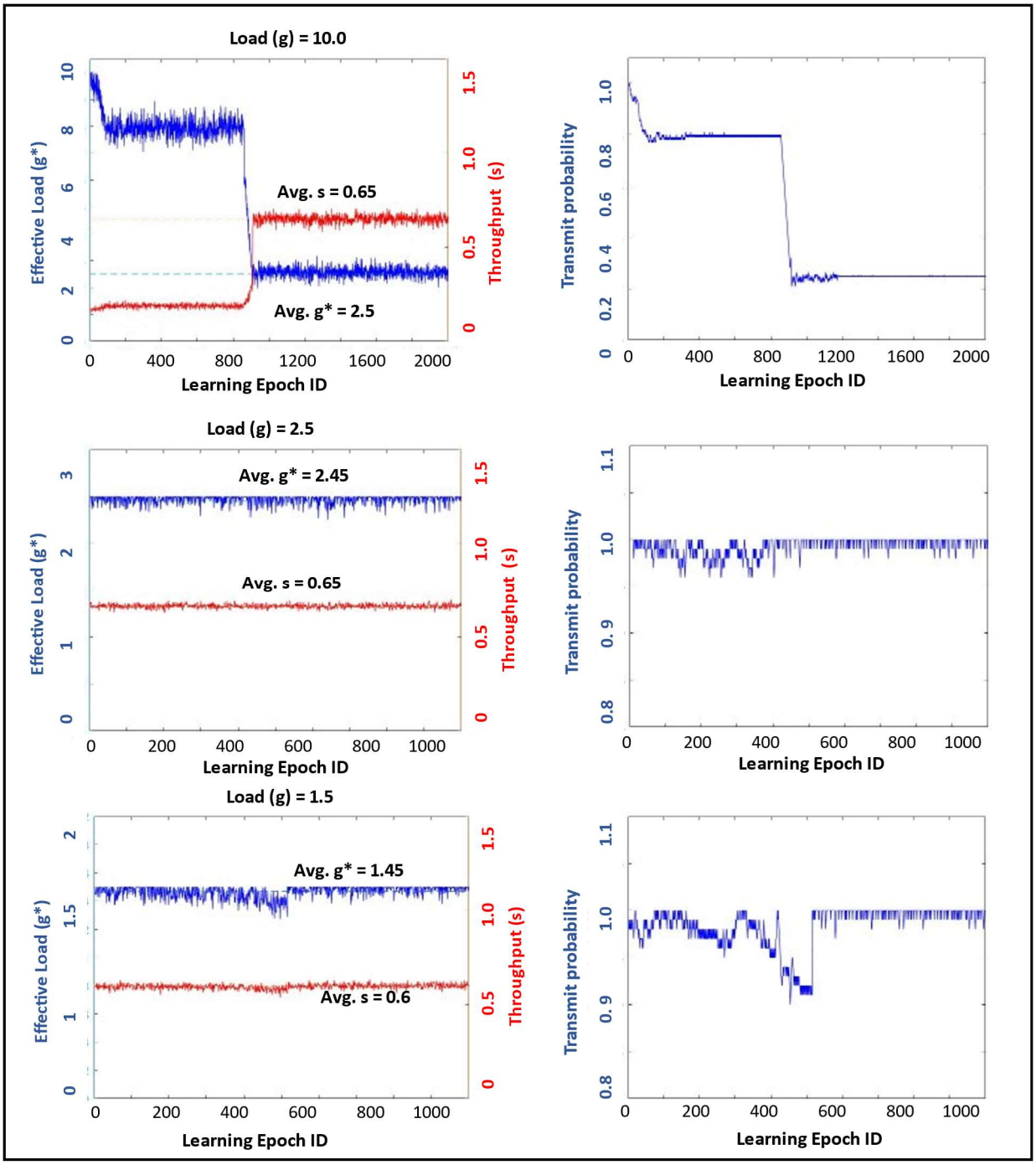

Figure 3.5: Convergence plots for RL-based MAC logic: effective load and throughput for three different initial loads using incremental action strategy.

As mentioned in Section 3.3, the RL-based MAC is implemented using two different action strategies by the RL agent: fixed action strategy and incremental action strategy. Figs. 3.4 and 3.5 show the convergence plots for these two cases respectively. It can be observed that the

learning convergence is faster for the fixed action strategy than that of the incremental one. This is because the search space for the optimal transmit probability is smaller when fixed actions are used as compared to the incremental actions. This is achieved at the expense of accuracy because the granularity of the transmit probability for fixed action strategy is less than that of the incremental action strategy.

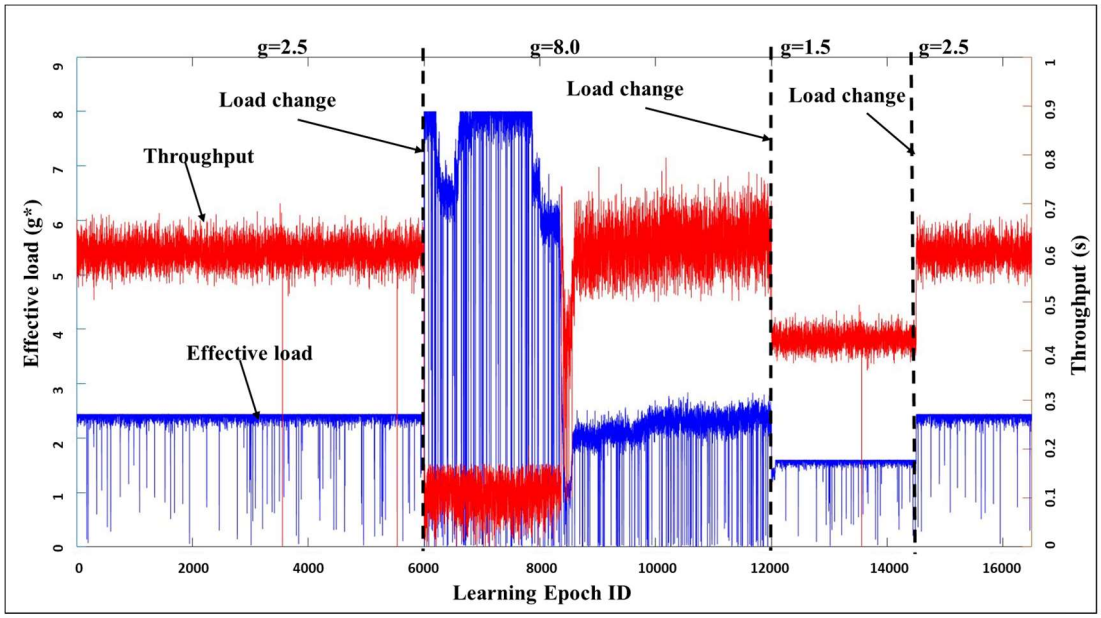

Figure 3.6: Convergence plot for RL-based MAC logic in dynamic load scenario.

As shown in Fig. 3.6, an important feature of the RL-synthesized MAC protocol is that it can adjust to dynamic traffic environments with time varying loads. When the traffic generated in the network changes, the protocol can adjust transmit probability accordingly, so that the optimal throughput is maintained. This is useful in scenarios in which application layer packet generation fluctuates over time. To summarize, the results in this section for a single-node scenario demonstrates the ability of a reinforcement learning-based MAC logic to attain the theoretically maximum throughput and to maintain that for higher application layer loads by controlling self-collisions. Moreover, it can adjust to time-varying loading conditions.

3.6 Multi-nodes Fully Connected networks

3.6.1 Performance in a Two-Node Network

Unlike for the single node implementation in Section 3.5 which uses the classical RL logic, we implemented the Decentralized Multi-agent Reinforcement Learning MAC (DMRL-MAC) for multi-nodes networks. This implementation is based on Hysteretic Q-learning as described in Section 3.4.

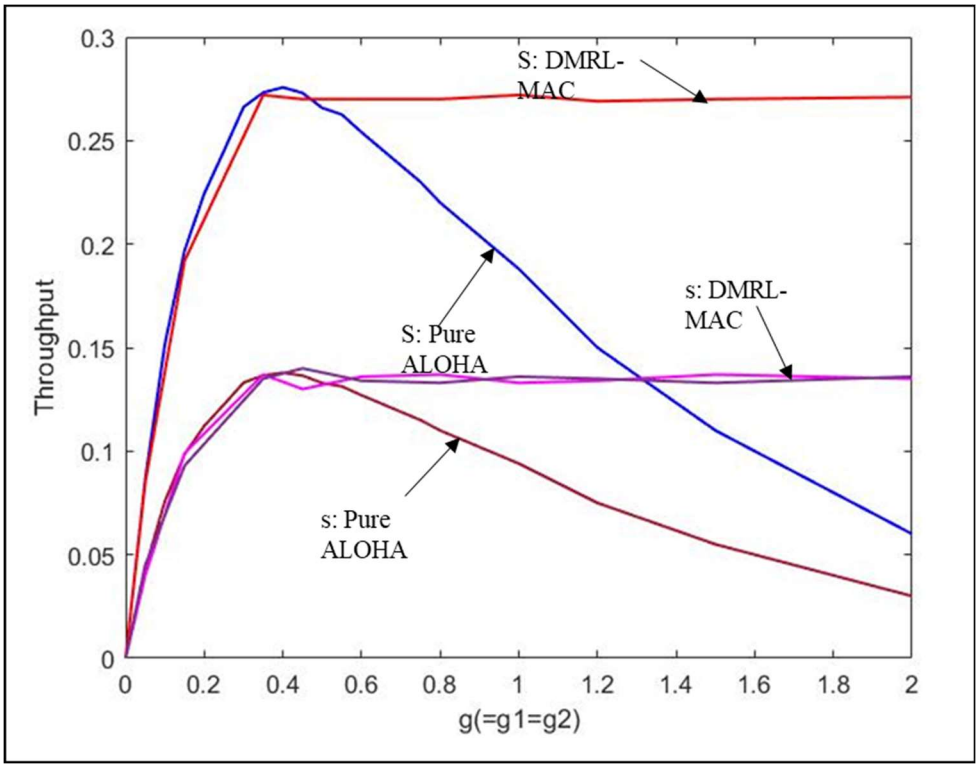

Figure 3.7: DMRL-MAC performance in a two-node network with homogeneous load.

Another key augmentation over the single-node case is that the state space in multi-node scenario contains inter-collision probabilities in addition to the probabilities for self-collisions. In other words, DMRL-MAC uses a 2-dimensional discrete state space with 6 and 4 equal discrete levels of self-collision and inter-collision probabilities respectively.

Homogeneous Loading: As shown in Fig. 3.7, for the two-nodes homogeneous loading case, the pure ALOHA protocol can achieve a maximum nodal throughput of $s_1 = s_2 = 0.135$ Erlangs at the optimal loading $g_1 = g_2 = \hat{g} \approx 0.4$. The figure also shows that the DMRL-MAC logic is able to learn to attain that maximum throughput, and then able to sustain it for larger application layer loads (i.e., g). Like in the single-node case, such sustenance is achieved via learning to adjust the effective MAC layer load (i.e., g^*) by prudently discarding packets from the application layer. This keeps both self-collisions and inter-collisions at check for higher throughputs.

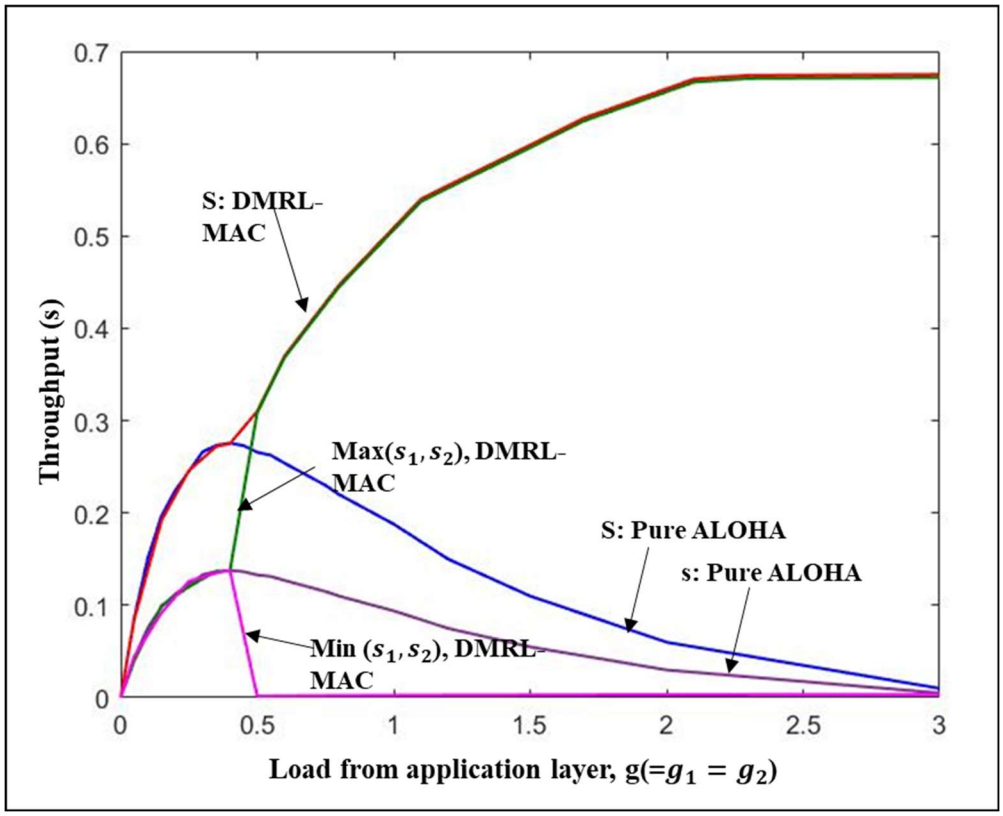

Figure 3.8: Performance of DMRL-MAC in a 2-node network with homogeneous load and a total throughput maximization strategy.

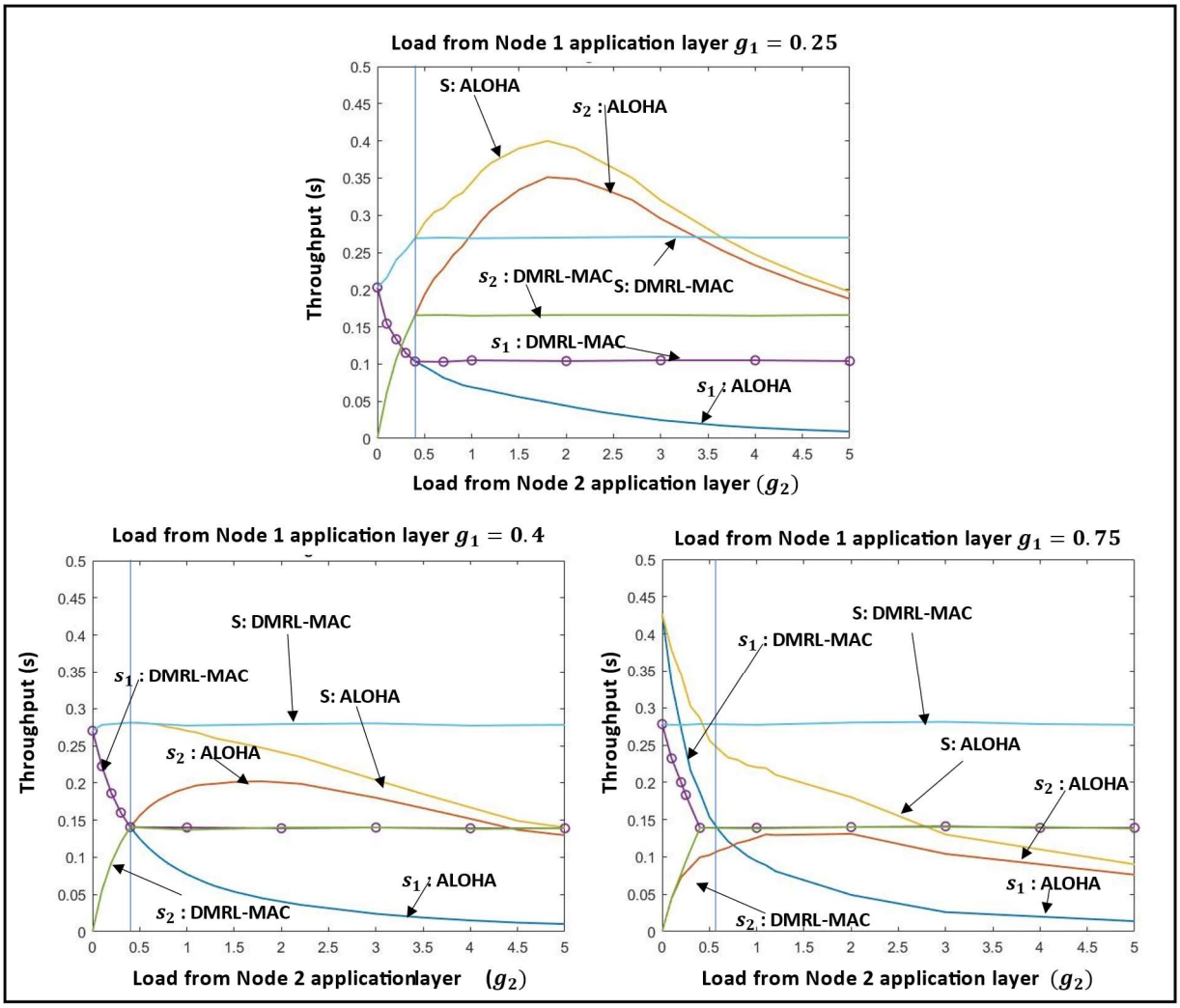

Figure 3.9: Performance of DMRL-MAC in a two-node network with heterogeneous load [s_1, s_2 are individual throughputs of nodes 1 and 2 respectively; S is the networkwide throughput; g_1, g_2 are load (in Erlangs) of nodes 1 and 2 respectively].

We investigated the performance of DMRL-MAC in a 2-node network with the objective of maximizing networkwide throughput, which is different from maximizing individual throughput in Fig. 3.7. This was achieved by setting $\mu_i = \sigma = 0$ in Eqn. (3.4). As shown in Fig. 3.8, for traffic $g_1 = g_2 < \hat{g}$, the individual node throughputs with DMRL-MAC mimic those of pure-ALOHA. With higher load, however, in the case of DMRL-MAC, one of the node's throughput goes to zero so that the other node is able to utilize the entire available throughput, which is the one-node throughput as shown in Fig. 3.8. This way, the network level throughput is maximized

at the expense of the throughput of one of the nodes which is chosen randomly by the underlying distributed reinforcement learning.

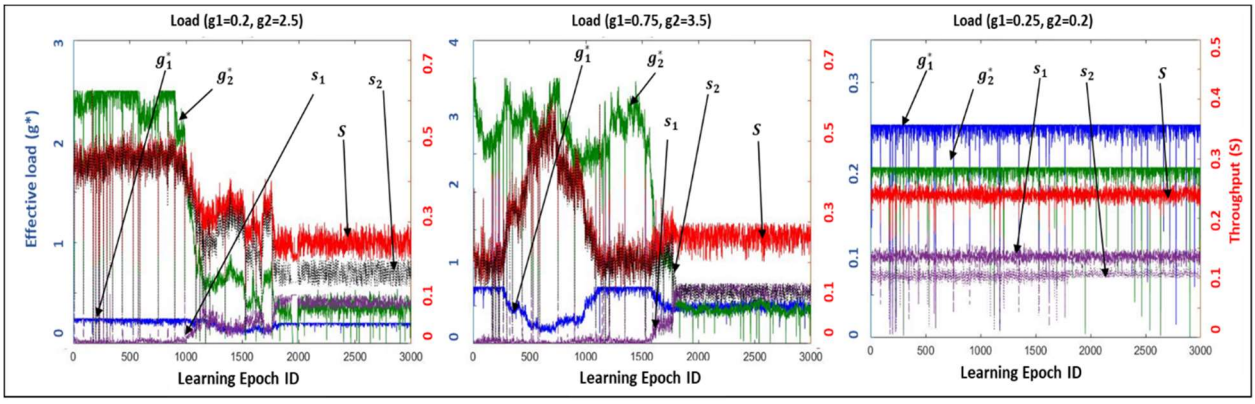

Figure 3.10: Convergence plots for DMRL-MAC effective load and throughput for three different initial heterogeneous loads [s_1, s_2 are individual throughputs of node 1 and node 2 respectively; S is the networkwide throughput; g_1, g_2 are loads (in Erlangs) from the application layers of node 1 and node 2 respectively; g_1^*, g_2^* are effective loads (in Erlangs) from the application layers of node 1 and node 2 respectively].

Heterogeneous Loading: Results in this section correspond to when the application layer data rates from different nodes are different. Fig. 3.9 shows the performance for three scenarios, namely, $g_1 < \hat{g}$, $g_2 = \hat{g}$, and $g_1 > \hat{g}$, where $\hat{g}$ is the effective load from the application layer, for which the optimal throughput is obtained for pure ALOHA. For a two-nodes network, the value of $\hat{g}$ is found out to be ≈ 0.4 Erlangs. Node 2's application layer load g_2 is varied from 0 to 5 erlangs. The behavior of the system can be categorized into three broad cases. Case-I: when $g_1 \leq \hat{g}\ and\ g_2 \leq \hat{g}$, DMRL-MAC mimics the performance of regular ALOHA. Case-II: when $g_1 \leq \hat{g},\ g_2 > \hat{g}$ or $g_1 > \hat{g},\ g_2 \leq \hat{g}$, the node with the higher load adjusts accordingly such that the optimal ALOHA throughput is maintained. Case III: when $g_1 > \hat{g}\ and\ g_2 > \hat{g}$, wireless bandwidth is fairly distributed, and both the nodes transmit such that the effective load boils down to $\hat{g}$. Thus, unlike the regular ALOHA protocol, the DMRL-MAC can learn to maintain

the optimal ALOHA throughput for higher traffic scenarios. It does so via learning to reduce both self- and inter-collisions by discarding packets from the application layer.

The convergence plots for the two-nodes network are shown in Fig. 3.10. It is observed that using the DMRL-MAC protocol, the nodes learn to adjust the transmit probability so that the optimal throughput is maintained, and bandwidth is fairly distributed even at higher loads with heterogeneous loads.

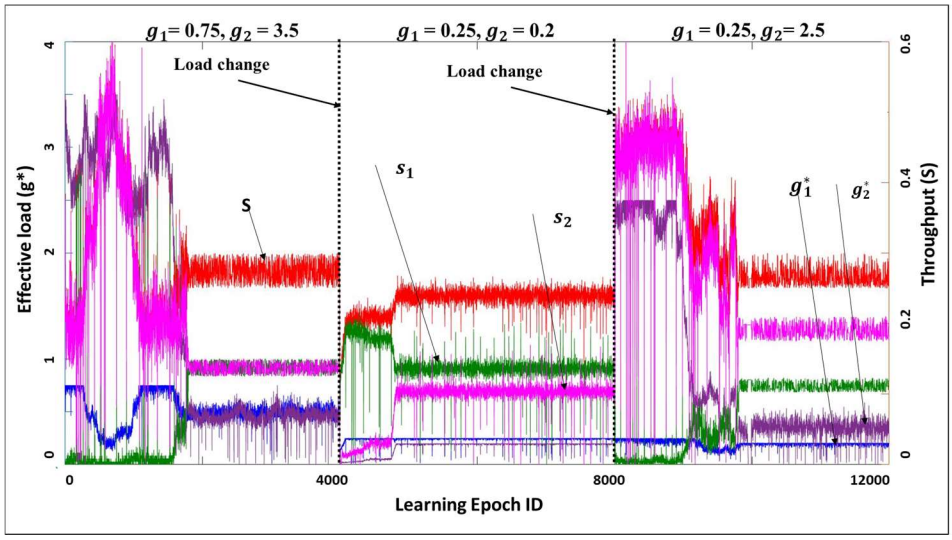

Figure 3.11: Convergence plot for DMRL-MAC for dynamic load.

As can be seen from Fig. 3.11, DMRL-MAC can learn to adapt as a response to changing network load over time. It can achieve and maintain the known optimal throughputs and fairly distribute the available bandwidth under heterogeneous loading conditions.

Prioritized Access: One notable feature of the proposed DMRL-MAC is that node-specific access priorities can be achieved by assigning specific values of the coefficients μ_i in Eqn. (3.4). In Fig. 3.12, the load-throughput plots are shown for two different values of μ_i: $\mu_1 = \mu_2 = 0$ and $\mu_1 = 2.0, \mu_2 = 0.1$. For $g_1 \leq \hat{g}$ and $g_2 \leq \hat{g}$, DMRL-MAC mimics the performance of Pure ALOHA for any values of μ_i. If $\mu_1 = \mu_2 = 0$, the system performs as ALOHA, that is, the individual

throughput for each node is equal to the ALOHA maximum. With increase in μ_1, node 1 gets the priority and the individual throughput for node 2 approaches towards zero. This kind of prioritized access is useful when data from specific sensors are more critical compared to others, especially when the available wireless bandwidth is not sufficient.

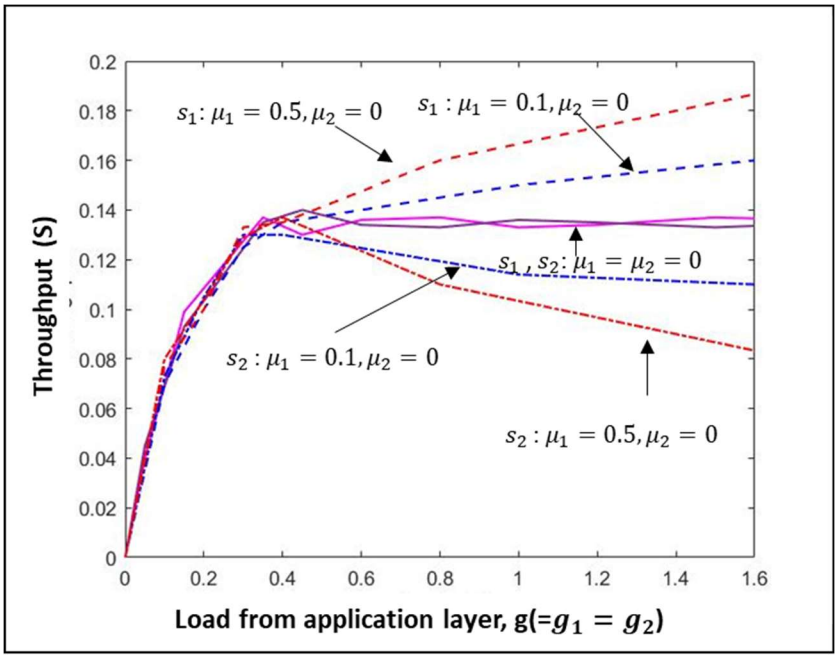

Figure 3.12: Load vs throughput plots for different values of μ_i (priority between the nodes) for a two-node network.

3.6.2 Performance in Larger Networks

Performance of DMRL-MAC for 3-nodes network is shown in Fig. 3.13. As shown for the simulated ALOHA performance, the maximum network wide throughput for a homogeneous load distribution occurs when $g_1 = g_2 = g_3 = \hat{g} \approx 0.25$ Erlangs. That throughput is $S = 0.26$, and that is with a fair distribution among the nodes. It can be observed that like the 1-node and 2-nodes scenarios, DMRL-MAC can learn the theoretically feasible maximum throughput and maintain that at higher loads by reducing both self- and inter-collisions.

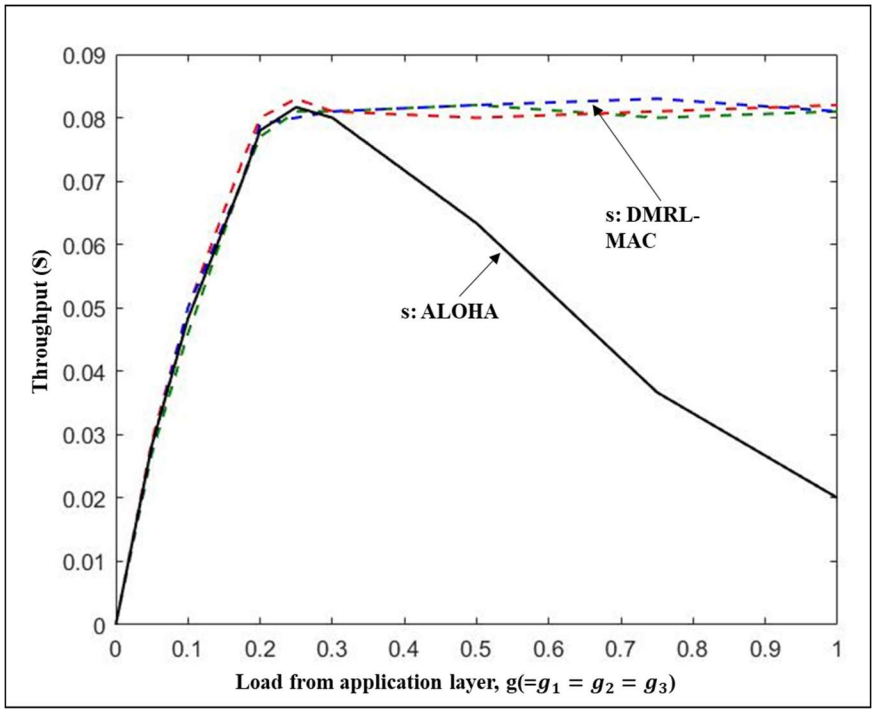

Figure 3.13: Performance of DMRL-MAC in a three-node network with homogeneous load.

For large networks with 2 or higher node-count, the learning hyperparameters in Eqn. (3.4) are made empirically dependent on the network size *N* as follows. We set $\sigma = \frac{1}{N-1}$ because with larger *N*, both the number of contributing terms in the expression of f_i in Eqn. (3.4) and the value of f_i itself go up. This effect is compensated by making σ (the coefficient of f_i) inversely proportional to the number of one-hop neighbors (which is $N-1$ for a fully connected network). After setting the value of σ, the parameter ρ in Eqn. (3.4) is determined empirically. It is observed that for a given σ, a range of values of ρ can be obtained for which the system converges. That range decreases with larger *N*. The relationship was experimentally found as: $\rho = 0.33 - 0.05 \times N$. Using this empirical relationship, the reward expression from Eqn. 3.4 can be rewritten as:

$$R_i = \{(0.33 - 0.05 \times \mathrm{N}) \times S + \sum_{\forall i} \mu_i \times s_i + \frac{f_i}{N-1}\} \tag{3.6}$$

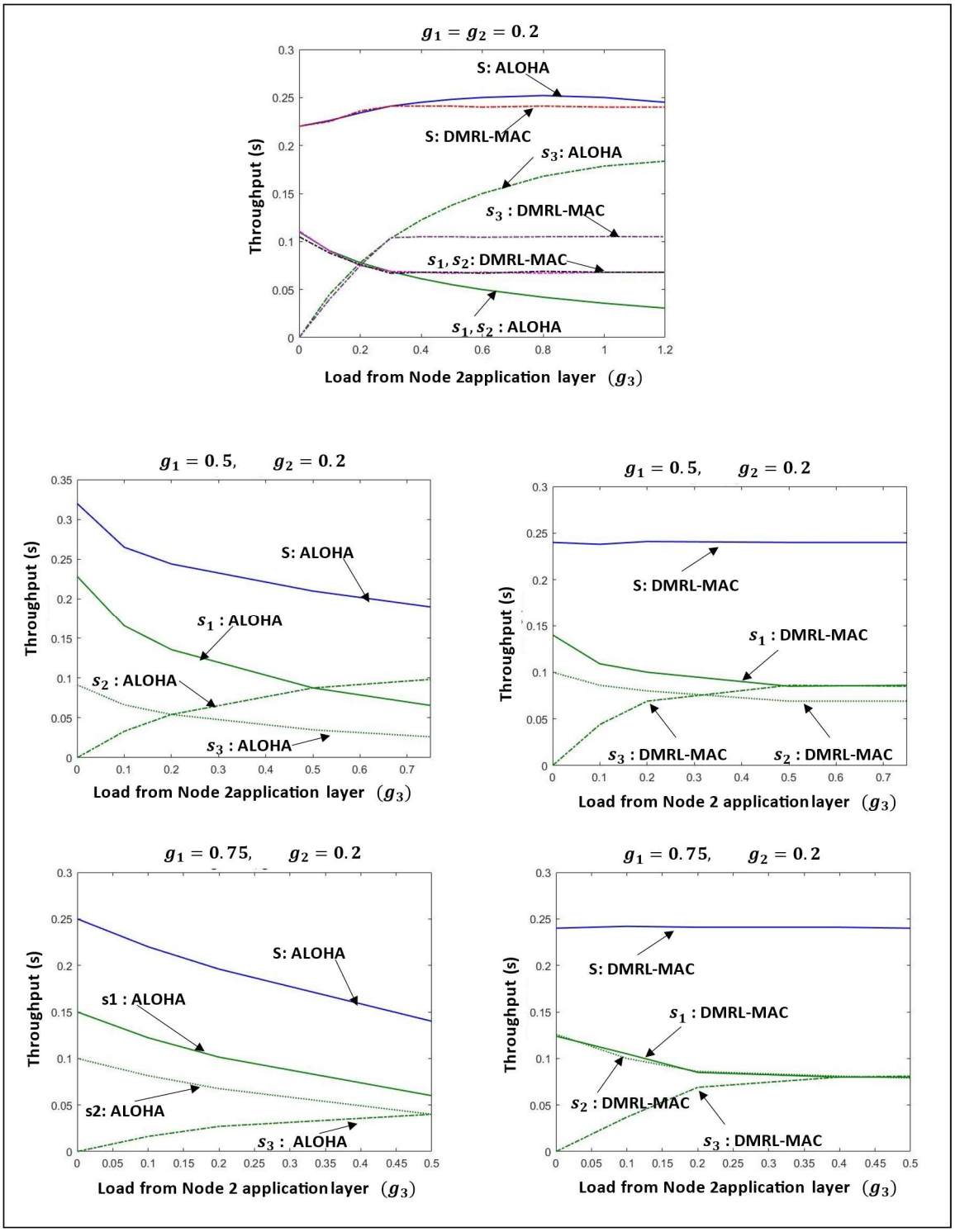

Figure 3.14: Performance of DMRL-MAC in a three-node network with heterogeneous load [s_1, s_2, s_3 are individual throughputs of node 1, node 2 and node 3 respectively; S is the networkwide throughput; g_1, g_2, g_3 are load (in Erlangs) from the application layers of node 1, node 2 and node 3 respectively].

Performance under heterogeneous loads is analyzed and reported in Fig. 3.14. Three different situations are studied, namely, $g_1 \leq \hat{g}, g_2 \leq \hat{g}$, $g_1 \leq \hat{g}, g_2 > \hat{g}$ *or* $g_1 > \hat{g}, g_2 \leq \hat{g}$, and $g_1 > \hat{g}, g_2 > \hat{g}$. In all these three cases, the throughput variation is studied by varying g_3. It can be seen that for $g_1 \leq \hat{g}, g_2 \leq \hat{g}$, *and* $g_3 \leq \hat{g}$, DMRL-MAC mimics the performance of regular ALOHA protocol. When the load in any of these nodes goes higher than the optimal value ($\hat{g}$), learning enables the node to adjust transmit probability so that the optimal ALOHA throughput is maintained by limiting both types of collisions.

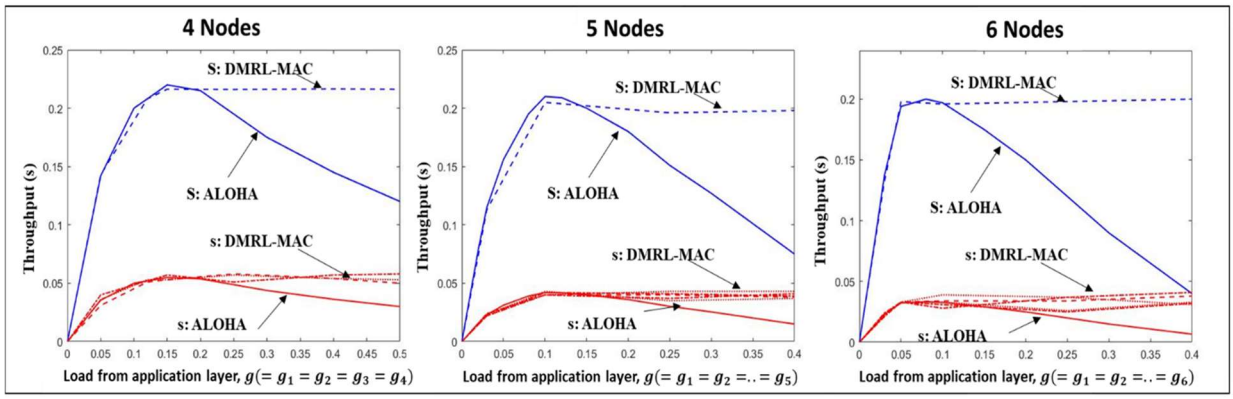

Figure 3.15: Performance of DMRL-MAC for homogeneous load in different network sizes.

Fig. 3.15 depicts the performance of larger networks with 4, 5, and 6 nodes. It shows that the desirable properties of DMRL-MAC in attaining the maximum ALOHA throughput and maintaining it at higher loads are still valid for such larger networks.

However, as shown in Fig. 3.16, the convergence becomes increasingly slower as the networks grow in size. The convergence time distributions in fact show that the learning almost stops working for networks with 7 or more nodes. This issue of scalability is handled by modifying the RL model, which is explained in the next chapter.

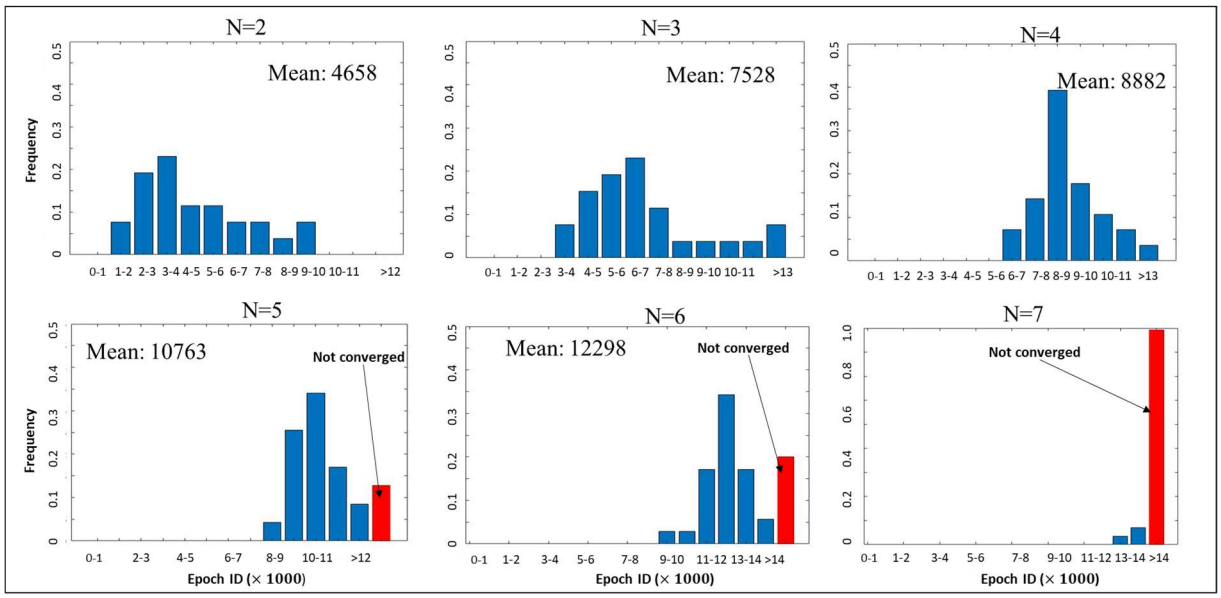

Figure 3.16: Convergence behavior for different number of nodes (pmf).

3.7 Protocol Synthesis for Partially Connected Topologies

The primary goal of this chapter is to report the key concepts of RL based wireless medium access and its performance in fully connected networks. Implementation of DMRL-MAC for a partially connected network is different from the fully connected case in the following ways. A node in this case have no throughput information about other nodes that are 2-hops and beyond and have only partial information about the throughput of 1-hop neighbors. A node can monitor the number of transmissions from its 1-hop neighbors that are overlapping and non-overlapping with its own transmissions. In the absence of any network-wide information, a node running DMRL-MAC treats: i) its immediate neighborhood (i.e., 1-hop) as the complete network, and ii) the estimated throughput of its 1-hop neighbors computed from monitored non-overlapping transmissions as their approximated actual throughputs.

Thus, the current framework of RL-based MAC protocol synthesis needs to be modified to make it generalized for mesh networks. More protocol refinements and experiments will be needed to

address learning in the absence of complete throughput information in the neighborhood. A more detailed study on this are presented in the next chapters.

3.8 Summary

In this chapter, the concept of protocol synthesis using multi-agent Reinforcement Learning is introduced. With the goal of making it more generalized and scalable, the framework is demonstrated here for fully connected networks with sparse connectivity. The learning-synthesized random access protocol (DMRL-MAC) allows the nodes to control the transmit probability so that the inter-collisions and self-collisions are reduced. As a result, the nodes can attain the benchmark ALOHA throughput and sustain it even for a higher traffic in the network. An important feature of this protocol is that it can deal with heterogeneity in the network, that may arise from three aspects: traffic conditions, performance requirement and topology. DMRL-MAC also allows the user to assign node specific priorities in the network based on QoS needs. Moreover, learning allows the nodes to self-adjust in a dynamic network environment with a time-varying traffic.

The immediate next step here would be to build on the current study and implement the learning-enabled protocol in a network with arbitrary topology. This is explored in the next chapter, which uses the protocol synthesis concept for developing a random-access protocol scalable for any topology with any network size and degree.

Chapter 4: MAC Protocol Synthesis for Mesh Networks using RL with Localized Information Sharing

In the previous chapter, we introduced the concept of network protocol synthesis using Reinforcement Learning. The idea of RL-enabled protocol synthesis was developed for fully connected networks with random access MAC arrangements. However, the proposed approach lacks generalizability in the sense that it relies on global network information which restricts its application for partially connected networks. In addition, the RL-synthesized DMRL protocol in chapter 3 suffers from scalability issues. That is, the performance of the MAC protocol degrades with increase in network size and degree.

In order to overcome these limitations and to make the framework generalized, we extend the concept of network protocol synthesis for arbitrary mesh networks. Building on the concept introduced in the previous chapter, we develop a decentralized learning approach that gives the desired performance even for partially connected topologies and that scales with network size and degree. The core objective here is still to make the nodes learn policies that can maximize network performance in terms of throughput and fairness. However, the RL-based protocol synthesis approach proposed in this chapter can work with local network information availability in partially connected mesh topologies and also in networks with large size and degree.

In this chapter, a decentralized, multi-Agent RL framework with localized network information is used as the basis for protocol synthesis. Decentralized behavior makes the nodes independently learn optimal transmission strategies without having to rely on full network level information and direct knowledge of the other nodes' behavior. The nodes learn to minimize packet collisions such that optimal throughput can be attained and maintained for loading conditions that are higher than what the known benchmark protocols (such as ALOHA) for IoT devices without complex

transceivers, while ensuring a fair share of wireless bandwidth even in heterogeneous traffic and topological conditions.

4.1 Motivation

Protocol design in wireless networks, as explained in chapter 1, is mostly driven by heuristics and past experience of network designers. A notable drawback of this design approach based on available standardized protocols is that the resulting logic often underperforms in application-specific non-standard scenarios caused by topology- and load-heterogeneity, and other factors. For example, with baseline ALOHA MAC, the throughput of a network starts falling at higher loads due to collisions. The situation is compounded in the presence of heterogeneity. In a network with arbitrary mesh topology, some nodes may be in more disadvantageous position as compared to others in terms of the collisions they experience. As a result, those nodes receive less share of the available wireless bandwidth. The root cause of these limitations is the fact that the nodes in the network are statically programmed with standardized protocol logic, and they lack the flexibility that can result in abilities to learn optimal behavior in specific scenarios.

In chapter 3, we introduced the concept of RL-enabled protocol synthesis for random-access MAC that can partially address these challenges for transceivers used in low-cost IoT devices and wireless sensor nodes. With the long-term objective of making the learning approach generalized, we started with an initial scenario of a network with its nodes running the simplest MAC logic without relying on complex and energy-expensive lower-level capabilities such as carrier sensing.

The key idea behind Reinforcement Learning (RL) for protocol design [24] is to formulate the protocol layer logic in each network node as a Markov Decision Process (MDP) and use RL as a

temporal difference method to solve the MDP. The solution of this MDP is a set of transmission actions taken by individual nodes, thus resulting in a network protocol. However, in decentralized RL, a key requirement is that an agent (i.e., a network node) needs to have as much network-level information as possible in order to avoid collisions and share wireless bandwidth in a fair manner. This becomes a problem in a partially connected network (as observed for the mechanism developed in Chapter 3) in which the information availability to a node is usually limited only to its immediate neighborhood.

An obvious solution to this problem is to employ centralized learning in which a learning hub maintains and updates network-level information, and learns the optimal transmission policies (i.e., protocol) for all network nodes. Apart from the fact that centralized learning is computationally expensive, such an approach requires individual nodes to frequently upload network information and download transmission policies to and from the centralized learning entity. Hence, this is not suitable for use in sensor and IoT nodes with limited networking resources and energy budgets. In this chapter, we develop an alternative decentralized approach that can work with partial/local network information availability in partially connected networks. Each network node acts as a learning agent, and they learn transmission policies using localized network information in a completely decentralized manner. The strategy is particularly scalable for topologies with partial connectivity where nodes do not have complete network level information. The approach is built on the model developed in chapter 3, which assumed full network-level information availability for fully connected and sparse networks. In this chapter, the decentralized learning mechanism is developed such that the RL agents (i.e., network nodes) rely only on the localized network information available in their immediate neighborhoods. It is shown that a decentralized learning model is feasible for training MAC layer logic to attain fair

and near-benchmark performance in the presence of loading and topological heterogeneities. It is also shown how the learning paradigm is scalable for large networks that give rise to additional levels of heterogeneities.

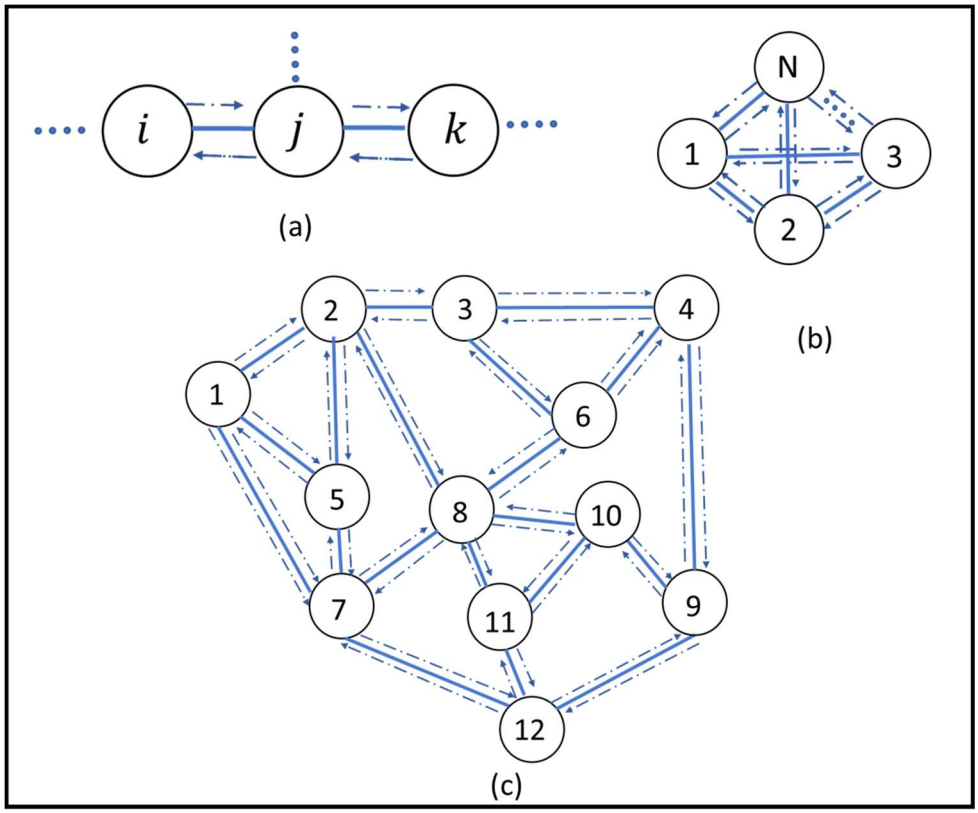

Figure 4.1: Network and data sharing model for (a) generalized, (b) fully connected and (c) partially connected mesh topology.

4.2 Network and Traffic Model

The proposed learning framework is developed and validated for general purpose multi-point to multi-point networks with arbitrary mesh topologies. In line with our initial objective of applying learning for low-complexity IoT/Sensor networks, nodes in that mesh topology are assumed to be without carrier sensing abilities. For all the networks in Fig. 4.1, the solid lines represent physical node connectivity, and the data flow is represented by the dashed lines. For the linear topology in Fig. 4.1 (a), node $'j'$ transmits packets for which nodes $'i'$ and $'k'$ are intended

receivers. Similarly, nodes $'i'$ and $'k'$ send packets destined to node $'j'$. The MAC layer traffic model is such that application layer packets generated in node $'j'$ is broadcasted to all its directly connected one-hop neighbor nodes. The traffic generation model in node $'j'$ is modelled as a Poisson process with mean g_j Erlang. It should be noted that in the partially connected topologies in Figs. 4.1 (a) and 4.1(c), individual nodes possess only topologically local network state information. For fully connected topologies (special case) such as the one in Fig. 4.1 (b), however, each network node possesses state information about the entire network. As will be shown later, the availability of network information is a strong determinant for the design and performance of the proposed learning framework.

A specific goal of the proposed paradigm is to enable protocol learning in the presence of topological heterogeneity in arbitrary mesh topologies such as the one shown in Fig. 4.1 (c). For example, node-8 in that topology suffers from more packet collisions as compared to the other network nodes, thus naturally leading to a smaller share of allocated bandwidth. One objective of the proposed learning is to implement fairness in that node-8 and other such topologically disadvantaged nodes should be able to receive a fair share of available bandwidth in spite of their topologically disadvantageous positions.

The network model incorporates a control information piggybacking mechanism for sharing relevant network state information within the two-hop neighborhoods of each individual node. Such state information is used by the decentralized RL agents within each node. Note that piggybacking over data packets allows information sharing without having to rely on higher layer protocols beyond two-hops.

4.3 Decentralized MAC Learning with Localized Information

The key concept is to model MAC layer logic as a Multi-Agent Markov Decision Process (MAMDP), and then use Hysteretic Q-Learning as a temporal difference solution method to find optimal transmission policies as the target MAC protocol. The resulting protocol synthesized using distributed reinforcement learning and localized network information is termed as Decentralized MAC Learning with Localized Information (DMLLI).

4.3.1 Learning using Two-hop Neighborhood Information

In a shared cooperative learning environment, it is desirable for each agent to have access to as much information as possible. However, in the presence of partially connected topologies and unreliable links, the available network information is usually limited. A distinctive feature of the proposed DMLLI is that the learning of individual agents (i.e., nodes) do not rely on the global network information. Instead, the reinforcement learning reward structure is designed such that only up to two-hop localized network information is used. Such information from two-hop neighborhood is gathered using an in-band control mechanism [77], [78], where the throughput information is shared by piggybacking it within the MAC layer Protocol Data Units (PDUs).

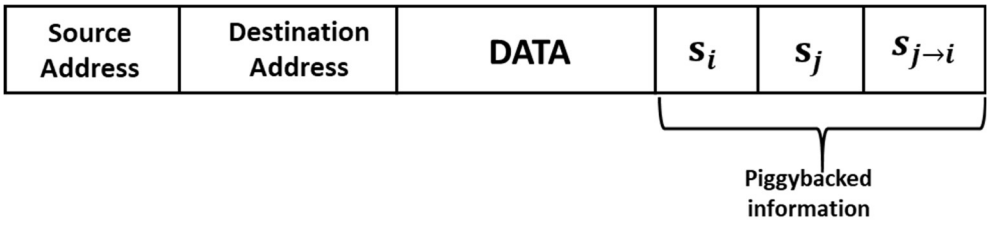

Figure 4.2: MAC layer PDU from node i.

The rationale behind using up to two-hop information is that transmissions from nodes that are up to two hops apart can result in collisions at a receiver node. Therefore, the throughput of a node can potentially be affected by all nodes that are within its two-hop neighborhood. Following is a description of the network information sharing model used in this paradigm.

Consider a scenario where s_i is the current throughput of node i and $s_{i\to j}$ is the part of throughput of node i for which the intended receiver is a one-hop neighbor node j. Node j computes $s_{i\to j}$ by keeping track of all the non-overlapping transmissions from i to j. Node j then periodically piggybacks $s_{i\to j}$ information within its outgoing MAC layer PDUs. Node i then computes its own throughput $s_i = \sum_{\forall i} s_{i\to j}$, where j represents all its 1-hop neighbors. Node i then periodically piggybacks its own throughput s_i and its one-hop neighbors' throughput s_j in its outgoing MAC layer PDUs. A typical MAC layer PDU from node-i is shown in Fig. 4.2.

The procedure above is periodically executed across all the two-hop neighbor nodes. In this way, the throughput information of a node is shared within two-hop neighborhoods. This localized information will be used to compute the RL reward functions in the learning framework.

It is to be noted that the concept of localized network information visibility for an agent (i.e., node) is different from the traditional Partially Observable Markov Decision Process (POMDP) [79], [80]. In POMDPs, the environment states are not completely visible to the agents. On the contrary, in this work, an agent has full access to the knowledge of its own state defined by its one-hop congestion status, and up to two hop information that affects collisions caused by its own transmissions. It is that the agent does not have access to the environment status of other agents that are part of the MAMDP process.

4.3.2 Reinforcement Learning Components

The RL environment is represented by the wireless network itself, and each node runs an independent learning agent. The working model for a 2-node network is depicted by the schematic in Fig 4.3. The incoming application layer load for nodes N_1 and N_2 are g_1 and g_2 (i.e., in Erlang) respectively. The packet arrival is a random process with Poisson distribution

with mean $g_i, i = 1,2$. Each learning epoch consists of the transmission of N_T packets with constant packet size. In each epoch, the learning-based MAC logic learns to set the packet transmission probability so as to reduce packet collisions. Such learning happens at an epoch level temporal granularity by iterating the transmission probabilities on an epoch-by-epoch basis.

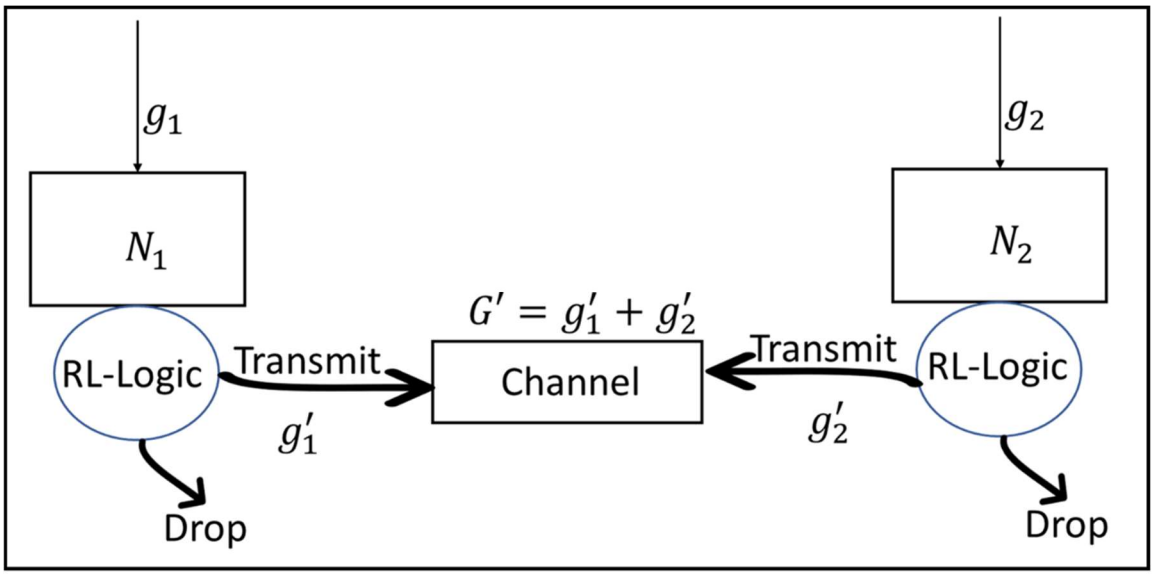

Figure 4.3: Schematic diagram for the working model of a 2-node network.

Action Space: In this RL framework, the actions taken by the MAC layer agent in a node is the transmit probability of a packet arrived from the application layer. To make the action space discrete, the transmit probabilities are discretized into a fixed number of possible actions. The action space for a node i is defined as: $\mathbf{A^i} = \{a_1^i, a_2^i, \ldots., a_{|\mathcal{A}|}^i\}$, where a_k^i = Prob [an application layer packet for node i is transmitted]= $P_{tx}^i = \frac{k}{|\mathcal{A}|}$. As a result of choosing such an action, the application layer load g_i for node i is modulated to get the effective load g_i^*, such that $g_i^*(a_k^i) = \frac{k}{|\mathcal{A}|} \times g_i$. This effective load g_i^* is the actual load with which packets are inserted into the wireless channel.

State Space: Environment state is defined by the local network congestion level as seen by a node

(i.e., the agent). The state at a learning epoch t is defined as the probability of collisions experienced by the node during that epoch. Since the learning is decentralized, each node maintains its own notion of state. Like actions, to keep the state space discrete, the states are discretized into fixed number of uniform discrete levels in the range [0, 1]. The notion of state of a node i at an epoch t is denoted as $\hat{s}^i(t) \in \hat{S}^i = \{\hat{s}_1^i, \hat{s}_2^i, \dots\dots, \hat{s}_{|\hat{S}^i|}^i\}$, where $\hat{S}^i$ is the state space for node i. Each state $\hat{s}_k^i$ represents a level of collision in the network as perceived by node i. The collision level is quantified using probability of packet collisions of a node $i\ as\ P_c^i$, which is defined as:

$$P_C^i = \frac{\text{Number of packet collisions for node } i}{\text{Number of transmitted packets by node } i} = \frac{N_c^i}{N_{tx}^i},$$

where Number of transmitted packets for node i $\left(N_{tx}^i\right)$

$=$ Number of application layer packets arrived in the

epoch - Number of dropped packets by node i

$$= N_T^i - N_D^i$$

Therefore, $P_C^i = \frac{N_c^i}{N_T^i - N_D^i}$

The states are mapped to collision probabilities as follows. For node i, at epoch $'t'$, the perceived state is $\hat{s}_k^i(t), if\ \{0.2(k-1) < P_C^i(t) \leq 0.2k\}$, where $P_C^i(t) = \frac{N_c^i(t)}{N_T^i(t) - N_D^i(t)}$.

On simplifying further, we observe that $N_D^i(t) = P_{tx}^i(t) \times N_T^i(t)$

Hence, $P_C^i(t) = \frac{N_c^i(t)}{\left(1 - P_{tx}^i(t)\right) \times N_D^i(t)}$

Rewards: The reinforcement learning reward function for node i is designed based on its observables, viz, the information about its own throughput (s_i) and its two-hop neighbors'

throughputs (s_j). Here j represents i's up to two-hop neighbor nodes. The quantities s_i and s_j are expressed in packets per packet duration or Erlangs. Node i computes its local two-hop neighborhood throughput as $S_i = s_i + \sum_{\forall j} s_j$. A temporal gradient-based reward function is used as follows. Since the primary objective is to maximize throughput, a high positive reward is assigned for a positive gradient of the throughput and vice-versa. The temporal gradient of the localized throughput of node i is computed as $\Delta S_i = S_i(t) - S_i(t-1)$. Along with throughput maximization, this mechanism also considers node-level throughput distribution fairness, which is captured by a fairness coefficient computed as:

$$f_i(t) = -\sum_{\forall k \neq i} |(1-\theta_i) \times s_i(t) - (1-\theta_k) \times s_k(t)| \tag{4.1}$$

$$k \epsilon \; onehop \; neighborhood \; of \; i.$$

The quantity θ_i represents priority coefficient of node i. A larger value of f_i represents a fairer system. For completely fair bandwidth distribution, $\theta_i = 0, \forall i$. A discrete time derivative of throughput fairness is computed as $\Delta f_i = f_i(t) - f_i(t-1)$. Using the throughput and fairness gradients, the temporally sensitive reward structure is defined by Eqn. 4.2 as follows.

$$R_i(t) = \begin{cases} +50, & \Delta S_i - \epsilon_s > 0, \Delta f_i - \epsilon_f > 0 \\ -50, & otherwise \end{cases} \tag{4.2}$$

In this equation, the coefficients ϵ_s and ϵ_f are used so that the agents do not get stuck in a near optimal solution. The coefficient ϵ_s is kept fixed at 0.005 throughout the experiments, whereas the coefficient ϵ_f is dependent on the size of the network. In addition to the reward from Eqn. (4.2), a learning recovery protection is provided by assigning a penalty (i.e, -100) in case the throughput s_i goes to zero.

4.4 Experiments and Results

The performance of proposed DMLLI-enabled MAC is evaluated for networks with generalized arbitrary mesh topologies. The performance is then compared with that of a known benchmark MAC (i.e., pure ALOHA [81], [82]) that does not rely on inter-node time-synchronization and hardware-level carrier sensing. The baseline learning hyperparameters for the DMLLI-MAC are tabulated in Table 4.1.

Table 4.1: Baseline experimental parameters

No. of packets per epoch	1000
$\lvert\mathcal{A}\rvert$	20
$\lvert\mathcal{S}\rvert$	5
α	0.9
β	0.1
γ	0.95
$\in$	$1.0 \times e^{-epoch\ ID/-2500}$

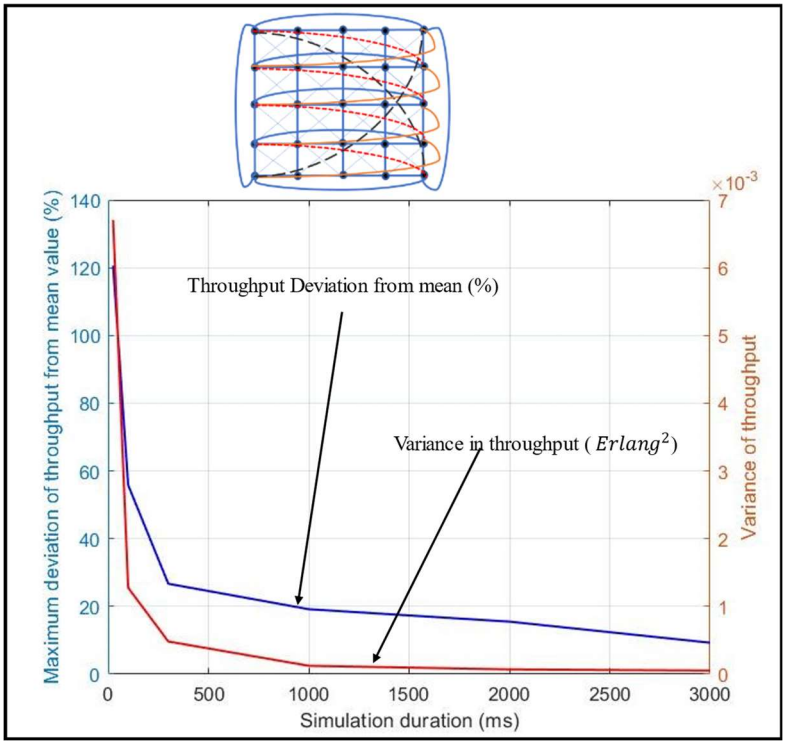

Figure 4.4: Maximum deviation (%) and variance of throughput for varying simulation duration.

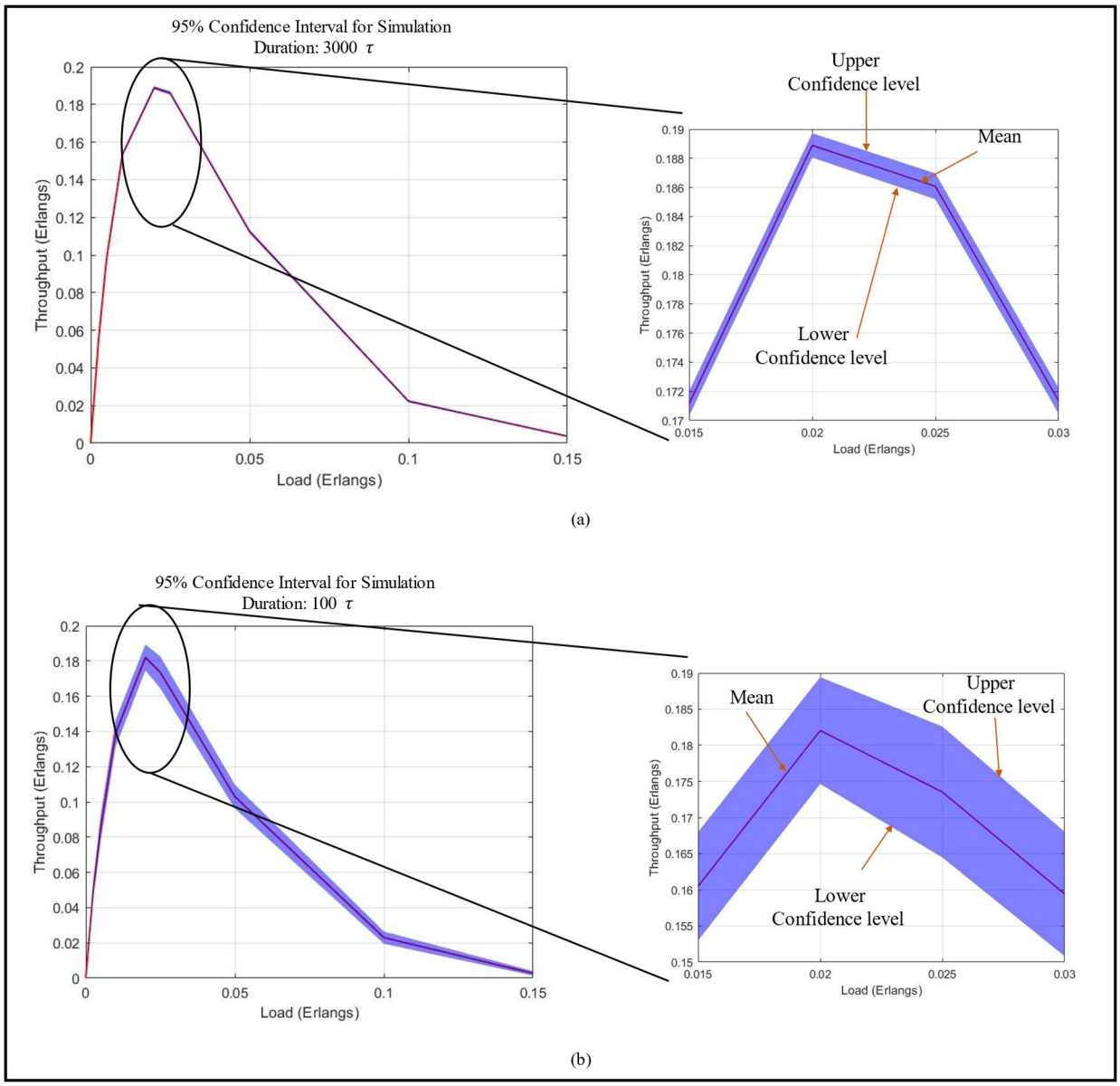

Figure 4.5: 95% confidence interval and mean throughput for (a) 3000 τ simulation duration (b) 100 τ simulation duration.

The simulation durations for the conducted experiments were chosen based on the following statistical analysis. Fig. 4.4 shows the statistical variation of results obtained from different runs of experiments on a large (i.e., 25-nodes) partially connected network. Fig. 4.4 shows the statistical variation of results obtained from different runs of experiments on a large (i.e., 25-nodes) partially connected network. It shows the variance and maximum deviation from mean of the experimental results over 100 runs for varying simulation duration. Variance of the dataset

decreases with increase in simulation duration, and it asymptotically tends to zero for about 3000 τ , where τ is packet duration. Similar trend is observed for the plot of percentage of maximum deviation from mean value and is within 9.27% for a simulation duration of 3000 τ. Figs. 4.5 (a) and (b) demonstrate the 95% confidence intervals and the corresponding mean throughput values for the 25-nodes partially connected network for simulation duration 100 τ and 3000 τ respectively. As expected, the confidence interval narrows down with increase in the simulation duration. Based on the above statistical analysis, 3000 τ was chosen to be the simulation experiment duration for all reported results in the chapter.

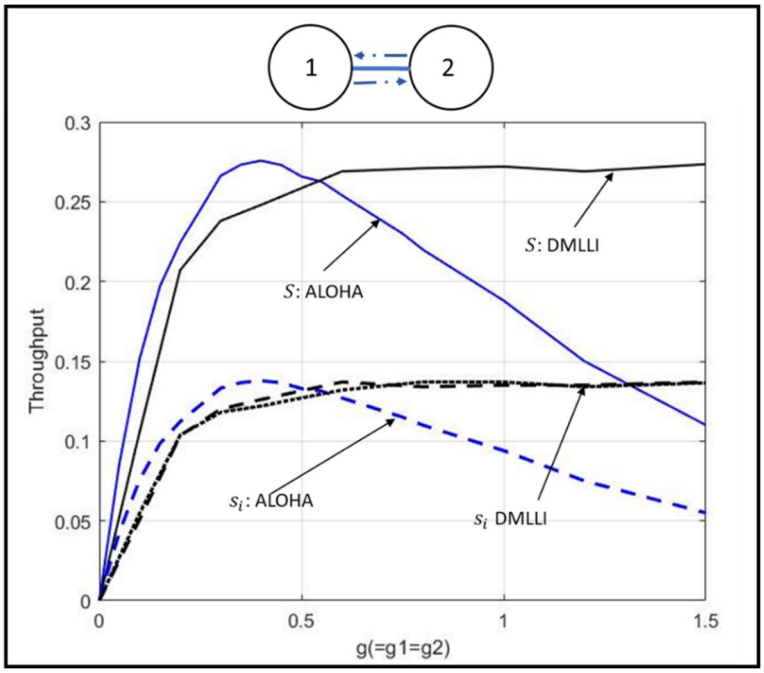

Figure 4.6: DMLLI performance in a two-node network [g_1, g_2 are load from the application layers of nodes 1 and 2 respectively; s_i, S are node-level and networkwide throughput].

4.4.1 Throughput Maximization with Fair Bandwidth Distribution

4.4.1.1 Fully connected Topology

Two-nodes Network: To understand the workings of the learning mechanism and its impacts on network dynamics, we start with a simple 2-node network. For enabling fairness, the priority

coefficients θ_i and θ_k in Eqn. (4.1) are set to zero. Fig. 4.6 depicts DMLLI's performance in comparison to ALOHA for homogeneous loads from both the network nodes. For ALOHA, throughput reaches the maximum for an optimal load, and then it falls. This is because of increased packet collisions at higher loads. Observe that with DMLLI, the learning-enabled MAC, the nodes learn to transmit packets with the optimal transmit probability such that: a) the maximum throughput reaches to the same observed for benchmark ALOHA, and b) packet collisions are kept under check so that the throughout does not go below the achieved maximum even after the load is increased. In other words, the nodes learn how to achieve the benchmark maximum, and unlike the benchmark protocol, it learns how to maintain that maximum even for high traffic loads.

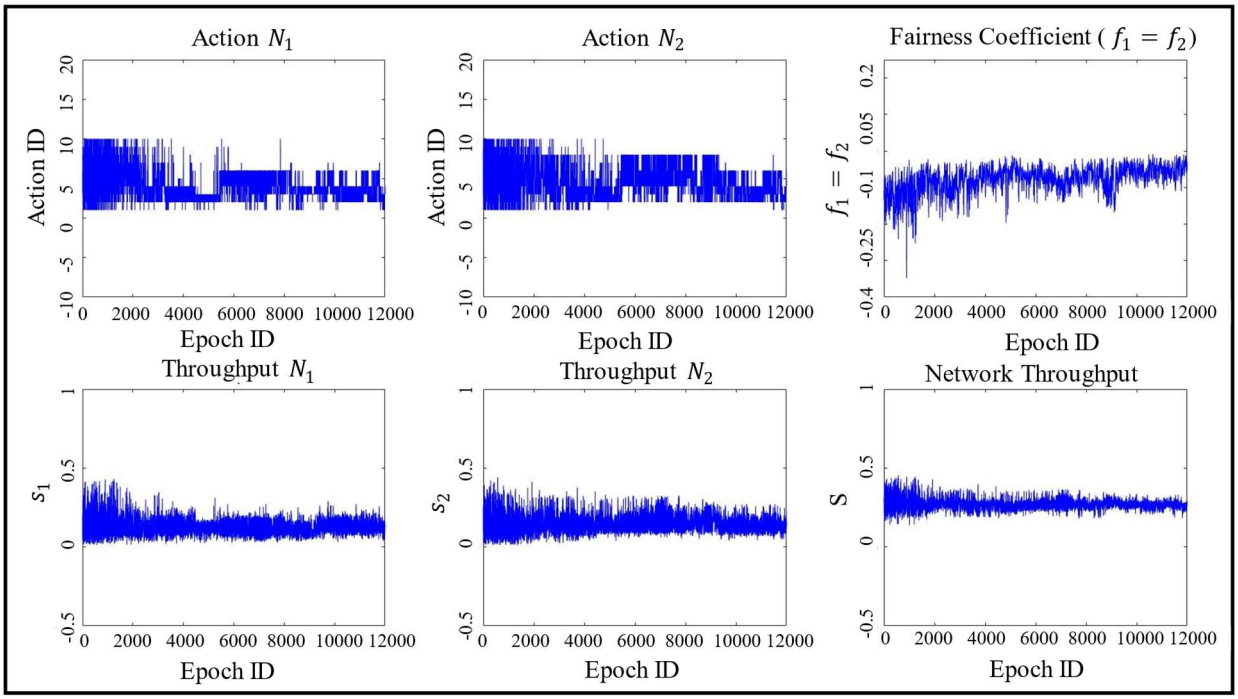

Figure 4.7: Convergence plots for actions, fairness coefficients, individual throughput, and network throughput.

The RL system dynamics in terms of actions, individual node throughput ($s_1 and\ s_2$), fairness coefficients ($f_1 and\ f_2$), and networkwide throughput (S) are presented in Fig. 4.7. The

convergence plots are shown for the case when the incoming application layer load ($g_1 = g_2 = 1.0$) is higher than the optimal value ($\widehat{g_1} = \widehat{g_2} \approx 0.4$), which represents the individual node-level load at which the benchmark ALOHA attains its maximum throughput. It can be observed from Fig. 4.7 that the nodes learn to take actions with action ID = 4 most frequently after convergence. This particular action corresponds to the transmit probability of $P_{tx} = 0.4$. The actions for both the nodes oscillate a bit around the optimal action, because of the stochasticity involved in action selection while learning. As a result of these oscillations from action selection as well as the stochasticity of packet generation, the throughput is slightly less than the ALOHA throughput for scenarios when the incoming application layer traffic load is lower than the optimal value (Fig. 4.6). Hence, the maximum throughput is attained at a higher load for DMLLI as compared to ALOHA.

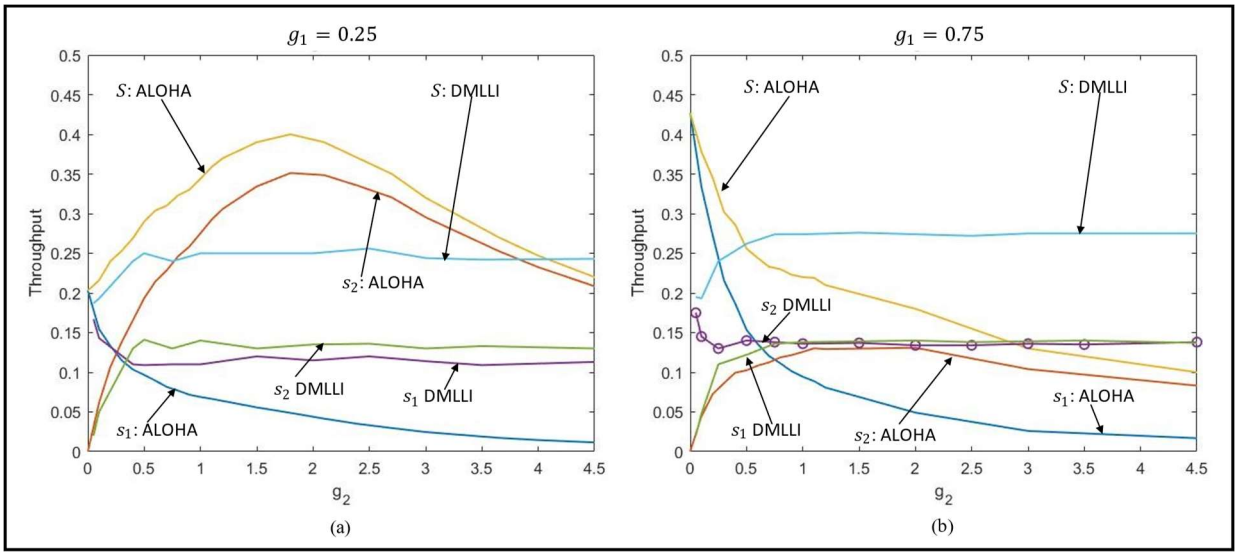

Figure 4.8: Performance of DMLLI in a two-node network with heterogeneous load [s_1, s_2 are individual throughputs of node 1 and node 2 respectively; S is the networkwide throughput; g_1, g_2 are load (in Erlangs) from the application layers of node 1 and node 2 respectively].

Another observation from Fig. 4.7 is that the fairness coefficients for both the nodes approach to zero, indicating that the bandwidth distribution becomes fairer as learning progresses. The

network throughput (S) is also seen to converge to a value of 0.27 Erlang, which is close to the maximum ALOHA-based network throughput of 0.273.

The learning-based framework in DMLLI protocol also allows the nodes to handle heterogeneous traffic as shown in Fig 4.8. Figs. 4.8 (a) shows the performance when node-1's application layer load g_1 is less than the optimal load ($\hat{g}_1$), and Fig. 4.8 (b) is when $g_1$is greater than $\hat{g}_1$. The key observation in both these cases is that in spite of the load heterogeneities, the nodes learn the optimal transmit probabilities such that the bandwidth is fairly distributed. The throughput of node 1 goes down with an increase in the load in node 2 in case of ALOHA. However, in DMLLI, the deviation in throughput among the nodes is significantly reduced. Moreover, like the case with homogeneous load distribution, the throughput of both nodes (and hence the network-wide throughput) with heterogeneous load is maintained for higher traffic as well.

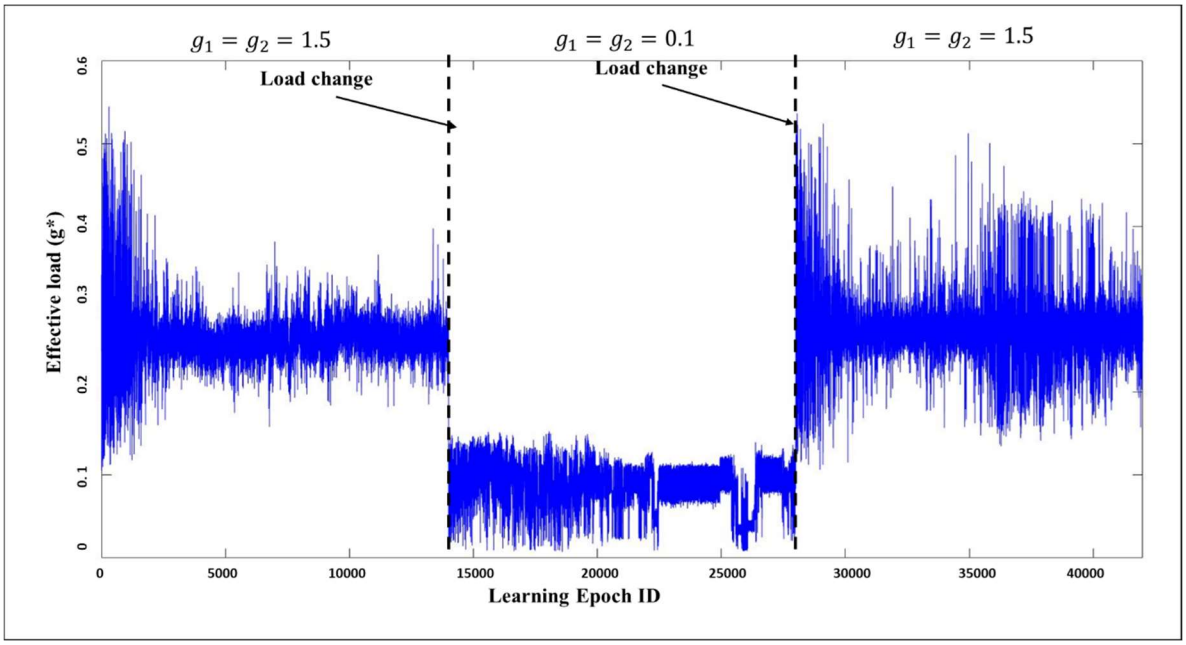

Figure 4.9: Performance of DMLLI in a dynamic environment.

The online learning abilities of the underlying RL framework allows the nodes to handle time-varying traffic in DMLLI as shown in Fig 4.9. Whenever the application layer load changes, the nodes learn to adjust to the new network traffic conditions and maintain the optimal throughput.

Initially, the application layer load in the network is $g_1 = g_2 = 1.0$, and the network throughput (S) converges to the optimal value of 0.27. There are changes in the incoming application layer load indicated by the dotted vertical lines, where the load changes to $g_1 = g_2 = 0.1$ and $g_1 = g_2 = 1.0$ respectively. It is observed that the RL framework allows the nodes to maintain the optimal throughputs even in these dynamic loading situations.

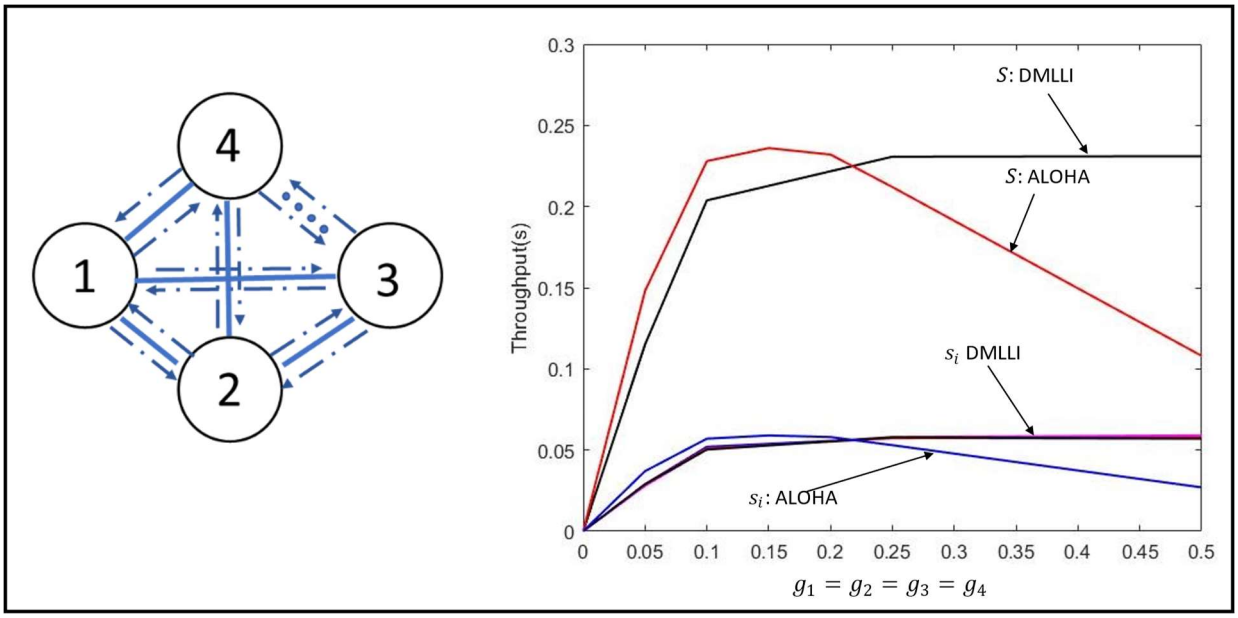

Figure 4.10: Load-Throughput for 4-node fully connected network; s_i, g_i are the throughputs and loads respectively for node i, S is the network throughput.

Large Networks: The load-throughput plots for homogeneous load distribution in a 4-node fully connected network is shown in Fig. 4.10. As in the 2-node case, the nodes here learn to attain the maximum possible (i.e., with ALOHA) throughput and to maintain it for higher loading conditions. Moreover, the bandwidth is distributed fairly among all the nodes.

We have analyzed the performance for larger networks with 12, 16 and 25 nodes in Figs. 4.11 (a), (b) and (c) respectively. These figures show that the desirable properties of DMLLI in attaining the maximum throughput and holding it for higher loads while maintaining fair bandwidth distribution continue to be valid for such large networks. The maximum throughput

of DMLLI is within a range of $[0, 1.0]\%$ of the maximum throughput attained by ALOHA in all these networks. The coefficient of variation among the nodes' throughput is measured to within the range of $[0.18, 0.23]$. This verifies the scalability of the protocol with network size.

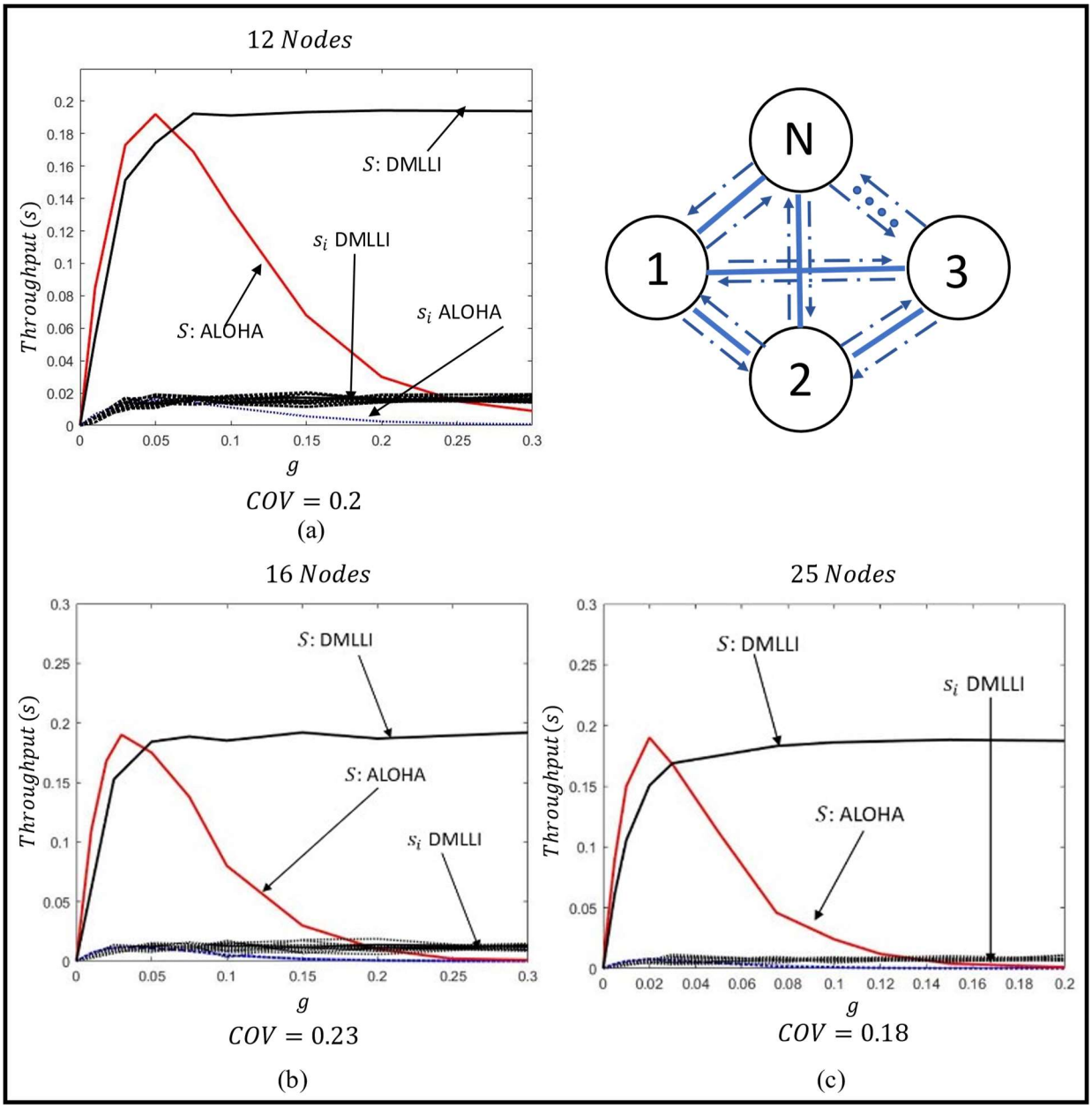

Figure 4.11: Performance of DMLLI-MAC in networks with (a) 12 nodes, (b) 16 nodes and (c) 25 nodes with heterogeneous load distribution [s_i denotes individual throughputs; S is the network-wide throughput; g denotes load (in Erlang) from the application layers].

It is to be noted here that the threshold coefficient ϵ_f defined in the reward function in Eqn. (4.2)

is assigned a value of 0 for networks with less than 12 nodes and a value of -0.025 for larger networks with more than or equal to 12 nodes. The logic behind using a threshold dependent on the network size can be explained using the convergence plots shown in Fig. 4.12. The figure shows the convergence of individual node throughput and network-wide throughput for a fully connected network with 12 nodes. It can be seen that even after convergence, there are oscillations (spikes) for the individual node throughput. These oscillations are the result of the stochasticity in action selection and Poisson distribution-driven packet generation of the system. As a result of these sudden impulses in the individual throughput, the fairness coefficients of the neighboring nodes go down penalizing these nodes. Thus, the neighboring nodes get penalized even for optimal transmission strategies. Hence, a threshold of -0.025 prevents the nodes to settle in a non-optimal solution.

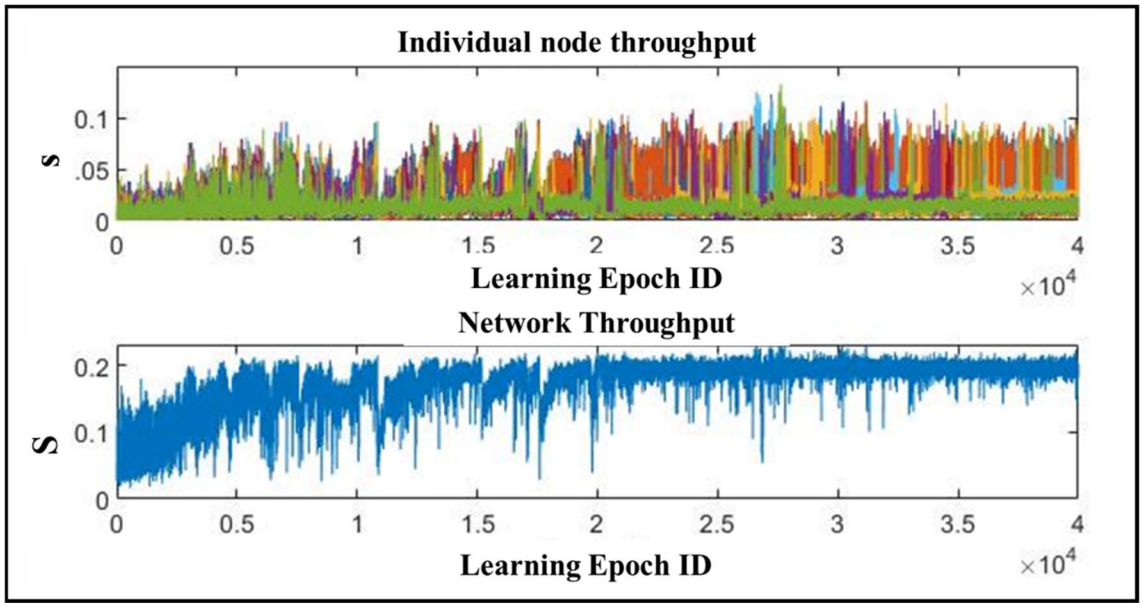

Figure 4.12: Convergence plots for individual node throughput and network-wide throughput for a 12 node fully connected network.

4.4.1.2 Partially Connected Topology

Unlike in the fully connected topologies discussed so far, in partially connected topologies as shown in Fig 4.1 (c), network information sharing in restricted up to only 2-hop neighborhoods.

3-Nodes Network: We begin our analysis with a simple 3-node network topology as shown in Fig. 4.13. The figure compares the performance of DMLLI with benchmark ALOHA for homogeneous load distribution. ALOHA throughput attains the maximum (i.e., $s_1 = s_2 = s_3 \approx 0.08$), when the optimum loads $\widehat{g_1} = \widehat{g_2} = \widehat{g_3} \approx 0.2$, and then decreases asymptotically due to increased packet collisions. In DMLLI, the nodes learn the optimal transmission strategy so that the number of collisions are reduced, and like the fully connected scenarios, the maximum throughput is maintained for higher network loading conditions.

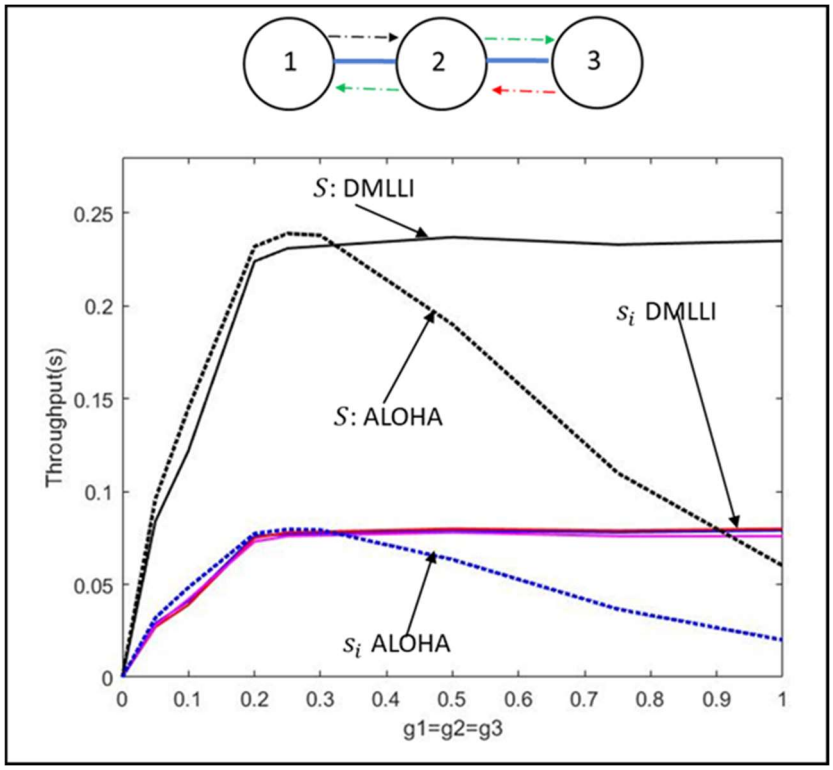

Figure 4.13: Load vs Throughput for 3-node linear network; s_i, g_i are the individual throughputs and loads for node i; S is the network throughput.

DMLLI is also able to support inter-node bandwidth distribution fairness for partially connected networks. With benchmark ALOHA, throughput is unevenly distributed in the presence of load heterogeneity. However, with DMLLI, the nodes learn in a decentralized manner to adjust their individual transmission probabilities such that the variation across individual throughputs is

greatly reduced. That is even when the generated traffic loads across the nodes are heterogeneous.

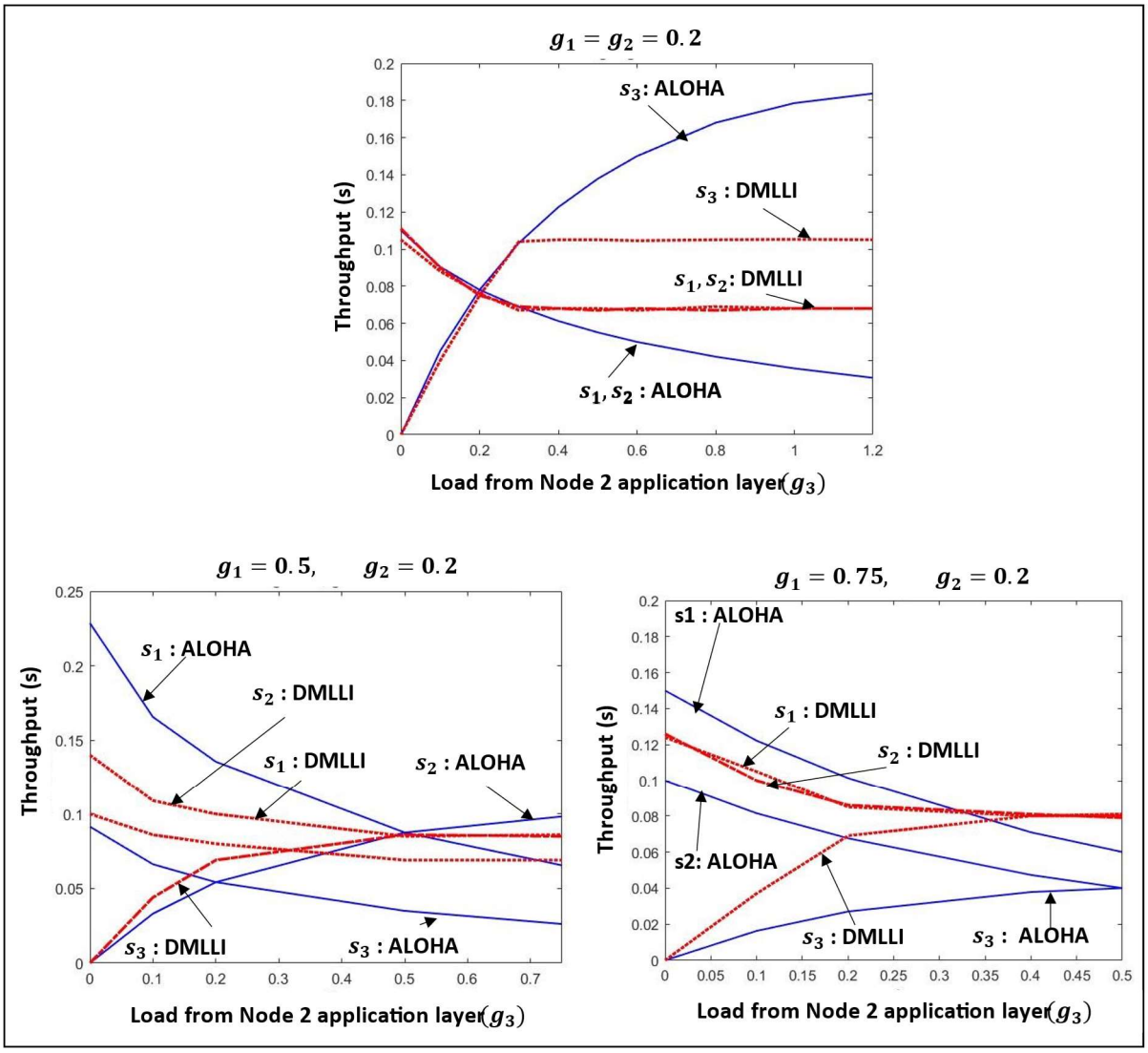

Figure 4.14: Performance of DMLLI in a 3-node linear network with heterogeneous load.

Such fairness is achieved by learning to maximize the fairness coefficient (f_i) in the reward function stated in Eqn. 4.2. This behavior of DMLLI is explained in Fig. 4.14, which shows the load-throughput plots in the presence of heterogeneous load. In each of the three plots, the loads from node-1 (g_1) and node-2 (g_2) are kept fixed at different values, and the node-level throughput variations are observed for varying load from node-3 (g_3). These represent the scenarios: $g_1 \leq \hat{g}, g_2 \leq \hat{g}, g_1 \leq \hat{g}, g_2 > \hat{g}$ *or* $g_1 > \hat{g}, g_2 \leq \hat{g}$, and $g_1 > \hat{g}, g_2 > \hat{g}$. The RL agents in the nodes

learn to adjust the transmit probabilities such that the available wireless bandwidth is fairly distributed. As before, the graph also shows how, unlike ALOHA, the throughout is held for higher network loads.

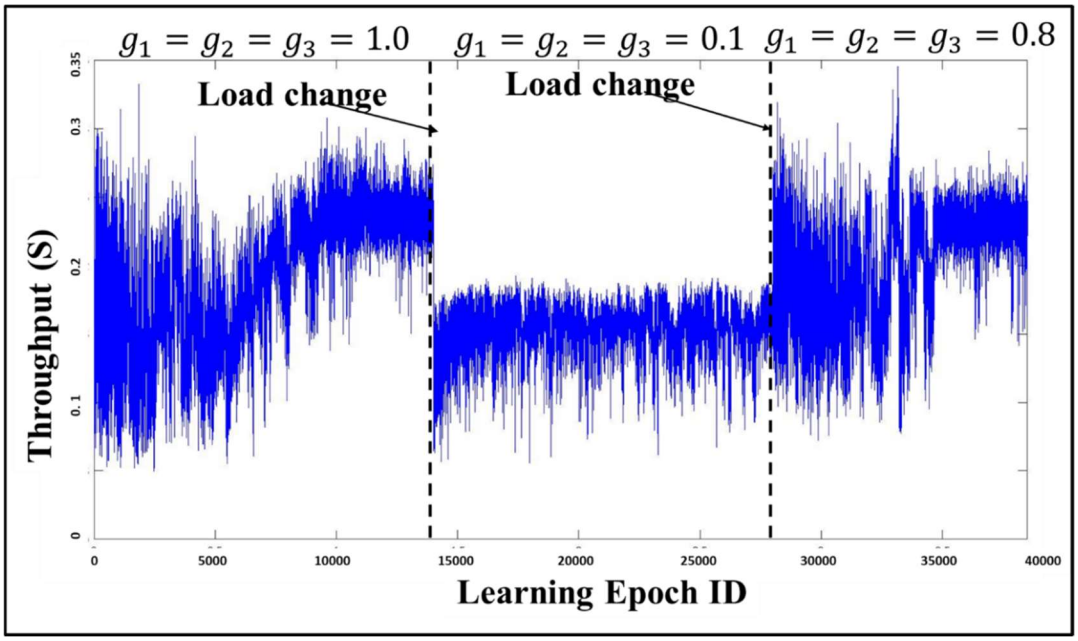

Figure 4.15: Performance of DMLLI in a dynamic environment.

Fig. 4.15 shows the online learning ability of DMLLI in order to adjust to time-varying loading conditions. Initially, the application layer load in the network is $g_1 = g_2 = g_3 = 1.0$, for which the network throughput (S) converges to the optimal value of 0.24. Subsequently, the load changes to $g_1 = g_2 = g_3 = 0.1$ and $g_1 = g_2 = g_3 = 0.8$ at two different points in time as indicated by the dotted vertical lines. It is observed that the RL framework allows the nodes to adjust transmission probabilities at those two change-points and maintain the optimal throughputs. This is achieved by the nodes learning to transmit packets with an optimal transmit probability according to the incoming application layer traffic.

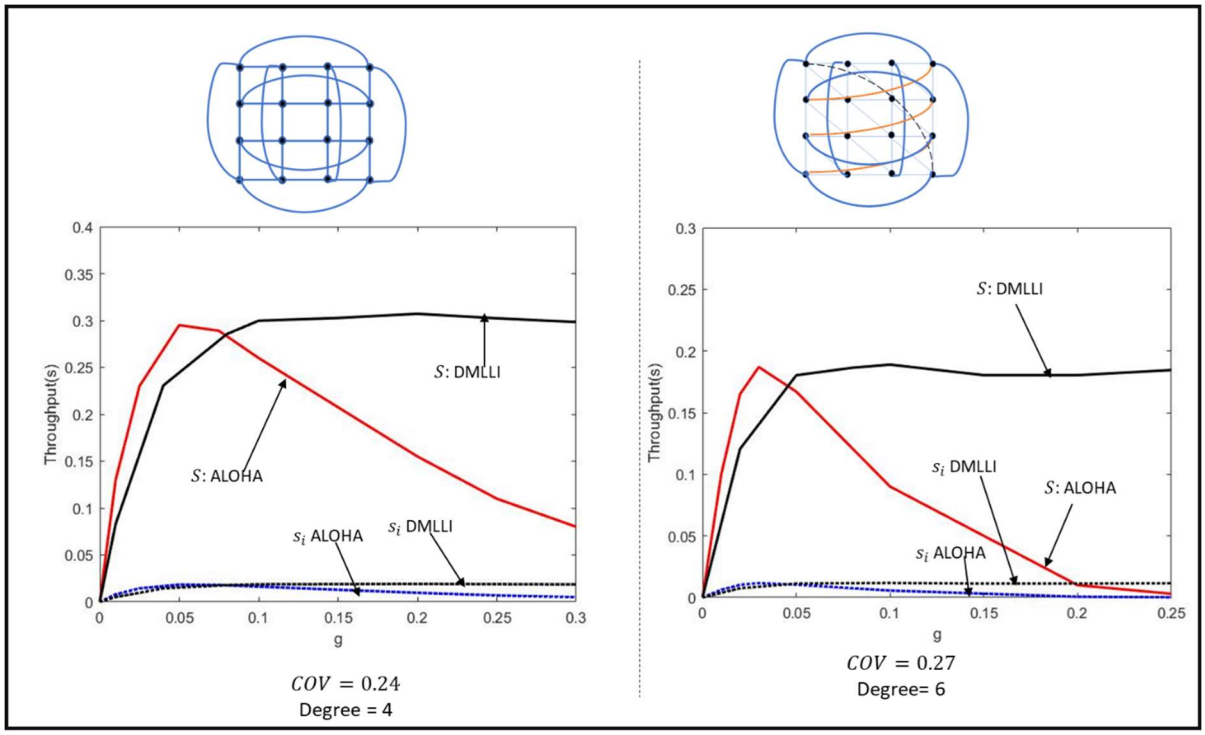

Figure 4.16: Performance of DMLLI in partially connected networks with 16 nodes [s_i denotes individual throughputs; S is the network-wide throughput; g denotes load (in Erlang) from the application layers].

Large Networks: Experiments were also performed with large partially connected networks with 16 and 25 nodes arranged in toroidal topologies as shown in Figs. 4.16 and 4.17. Contrary to the fully connected case, the information availability for a node is limited only to its two-hop neighbors. The throughput results in Figs. 4.16 and 4.17 validate DMMLI's learning abilities to attain the benchmark throughput (i.e., of pure ALOHA), and to maintain it for higher loading situations in large networks. The ability to fairly share the network bandwidth is also observed in these figures. The coefficient of variation among the nodes' throughput is within a range of $[0.23, 0.27]$ and $[0.21, 0.29]$ for the 16-node and 25-node networks, respectively.

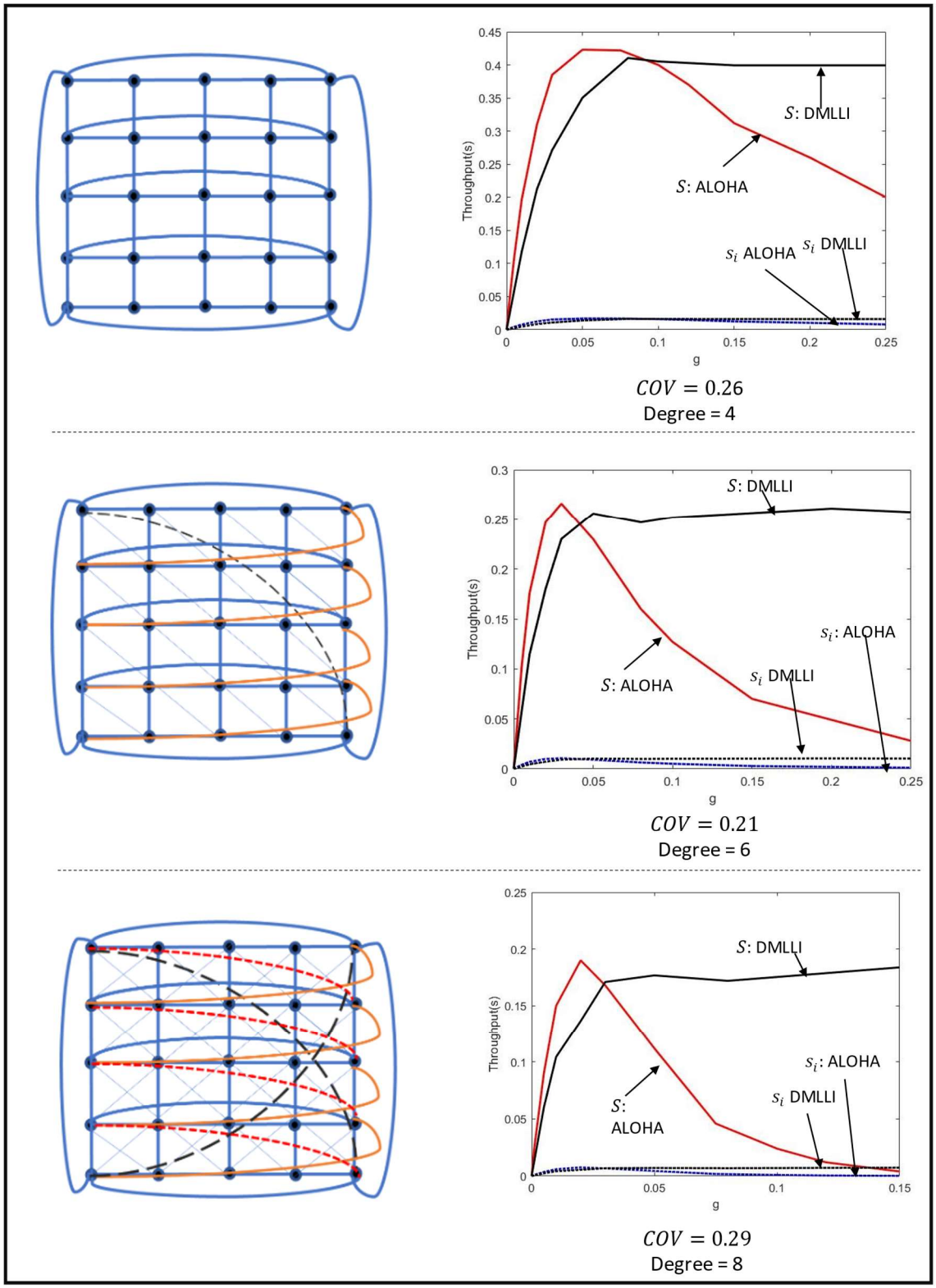

Figure 4.17: Performance of DMLLI in partially connected networks with 25 nodes [s_i denotes individual throughputs; S is the network-wide throughput; g denotes load (in Erlang) from the application layers].

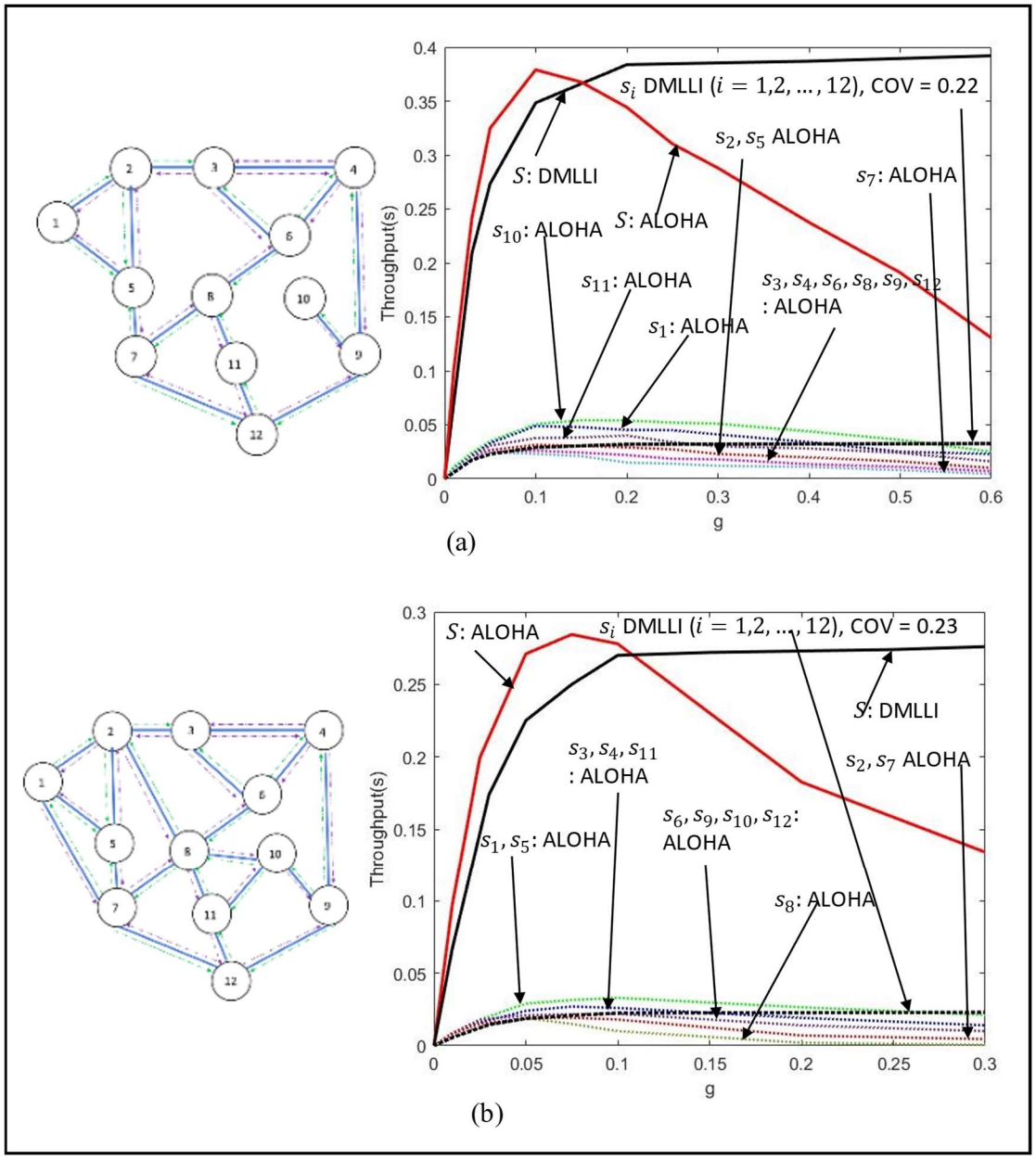

Figure 4.18: Performance of DMLLI in heterogeneous mesh networks with 12 nodes.

Heterogeneous Mesh Topologies: Performance of DMLLI is analyzed in networks with arbitrary mesh topologies. The major drawback of using the ALOHA family MAC logic in these networks is that the throughput is not fairly distributed because of the inherent topological heterogeneities. For example, in the 12-nodes mesh topology in Fig. 4.18 (b), node 8 is in a topologically disadvantageous position as compared to the rest of the nodes, and hence, its throughput with

ALOHA and its derivatives is less than the other nodes. The situation compounds with increase loading conditions. The learning mechanism in DMLLI is shown to address such unfairness.

Figs. 4.18 (a) and (b) compare the performance of DMLLI and pure ALOHA in two 12-node arbitrary mesh networks with maximum degrees of 3 and 5 respectively. It is observed that with DMLLI, the nodes learn to adjust their individual transmission probabilities such that the available wireless bandwidth is fairly distributed across all nodes despite their heterogeneous topological positions. Furthermore, unlike with ALOHA, the attained maximum throughputs hold for larger application layer loads.

4.4.2 MAC Learning with Access Priorities

The learning in DMLLI can incorporate node-specific access priorities by assigning node-specific priority coefficients θ_i in the reward function in Eqns. 4.1 and 4.2. By assigning priority, a specific node can receive higher share of the available wireless bandwidth. This strategy is particularly useful when data from certain sensors/IoTs are more critical than the others and the available wireless bandwidth is limited.

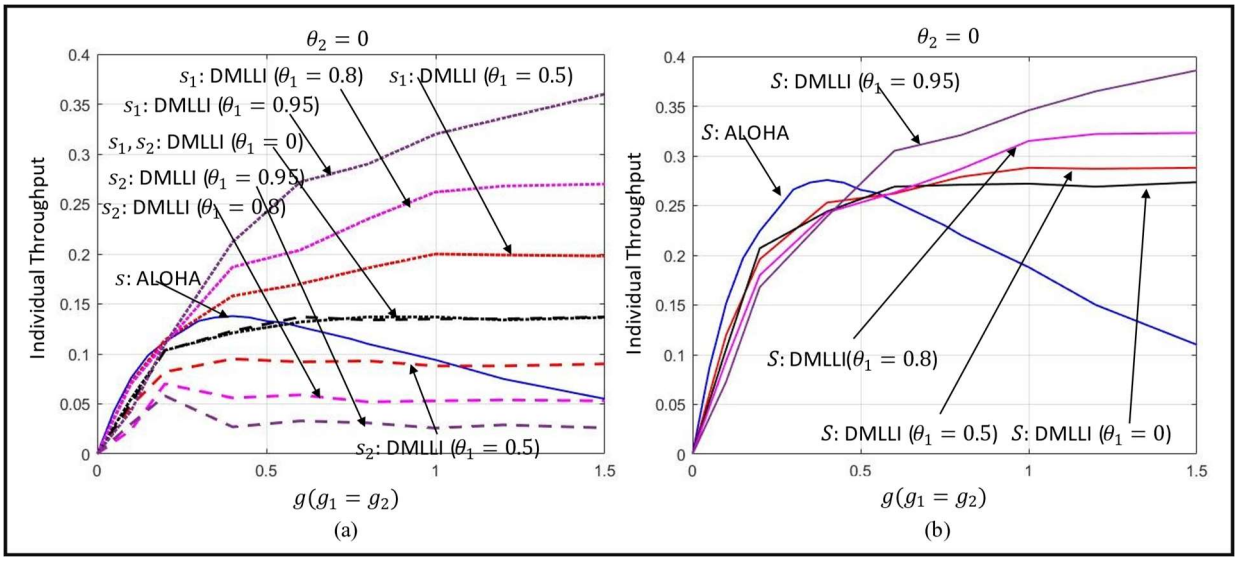

Figure 4.19: Assignment of node-level access priority among nodes.

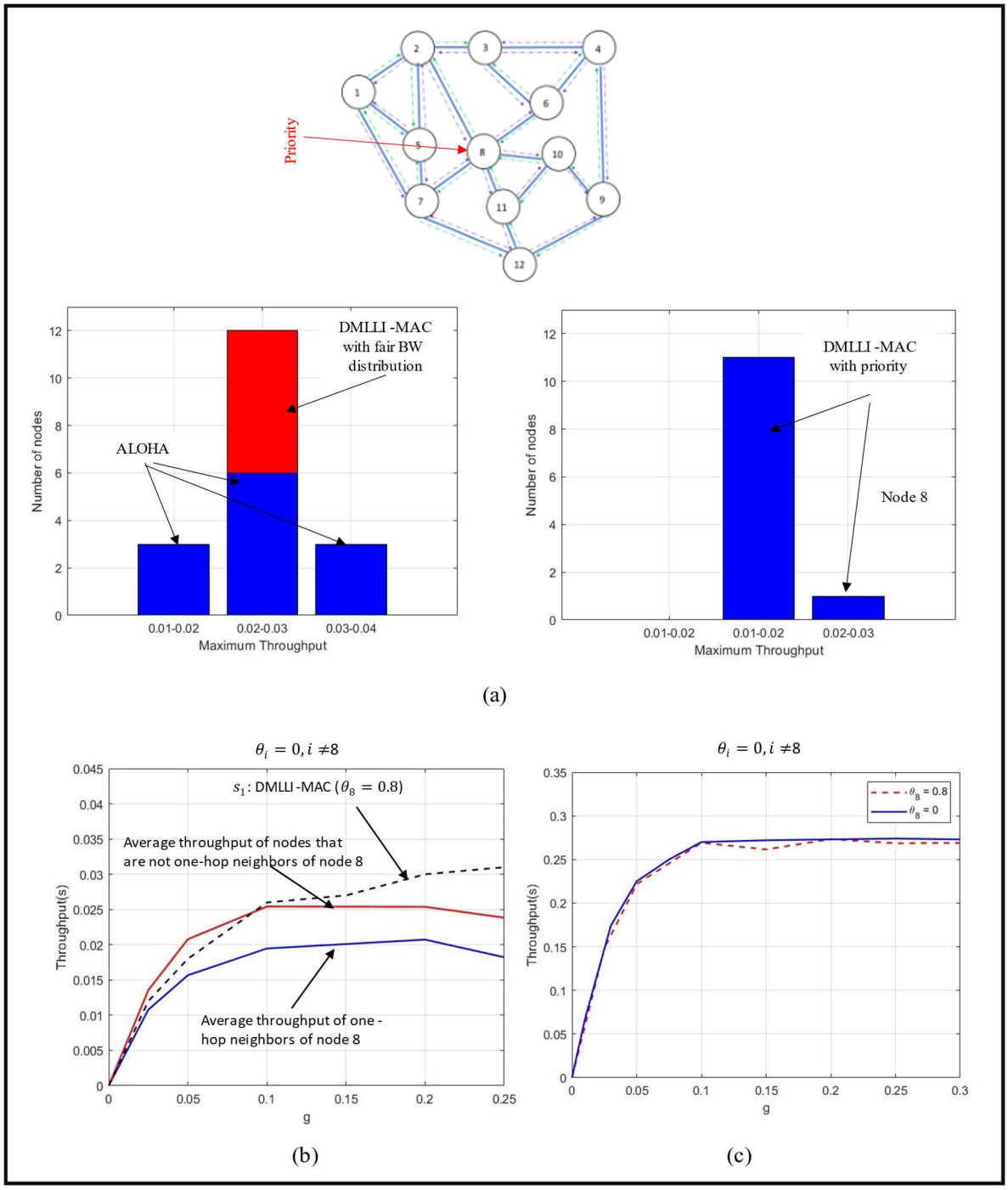

Figure 4.20: Assignment of node-level access priority to node 8 in arbitrary mesh topology.

2-Nodes Network: For assigning priority, the value of priority coefficient θ_2 is kept fixed and equal to zero. The value for node 1, θ_1, is then varied to assign different access priorities to node 1. Thus, the fairness coefficients from Eqn. (4.1) for nodes 1 and 2 respectively become:

$$f_1(t) = -|(1-\theta_1) \times s_1(t) - s_2(t)|$$

$$f_2(t) = -|s_2(t) - (1-\theta_1) \times s_1(t)|, where\ 0 \leq\ \theta_1, \theta_2 \leq 1.$$

The reward function aims at maximizing f_1 and f_2, and the maximum possible values are $f_1 = f_2 = 0$. After convergence, both the nodes jointly learn to transmit such that the throughput of node 2 becomes a fraction of node 1's throughput. In this way, node 1 gets priority in terms of the bandwidth share as compared to node 2.

Fig. 4.19 (a) shows the load-throughput plot for different values of θ_1. It is observed that when $\theta_1 = \theta_2 = 0$, as expected, the bandwidth is fairly distributed. With increase of θ_1, node 1 gets a higher share of the bandwidth. Moreover, the difference in throughput between node 1 and 2 is proportional to the difference between the fairness coefficients θ_1 and θ_2. Fig. 4.19 (b) shows the variation in network-wide throughput with load for different values of θ_1. The network throughput increases with increase in θ_1. This is because, with the assignment of priority to one of the nodes, the other node transmits a smaller number of packets and hence the collisions are reduced, thus resulting in an increase in network throughput.

Arbitrary Mesh Topology: Fig. 4.20 demonstrates the effects of node-level access prioritization for a 12-node network running DMLLI-based access learning. Priority is assigned to node 8 which is in the most disadvantageous topological position due to its highest number of bandwidth-competing neighbors. Fig. 4.20 (a) shows the distribution of node-throughputs for three cases: first, ALOHA, second, DMLLI with fair bandwidth distribution ($\theta_i = 0, \forall i$), and finally, DMLLI with priority assigned to node 8. The key observation here is that by assigning priority to node 8 ($s_8 = 0.8$), node 8 gets higher share of bandwidth as compared to the rest of the network. It is to be noted that when assigned priority node 8 siphons bandwidth out from its directly connected nodes which is evident from Fig. 4.20 (b). This also demonstrates that assigning priority to a node affects only its local neighborhood, and not beyond. It should also

be observed that on assigning access priority to node 8, the variation of network-wide and node throughputs remains the same as in the case with fair bandwidth distribution. This is because, node 8, on assigning priority, try transmitting with higher transmit probability. But, owing to its topological position, it suffers more packet collisions with its neighboring nodes. As a result, the

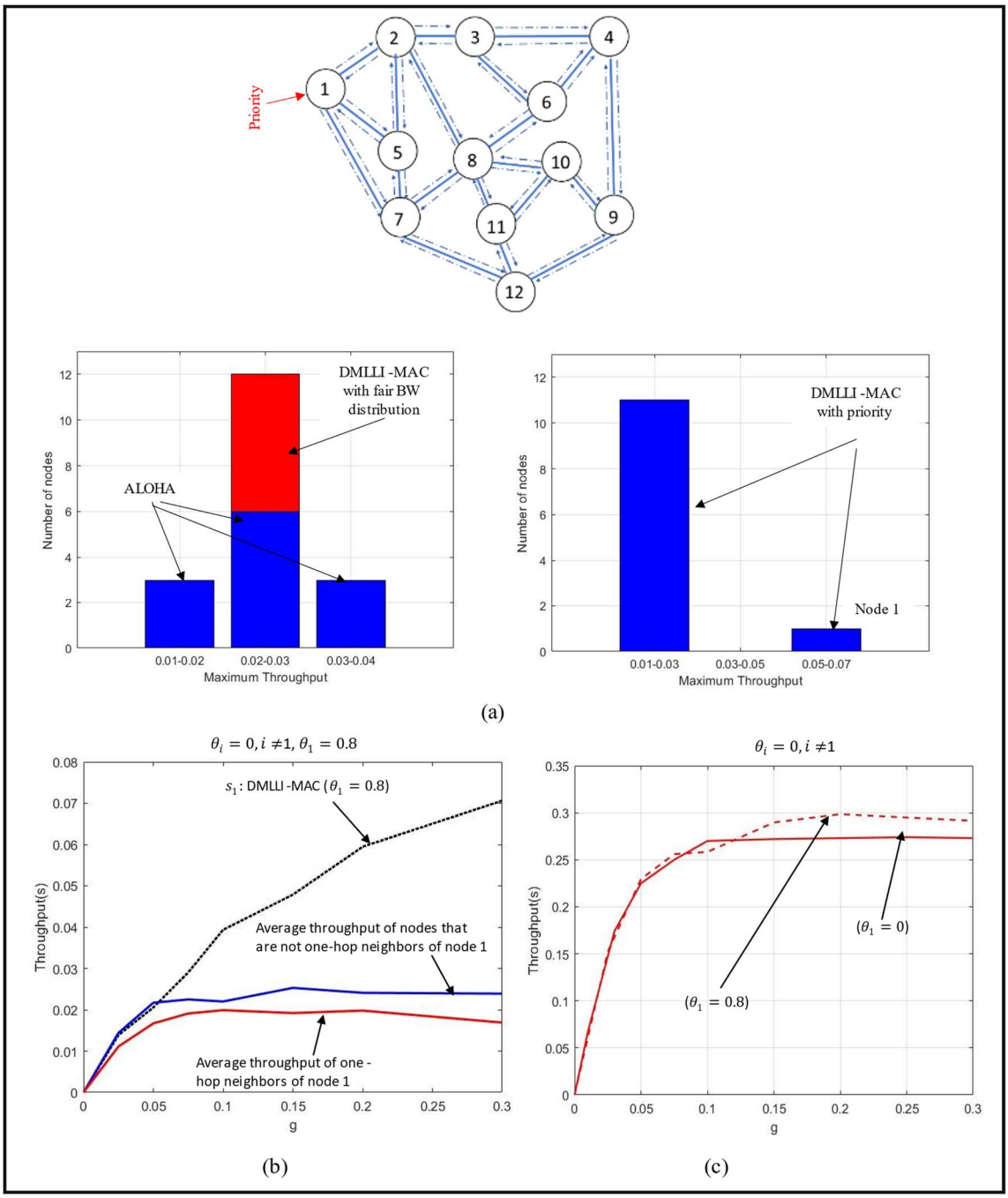

Figure 4.21: Assignment of node-level access priority to node 1 in arbitrary mesh topology.

network throughput does not increase on assigning priority in this case. This behavior is shown in Fig. 4.20 (c).

The next experiments are with assigning access priority to node 1, which unlike node 8, is in a relatively topologically advantageous position due to lesser number of neighbors. The results are shown in Fig 4.21. The distributions of nodes with throughput for three different scenarios: ALOHA, DMLLI with fair bandwidth distribution, and DMLLI with prioritized access to node 1 are shown in Fig. 4.21 (a). Node 1 gets a higher share of network bandwidth with access-priority. The fact that assigning priority to a node only affects its local neighborhood is valid here as well and can be observed from Fig. 4.21 (b). The increase in throughput for node 1 is at the expense of throughput loss of its directly connected neighbors. Furthermore, in this scenario, network-wide throughput also increases with assignment of priority to node 1. This is because, unlike in the previous case, the prioritized node here has fewer 1-hop bandwidth competitors. As a result, the increase in node-1's throughput does not increase collisions so much that the overall network throughput goes down. In fact, the throughput increases.

Table 4.2: $\frac{S_{DMLLI-MA}}{S_{ALOHA}}$ ratio for different packet error probability

Packet Error Probability	**0**	**0.01**	**0.05**	**0.1**
$\frac{S_{DMLLI-MAC}}{S_{ALOHA}}$ (12-nodes partially connected network)	1.8319	1.9183	1.8595	1.8214
$\frac{S_{DMLLI-MAC}}{S_{ALOHA}}$ (3-nodes fully connected network)	4.1217	4.7205	4.5243	4.8499

4.4.3 Effect of Channel Unreliability

Experiments in this subsection demonstrate the effects of channel unreliability on MAC protocol learning performance. Figs 4.22 (a) and (b) show the throughput variation for varying packet error probabilities in a 3-nodes fully connected network and 12-nodes partially connected network (Figs 4.21). The following observation can be made. In Fig. 4.22, as expected, for both ALOHA and DMLLI-MAC, their throughputs go down because of packet loss on unreliable channels. However, it can be observed from Table 4.2. that the throughput ratio ($\frac{S_{DMLLI-MA}}{S_{ALOHA}}$), after learning convergence, remains in the same ballpark value for different values of packet error probabilities. This indicates that the impacts of channel errors on DMLLI-MAC are no worse than those on the baseline ALOHA protocol.

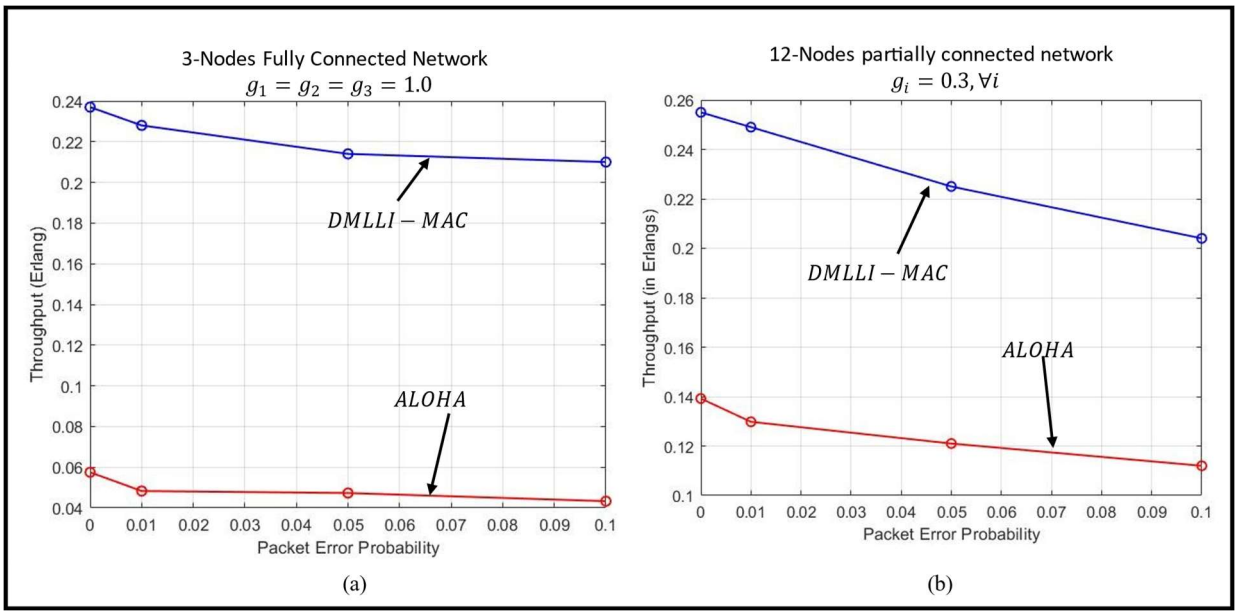

Figure 4.22: Effect of channel unreliability on DMLLI-MAC and ALOHA throughput on (a) 3-nodes fully connected network, (b) 12-nodes partially connected network.

4.5 Summary

To summarize, this chapter proposes a scalable, decentralized Reinforcement Learning (RL) framework for MAC layer wireless network protocol synthesis, in the absence of localized network information visibility. The mechanism is targeted for low-complexity wireless

transceivers in that it does not rely on complex lower-level hardware support, such as, carrier sensing and time-synchronization. The salient features of the mechanism are as follows. First, it is capable of achieving the maximum network-wide throughput that can be obtained by the known benchmark ALOHA family of protocols. Second, unlike ALOHA, the maximum throughput can be sustained for high network loading scenarios. Third, the synthesized MAC protocol ensures a fair node-level wireless bandwidth distribution in the presence of loading and topological heterogeneities. Fourth, by adjusting a set of priority coefficients in the Reinforcement Learning reward function, access priorities can be assigned to specific network nodes. Finally, the learning is shown to make the nodes adjust their transmission strategies in order to adapt with time-varying traffic conditions. Detailed simulation experiments were performed to demonstrate the functional and performance validity of the proposed paradigm in a wide variety of network types, loading, and operating conditions.

Now that we have established the concept of protocol synthesis for random access MAC, a natural extension of this work would be to broaden the applicability of the learning paradigm for wider MAC applications, such as, networks with time-slotting capability. Building on the idea and insight developed in this chapter, we present a learning framework designed for networks with TDMA-based MAC arrangements in the next chapter.

Chapter 5: TDMA Slot Allocation using Multi-Armed Bandits

Till this point in this thesis, we have explored the concept of reinforcement learning (RL)-enabled network protocol synthesis, particularly focusing on random access MAC arrangements. Moving forward, the next step in this research involves extending this protocol synthesis approach to networks with time slotting capacity. Specifically, we delve deeper into developing and implementing RL-driven protocols in networks with TDMA-based access schemes. We will investigate how RL and its variants can be leveraged for performance improvement in such networks.

With the high-level objective of demonstrating the concept of protocol synthesis for TDMA MAC arrangements, in this chapter, we specifically focus on the problem of slot allocation. A decentralized learning framework for MAC slot allocation is developed for resource-constrained networks using Multi-Armed Bandits (MAB), a variant of RL. In this chapter, we particularly deal with networks that do not possess time-synchronization ability. As will be demonstrated later in this chapter, MAC slot allocation in the absence of time synchronization is a challenging problem, and it is a notable feature of the proposed learning mechanism. The non-reliance on network time synchronization makes the proposed learning mechanism feasible for low cost and low complexity transceivers for wireless sensor networks and IoT devices. MAC slot allocation is formulated as an MAB problem where the nodes act as independent learning agents and learn transmission policies that ensure collision free transmissions.

5.1 Motivation

Accurate time synchronization in wireless networks can be expensive to realize especially in low-cost nodes with limited processing and communication resources. Moreover, the MAC layer

performance in such networks can be very sensitive to even slight perturbations in the quality of time synchronization [21]. Hence, this work aims to develop a learning-based time asynchronous TDMA framework useful for low-cost transceivers in Wireless Sensor Networks (WSNs) and Internet-of-Things (IoTs).

We propose a decentralized MAB learning framework for MAC slot allocation. As already explained earlier, the advantage of using distributed learning is that all the network nodes, which are learning agents, learn independently without explicitly sharing the learning policies with each other. This is specifically useful in partially connected network topologies, where the nodes have limited network information visibility. This also makes the framework scalable with network size since the learning is done independently in each node, and its performance depends on network degree rather than the network size. Note that centralized learning, in which a centralized agent, with access to complete network level information, can learn optimal node behavior (i.e., a protocol) and downloads it to the individual nodes, typically requires additional network resources in terms of a separate channel to share the learned policies to the nodes. In addition, it usually puts a heavy burden on the central server in terms of computation perspective [24], [25].

There are several papers [48, 42, 38] that use Multi-armed Bandits and Reinforcement Learning for wireless MAC slot allocation. These works in general rely on network time synchronization. The approaches in [83] [84] propose centralized slot allocation without time synchronization. Such centralized approaches are often not suitable in wireless networks because of the drawbacks mentioned earlier. In this work, it is shown that wireless MAC slot allocation is feasible even in the absence of time synchronization and centralized controllers. This can be achieved using distributed Multi-armed Bandit strategy while trading certain amount of wireless bandwidth for fast allocation convergence.

One major challenge of using learning in time-asynchronous scenario is slow convergence rate as compared to that in a time-synchronized network. Moreover, the convergence time increases with the use of distributed learning due to the inherent limitations of timely information visibility as compared to centralized learning. Hence, in order to accelerate the convergence behavior, a novel concept of hysteretic MAB, that uses two learning rates, has been explored. It is shown when compared to the native MAB approaches, Hysteretic MAB reduces the convergence time up to 63% for time asynchronous network and up to 56% for networks with time synchronization. The MAC slot allocation has been modelled as a Multi-Agent Multi-Armed Bandits (MAMAB) problem, where each node acts as a 'k-armed bandit' agent. An arm represents a MAC transmission slot that the agent can choose. Each node maintains its own notion of a fixed duration transmission frame with *k*-slots. Since time is not synchronized, the frames for the nodes are also not synchronized. Each node can independently select a slot, which is an arm in MAB. The learning framework allows the nodes to learn arm selection policies, together that constitutes a transmission slot selection protocol, in a distributed manner. After learning convergence, the system ensures that there are no overlapped transmissions (i.e., collisions) in the system. It is shown that the learning mechanism gives the desired performance for networks with both fully connected and partially connected arbitrary mesh topologies. The nodes in a partially connected topology learn to reuse bandwidth spatially by choosing fully to partially overlapping transmission slots when they are outside their mutual realm of influences.

5.2 Network and Traffic Model

The proposed mechanism is developed for generalized multi-point to multi-point networks with arbitrary mesh topologies (i.e., fully connected and partially connected) and network traffic patterns. From a learning standpoint, the main difference between the two connectivity modes is

the amount of slot allocation information availability at each node. While for the fully connected scenario, each network node possesses the current MAC slot information for all other nodes in the network; for the partially connected case, a node knows slot allocation information only within its local neighborhood. Figs. 5.1 (a) and (b) shows two examples of fully and partially connected networks.

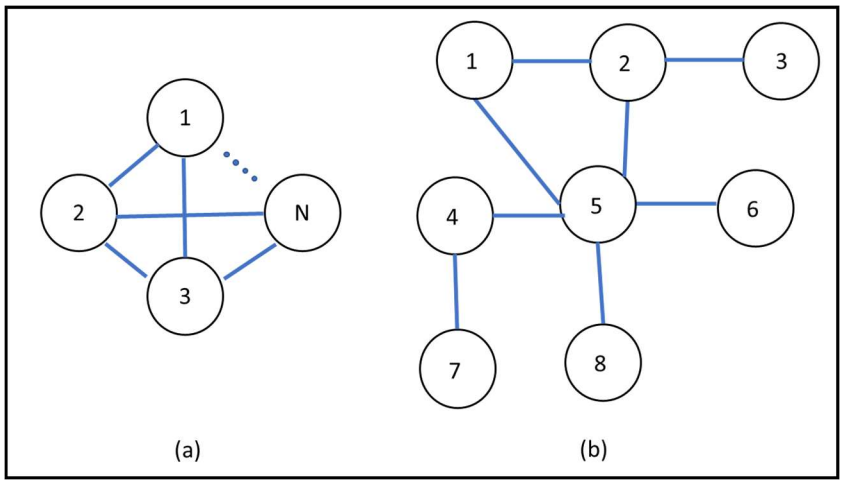

Figure 5.1: (a) Fully connected and (b) Partially connected network topologies.

Two different packet generation models, namely, Constant Bit Rate (CBR) and Poisson Distributed random have been used. The MAC layer traffic load model is created such that a packet generated in a node is broadcasted to all its one-hop neighbors.

The network nodes are assumed to be not time synchronized. This is a crucial feature since MAC slot allocation in the absence of time synchronization is a challenging problem, and it is a notable feature of the proposed learning mechanism in this work. The network model includes the availability of piggybacking for sending control information using a small part of the data packets. Such control information is used for sharing feedback on packet transmissions that end up in packet collisions. Such piggybacking-based information sharing allows the framework to be not dependent on the abilities of direct collision detection, which is especially meaningful for the low-complexity wireless transceivers in IoT/Sensor nodes.

5.3 Asynchronous TDMA MAC Operation

Asynchronous Frames and Mini-slots: Similar to regular TDMA, the target mechanism would work with fixed size frame abstraction. Frames, however, in this model are not synchronized across the network nodes. The notion of frame is local to each node. A node decides the time of start of its own frame, and the frame end time is decided based on the fixed frame duration, denoted by T_{frame}. The node does not know about the start times of the other network nodes' frames. Within a frame, a node can schedule a packet transmission only in certain discrete time instances away from its frame start time. The intervals between those time instances are referred to as mini-slots, the duration of which is an integer submultiple of the packet duration, and is equal at all nodes. The relationship between mini-slot duration (T_{mini}) and packet duration (τ) can be expressed as:

$$\tau = n_M \times T_{mini}, \text{ where } n_M \in I \tag{5.1}$$

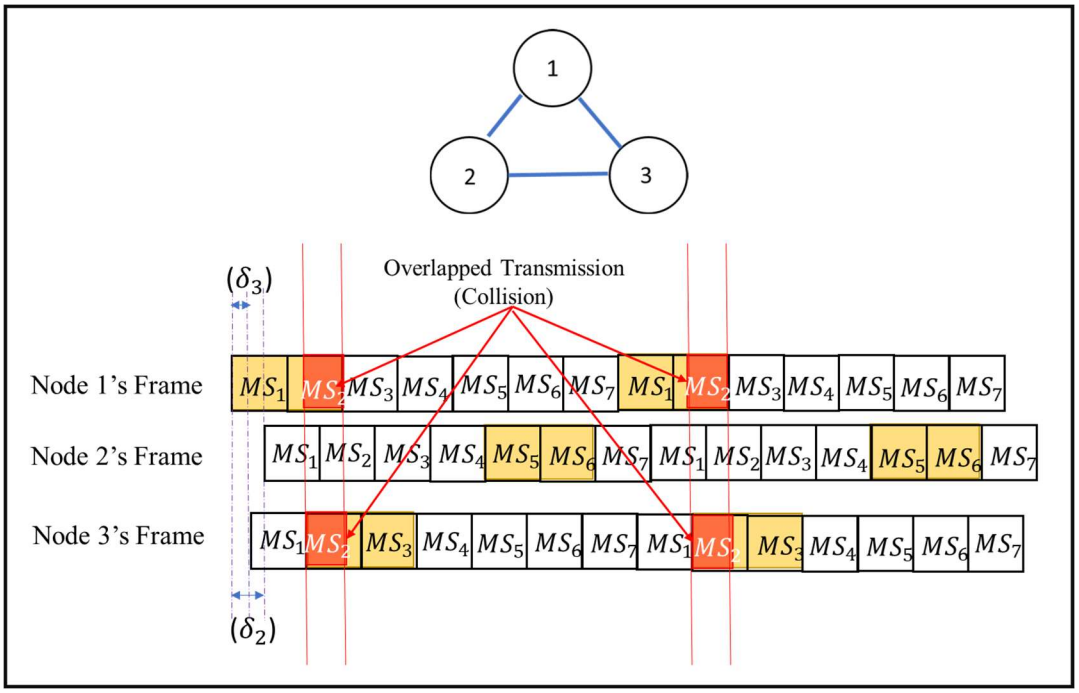

Figure 5.2: Frame and mini-slot structure in a 3-nodes fully connected network.

To be noted that since frames across nodes are not necessarily synchronized, the mini-slot time boundaries across the nodes are not necessarily aligned. As a special case, when the frames are synchronized and the mini-slot size equals the packet duration, it becomes a standard TDMA system.

This arrangement of mini-slot based asynchronous TDMA is shown for a 3-nodes fully connected network in Fig. 5.2. Note that frames of node 2 and node 3 lag from node 1's frame by δ_2 and δ_3 durations. In this specific example, the frame size equals 7 mini-slots and a mini-slot duration is half of packet duration (i.e., $n_M = 2$). A node can select any of these 7 mini-slots within its own frame as the starting point of its packet transmission. The figure depicts a situation where nodes 1, 2 and 3 select mini-slots 1, 5 and 2 in their own respective frames for packet transmissions. The nodes periodically transmit in those mini-slots in subsequent frames. Observe that the packets from nodes 1 and 3 get collided because of their time-overlapped transmissions (indicated by red), whereas packets from node 2 are successfully transmitted. The objective of this paper is to develop a distributed learning-based mechanism (presented in Section 5.4), where the nodes can learn to select collision-free transmission mini-slots with fast learning convergence.

Bandwidth Redundancy: Any learning for mini-slot selection would require nodes to perform certain amount of iterative search for a collision free transmission mini-slot within its own frame. Since the targeted learning is distributed in that each node performs its own independent search, short term collisions and scheduling deadlocks can occur. This can be mitigated by making the frames longer than the absolutely minimum required length, leading to certain amount of bandwidth redundancy. This redundancy can be expressed by a factor *K*:

$$K = \frac{Frame\ Size\ (in\ mini-slots)}{Minimum\ frame\ size\ (in\ mini-slots)} \tag{5.2}$$

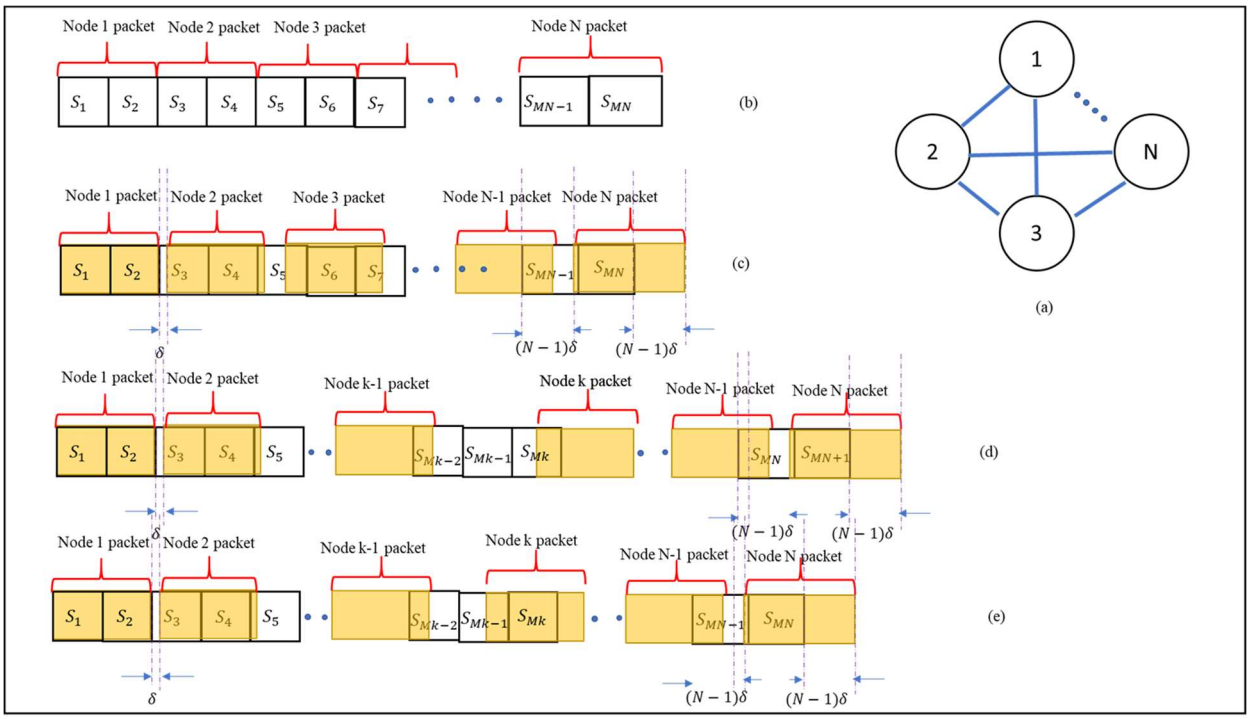

Figure 5.3: (a) N-nodes fully connected topology, (b) Slot allotment with time synchronization, (c) With frame of reference of Node 1, slots occupied by packets from other nodes without time synchronization when maximum frame lag is less than the slot duration (d) and (e) With frame of reference of Node 1, slots occupied by packets from other nodes without time synchronization when maximum frame lag is more than the slot duration.

The minimum frame size in the denominator of Eqn (5.2) represents the absolute minimum frame size that is possible in the presence of time synchronization. The requirement of a larger frame size in the absence of time synchronization can be demonstrated in terms of an *N*-node fully connected network as shown in Fig. 5.3 (a). The minimum frame size (i.e., expressed in number of mini-slots) required for a time synchronized network is $N \times n_M$, where n_M is the number of mini-slots per packet duration (from Eqn. 5.1). The frame structure in a time synchronized network is shown in Fig. 5.3 (b) (With n_M=*2*). A time asynchronous scenario is shown in Fig. 5.3 (c). The positions of mini-slots of all nodes with respect to node 1's frame are shown in Fig.

5.3 (c). The frame of node i lags from that of node 1 by $(i-1) \times \delta$ durations. There are two distinct scenarios as follows.

Case I: Maximum frame lag with respect to node 1 is less than mini-slot size (i.e., $(N-1)\delta < T_{mini}$). In this case, the mini-slot scheduled for packet transmission for node N starts at $(N-1) \times n_M \times T_{mini} + (N-1)\delta$ and ends at $N \times n_M \times T_{mini} + (N-1)\delta$. Since $(N-1)\delta < T_{mini}$. Hence the minimum frame size that will ensure a collision free transmission is $N \times n_M + 1$ mini-slots.

Case II: Maximum frame lag with respect to node 1 is more than the mini-slot size. Let the frame lag of the k^{th} node with respect to node 1 exceeds the mini-slot size (i.e., $(k-1)\delta > T_{mini}$). In this case, there is a free mini-slot worth of duration where no node is transmitting (i.e., in Fig. 5.3 (d)). Hence, each node $j, \forall j \geq k$, can transmit in one mini-slot prior to its current mini-slot (i.e., in Fig. 5.3 (e)). Hence, even in this case, mini-slot scheduled for packet transmission for node N starts at $(N-1) \times n_M \times T_{mini} + (N-1)\delta$ and ends at $N \times n_M \times T_{mini} + (N-1)\delta$. Therefore, the minimum frame size needed for a collision free transmission is $N \times n_M + 1$ mini-slots.

Thus, for a fully connected *N*-nodes time-asynchronous network, the minimum frame size required for collision-free transmission is $N \times n_M + 1$ mini-slots, which is one mini-slot more than that in a time-synchronous scenario. For a partially connected network, the frame size depends on the maximum network degree, rather than the network size. As explained in the following section, this minimum frame size and the adopted frame size, indicated by the redundancy factor K in Eqn. 5.2, play a crucial role in the proposed learning mechanism and its convergence behavior.

5.4 Slot Allocation using Decentralized MAB

This section details the proposed Multi-Armed Bandit (MAB) based distributed learning mechanism for transmission scheduling within the MAC framework described in Section 5.3.

5.4.1 Introduction to MAB

Multi-Armed Bandits (MAB) is a special class of Reinforcement Learning in a non-associative setting [85] [86] [76]. It is applicable in situations which do not involve learning in different states of a system. In other words, there is no concept of state as in generalized reinforcement learning [12]. A much-explored variant of MAB is the 'k-armed bandit' problem, where the learning agent (bandit) has k possible arms or possible actions to choose from. Each of the k actions has an associated stochastic reward the distribution of which is not known to the learning agent. After an arm/action is chosen, the agent gets a sample of that reward following the unknown distribution. In other words, it gets feedback in terms of a numerical reward on how good or bad a selected action is. The agent's goal is to maximize the total accumulated reward over an infinite time horizon by learning to estimate the reward distribution of the possible actions.

Formally stated, the value for an action a is denoted as: $q_*(a) \leftarrow E[R_t \,|A_t = a]$. At each timestep t, the value of each action a is estimated iteratively as $Q_t\,(a)$. The model is said to converge when the estimate $Q_t(a)$ becomes close to the true value for action a, that is, $q_*(a)$. A simple approach to estimate the action value at instant t is to find the average of all the rewards received till instant t:

$$Q_t(a) = \frac{Sum\ of\ Rewards\ when\ a\ was\ taken\ prior\ to\ t}{Number\ of\ times\ a\ was\ taken\ prior\ to\ t} \tag{5.3}$$

As the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$. Thus, the estimate of action a at time instant t is given by:

$$Q_t = \frac{R_1 + R_2 + \cdots + R_{t-1}}{t-1}$$

$$\Rightarrow Q_{t+1} = \frac{1}{t}\sum_{i=1}^{t} R_i$$

$$\Rightarrow Q_{t+1} = Q_t + \frac{1}{t}[R_t - Q_t]$$

This leads to the update rule of the algorithm:

$$New\ estimate = Old\ esimate + Step\ size \times [Reward - Old\ estimate] \qquad (5.4)$$

For a stationary problem, meaning if the reward distributions for the actions are not time-varying, the step size decreases per step. This provides equal weightage to the rewards in each step. However, for a nonstationary situation, it is useful to provide more weightage to the recent rewards. This is achieved by making the step size constant, that is, $\alpha_n = \alpha$.

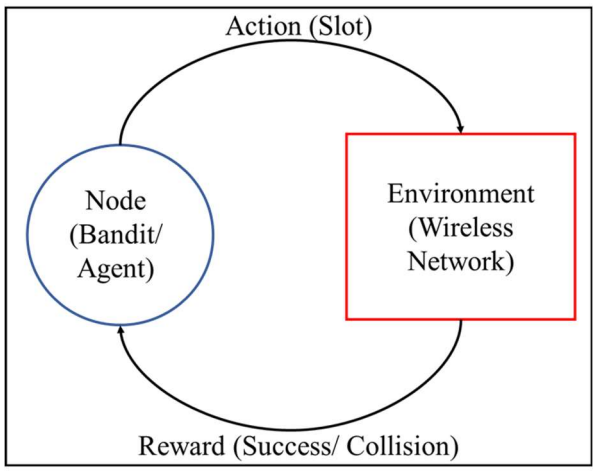

Figure 5.4: TDMA scheduling as a Multi-Armed Bandits Problem.

5.4.2 Formulating TDMA MAC scheduling as MAB

Transmission scheduling problem in this context boils down for each node to be able to choose a mini-slot at which the node can transmit in all subsequent frames without colliding with the

transmissions from the other network nodes. Such collision-free mini-slots should be selected locally at each node in a fully distributed manner, and that is without any centralized allocation entities and network time synchronization. The selection policy is modeled as a Multi-Armed Bandit problem. As shown in Fig. 5.4, each node acts as an 'F-armed bandit', where F is the frame size in number of mini-slots. Thus, the action of the bandit is to select a mini-slot, representing an arm, from an action pool of F mini-slots, which is preset based on network size/degree as explained in Section 5.3.

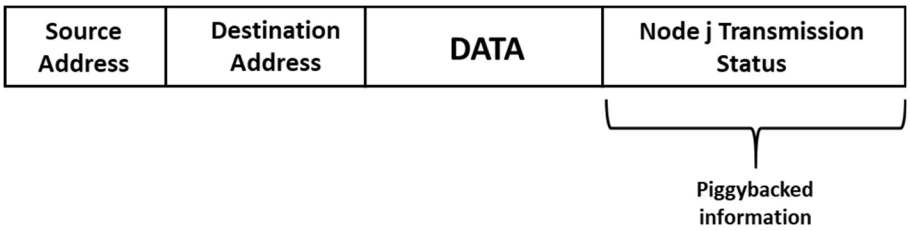

Figure 5.5: MAC layer PDU from node i.

The environment here is the wireless network with which the nodes/agents interact through their actions of choosing transmission mini-slots (i.e., the bandit arms). The reward associated with an action is formulated such that a node or an agent receives a penalty if it selects a mini-slot that overlaps with transmissions from other nodes, leading to collisions. Conversely, an action is rewarded for a collision-free transmission. The reward function for node i in decision epoch t is formulated as:

$$R_i(t) = \begin{cases} +1, & success \\ -1, & collision \end{cases} \tag{5.5}$$

As presented in Section 5.2, each node exchanges control information with its one-hop neighbors using an in-band piggybacking mechanism. Fig. 5.5 shows a MAC layer PDU from node i, destined to its one-hop neighbor j. Thus, the PDU of node i carries the information regarding the status of transmission (success/collision) from all its one-hop neighbors in the previous frame. In

this way, node j will know about its own transmission status from i. However, if packet from node i to j gets collided, node j may not know about its previous transmission, that may delay convergence. It is shown in the results in Section 5.5 how piggybacking relevant control information over MAC layer PDUs makes the proposed mechanism work in the absence of collision detection. From the transmission status information (success/collision) in the piggybacked packet, a node can compute the reward value from Eqn (5.5).

Using the actions and the reward function mentioned above, each learning agent (i.e., a node) learns a transmission policy to avoid collision in a distributed manner. A special concept of distributed Multi-Armed Bandits (MAB), called Hysteretic MAB, has been used as the learning framework for faster convergence. The update rule for Hysteretic Learning has been proposed for Reinforcement Learning context in [14] which has been leveraged here for applying it in Multi-Armed Bandit problem as a novel concept.

5.4.3 Hysteretic MAB

The concept of Hysteretic Q-Learning proposed in [14] is adapted here for MAB in order to apply it in a non-associative setting. With Hysteretic MAB, each node or bandit acts as an independent learning agent towards the goal of distributed learning of collision-free transmission mini-slots. In such a distributed and multi-agent setting, the reward received for an action in a decision epoch depends not only on the arm chosen by the agent itself, but also on the arms (i.e., transmission mini-slots) chosen by other nodes/agents. In other words, even if the agent chooses an arm that is rewarding, it may still get penalized if the arms chosen by the other agents turn out to be unfavorable. Hence, the idea used in Hysteretic MAB is to 'reward more and penalize less' for the agents' actions/arm selections. This is achieved by using two different learning rates in the MAB update rule (Eqn. 5.4). The value update equation for a Hysteretic MAB agent are:

$$\delta_t = r_t - Q_t(a)$$

$$Q_t(a) \leftarrow \begin{cases} Q_t(a) + \alpha \times \delta_t, & if\ \delta \geq 0 \\ Q_t(a) + \beta \times \delta_t, & else \end{cases} \tag{5.6}$$

The quantities $Q_t(a)\ and\ r_t$ are the values and reward for the a^{th} arm chosen in epoch t. There are two learning rates α and β, and the parameter δ controls which learning rate should be used in a particular epoch. Positive and negative values δ lead to the use of α and β as the learning rates respectively. The parameter δ is positive when the actions taken were beneficial for attaining the desired optimum of the system, and vice-versa. Hence, β is chosen such that it is always less than α in order to assign less importance to the penalties.

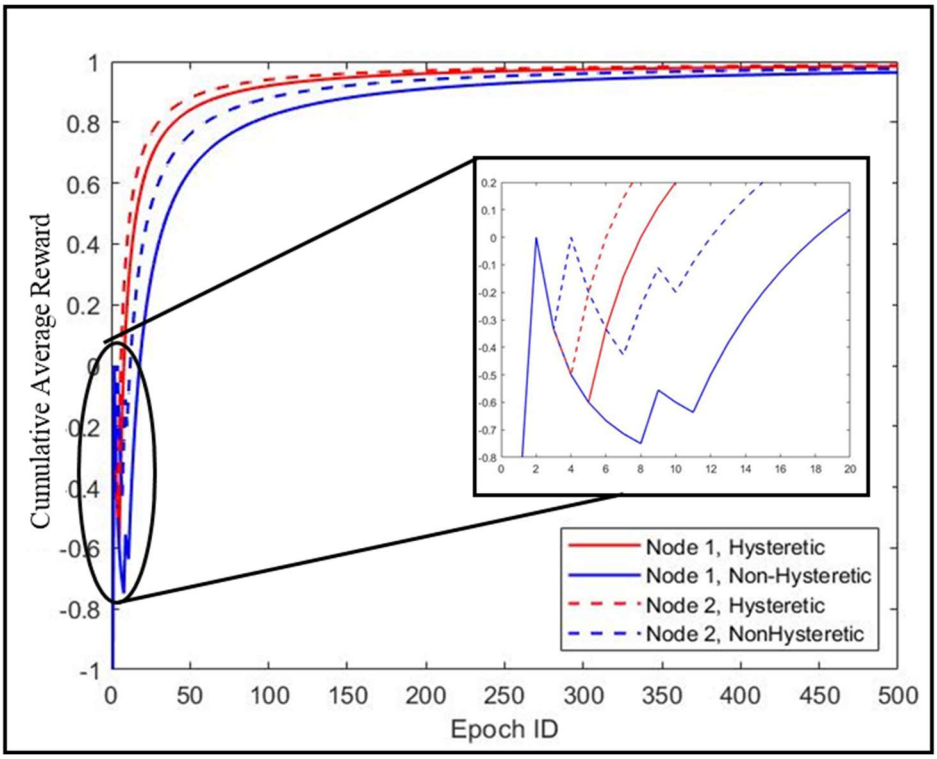

Figure 5.6: Cumulative average reward for two agents using Hysteretic and regular MAB.

The benefit of using two-learning rates in Hysteretic MAB can be observed from Fig. 5.6 which plots the cumulative average reward received by the agents in a two-node network setting. The action space and reward function used are the same as discussed earlier. It can be observed that

although the average rewards received using both hysteretic and non-hysteretic MAB converge to the maximum possible reward (i.e., +1), convergence is sooner for the hysteretic MAB as opposed to the non-hysteretic case. The reason behind this is that an agent's actions are severely penalized even for bad actions taken by the other agent in non-hysteretic update rule. Hysteretic MAB prevents the action value from going down because of penalties and thus accelerates the convergence speed.

5.5 Experiments and Results

The performance of the Hysteretic MAB-based protocol is evaluated for both fully connected and partially connected arbitrary mesh networks. The learning behavior of Hysteretic MAB is compared with the regular (non-Hysteretic) MAB. The results are also shown for how Hysteretic MAB performs in a time-synchronized network. The baseline experimental parameters, viz, rewarding learning rate, penalizing learning rate and packet duration per mini-slot, represented by α, β and M, take values of 0.99, 0.1 and 1 respectively.

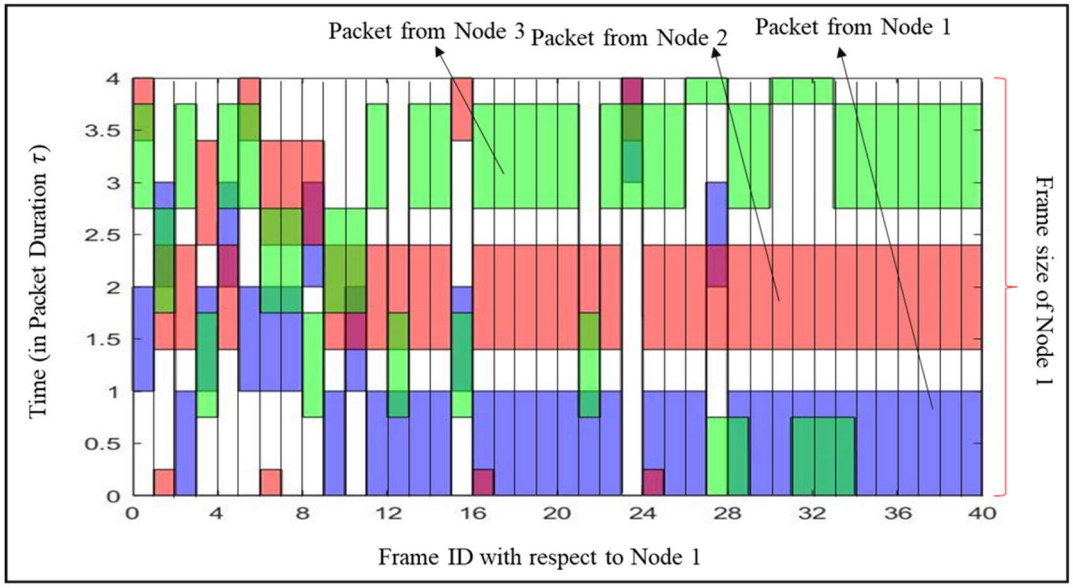

Figure 5.7: Convergence behavior of MAB in a three-node fully connected network.

5.5.1 Learning with Collision Detection Enabled

To understand the operations and to gain insights to the MAB-based learning paradigm, we first experiment with network nodes with collision detection ability. The learning-based framework is then validated in more practical scenarios without collision detection. Instead, a control information exchange mechanism is used via MAC layer PDU piggybacking.

1) Fully Connected Scenario:

The convergence of the proposed learning framework for a 3-nodes fully connected network with constant data rate $\lambda = 1$ packet/frame and frame scaling factor $K = \frac{4}{3}$ is shown in Fig. 5.7. Packet transmission by the nodes with node 1's frame as the frame of reference is plotted in the figure, where frames of nodes 2 and 3 lag the frame of node 1 by $0.4\,\tau$ and $0.75\,\tau$ respectively, where τ represents packet duration. It is observed that there are overlapped packet transmissions among

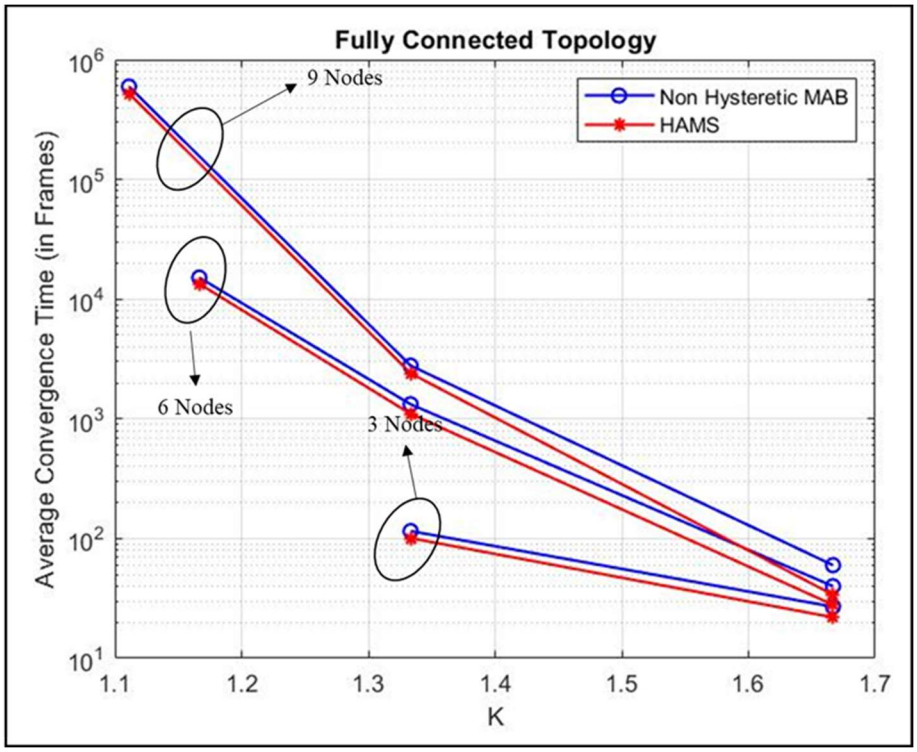

Figure 5.8: Average convergence of Hysteretic learning-based protocol for 3, 6 and 9 nodes fully connected networks.

the nodes initially. However, over time, all the nodes learn to pick transmission times in a distributed manner so that no overlapped transmissions take place, and hence collisions are avoided. Note that the learning here happens without network time synchronization.

In Fig. 5.8, average learning convergence time for different values of K are plotted for fully connected networks with 3, 6 and 9 nodes respectively. As discussed in Section 5.3, the frame scaling factor K represents the redundant bandwidth required in this mechanism when the network is not time synchronized. It can be observed that the convergence time decreases with increase in the value of K for all three networks. It is because, with increase in K, the number of feasible solutions of the MAB problem increases and hence the probability of finding a collision-free transmission strategy increases. Another observation is that for a fully connected network,

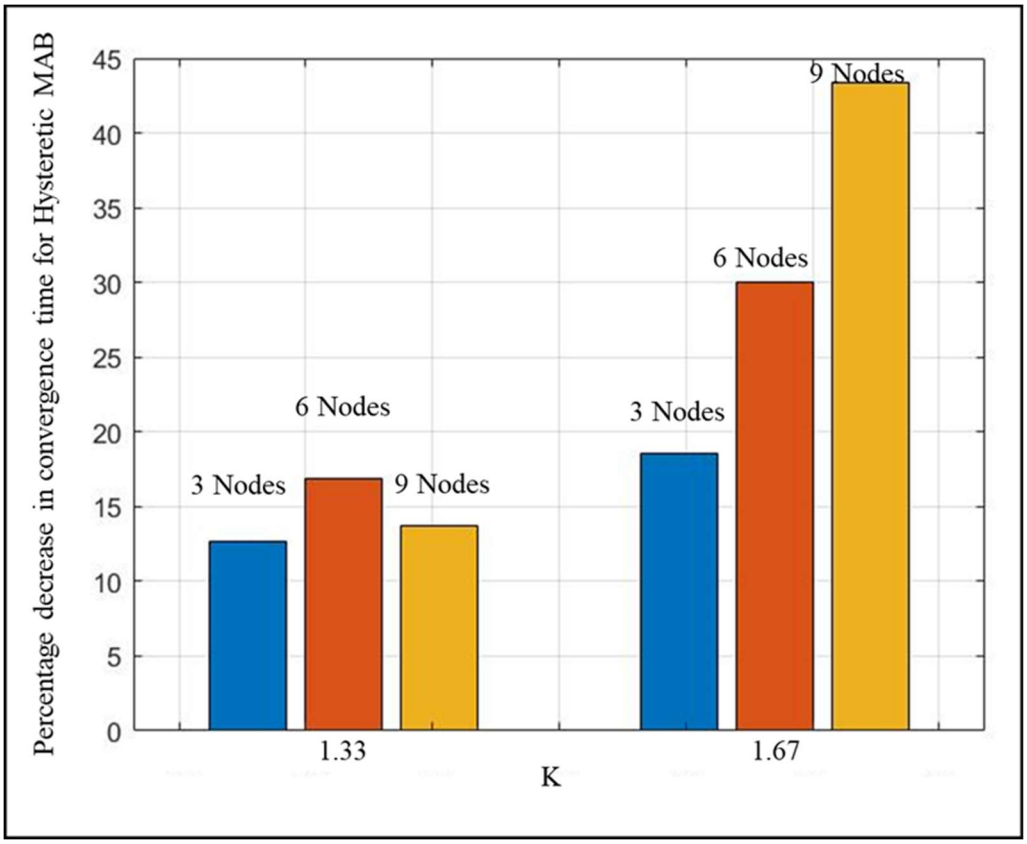

Figure 5.9: Convergence speed comparison of Hysteretic MAB with regular MAB for 3, 6 and 9 nodes fully connected networks.

convergence time increases with the network size. With increase in network size, the number of mini-slots per frame, and hence the number of actions available to each MAB agent (N_a) increases. The search space for the entire system, represented by $N_a \times N$, where N is the network size, increases non-linearly with the number of nodes. Thus, for a fixed number of feasible solutions, the probability of finding a solution by the nodes decreases with the increase in network size, and hence convergence time increases. However, it will be shown in the next section that for fully connected networks, convergence time is more dependent on network degree rather than the network size.

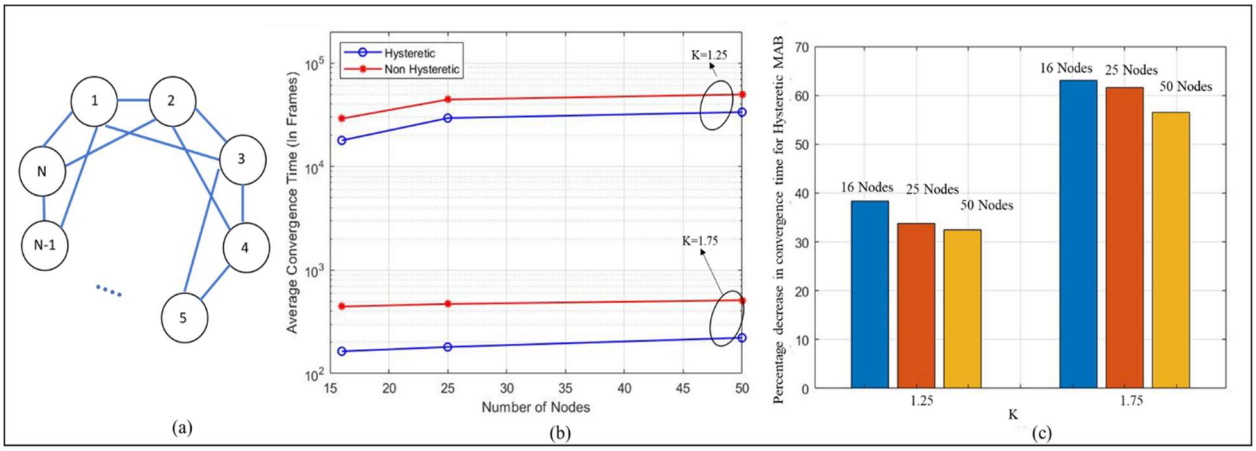

Figure 5.10: (a) Partially connected topology with degree 4, (b) Average convergence of the learning-based protocol for 16, 25 and 50 nodes partially connected networks, (c) Convergence speed comparison of Hysteretic MAB with regular MAB for partially connected networks.

For all three networks involving Fig. 5.8, convergence is faster for learning using Hysteretic MAB as compared to the non-Hysteretic learning. The percentage decrease in convergence duration using Hysteretic MAB for $K = 1.33$ and $K = 1.67$ is plotted in Fig. 5.9. It is observed that convergence speeds up by $13\% - 44\%$ when Hysteretic MAB is used. This is achieved by giving less importance to penalties than rewards in Hysteretic MAB as explained in Section 5.4.3.

2) Partially Connected Topology:

Fig. 5.10 (a) shows the average convergence time, plotted for different network sizes, all with nodal degree of 4. The value of $K = 1.25$ and 1.75, and $\lambda_i = 1$ pkt/frame , $\forall i$. The plots in Fig. 5.10 (b) demonstrate that the convergence time does not blow up with increase in network size as long as the nodal degree is kept constant, which makes learning scalable with network size. Another observation from Fig. 5.10 (b) is that faster convergence using Hysteretic MAB as compared to that of traditional MAB (non-Hysteretic) also holds for partially connected network topologies. The percentage reduction in convergence time using Hysteretic MAB in the networks in Fig. 5.10 (c). Convergence time decreases by $32 - 63\%$ in these topologies when Hysteretic MAB is used, as compared to traditional MAB. The reason behind this performance improvement is that the distributed learning agents are not severely penalized by the bad action selection by other agents, due to the use of two learning rates in Hysteretic MAB.

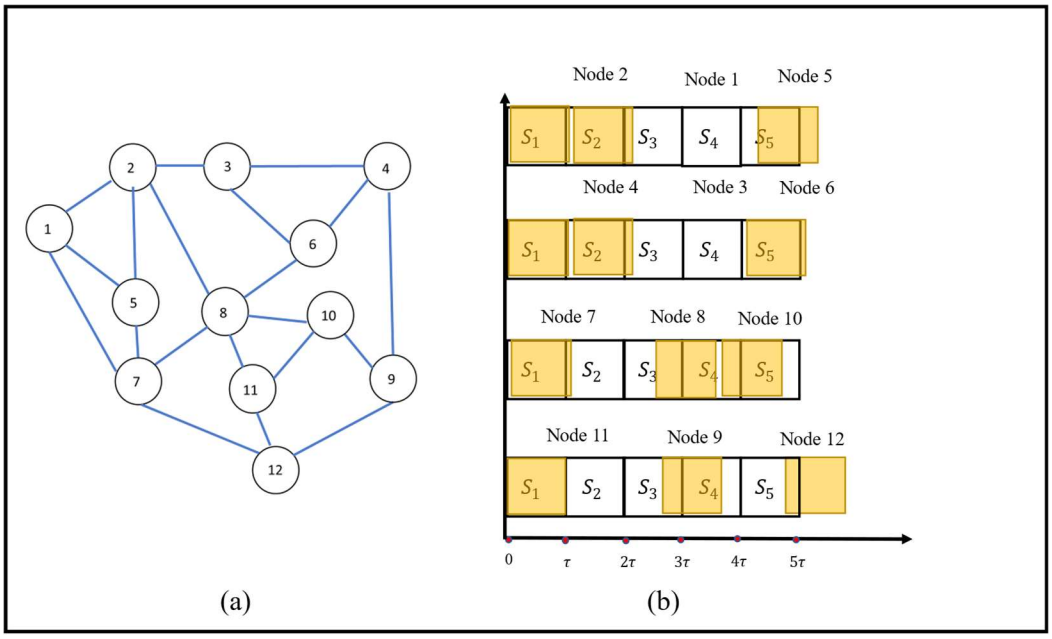

Figure 5.11: (a) 12-nodes arbitrary mesh network (b) Spatial reuse of channel in a partially connected network.

The MAB learning mechanism also allows the nodes to learn transmission schedules such that there is spatial reuse by the nodes that are more than one-hop distance away from one another. The learnt spatial reuse by the nodes in a 12-nodes arbitrary mesh network (Fig. 5.11 (a)) is demonstrated in Fig. 5.11 (b). The figure shows post-convergence packet transmission schedules. All the frames in Fig. 5.11 (b) are with reference to the frame of node 1. It can be observed that the nodes 2, 4, 7 and 11 learn overlapped transmissions allowed by their more than one hop topological separations. Similarly, other nodes that are more than one-hop distance apart (for e.g., 1, 3 and 10) learn overlapped transmissions exploiting spatial reuse. Spatial reuse is a key feature in slotted MAC scenarios for keeping the frame size small, thus keeping it scalable with network size.

5.5.2 In-band Information Exchange in the Absence of Collision Detection

In more practical network scenarios with nodes not capable of wireless collision detection, an in-band information exchange mechanism is used for exchanging overlapping transmission information. The technique of data packet piggybacking as explained in Section 5.4 is leveraged for such information exchange.

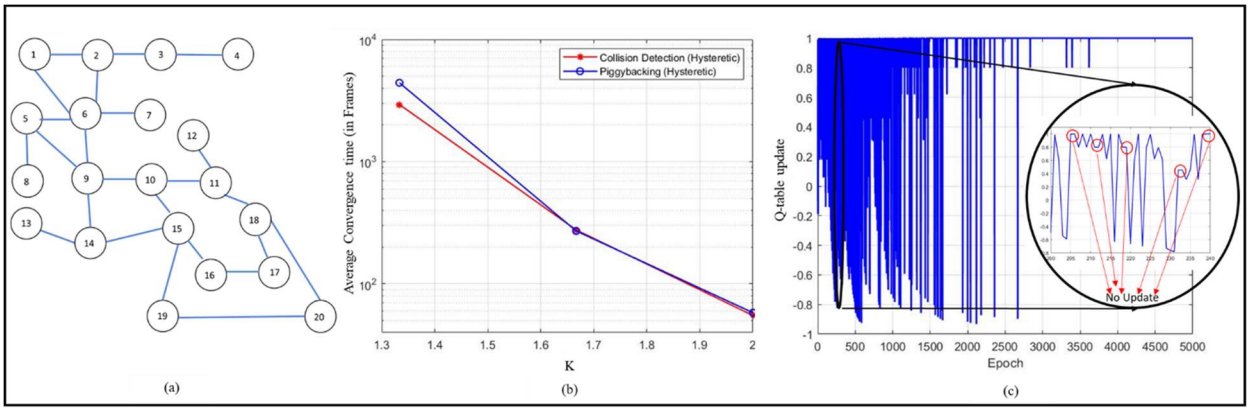

Figure 5.12: (a) Arbitrary mesh network with maximum degree 5, (b) Average convergence of Hysteretic MAB based protocol with piggybacking and collision detection ability, (c) Q-table updates when piggybacking mechanism is used.

Fig. 5.12 (b) presents the average convergence time using piggybacked information as compared to that in the presence of collision detection in ($\lambda_i = 1$ pkt per frame). Experiments are performed in an arbitrary mesh network shown in Fig. 5.12 (a). It is observed that for lower value of K (i.e, 1.33), the convergence time with piggybacking is higher as compared to convergence with collision detection. The reason for this behavior can be explained by the Q-table updates shown in Fig 5.12 (c). There are instances where there is no update in the Q-table as shown by the red circles in the figure. This is because when a packet destined to a node i from one of its one-hop neighbor gets collided, then node i does not get to know about its transmission status (success/collision) in its previous learning epoch, which was piggybacked in the collided packet. This delays the convergence for lower value of K. However, for larger K values, i.e., larger frames, the collisions in the network are less because of larger available bandwidth and hence the convergence speed of learning using piggybacking is close to that in the presence of collision detection ability.

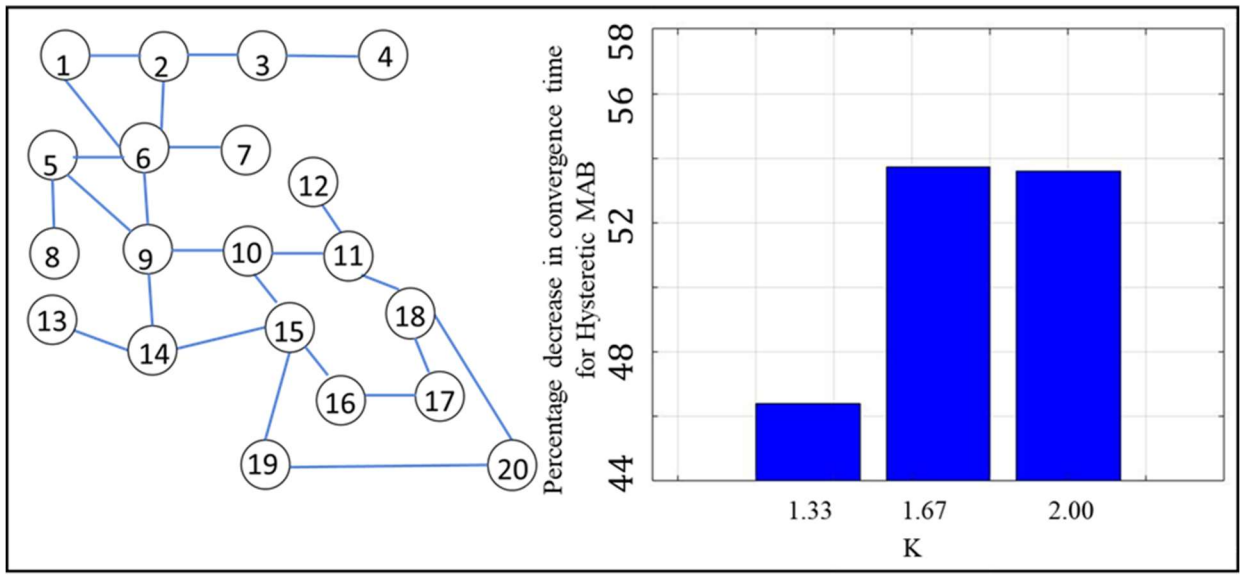

Figure 5.13: Convergence speed comparison of Hysteretic MAB with regular MAB for mesh networks.

As shown for the results in Section 5.5.1, expediting learning convergence using Hysteretic MAB also holds with piggybacking-based information sharing. It is shown in Fig. 5.13 that the percentage reductions in convergence time using Hysteretic MAB are between $46\%\ to\ 54\%$ when the values of frame scaling factor K is chosen to be 1.33, 1.67 and 2.

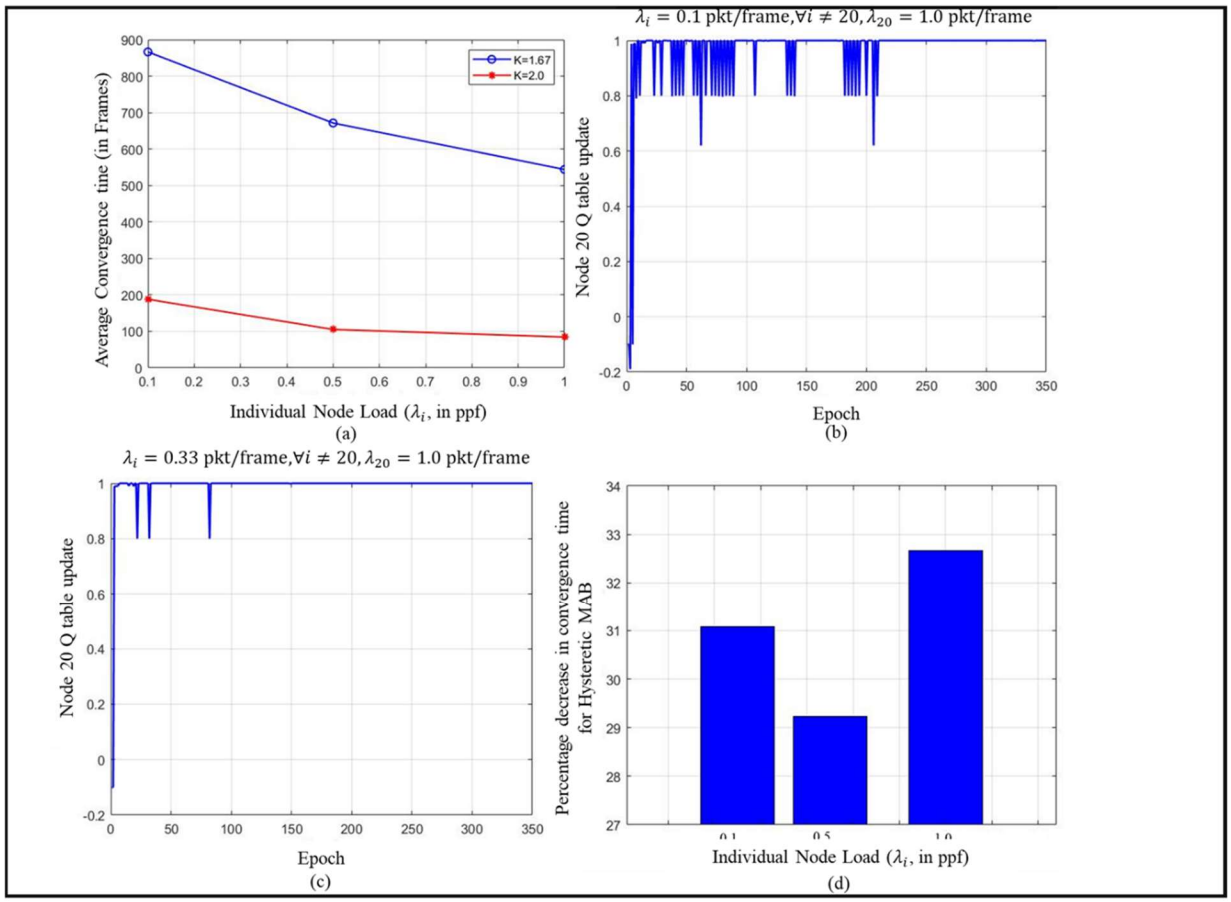

Figure 5.14: (a) Convergence time variation with network load, (b) Q-table updates for node 20 with $\lambda_i = 0.1$ pkt/frame,$\forall i \neq 20,\ \lambda_{20} = 1.0$ pkt/frame, (c) Q-table updates for node 20 with $\lambda_i = 0.33$ pkt/frame,$\forall i \neq 20, \lambda_{20} = 1.0$ pkt/frame, (d) Performance comparison of Hysteretic MAB with regular MAB for varying traffic.

Fig. 5.14 (a) depicts the impacts of network load on learning convergence for a network with the mesh topology shown in Fig. 5.12 (a). It can be observed that for both the values of frame scaling factor $K = 1.67$ and $K = 2$, the convergence time decreases with increase in the network traffic. This can be explained using the Q-table update plot for node 20 as shown in Fig. 5.14 (b). This

is shown for a scenario where the packet generation load in node 20 is $\lambda_{20} = 1$ pkt/frame, and at all other nodes $\lambda_i = 0.1$ pkt/frame,$\forall i \neq 20$. The figure shows that the Q-values for node 20 converges to 1.0 within just 20 learning epochs, but there are many intermittent downward spikes after that. These spikes are the result of node-20's transmissions colliding with those from other nodes. It is evident that node 20 is unable to learn a fully collision free transmission time because the piggybacked information is not readily available since not enough packets are being transmitted by those nodes. With increase in network traffic (i.e., $\lambda_i = 0.33$ pkt/frame,$\forall i \neq 20$), the occurrence of these downwards spikes drastically reduces, and as a result, as depicted in Fig. 5.14 (c), learning convergence is expedited.

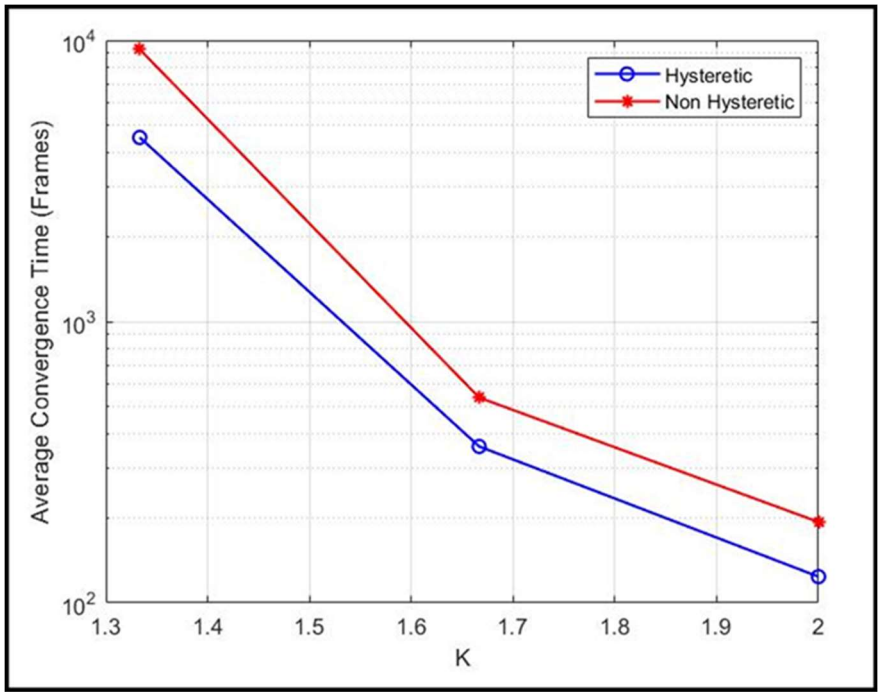

Figure 5.15: Average convergence of the proposed learning framework for Poisson distributed traffic ($\lambda_i = 1.0\ pkt\ per\ frame$).

To be noted that the Hysteretic MAB can expedite learning convergence compared to the regular MAB even under low network traffic. The percentage reduction of convergence time in different

traffic conditions for K=2 is shown in Fig. 5.14 (b). A reduction in convergence time by 31%-33% is achieved by introducing the Hysteretic features to the traditional MAB.

All the results presented so far are for constant packet rate traffic. Fig. 5.15 depicts the benefits of Hysteretic MAB over classical MAB when the network traffic is Poisson distributed. The figure shows that the benefits of Hysteretic MAB, as described in Section 5.4.3, holds for Poisson distributed traffic as well.

5.5.3 Performance in Time Synchronous Networks

Finally, the learning algorithm is implemented in networks with time synchronization in order to understand the effectiveness of Hysteretic MAB under time synchronization. Fig. 5.16 (a) shows the performance for fully connected networks with 3, 6 and 8 nodes. Fig. 5.16 (b) shows the performance for the arbitrary mesh network in Fig. 5.12 (a). For both, the value of the frame scaling factor K is set to 1. For both cases, it can be seen that Hysteretic learning outperforms the regular MAB by speeding up convergence up to $9\% - 56\%$.

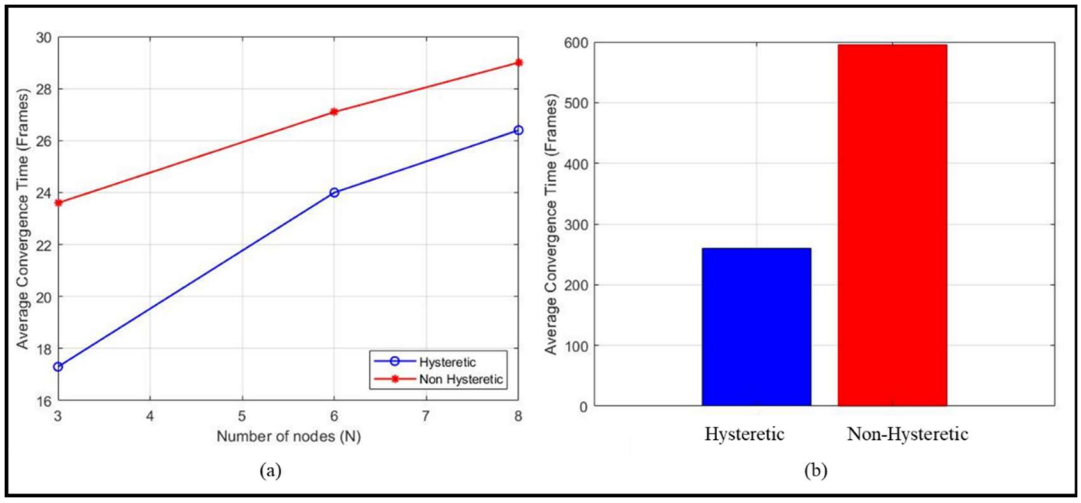

Figure 5.16: Performance comparison of Hysteretic MAB with regular MAB for time-synchronized networks (a) Fully connected, (b) 20-nodes Arbitrary mesh network (Fig. 5.12 (a)).

5.6 Summary

In this chapter, we have demonstrated the protocol synthesis concept for networks with TDMA-based MAC arrangements. A decentralized MAB-based learning framework for performing collision-free MAC scheduling in the absence of time synchronization is proposed. Transmission slot allocation policy in wireless MAC has been modeled as a distributed Multi-Armed Bandits Problem, where the nodes behaving as the learning agents, learn transmission policies independently such that there is no collision post convergence. A novel mechanism for distributed Multi-Armed Bandits, termed Hysteretic MAB, has been introduced and leveraged to accelerate learning convergence. Performance of the proposed mechanism is validated by simulations on both fully connected and heterogeneous arbitrarily connected mesh networks. The nodes learn to spatially reuse wireless bandwidth in partially connected topologies, and the developed system is scalable with network size and topological diversities. Although the learning framework is mainly designed for low-complexity networks without time synchronization capability, it is shown that the Hysteretic MAB-based approach outperforms the regular MAB-based slot allocation even in time-synchronous TDMA systems.

It follows from the discussion in this chapter that to assign collision-free transmission mini-slots in networks without time synchronization capability, the minimum frame length should be at least one mini-slot more than the absolute minimum frame size. In other words, the frame-scaling factor (K) should be greater than 1. This leads to a certain amount of bandwidth redundancy because of the extra mini-slot in the frame without any packet transmission. Higher the value of K, higher is the bandwidth redundancy in time-asynchronous network. Moreover, it follows from the discussion in this chapter that there is a tradeoff between learning convergence time and bandwidth redundancy. This calls for the requirement of a mechanism that can (a) reduce the

bandwidth redundancy required by the framework for networks without time synchronization, (b) handle the trade-off between learning convergence speed and bandwidth redundancy. In the next chapter, we build on this MAB-based slot allocation scheme to develop a framework that can improve bandwidth usage efficiency and manage convergence speed and bandwidth redundancy trade-off.

Chapter 6: Adaptive Slot Defragmentation for Bandwidth Efficiency

6.1 Motivation

It transpired generally from the discussion in chapter 5 that a large frame scaling factor K leads to faster convergence across all scenarios. However, it comes with a price of additional bandwidth overhead. In this chapter, a decentralized slot defragmentation mechanism is introduced to reduce such capacity overhead. The goal of this defragmented backshift mechanism is to reduce the wastage of bandwidth because of large frame scaling factor K used to speed up convergence. This step is performed by all the nodes independently after they have found a suitable mini-slot for transmission post MAB convergence (Fig. 6.1).

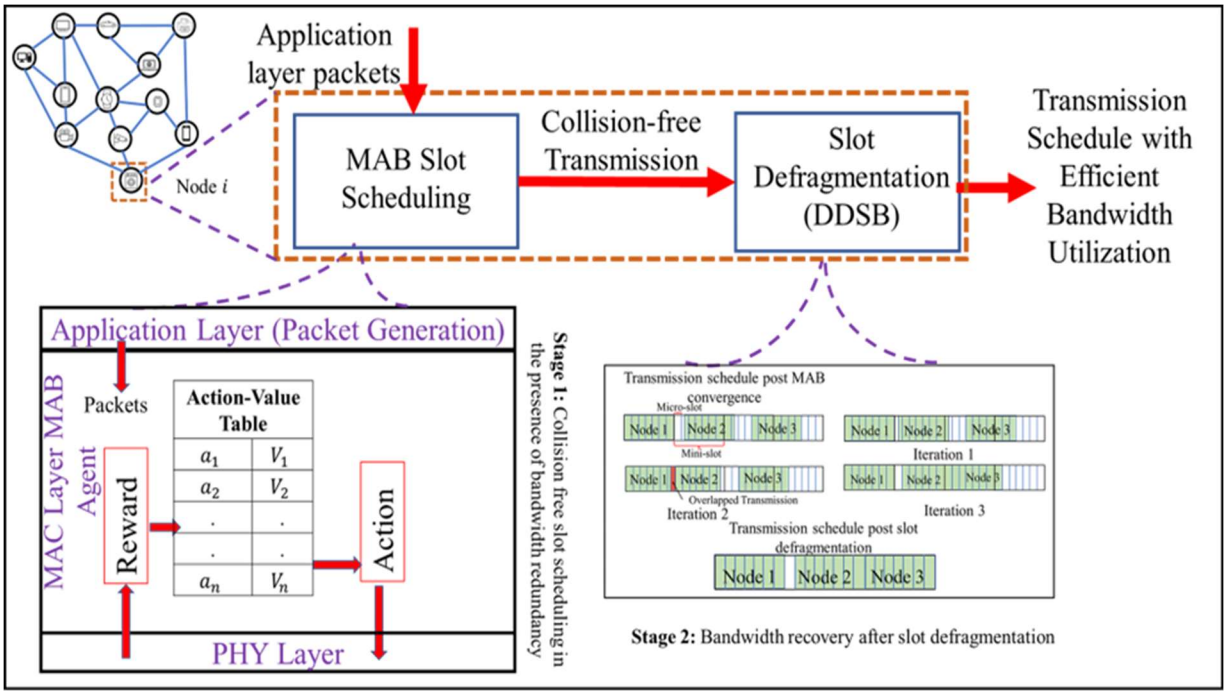

Figure 6.1: High-level working model of the proposed scheduling framework with efficient bandwidth usage.

6.2 Decentralized Defragmented Slot Backshift (DDSB)

This concept is implemented by discretizing each mini-slot (refer to chapter 5) within a frame into 's' number of micro-slots. After MAB convergence, each node shifts its packet transmission by one micro-slot back in time till it experiences a collision. Once a node experiences a collision it undoes its previous action to find its new transmission micro-slot. In this way, the nodes make an estimate of the unused space in the frame and try to reduce it in a decentralized manner. This mechanism of defragmentation is explained using Fig. 6.2 for a 3-nodes fully connected network. The figure shows how the frame structure (with reference to node 1) evolves over 5 iterations of defragmentation mechanism for $K = 1.33$ and $s = 7$. In this figure, node 1 does not shift its transmission since it is transmitting at the beginning of the frame. Nodes 2 and 3 backshift their transmissions by one micro-slot per iteration. In iteration 2, nodes 1 and 2 experience a collision and hence node 2 undoes its previous action by shifting by one-micro-slot forward in iteration 3. But node 1 does nothing in iteration 3, since it experienced a collision without any micro-slot

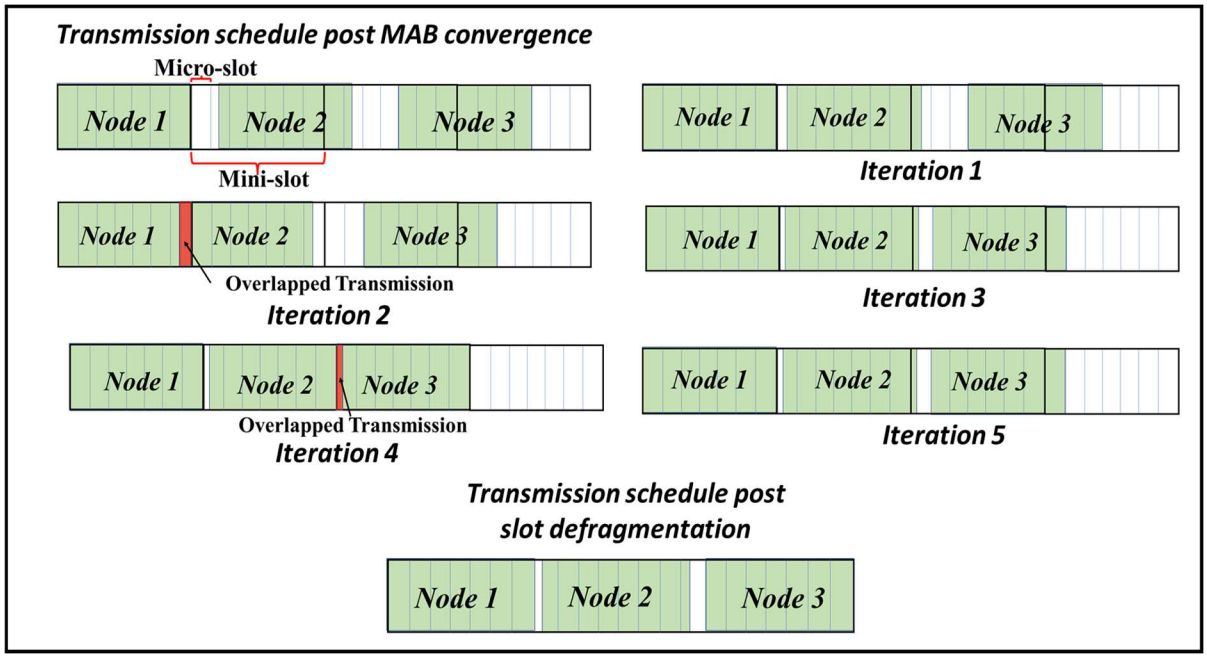

Figure 6.2: Defragmented backshift operation in a 3-nodes fully connected network.

Initialize: $\mu_{shift_i} = 0, c_i = 0$ // μ_{shift_i}: Number of micro-slot that node i has shifted; c_i: Status of the micro-slot search (1, $if\ search\ is\ complete$, $else$, 0)
If (! Tx in the beginning of frame), do:
    Shift to previous micro-slot
    $\mu_{shift_i} + +$
    Check Collision
    If (Collision ==TRUE):
        Check action in the previous frame $a(t-1)$
        If ($\mu_i\ (t) > \mu_i(t-1)$):
            Shift to next micro-slot
            Check Collision
            If (Collision ==TRUE):
                Shift to previous micro-slot
            End If
        Else If ($\mu_i(t) < \mu_i\ (t-1)$):
            Shift to next micro-slot
        End If
        Set $c_i = 1$
        Piggyback c_i, μ_{shift_i}
        Check the value of c_j, $\forall j \in one-hop\ neighbor$
        If ($c_j == 1\ (\forall j \in one-hop\ neighbor)$)
            Find new frame size:
            $F_{shru}\ \ (t) = \max\left\{\mu_{shift_i}(t), \mu_{shift_j}(t)\right\}, \quad j \in one-hop\ neighbors\ of\ i$
            If ($F_{shrunk}\ (t) == F_{shru}\ \ (t-1)$):
                $Frame\ Size \leftarrow\ Frame\ Size - F_{shrunk}$
                $\mu_i\ (t) =\ \mu_i\ (t-1) - F_{shrun}$
                Ignore all collisions
            Else:
                Continue
            End If
    Else:
        Do Nothing
        Set $c_i = 1$
        Piggyback c_i, μ_{shift_i}
        Check the value of c_j, $\forall j \in one-hop\ neighbor$
        If $c_j == 1\ (\forall j \in one-hop\ neighbor)$
            Find new frame size:
            $F_{shrunk}(t) = \max\left\{\mu_{shift_i}(t), \mu_{shift_j}(t)\right\}, \ j \in one-hop\ neighbors\ of\ i$
            $Frame\ Size \leftarrow\ Frame\ Size - F_{shrunk}$
            $\mu_i\ (t) =\ \mu_i\ (t-1) - F_{shrun}$
        End If
    End If

Algorithm 6. 1. Slot Defragmentation

frame size from the μ_{shift} values from its shift in its previous frame. Similarly, packets from nodes 2 and 3 collide in iteration 4 because of backshift operation of node 3. Node 3 shifts forward its transmission by one micro-slot and it knows that it has found its suitable micro-slot for transmission. In this example, the new frame size as shown in the figure reduces by 21% as a result of slot defragmentation mechanism. To be noted that the bandwidth redundancy left after slot defragmentation is because of the time lag existing among the nodes as a result of the absence of network time synchronization.

Once a node finds a stable micro-slot, it piggybacks control information to all its one-hop neighbors indicating that it is stable. In addition, each node also piggybacks the information indicating the number of micro-slots it has shifted (μ_{shif}) to find its stable position. Thus, a node knows that its one-hop neighbors have found the stable micro-slots and it computes the new

$$F_{shrunk} = \max\{\mu_{shift}(i), \mu_{shift}(j)\},$$
$$j \in \text{one} - \text{hop neighbors of } i \tag{6.1}$$
$$New\ Frame\ Size = Old\ Frame\ Size - F_{shrun}$$

The pseudo code logic for slot defragmentation executed by each node i is given in Algorithm 6.1.

6.3 Experiments and Results

Experiments are performed for both fully connected and partially connected networks to validate the concept. Fig. 6.3 (a) demonstrates the effect of slot defragmentation mechanism on 9 nodes fully connected network with $K = 1.67$. The number on each packet in the figure indicates the transmitter of the packet. The figure shows the transmission schedules of the nodes in three

stages. First is the transmission schedule after MAB convergence. This is the time when defragmentation operation begins. It is observed that there are significant amount of time gap among the packet transmissions in a frame because of high value of K. The second plot in the figure is an intermediate stage when the nodes individually have found a stable micro-slot by defragmentation mechanism. The last plot shows the final stage when the nodes have computed the new frame size using Eqn. 6.1 and have started transmitting according to the new frame size. The bandwidth redundancy as a result of this operation reduces from 67.7% to 3.3% and this is

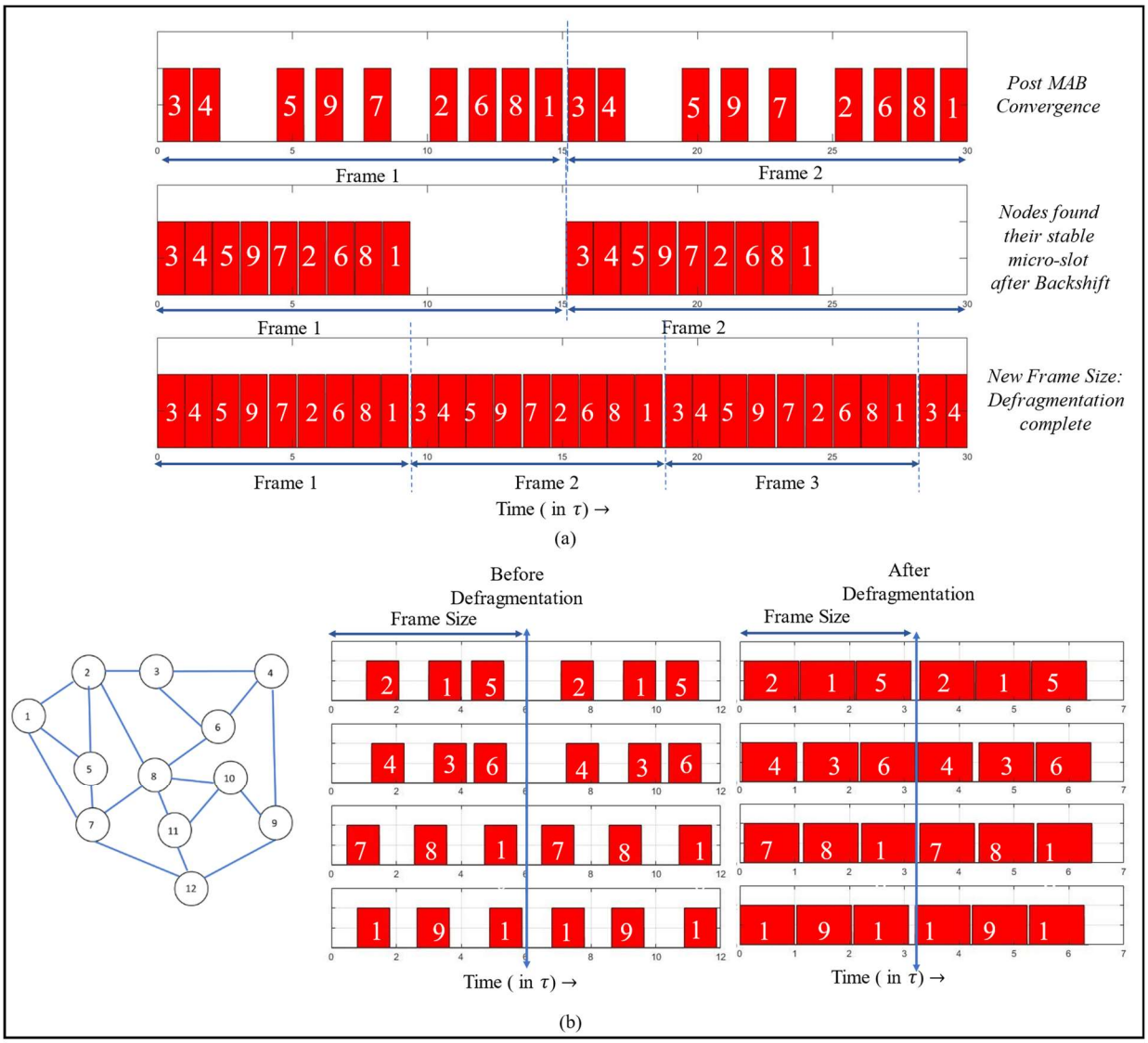

Figure 6.3: Slot Defragmentation in (a) 9-nodes fully connected (b) 12-nodes partially connected network.

achieved over 25 iterations, where each iteration corresponds to a frame duration. To be noted that the bandwidth redundancy of 3.3% at the end of defragmentation is because of the existing temporal lag between the frames because of time asynchronization and can be reduced by increasing the discretization of micro-slots. But increase in micro-slot granularity would reflect in increasing defragmentation convergence time. This excess bandwidth goes to zero in a time synchronous network, post defragmentation.

Similar observation can be visualized on experimentation of this framework in a partially connected topology. The reduction in bandwidth in a 12-nodes partially connected topology for $K = 2$ is shown in Fig. 6.3 (b). The excess bandwidth after defragmentation reduced from 100% to 7.11% in that topology.

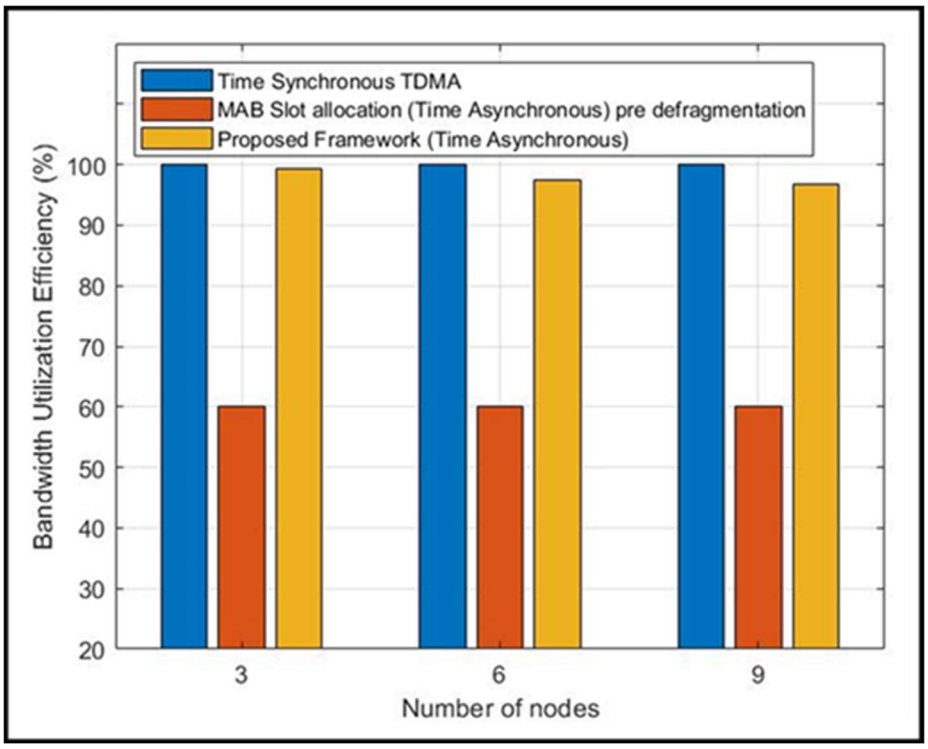

Figure 6.4: Bandwidth utilization efficiency by the proposed framework.

To understand the efficiency of the proposed framework, the performance of the allocation

mechanism in terms of bandwidth utilization efficiency has been compared with the ideal scenario in the presence of time synchronization. Bandwidth utilization has been compared as $U = \frac{D}{T} \times 100\%$, where D is the actual duration within a frame of duration T that is used for packet transmission. This has been plotted for three fully connected networks of size 3, 6 and 9 nodes (with bandwidth redundancy factor K=1.67) in Fig. 6.4 for three different scenarios: time synchronous TDMA (benchmark), time asynchronous slot allocation by MAB and proposed framework (MAB and defragmentation) without time synchronization. It can be seen that the proposed mechanism achieves bandwidth utilization efficiency in the range of 97-99% that is close to the time synchronous TDMA. The figure also shows the significance of the defragmentation technique post-MAB convergence that improves the bandwidth utilization efficiency by 37-39%.

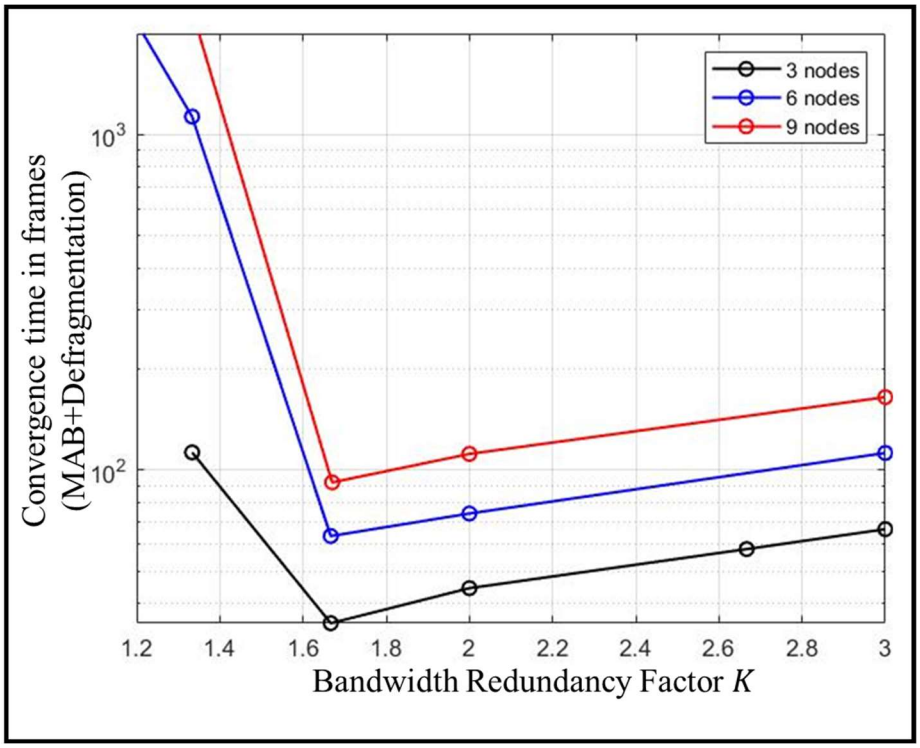

Figure 6.5: Convergence time variation with K for fully connected networks.

Fig. 6.5 depicts the additive time for stage-1 MAB convergence and stage-2 defragmentation convergence. Larger K values speed up MAB convergence while slowing down the defragmentation process. The latter is because with a larger frame length, the number of iterations that a node has to backshift its transmission micro-slot to find a suitable micro-slot increases. Thus, the search space to find the suitable transmission micro-slot increases with K. As can be seen in Fig. 6.5, the total convergence duration (MAB and slot defragmentation) initially goes down with increase in K, reaches a minimum, and then goes up again. This is because for small K, MAB convergence time is significantly higher than defragmentation convergence and hence the total convergence is largely affected by the MAB learning convergence. However, for larger K values, defragmentation convergence time overpowers MAB convergence time, and thus, total convergence time increases with K. These results indicate that an optimum value of K exists that gives the minimum total convergence time of the proposed learning framework.

6.4 Summary

This chapter proposes a decentralized slot defragmentation approach for handling the trade-off between bandwidth efficiency and convergence time of learning for TDMA slot allocation in the absence of time synchronization. This framework aims to solve the limitations of the Multi-Armed Bandits slot selection policies detailed in chapter 5 in terms of the bandwidth usage inefficiency in resource constrained networks. The developed slot defragmentation operation, in conjunction with MAB-enabled slot scheduling, improves bandwidth utilization while guaranteeing collision free transmissions in the absence of network time synchronization. This decentralized mechanism also manages the convergence-bandwidth redundancy trade-off by algorithmic reduction of excess bandwidth required for faster MAB convergence.

The discussion till this point in this thesis is centered around reducing collisions (in random

access schemes) and collision-free slot allocation (in TDMA-based access schemes), so that network throughput and bandwidth usage can be maximized. While doing so, other access performance parameters, such as, energy efficiency, packet delay are ignored. However, for resource constrained networks, judicious energy management is important, along with maximizing other network performance parameters. In the next chapter, a learning-based framework is developed as an extension of the MAB-based slot allocation logic in order to improve energy efficiency and network throughput while minimizing access delay and packet losses.

Chapter 7: Protocol Synthesis for Flow and Energy Management using multi-tier Learning

Thus far in this thesis, we have been interested in developing learning-driven MAC protocols mainly focusing on improving network throughput. While doing so, we have not considered the energy constraints of the sensor nodes. However, resource-constrained sensor networks typically operate with limited energy reserves. Efficient energy management in such networks is crucial for enhancing network lifetime, without compromising on performance. Building on the Multi-Armed Bandit-based slot allocation mechanism developed in chapter 5, we extend the study to develop a multi-tier learning framework for flow and energy management in resource-constrained wireless networks.

This chapter presents a learning-enabled framework for MAC sleep-listen-transmit scheduling in wireless networks. The developed paradigm is shown to work in the absence of network time-synchronization and other complex hardware features, such as carrier-sensing, thus making it suitable for low-cost transceivers for IoT and wireless sensor nodes. The framework allows wireless nodes to learn policies that can support throughput-sustainable flows while minimizing node energy expenditure and sleep-induced packet drops and delays. Each node independently learns a scheduling policy without explicit communication with other network nodes. The trade-off between packet drops and energy efficiency is analyzed, and an application-specific solution is proposed for handling the trade-off.

7.1 Motivation

Efficient scheduling for turning wireless transceivers on and off is an effective energy management mechanism in embedded networks of resource-constrained sensors and IoTs. In

traditional approaches [22, 23], such schedules are typically pre-programmed in nodes based on manual settings and heuristics that are designed based on specific target network and traffic scenarios. As a result, such schedules cannot adapt well to network and traffic dynamics and time varying heterogeneity. Shortcomings are usually manifested in terms of not being able to maintain the desired balance between network performance (i.e., throughput and delay) and energy efficiency. The proposed framework in this chapter overcomes these by learning appropriate sleep-listen-transmit policies and by adapting them in the presence of heterogeneous network and traffic conditions.

Learning in this framework happens in three distinct stages. In the first stage [27], using Multi-Armed Bandits (MAB) learning, nodes independently learn to select collision-free transmission slots. This learning module is equivalent to the one developed in chapter 5. The second stage uses Reinforcement Learning (RL) in order to learn whether to transmit or to sleep in the transmission slot selected in stage-I. This learning happens in a per-flow context in which multiple data flows may exist within each network node. The objective of the learning agent in this tier is to save energy by judicious sleeping, while keeping end-to-end packet delay low. Finally, in stage-III, another Multi-Armed Bandit agent learns whether to sleep or remain awake during a slot that was decided to be a non-transmitting slot by the MAB agent in stage-I. This agent ensures that the node remains awake only on the slots it is intended to receive packets from one of its neighbors, thus minimizing the idling energy expenditure. The objective of the coordinated learning by these three agents is to maximize per flow throughput while keeping energy expenditure minimum via judicious sleep-wake decisions. In addition to being adaptive to network and traffic heterogeneity, this decentralized learning architecture is able to handle the trade-offs between throughput and energy efficiency based on any application-specific

requirements.

7.2 Network and Traffic Model

The network model used in this work is a generalized network with arbitrary configurations of end-to-end flows in mesh topologies. Fig. 7.1 shows an example of the network model where there exists two flows originating from source nodes S_1 and S_2 and destinations D_1 and D_2 respectively. It is assumed that after the flows are set up at the routing layer, each node is aware of all the flow that passes through it, including the originating ones.

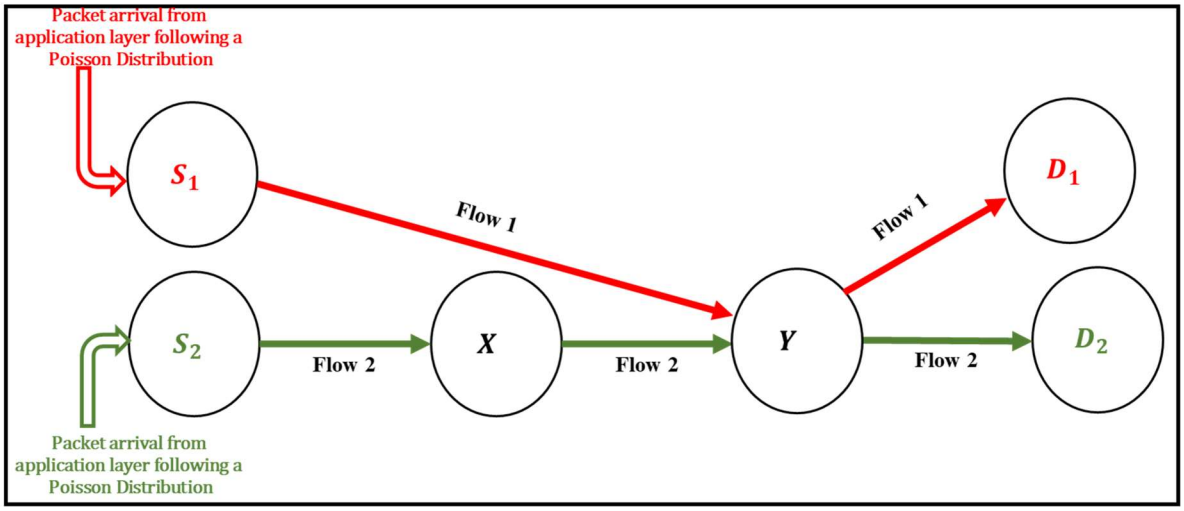

Figure 7.1: Network and Traffic Model.

The packet generation process at a flow source node follows a Poisson distribution with mean λ, repressing the flow rate. The maximum flow rate is kept equal to $\frac{1 \text{ packet per flow}}{\text{Flow}_{\max}}$, where $\text{Flow}_{\max}$ represents the maximum number of flows passings thorough a node. The core MAC arrangement of the framework is TDMA, which exclusively works with fixed packet size [87] . Each node maintains an M/D/1/∞ buffer/queue for each flow, where the Poisson distributed queue arrival rate is the flow rate, and the queue service rate is determined by the actuated per-flow transmission policy.

Access in a node is slotted but are not necessarily synchronized across different nodes. The proposed learning mechanism is shown to work both in the presence and absence of network time synchronization. This is a crucial feature since MAC slot allocation and sleep scheduling in the absence of time synchronization is in general a challenging problem. Each node maintains a hierarchy of learning agents that first learns to find collision-free transmission slots, and then determines transmit/sleep/listen policies so as to strike a balance between per-flow throughput and nodal energy efficiency.

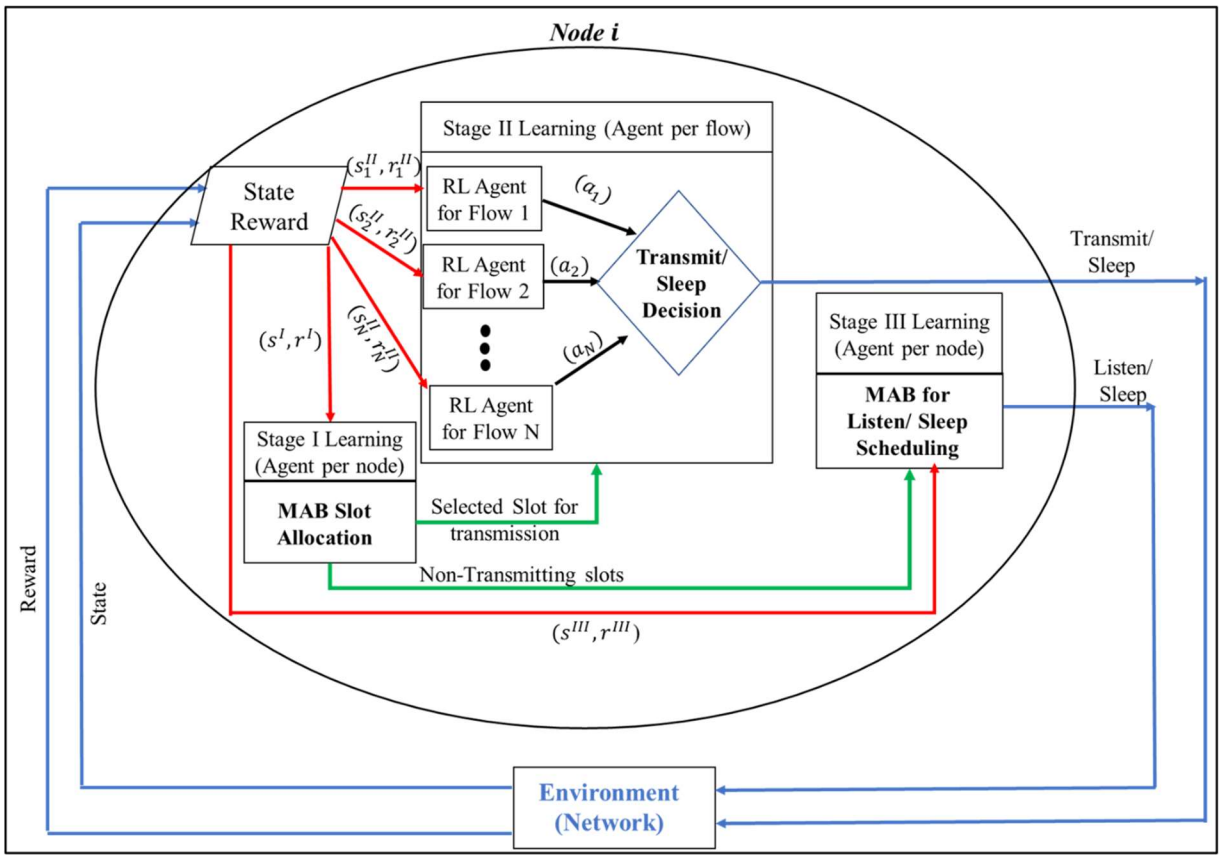

Figure 7.2: Different Stages of the learning modules from an individual node's perspective.

7.3 Flow-Specific Wireless Access and Sleep-Awake Scheduling using Reinforcement Learning

The holistic goal of the proposed framework is to make the wireless nodes learn a transmit/listen/sleep policy that can support flow-specific throughputs while minimizing node-

level energy consumption. This is achieved by a 3-stage learning framework, which is shown from an individual node's perspective is shown in Fig. 7.2. In stage-I, per-node Multi-Armed Bandits based learning is used to make the nodes independently learn to select a collision-free transmission slot. In stage-II, per-flow Reinforcement Learning is used for deciding whether to transmit or sleep in the parent node's transmission slot that was learnt in stage-I. Finally, a per-node Multi-Armed Bandit agent learns an efficient sleep-listen schedule on the transmission slots of its 1-hop neighbors in order to minimize the energy wastage due to idle-awake slots. And it is done while ensuring minimum number of missed packet receptions due to oversleeping. The functional details of each of these learning modules are presented below.

7.3.1 TDMA Slot Self-selection Using Node-specific Multi-Armed Bandit

MAC slot allocation in this context refers to each node being able to self-select a collision-free slot within a network-wide agreed upon frame structure. Such slots are selected by each node using a local and independent learning process without the need for any centralized arbitration. The selection policy is modeled as a Multi-Armed Bandit (MAB) problem, where each node acts as an 'F-armed bandit,' where F is the frame size in number of slots (refer to chapter 5). The action of the bandit is to select a slot, representing an arm, from an action pool of F slots. A slot is of one packet duration, and it is globally preset based on the network degree. The MAB environment here is the wireless network with which the nodes/agents interact through their actions of selecting transmission slots (i.e., the bandit arms). The reward associated with an action is formulated such that a node/agent receives a reward or penalty depending on if the selected slot is collision-free or not. The reward function for node i in decision epoch t is formulated as:

$$R_i(t) = \begin{cases} +1, & \text{success} \\ -1, & \text{collision} \end{cases} \tag{7.1}$$

Using the actions and this reward function, the MAB learning agent (i.e., the first stage of learning in Fig. 7.2) in each node learns to self-select a collision-free slot using the MAB update Eqn. (7.2), where $V_t(a)$ denotes the value of the arm a; α_t, r_t are the learning rate and the reward received at instant t.

$$V_{t+1}(a) = V_t(a) + \alpha_t \times [r_t - V_t(a)] \tag{7.2}$$

Note that during this stage of learning, the sleep-awake schedulers (Stage II and Stage III) are inactive. In other words, the node is always on. Whenever there is a packet in the queue, it transmits. Similarly, the node is always on to receive packets from its one-hop neighbors.

7.3.2 Transmission Decisions using Flow-specific RL based Learning

Once a node selects a collision-free transmission slot using MAB, its next course of action is to decide whether it should transmit or sleep in that slot in each frame. The goal here for a node is to find an efficient transmit/sleep schedule for each of its flows such that the data rates for the flows are supported while maximizing transceiver sleeping for energy conservation. Such behavior is learnt by deploying a Reinforcement Learning (RL) agent (i.e., stage-II learning in Fig. 7.2) for each flow.

In this framework, there is an RL agent associated with each flow through a node and the environment is the wireless network itself with which the agent interacts via actions. The modeling of state-action space and the reward function for the RL-agent in this work is explained below.

RL-based Sleep-Transmission Scheduling: A learning decision epoch for each flow-specific RL agent is set to m frame durations, which is a hyper-parameter to be explored. Each frame comprises of F slots, which is globally preset based on network degree. Each agent's action is a vector of length m in which each element represents either a *Transmit* or a *Sleep* as action. Thus,

the action space $\mathcal{A}$ is of size 2^m. The environment state space for an agent is determined by the congestion level as perceived by the agent. The state is coded by the change in the length of the queue dedicated for the flow over the decision epoch of m frames. Formally stated, the state is $\Delta Ql(t) = Ql(t) - Ql(t-1)$, which is the change in the length of the queue for the i^{th} flow in the t^{th} epoch. In order to keep the state space discrete, the changes in queue length are quantized into l uniform discrete intervals, where l is a hyperparameter to be explored empirically. The state space for the RL agent of flow i is of size l and can be denoted as $\hat{S}^i = \{\hat{s}_1^i, \hat{s}_2^i, \dots\dots, \hat{s}_l^i\}$.

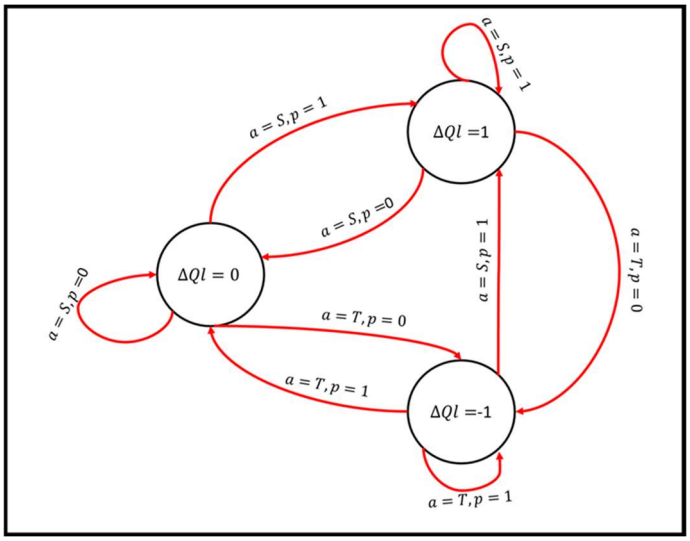

Figure 7.3: State transition for a flow as a result of its agent's action.

An example of state transition for an RL agent associated with a specific flow as a result of its actions is shown in Fig. 7.3. Depending on the action taken by the RL agent, its state may change in terms of the changed length of the flow-queue corresponding to the agent. For simplicity in the figure, we have considered $m = 1$, and it is assumed that p packets arrive in a decision epoch of m frames. Thus, there are two possible actions: {Sleep (S) and Transmit (T)}. Based on the action selected by the agent and the number of packets received, the queue length changes, and this results in a state transition which is governed by the change in queue length (ΔQl).

The action taken by an agent i is evaluated based on a numerical reward received after each

decision epoch, which is computed from the reward function shown in Eqn. (7.3).

$$R_i(t) = -\rho \times \Delta Ql_i^*(t) \times S(t) \tag{7.3}$$

Here the term $\Delta Ql_i^*(t)$ denotes the sample average of change in queue lengths for the action a_t at the t^{th} epoch. The negative sign indicates that a positive gradient in the queue length is to be penalized and vice versa. This is because, an increasing queue length for the queue related to the flow i means that the agent associated with i has decided to sleep on the transmission slot of its parent node more than the required duration, leading to an unstable queue and subsequent packet delay.

For a non-positive gradient of queue length (i.e., throughput of the flow is sustained by needed transmissions), the RL-agent associated with the flow should try to minimize the awake time of the parent node of the flow for saving energy. On the other hand, for an increasing queue length, the flow-agent should try to increase the awake time. This is captured by the term $S(t)$, which defines the number of slots in which the node sleeps in the t^{th} epoch. A higher number of sleeping slots is beneficial for energy saving and hence multiplying $S(t)$ in the reward reinforces the agent to sleep more for a negative value of $\Delta Ql_i^*(t)$ in order to save energy. A scaling factor $\rho = \rho_{min} \times e^{t/\rho_r}$ is multiplied with the reward function which gives more importance to the recent samples of reward as compared to the older ones.

Using the RL model discussed above, each flow-agent learns a transmit/sleep policy that can support the data rate of the flow while minimizing the energy consumption. In other words, the agent associated with the flow i learns a suitable service rate of the queue for flow i in order to achieve the above-mentioned objectives. Learning is accomplished using a tabular Q-Learning [13, 88]. An agent (associated with each flow) interacts with the environment (the wireless

network) repeatedly to learn the best possible set of actions (transmit/sleep decisions) to maximize the long-term expected reward. For all state-action pairs, the agent maintains Q values which are updated using Eqn. (7.4). Here, $Q(s,a)$ represents the Q value for the current state-action pair (s,a); r is the reward received and the hyperparameters α and γ represent learning rate and discount factor, respectively.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r(s,a) + \gamma \times \max_{\forall a' \in A} Q(s,a') - Q(s,a)\right] \tag{7.4}$$

To be noted that since reward captures the state in terms of ΔQl (meaning, the value of the state is already captured in the reward), so in the update equation, the maximum of the Q-values of the current state ($\max_{\forall a' \in A} Q(s,a')$) is considered. This means, whatever the next state is, if the current state is suitable for the system, the q-table for that state is updated with a low discounted reward. Still, the immediate reward is based on the action taken in that state. This is done to speed up the learning convergence, as found empirically.

The Q-values are initialized to zero to make sure that equal importance is given to all the actions. Initial random action selection is implemented by ϵ- greedy action selection policy. As the Q-values are adaptively updated, the agent prefers the action with the highest Q value for a specific state with the goal of maximizing long-term reward.

After learning convergence, the actions of a flow-agent may remain oscillatory because of non-uniqueness of the possible solutions. But this is not desirable for the sleep/listen scheduling of the downstream node in the flow, as it will not be able to learn from the aperiodic packet arrival intervals in the presence of such oscillations. In order to overcome this, the agent picks the action with one of the highest Q-values (i.e., in the presence of multiple highests) and sticks to it for the lifetime of the flow. This is achieved by computing the moving variance ($MV(\mu', w)$) of the

estimated service rate (over m frames) of the queue over a sliding window of w epochs. If $MV(\mu', w) < \epsilon_\mu$, the action $a^* = argmax_{\forall a}\{Q(s,a)\}$ is chosen for post-convergence epochs. The details on hyperparameters $\rho_{min}, \rho_r, w, \epsilon_\mu$ are given in Section 7.4.

Inter-flow Transmission Conflict Coordination: Each flow-level learning agent in stage-II operates independently in order to learn whether to sleep or transmit in its parent node's transmission slot that was selected using MAB learning in Stage-I. Since there can be multiple flows through a node, multiple such agents may learn to transmit in that slot, thus leading to inter-flow transmission conflicts. Such conflicts can be handled using a node-wise transmit/sleep decision maker (shown in Fig. 7.2) that assigns the transmission slot to one of those conflicting flows either in a round robin manner or randomly (uniformly). Selecting flows using a uniform distribution, in the long run produces similar long-term performance like round robin policy. Moreover, the rewards for RL are computed over a decision epoch long enough (controlled by parameter m) to handle any short-term variability arising from random flow-coordination.

7.3.3 Node-specific Multi-Arm Bandits for sleep-awake learning in non-transmission slots

A node-specific MAB learning agent decides whether to sleep or to listen on the node's non-transmission slots. The objective is to stay awake during an appropriate subset of those slots in order to ensure reception of neighbor nodes' transmitted packets that are intended for the node. And to save energy during the remaining subset of those slots when no packets are transmitted for the node. This is the stage-III learning as shown in Fig. 7.2. The MAB agent in this stage has two possible arms or actions, namely, Sleep (S) and Listen (L), in each learning epoch which is set to one slot. In other words, on each non-transmission slot in a frame, the node-specific agent

decides whether to sleep or listen. This action is evaluated using a numerical reward which is computed from the reward function for agent i at epoch t given in Eqn. (7.5).

$$R_{i,s}(t) = \begin{cases} +1, & for\ successful\ packet\ received\ at\ t \\ 0, & for\ awaking\ up\ idle\ at\ t \\ \frac{I_t(s)}{W_t(s)}, & for\ sleeping\ at\ t \end{cases} \tag{7.5}$$

Here $I_t(s)$ represents the number of instances the node remained awake without receiving any packet in slot s, till the t^{th} epoch from the start of learning. $W_t(s)$ is the total number of instances the node remained awake in slot s, till the t^{th} epoch from the start. Thus, the quantity $\frac{I_t(s)}{W_t(s)}$ represents the average duration of idle listening in slot s, that is the duration for which the node remained awake although there was no packet for which the node i was the intended receiver. The physical intuition of the reward in Eqn. (7.5) can be understood as follows. The action of a node remaining awake in a slot s in an epoch t (i.e., $a_t = L$) is rewarded if there is a successful packet reception in that epoch. That is, the node i remained awake in a slot s, and there was a packet transmission from a one-hop neighbor of i for which i is the intended receiver. The node-level agent i receives no reward for remaining awake in a slot s, if there is no packet intended to it in that slot. This is because it leads to unnecessary energy consumption due to remaining awake without receiving any packet. Moreover, the action of sleeping on a slot s (i.e., $a_t = S$) is rewarded based on the estimate of the average idle-listening period ($\frac{I_t(s)}{W_t(s)}$) in that slot. In other words, higher the value of $\frac{I_t(s)}{W_t(s)}$, higher is the probability that the node was awake in a slot s without receiving any packet. Hence, higher is the benefit of sleeping in that slot.

The working of the MAB learning model can be explained analytically as follows. From the

definition of the value of an arm in Multi-Armed Bandits [12], we can obtain the value of the arm '*Listen (L)*' for slot s as follows.

$$V_s(t|a=L) = V_s(t-1|a=L) + \alpha_t \times [R_{i,s}(t-1|a=L) - V_s(t-1|a=L)]$$

This is equivalent to

$$V_s(t|a=L) = \alpha_t \times [R_{i,s}(t|a=L) + R_{i,s}(t-1|a=L) + R_{i,s}(t-2|a=L) + \cdots . + R_{i,s}(0|a=L)]$$

Now choosing an adaptive learning rate $\alpha_t = \frac{1}{W(s)}$, we obtain:

$$V_s(t|a=L) = \frac{1}{W(s)} \times [R_{i,s}(t|a=L) + R_{i,s}(t-1|a=L) + R_{i,s}(t-2|a=L) + \cdots . + R_{i,s}(0|a=L)]$$

From the reward definition in Eqn. (7.5), $R_{i,s}(t|a=L)$ can take values 1 or 0, depending on whether it received a packet or not. This means, as $t \to \infty$:

$$\sum_{k=0}^{t} R_{i,s}(k|a=L) \to W(s) \times \lambda_s$$

$$\Rightarrow V_s(t \to \infty|a=L) \to \lambda_s$$

where λ_s denotes the rate of the flow associated with slot s. Similarly, the value of the arm '*Sleep (S)*' converges to

$$V_s(t \to \infty|a=S) \to (1-\lambda_s)$$

Now, at time instant t, an arm a is chosen for slot s with probability

$$P_s(a_t=a) = \begin{cases} V_s(t|a_t=a)\left(1 + \dfrac{1-V_s(t|a_t=a)}{V_s(t|a_t=a)}\delta_p\right), a = L \\ V_s(t|a_t=a)\left(1 - \dfrac{1-V_s(t|a_t=a)}{V_s(t|a_t=a)}\delta_p\right), a = S \end{cases} \quad (7.6)$$

Here δ_p $(0 \le \delta_p \le 1)$ is a tunable priority coefficient, an increase in which prioritizes reduced

missed packet reception over increasing energy efficiency. At $\delta_p = 1$, the missed reception rate is reduced to zero. In this way, a node learns an efficient sleep-listen scheduling policy with an aim of striking a right balance between missed packet receptions and energy savings based on application-specific requirements.

Using the learning frameworks detailed above, each node learns a transmission/sleep/listen policy that provides a data-rate sustainable flow support for all the flows through it, minimizes sleep-induced missed packet receptions, and maximizes energy savings. This learnt behavior gives rise to the proposed Energy-Efficient Sleep Scheduling-MAC (ESS-MAC). To be noted that in Stages-I and III, the actions are not dependent on the state of the environment. For Stage I, the actions are evaluated based on whether the transmission was successful. For Stage III, the actions are evaluated based on whether the node was awake in correct reception slots. However, in stage II, the transmission decisions are situation dependent, which justifies the use of Reinforcement Learning in that stage. In other words, for a high queue length, it should transmit more even if the goal is to minimize energy consumption. Whereas it is the reverse for a lower value of queue length.

7.4 Experiments and Results

Experiments are performed to analyze the performance of the proposed ESS-MAC protocol using a MAC layer simulator with embedded learning components. The simulation kernel performs event scheduling in terms of packet generation, transmissions, and receptions. To implement the proposed ESS-MAC protocol, the Reinforcement Learning and Multi-Armed Bandits update equations are embedded on top of the MAC layer functions. The baseline experimental

parameters are tabulated in Table 7.1. The post- convergence performance of the learning-based MAC protocol is evaluated based on the following performance metrics.

Missed Reception Rate (P_{miss}): This is defined as the average number of packets missed by the receiver in a frame because of oversleeping, which is computed as: $P_{miss} = \frac{|R_x \cap S|}{|R_x|}$, where R_x is the set of slots in a frame where it is supposed to receive packet and *S* is the set of slots in a frame where the node actually sleeps. It captures the intersection of those two, and it needs to be minimized.

Listening Energy Efficiency (η_L): This takes into account the number of correct sleeps in a frame that leads to energy saving and also the bad sleeps that leads to missed packet receptions. This is computed as $\eta_L = \frac{|\overline{R_x} \cap S| - |S - (\overline{R_x} \cap S)|}{|\overline{R_x}|}$ and needs to be maximized. In this equation, the expression $|\overline{R_x} \cap S|$ and $|S - (\overline{R_x} \cap S)|$ capture the number of good and bad sleeps (i.e., oversleeping) in a frame respectively.

Table 7.1: Baseline Experimental Parameters

Parameter	**Value**
m	10
ρ_{min}	0.3
ρ_r	1000
α	0.3
γ	0.9
w	100
ϵ_μ	0.01
l	9

Throughput (s_i): Throughput of a node i is computed as the average number of packets in a frame successfully received by the intended receiver. This is computed as $s_i = \lambda_i \times (1 - P_{miss})$. Here λ_i is the packet generation rate of node i's application layer.

In addition, performance is evaluated in terms of Idle Listen Probability and Off-Probability that denote the average number of slots a node remains idle and transceiver off in a frame respectively. Performance of the flow controlling agent is evaluated on the basis of end-to-end delay in a flow and also the transmission energy efficiency (η_{tx}) computed by the average queue utilization ratio ($\eta_{tx} = \frac{\lambda}{\text{Average } \mu}$). For a given rate λ, energy efficiency is high for a low service rate μ, as the node has to remain awake for low duration. However, the service rate should be always higher than the flow rate λ, otherwise the queue length will not be stable and hence delay will go up.

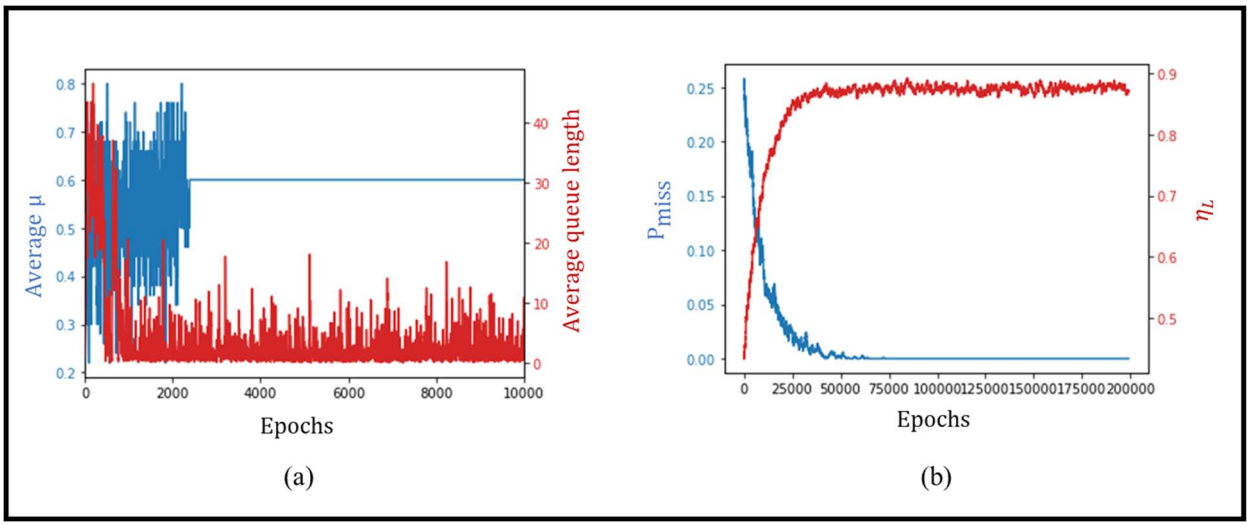

Figure 7.4: Convergence behavior of (a) Flow-controlling RL agent, (b) Sleep-Listen Scheduling MAB agent.

7.4.1 Performance Analysis on a Single Flow Network

In order to test feasibility and gain insights of the proposed mechanism, performance of the developed framework is first studied in a simple scenario of only a single flow in a 2-node network. Node-1 is a source in which packets are generated using Poisson-distribution with mean rate $\lambda = 0.5$. Packet transmissions are done based on a learnt transmit/sleep policy (i.e., access protocol) as described in Section 0. Node-2 is the receiver that learns a sleep-listen policy (i.e.,

as proposed in Section 7.3.3) in order to maximize energy efficiency while minimizing packet drops due to oversleeping. Fig. 7.4 shows the convergence behavior of both the learning agents, viz, the flow-specific transmit-sleep scheduling RL-based learning agent in stage-II and the sleep-listen scheduling MAB-based agent in stage-III (Fig. 7.2).

Fig. 7.4 (a) plots the convergence behavior in terms of average μ and average queue length for the flow. It can be observed that after learning converges, average μ settles down at 0.6. This is because, the epoch size $m = 10$ frames which means the possible values of μ have a step size of 0.1. Since the packet arrival rate $\lambda = 0.5$ and for a stable queue $\mu > \lambda$ is a requirement, the minimum feasible value for the queue service rate μ should be 0.6. To be noted that although $\mu > 0.6$ would allow the queue to be stable it would lead to unnecessary awake duration, thus affecting energy efficiency. Similarly, Fig. 7.4 (b) shows the convergence behavior of the sleep-scheduling agent in terms of sleep-induced missed reception rate (P_{miss}) and energy efficiency (η_L). This is for the packet miss reduction parameter $\delta_p = 1$ and it is observed that as learning

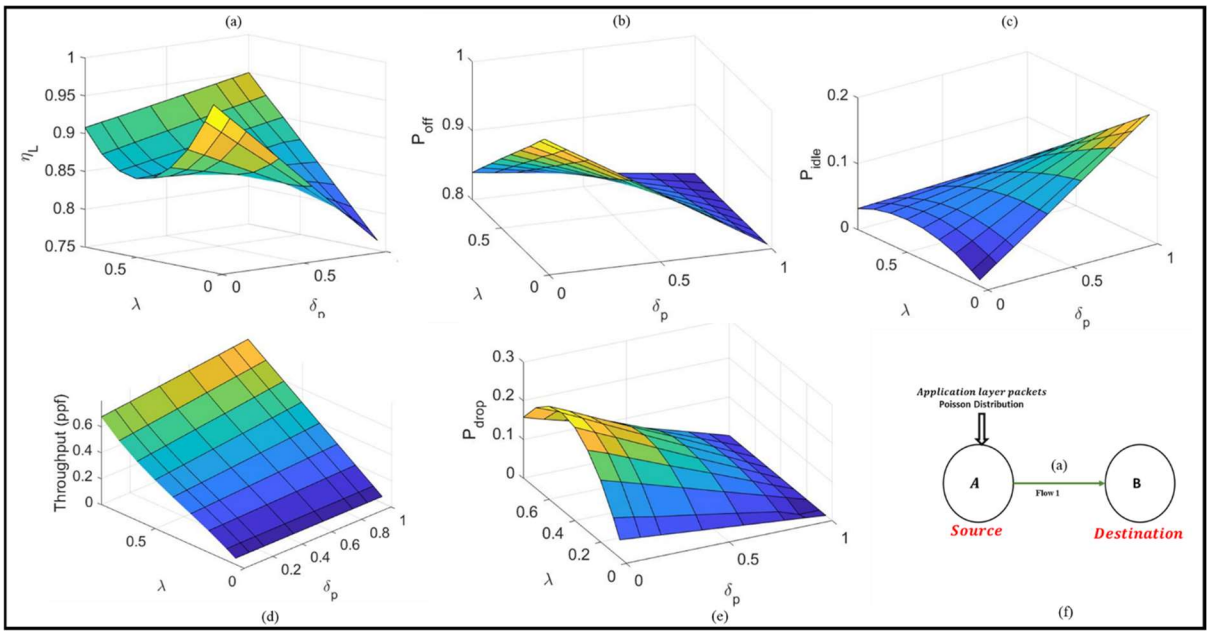

Figure 7.5: Performance analysis of ESS-MAC in a single flow scenario.

progresses, missed reception rate keeps on reducing and efficiency increases. Missed reception rate due to bad sleeps goes to zero after learning convergence. It is to be noted that stage-III learning depends on stage II learning and as can be seen from Fig. 7.4, stage III learning convergence time is higher than Stage-II learning. In other words, if a bad action is taken in the initial learning epochs of stage II, this leads to poor action selection by the MAB agent in stage III. This makes the learning in stage III dependent on stage II.

The effects of throughput priority coefficient δ_p and flow rate λ on performance are demonstrated in Fig. 7.5. The following observations can be made. First, for lower values of flow rate λ, efficiency (η_L) goes down with increase in δ_p, whereas, for higher λ, efficiency increases with increase in δ_p (Fig. 7.5 (a)). This can be explained using Fig. 7.5 (b), that shows that the node 'off probability' (P_{Off}) decreases with increase in packet drop reduction priority factor (δ_p) and the node learns to remain Off with high probability for lower flow rate, so that it can save energy when the packet arrival is sparse. Hence, when the flow rate is low, low δ_p yields higher efficiency. Another observation from Fig. 7.5 (c) is that the idle awake probability (P_{idle}) is inversely related to efficiency. Meaning, P_{idle} increases with increase in δ_p for lower λ, whereas, for higher λ, P_{idle} decreases with increase in δ_p. This is because of the fact that it is beneficial to sleep more if the flow rate is low and lower δ_p leads to lower idle-awake and vice-versa. Fig. 7.5 (d), (e) show the variation of throughput and missed reception rates (P_{miss}) with flow rate λ and priority coefficient δ_p. The first observation from these figures is that average missed reception probability resulting from bad sleep decisions goes down with increase in δ_p, because the node remains on with high probability for higher δ_p. To be noted that for any flow rate, P_{miss} goes to zero for $\delta_p = 1$. As a consequence, throughput increases with increase in δ_p and reaches the

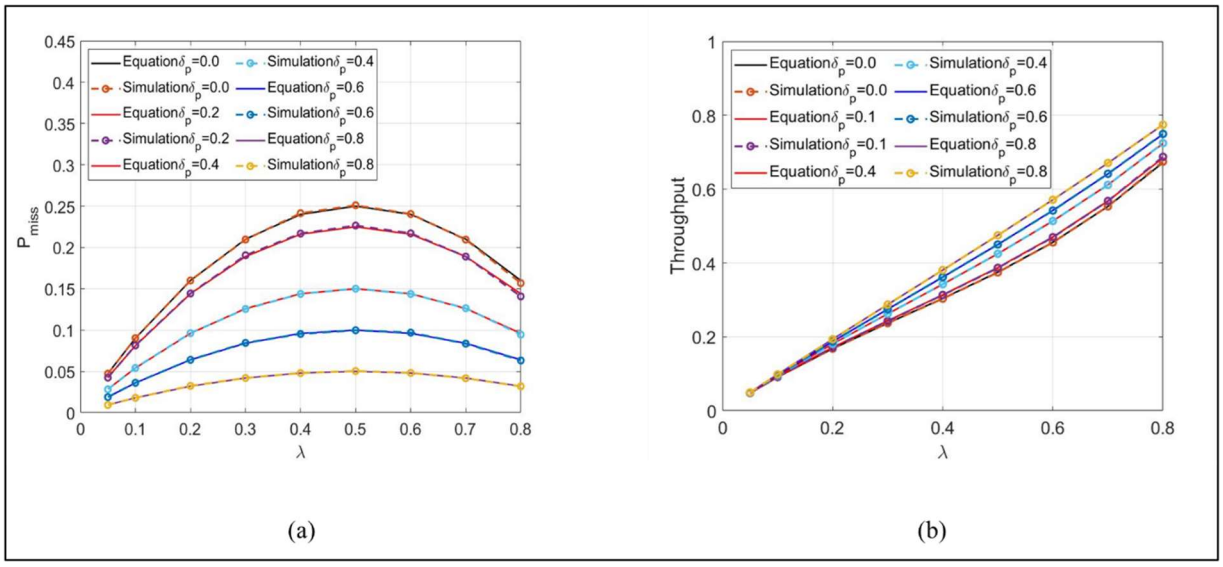

Figure 7.6: Comparison of analytical model for sleep-induced packet drops and throughput with simulations.

maximum possible value of λ for $\delta_p = 1$. Another observation is that sleep-induced missed receptions increase with λ and then decrease for $\delta_p \neq 1$. This behavior can be explained mathematically as follows.

$$P_{miss} = P_{off} \times \lambda$$

$$= (1 - P_{on}) \times \lambda$$

$$= (1 - V(a_t = L) \times \left(1 + \frac{1 - V(a_t = L)}{V(a_t = L)} \delta_p\right)) \times \lambda$$

After convergence, $V(a_t = L) \rightarrow \lambda$. This leads to:

$$P_{miss} = (1 - \lambda \times \left(1 + \frac{1 - \lambda}{\lambda} \delta_p\right)) \times \lambda \tag{7.7}$$

$$\text{Throughput}, s = \lambda \times (1 - P_{miss})$$

$$= \lambda \times (1 - (1 - \lambda \times \left(1 + \frac{1 - \lambda}{\lambda} \delta_p\right)) \times \lambda)$$

This analytical model is validated using simulations from the plot shown in Fig. 7.6 (a). The behavior of the plot can be understood by looking at P_{miss} as a product of two terms: off

probability and flow rate. With increase in flow rate, the MAB agent learns to remain off with lower probability, and hence P_{off} goes down. That explains the inflection point in the graph of P_{miss} vs. λ in Fig. 7.5 (e). For the same reason, throughput also follows a non-linear relationship with λ (Fig. 7.6 (b)).

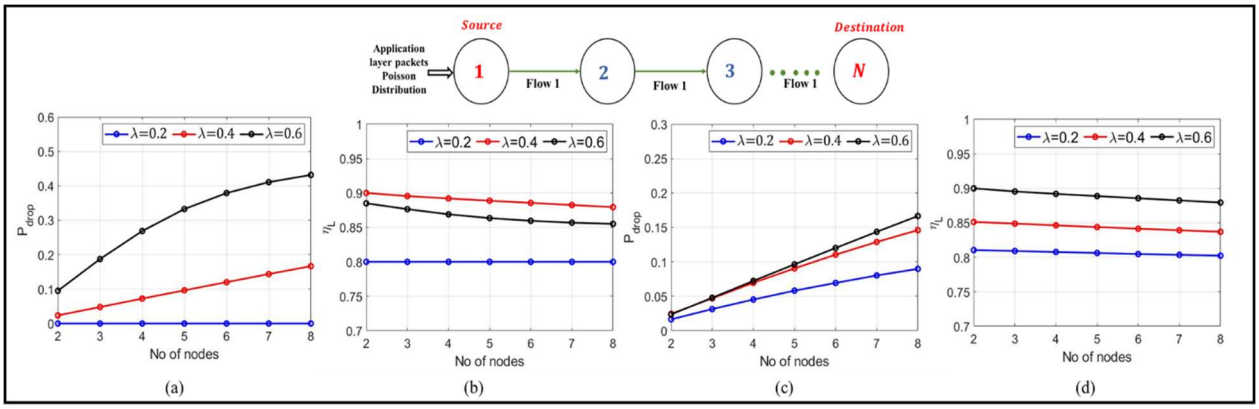

Figure 7.7: Effect of increase in flow length on packet drop rate and efficiency.

The effect of increasing the number of intermediate nodes in the flow is presented in Fig. 7.7 (a) for three different values of δ_p $(0.6, 0.9, 1.0)$. It is observed that with the increase in number of intermediate nodes in the flow, a greater number of nodes contribute to the packet losses due to missed receptions, and hence the overall missed reception rate increases. However, missed reception rate remains constant at zero for $\delta_p = 1$. Another observation is that for lower values of δ_p, the flow rate is disrupted because of missed receptions and hence the missed reception probability of a downstream node is lower than that of an upstream node. This effect is captured by the non-linear behavior of the network missed reception probability in the figure.

The efficiency plot shown in Fig. 7.7 (b) suggests that efficiency remains in the range of 80-90% for different number of intermediate nodes. The increase in missed receptions and decrease in efficiency with an increase in flow length holds for different flow rates and $\delta_p = 0.9$. This can be seen from Fig. 7.7 (c) and 7.7 (d).

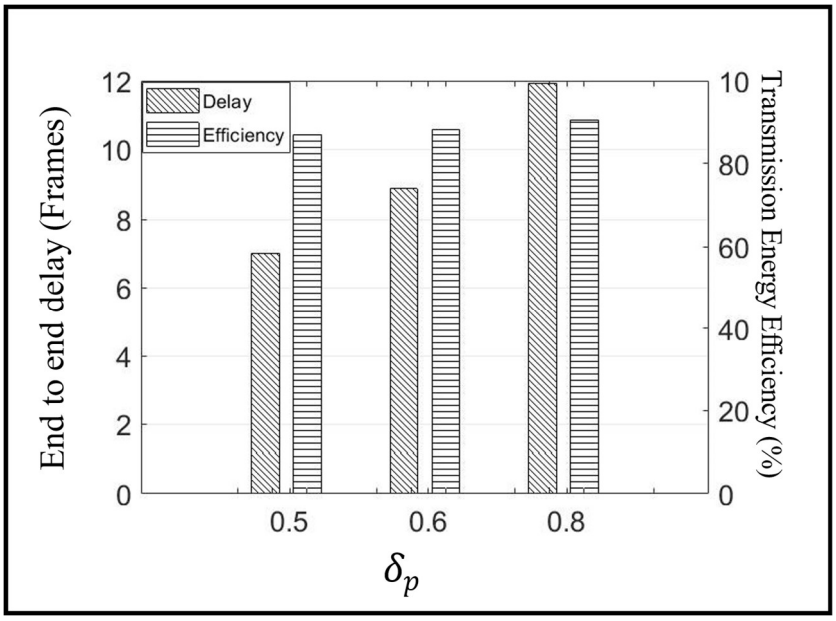

Figure 7.8: Effect of δ_p on Packet delay and Transmission Energy Efficiency.

To understand the effect of δ_p, packet miss reduction parameter, on the performance of the flow-controlling agent (Tier-II), end-to-end delay and transmission energy efficiency are plotted for $\delta_p = 0.5, 0.6, 0.8$ in Fig. 7.8. The figure demonstrates that both end-to-end delay and transmission energy efficiency (η_{tx}) increase with increase in δ_p. This is because Delay $\propto \frac{1}{\mu\text{-}\lambda}$ and $\eta_{tx} = \frac{\lambda}{\mu}$. With increase in δ_p, missed reception rate goes down, which in turn increases λ, causing delay and η_{tx} to go up. Note that the transmission energy efficiency (η_{tx}) in this figure is different from the reception energy efficiency (η_L) in Figs. 7.5 and 7.7. The quantity η_{tx} denotes the energy efficiency of the Stage-II learning, whereas η_L denotes the energy efficiency of the Stage-III learning.

To summarize, δ_p is a tunable parameter which controls the trade-off between missed packet reception rate, efficiency, and delay, and it should be chosen based on the application requirements. For an application that cannot afford packet loss, the value of δ_p should be kept at its highest value 1. On the other extreme of applications with stringent energy budget and relaxed

delay requirements, δ_p should be kept close to zero.

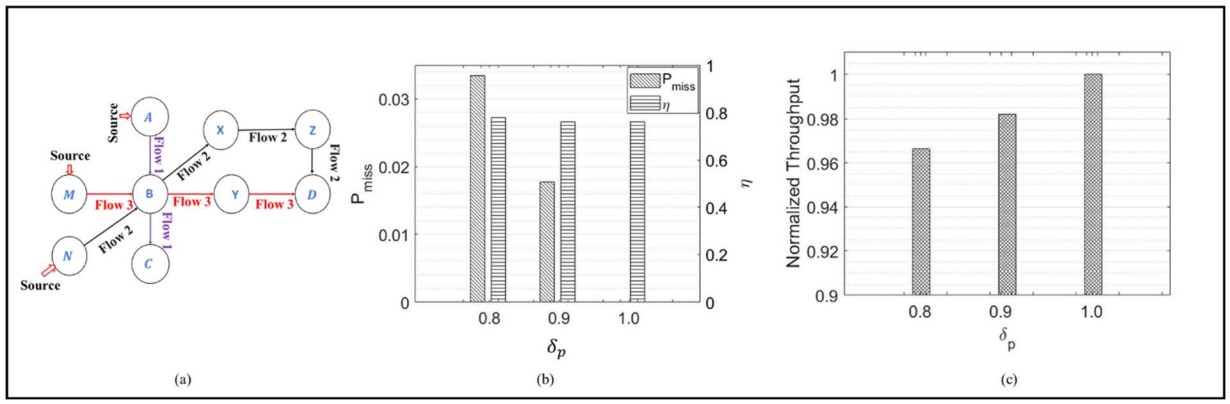

Figure 7.9: (a): Mesh network, (b): Missed Reception rate and efficiency of ESS-MAC on mesh topology, (c) Normalized throughput of ESS-MAC on mesh topology.

7.4.2 Performance Analysis on Arbitrary Mesh Network

To show the overall working of the learning-based ESS-MAC protocol, experiments are performed in the 9-nodes arbitrary mesh network with 3 flows with heterogeneous flow rates of $\lambda_1 = 0.1$, $\lambda_2 = 0.4$ and $\lambda_3 = 0.3$ packets per frame (Fig. 7.9 (a)). The missed reception rate (P_{miss}) and energy efficiency (η) for different values of δ_p are shown in Fig. 7.9 (b). The energy efficiency here represents the overall efficiency capturing both transmission and listening, which is computed as $\eta = \frac{\eta_L \times |\bar{T}| + \eta_{tx} \times |T|}{|T| + |\bar{T}|}$, where $T, \bar{T}$ represent the sets of slots in a frame where the node transmits and does not transmit respectively. In other words, η captures the energy efficiency of the entire framework. Similar to the findings for single flow scenario in Section 7.4.1, reduction in sleep-induced packet losses and increase in efficiency with increase in the value of packet miss reduction parameter (δ_p) still hold for the mesh topology. In addition, missed packet reception goes to zero for $\delta_p = 1$ with an energy efficiency of around 80%. Fig. 7.9 (c) shows the network throughput variation with different values of δ_p. It shows that with increase in δ_p, throughput increases. This is because, with increase in δ_p, packet misses resulting from oversleeping reduces, and hance throughput increases.

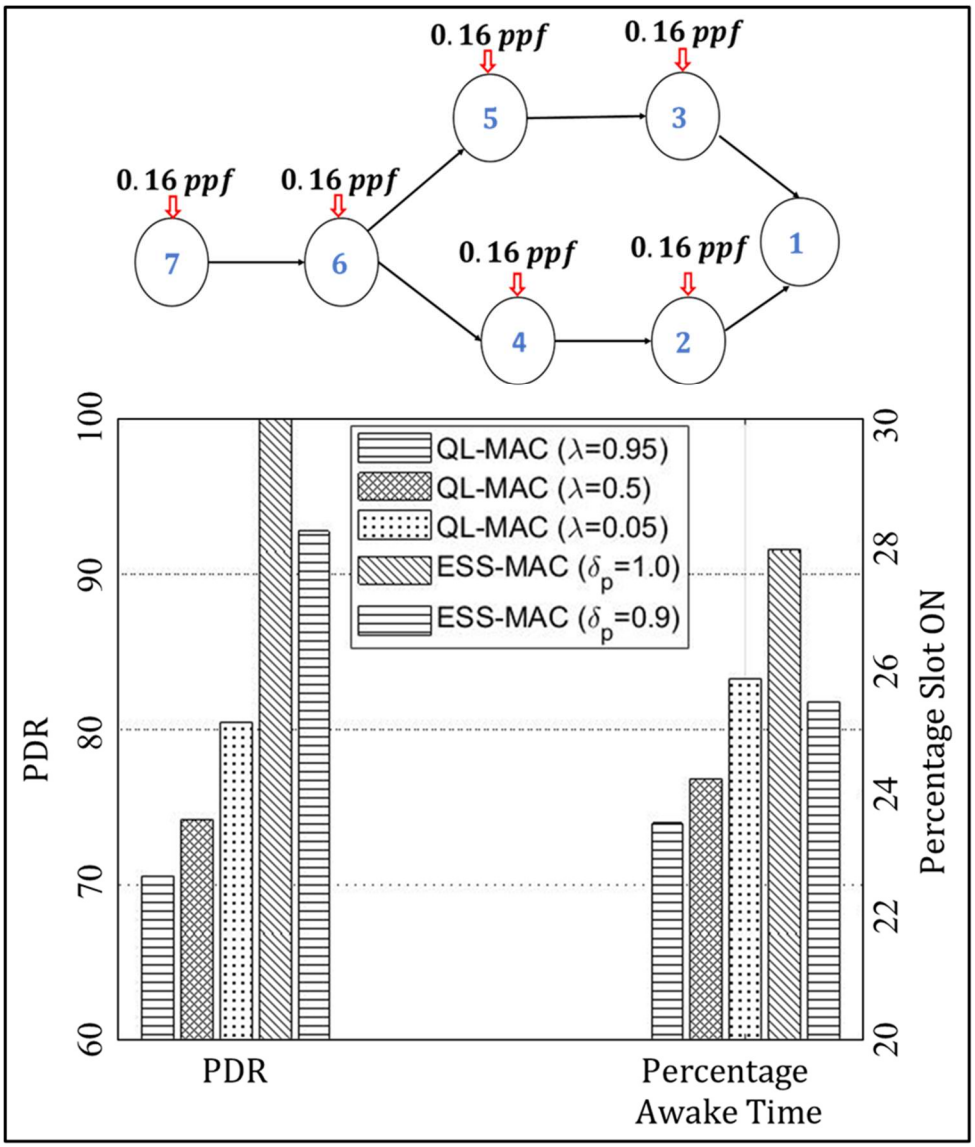

Figure 7.10: Performance Comparison of ESS-MAC with QL-MAC.

7.4.3 Performance Comparison with QL-MAC

The performance of the proposed framework is compared against the existing state-of-the-art RL-based sleep scheduling protocol QL-MAC [89]. The working concept and major limitations of the same have been summarized in chapter 2 (Section 2.2). Fig. 7.10 shows the comparison of the proposed ESS-MAC protocol with the existing QL-MAC protocol for the 7-nodes mesh topology. The parameter λ in QL-MAC controls the trade-off between packet delivery rate and energy consumption. In this work, performance of ESS-MAC is compared with QL-MAC for

three different values of λ (low, medium and high). The values of λ are selected based on what is suggested by the authors in the paper [89]. It is observed that ESS-MAC can achieve higher packet delivery ratio (PDR) for lower values of node awake probability. This indicates that ESS-MAC is more reliable than QL-MAC for the similar energy efficiency. Moreover, ESS- MAC allows 100% PDR for a minimal compromise on energy efficiency for $\delta_p = 1$, whereas QL-MAC is not able to achieve a 100% PDR. This makes ESS-MAC useful in applications that cannot afford any packet loss.

7.4.4 Learning Adaptability to Dynamic Network Traffic

Experiments were performed to analyze the adaptability with dynamic network traffic. Fig. 7.11 (a) shows the performance of the protocol for a time-varying flow rate in a linear network, where the initial data rate is 0.33 ppf. Data rate for the flow was changed at the time instances shown by dotted lines in the figure. It was observed that the learning agent is able to adjust its learnt policy according to the time-varying flow rate, so as to minimize the energy consumption while maintaining a stable queue. However, it is observed that learning convergence after the flow rate changes are generally slow. This is because when the scheduling policy is learnt for a particular flow rate, it assigns penalties for actions not favoring that flow rate. Hence, after learning convergence, the Q table is biased towards that specific flow rate. However, the penalized action may be suitable for a different flow rate. This makes the learning convergence to be slow in scenarios of dynamic network traffic. To be noted that learning recovery time is dependent on the original (λ_{old}) and final flow data rate (λ_{new}) as can be observed from Fig. 7.11 (b). If the change in load is such that λ_{old} and λ_{new} has the same optimal solution, then the convergence is faster, since the learnt Q-table is already biased to the optimal solution. With increase in the change in load, recovery convergence time increases. This problem of high learning recovery

time to dynamic traffic can be handled by incorporating the estimated flow data rate in the RL state definition and by using a Deep Neural Network to estimate the Q-function, which is presented in next chapter.

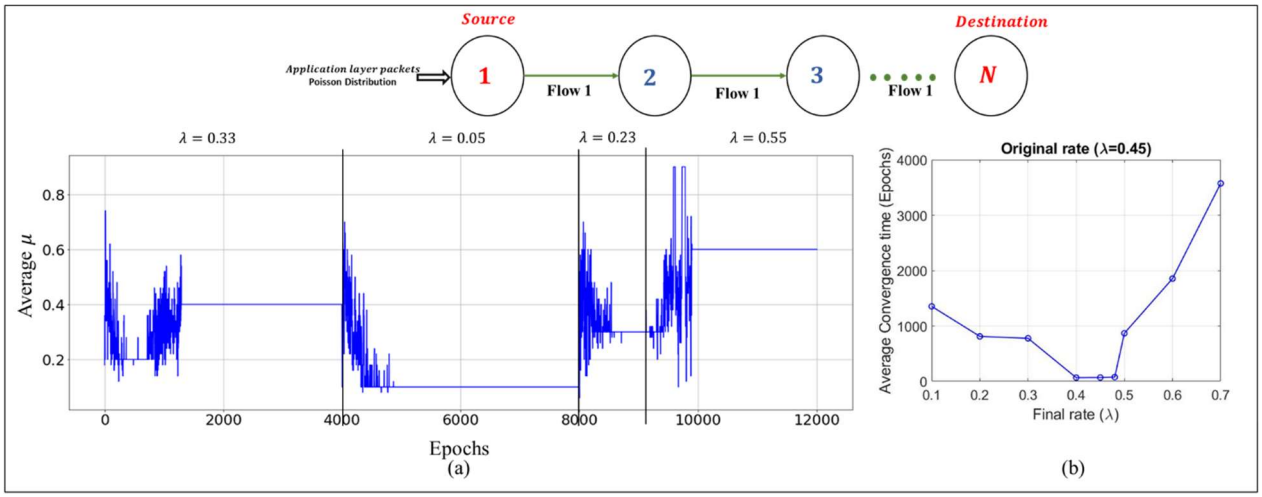

Figure 7.11: (a): Learning adaptability to dynamic network traffic at node 2, (b) Learning recovery time variation with dynamic load.

7.5 ESS-MAC for Networks without Time Synchronization

Accurate time synchronization over wireless can be expensive, especially with low-cost hardware and limited processing and communication resources. Moreover, the MAC layer performance in such networks can be very sensitive to even slight perturbations in the quality of time synchronization [21]. Adaptations of the proposed learning-based MAC framework for networks without time synchronization are presented in this section. Asynchronous TDMA MAC operation has already been explained in chapter 5.

7.5.1 Mini-slot Allocation in Asynchronous TDMA

Transmission scheduling problem in this context boils down to each node to be able to choose a mini-slot at which the node can transmit in all subsequent frames without colliding with the transmissions from the other nodes. Such collision-free mini-slots is selected locally at each node

independently, and that is without any centralized allocation entities and network time synchronization. The selection policy is modeled as a Multi-Armed Bandit (MAB) problem, as formulated in chapter 5.

7.5.2 Learning Transmit-Listen-Sleep Policy

Once a node finds its non-overlapping transmission mini-slots using the MAB model formulated above, the next aim is to find an efficient transmit/sleep schedule for providing a load-sustainable support for all the flows through it. The Reinforcement Learning model used here is the same as what was used for the networks with time synchronization capability as explained in Section 7.3. There is a learning agent per flow (Stage-II learning agent in Fig. 7.2) and the action is to decide on whether the node should transmit or sleep in the allocated mini-slots for the node. The state space and the reward function are the same as in the time-synchronized scenario and follow Eqn. 7.3. Using this RL model, a node finds its transmit/sleep duty cycle so as to spend energy judiciously while keeping end-to-end packet delay at an acceptable level.

The next task for a node is to decide whether to listen or sleep in the non-transmission mini-slots decided in tier-I. The nodes learn a sleep-listen policy using the MAB-based model of the learning agent in Stage-III (Fig. 7.2) as discussed in Section 7.3. The only difference is that the decision epoch here is at the mini-slot level as opposed to the slot level in the time-synchronous case. Since time is not synchronized, a node should be awake in multiple mini-slots to successfully receive a packet sent from its neighbor. This is because, a packet occupies multiple mini-slots (depending on the value of n_M (Eqn. 5.1)) and the receiver has to remain awake on all these mini-slots to successfully receive the packet. This behavior is illustrated in Fig. 7.12, where node i is transmitting packets to node j. Since time is not synchronized, the frame of node j lags that of node i by an amount Δ_i. In this figure, node i transmits packets at the second mini-slot in

its frame, and since $n_M = 2$, the packet occupies the second and the third mini-slots of the frame of node i. From node j's perspective, for successful reception of that packet, it should be on for that duration. Meaning, it should be awake for mini-slots 1, 2 and 3 in its frame. This means that the listening node has to remain awake for 1 mini-slot worth of duration more than the actual packet duration for this time-asynchronous case.

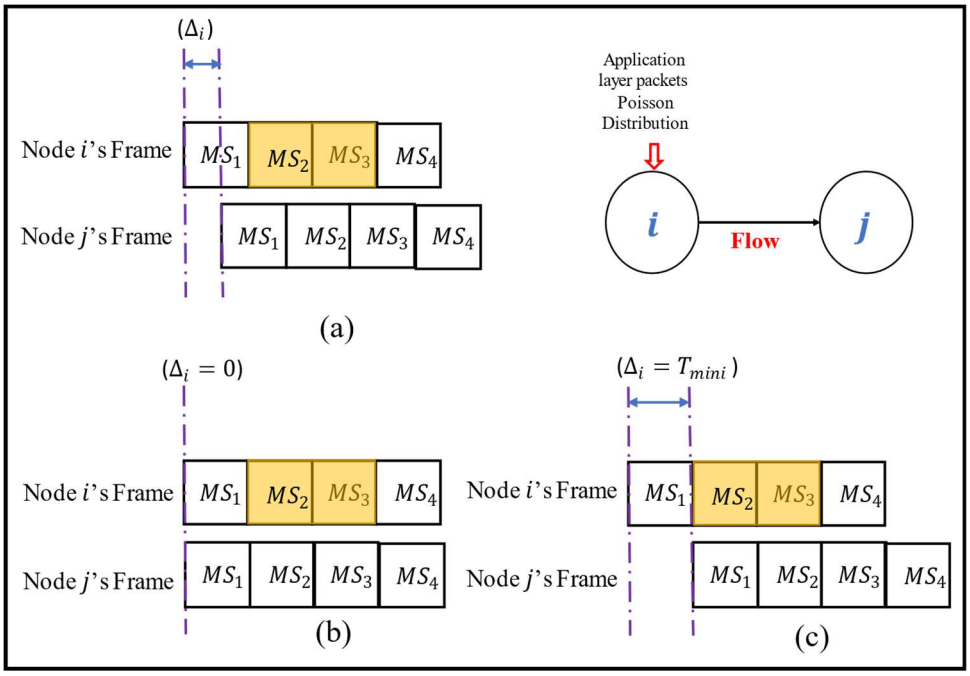

Figure 7.12: (a) Sleep-listen scheduling in time-asynchronous networks (b) $\Delta_i = 0$ (c) $\Delta_i = T_{mini}$.

Considering the two extreme cases, $\Delta_i = 0$ and $\Delta_i = T_{mini}$, as shown in Fig. 7.12 (b) and (c) respectively, the receiver node j does not need to be awake for extra duration than the packet duration for successful reception. This is because these two cases correspond to the time synchronization case at the mini-slot level of temporal granularity. This concept can be extended for any value of n_M. Thus, for a node to successfully receive a packet destined to itself, it has to remain awake for a duration D, where $\tau \leq D \leq \tau + T_{mini}$ (τ, T_{mini} are the packet duration and mini-slot duration respectively).

Using the above Multi-Armed Bandit model, experiments are performed for networks in the absence of time synchronization. Performance in terms of energy efficiency (η_L), throughput, and sleep-induced missed reception probability are shown in Figs. 7.13 (a), (b) and (c) respectively. These performance metrics are computed using the same way as in the time-synchronous case (See Section 7.4). The following observations can be made. First, for lower values of flow rate λ, efficiency (η_L) goes down with the increase in δ_p, whereas, for higher λ, efficiency increases with increase in δ_p. This means that, it is beneficial to sleep more when the flow rate is low. Hence, a smaller value of δ_p increases efficiency for lower λ values. The converse is true for higher flow rates.

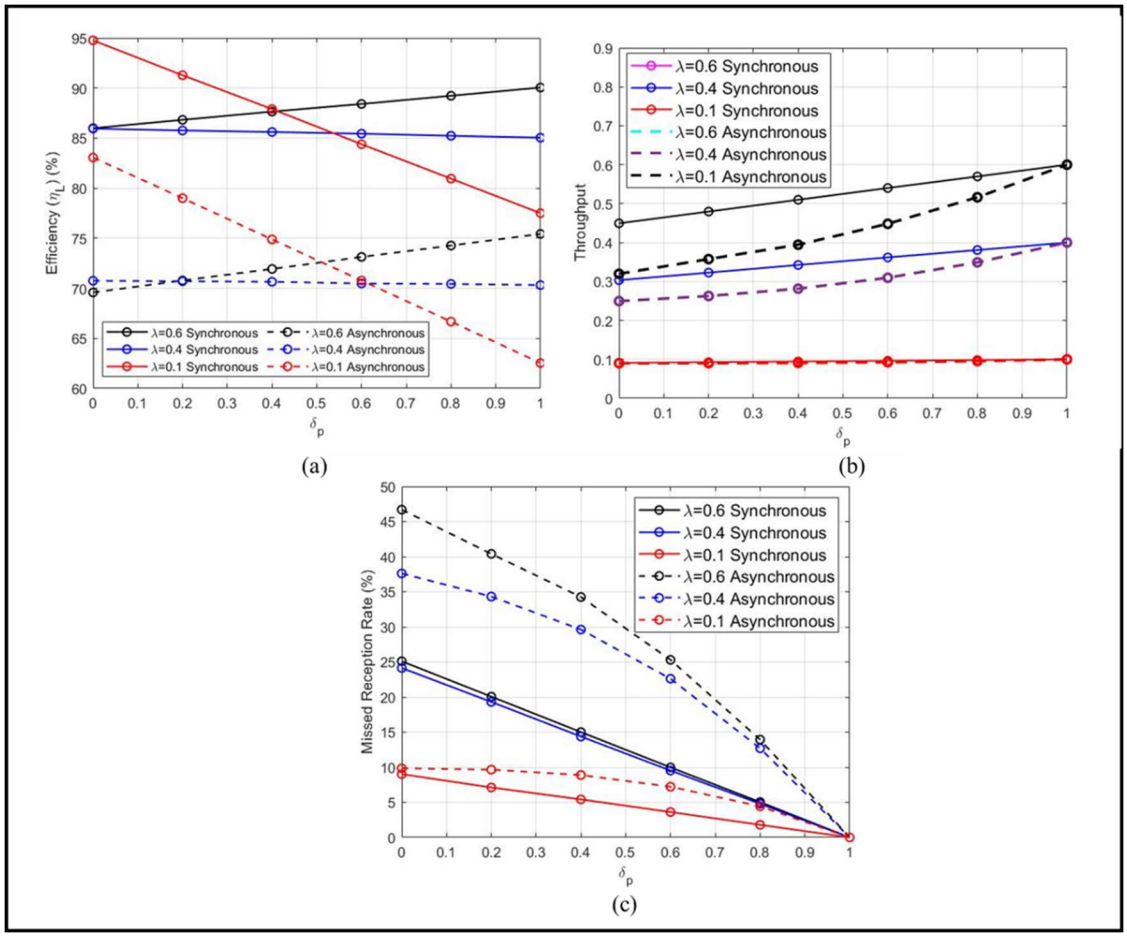

Figure 7.13: Performance evaluation of ESS-MAC Protocol on networks in the absence of time synchronization.

Second, throughput increases with increase in δ_p values and reaches the maximum value of λ for $\delta_p = 1$. This also shows that sleep-induced packet reception losses is zero for $\delta_p = 1$. Third, for the same operating conditions, efficiency and throughput are low, and the packet miss rates are high for this time-asynchronous case as compared to the time-synchronized networks. This is because, for the time-asynchronous case, the node has to be awake in multiple mini-slots for successful reception of a packet, unlike in the time-synchronous case. For lower δ_p, when a node decides to sleep in a mini-slot, there is a probability that it will miss a packet whose transmission started in the prior mini-slot. This is not the case in a time-synchronized network, where sleeping in a slot only affects the missed reception in that slot. This explains the high missed receptions and low throughput in time-asynchronous case. Also, the sleep-induced packet losses, and hence throughput, follows a non- linear relationship with δ_p for networks in the absence of time synchronization. This non-linear behavior can be explained from the analytical model as follows.

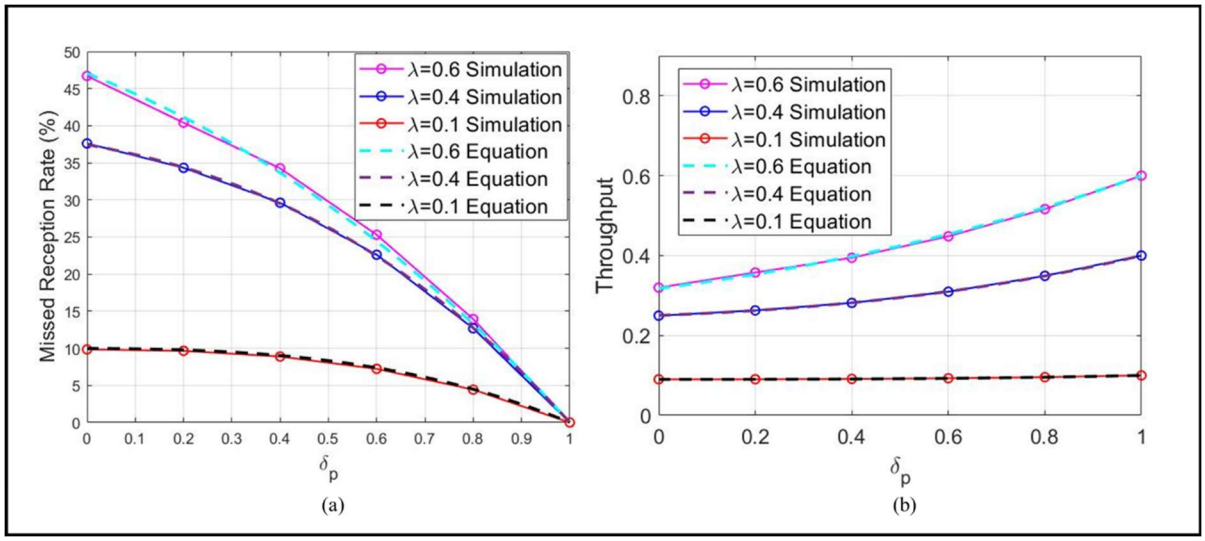

Figure 7.14: Comparison of analytical and simulation model for (a) missed reception rate and (b) throughput.

P_{miss}^{async} = Probability that the node is off in at least one mini-slot

where it is supposed to receive packets

$$= \begin{cases} \left(1 - P_{on}^{n_M+1}\right) \times \lambda, & if\ n_M \geq 1 \\ (1 - P_{on}) \times \lambda, & otherwise \end{cases}$$

$$= \begin{cases} \left(1 - \left(V(a_n = L) \times \left(1 + \frac{1 - V(a_n = L)}{V(a_n = L)} \delta_p\right)\right)^{n_M+1}\right) \times \lambda, if\ n_M \geq 1 \\ \left(1 - \left(V(a_n = L) \times \left(1 + \frac{1 - V(a_n = L)}{V(a_n = L)} \delta_p\right)\right)\right) \times \lambda \quad otherwise \end{cases}$$

After convergence, $V(a_n = L) \rightarrow \lambda$.

This gives:

$$P_{miss}^{async} = \begin{cases} \left(1 - \left(\lambda \times \left(1 + \frac{1 - \lambda}{\lambda} \delta_p\right)\right)^{n_M+1}\right) \times \lambda, if\ n_M \geq 1 \\ \left(1 - \left(\lambda \times \left(1 + \frac{1 - \lambda}{\lambda} \delta_p\right)\right)\right) \times \lambda \quad otherwise \end{cases} \tag{7.8}$$

The power term $(n_M + 1)$ in the equation gives rise to the non-linear behavior in missed reception probability. Similarly, the throughput can be obtained as:

$$\text{Throughput, } s_{async} = \lambda \times (1 - P_{miss}^{async}) \tag{7.9}$$

The missed reception probability expression derived in Eqn. (7.8) and the throughput in Eqn. (7.9) have been validated using simulation and is shown in Fig. 7.14. To be noted that for $n_M < 1$, the missed reception probability equals the missed reception probability (Eqn. 7.7) in the time-synchronous networks. But choosing the mini-slot scaling index $n_M < 1$ is not practical, as it makes the mini-slot duration more than the packet duration, that will lead to poor bandwidth utilization. However, even for $n_M \geq 1$, there exists a trade-off between the sleep-induced packet losses and bandwidth redundancy that comes from the learning in tier-I. This bandwidth redundancy comes from the requirement of one additional mini-slot for networks in the absence

of time synchronization. This trade-off is depicted in Fig. 7.15 for $\delta_p = 0.8$ and $\lambda = 0.4$, where missed reception probability goes down and bandwidth redundancy goes up with increase in n_M values. Hence, based on the application-specific requirements and networking resource constraints in terms of bandwidth and throughput, an appropriate value of n_M can be chosen.

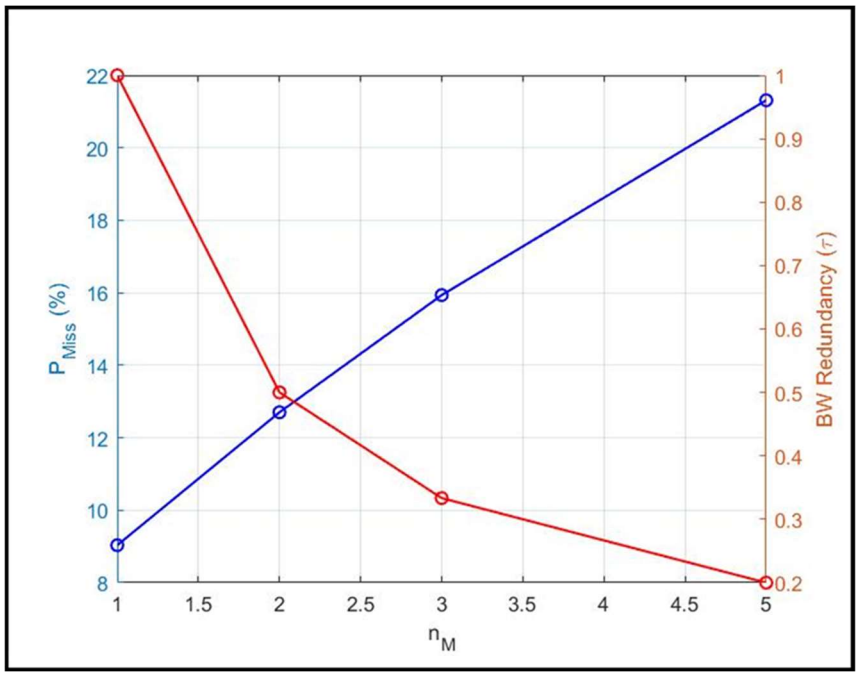

Figure 7.15: Trade-off between missed detection rate and bandwidth redundancy.

To summarize, it is shown that ESS-MAC can be used for sleep scheduling in networks without time synchronization. However, the throughput and energy efficiency of ESS-MAC in the absence of time synchronization is lower than that in a time-synchronized network. This performance difference decreases with increase in δ_p. Also, there exists a trade-off between throughput and bandwidth redundancy in networks without time synchronization. By suitable choice of n_m, this trade-off can be managed based on application-specific requirements.

7.6 Summary

A learning-based framework is proposed in this chapter for efficient sleep-transmit-listen scheduling in wireless networks. It is shown how the proposed mechanism allows wireless nodes

to learn policies that can support throughput-sustainable flows while minimizing sleep-induced packet loss, packet delay, and idle awake duration. The trade-off between energy efficiency and missed packet reception is studied, and it is shown how the trade-off can be tuned based on application-specific requirements. An analytical model for the MAB-based scheduling is developed and validated against results from extensive simulation experiments. Finally, by means of experimentation with mesh networks, the framework is generalized for arbitrary topologies. Notably, the proposed framework is shown to work in the absence of network time synchronization, carrier-sensing, and other complex lower-layer hardware support, thus, making it suitable for low-cost transceivers for wireless IoT and sensor networks.

Note that although the online learning abilities of RL make the system adaptive to time-varying traffic scenarios, it is observed that learning recovery time to adapt to time-varying traffic is slow. It restricts the usage of the protocol only to slow-varying flow rate. Moreover, because of using tabular RL algorithms, the proposed framework has scalability issues with network size, degree, and the RL state and action spaces. In the next chapter, we propose a Contextual Deep Reinforcement Learning framework for network flow and energy management with an aim to address the limitations mentioned above.

Chapter 8: Energy Management using Contextual Learning

The learning-enabled architecture for network flow and energy management developed in chapter 7 has the limitations of slow adaptation to dynamic traffic conditions. In addition, the synthesized protocol exhibits scalability challenges concerning network size and degree. In this chapter, we present a learning framework for data flow and energy management in wireless networks with an aim to solve these limitations. The overall goal of the developed mechanism is to make the wireless nodes learn an efficient and data-rate adaptive transmit-sleep-listen schedule. The learnt schedule can provide throughput-sustainable support for the active flows in a network, while minimizing the energy expenditure and sleep-induced packet drops. This is achieved using Contextual Deep Q-Learning (CDQL), that makes the system adaptive to dynamic and heterogeneous network traffic conditions.

8.1 Motivation

As mentioned above, a major limitation of the ESS-MAC proposed in chapter 7 and RL based sleep-awake scheduling in general [37] [89] [30], is that these mechanisms lack scalability with network size, degree, and the RL state and action spaces. These adversely affect the performance of the RL-synthesized schedules. Additionally, their slow adaptations to changing network conditions may pose practical implementation issues. The problem of scalability can be handled using deep Reinforcement Learning [90], which uses an Artificial Neural Network model as a function approximation entity for learning appropriate RL actions. The problem of adaptability to changing network conditions can be further addressed using a contextual learning model. The latter uses 3-dimensional context-state-action space as opposed to the 2-dimensional state-action space used in traditional RL models, thus offering context-sensitive learning options.

In this chapter, a multi-dimensional function approximation model using Contextual Deep Q-Learning (CDQL) is developed to address the scalability and adaptability issues discussed earlier. As in the framework proposed in chapter 7, learning here takes place in three distinct tiers. Tier-*I* learning is executed using a *per-node* Multi-Armed Bandit (MAB) model which is responsible for ensuring collision free TDMA transmission scheduling. Based on the active flows and their data rates in the node, the next course of action is to decide whether to transmit or sleep in its allocated slot. This is accomplished by a *per-flow* Contextual Deep Reinforcement Learning (CDRL) model in tier-*II* learning. The primary objective of this tier is to learn a suitable on-off transceiver duty cycle so that the energy expenditure can be minimized while keeping the end-to-end delay under check. Finally, for the non-transmission slots, sleep-or- listen decisions are learnt by a per-node tier-*III* learning agent. The goal of this stage is to learn an efficient sleep-listen schedule for receiving packets sent by the node's one-hop neighbors. The objective is to reduce the listening energy expenditures while reducing the missed packet receptions due to inappropriate sleep decisions. The system level goal of this 3-tier coordinated learning framework is still to maximize throughput, while keeping the energy expenditure and end-to-end delay under check. In addition to learning adaptively with changing network traffic conditions, this CDRL-based mechanism allows controlling and fine tuning the balance between throughput and energy efficiency in an application-specific manner.

8.2 Contextual Deep Q- Learning Model for Sustainable Flow and Efficient Sleep-Awake Scheduling

The goal of the developed learning-based framework is to make the wireless nodes learn a traffic-adaptive transmit/ sleep/ listen schedule that can support sustainable data flows, while minimizing energy consumption. This goal is accomplished through a three-stage reinforcement

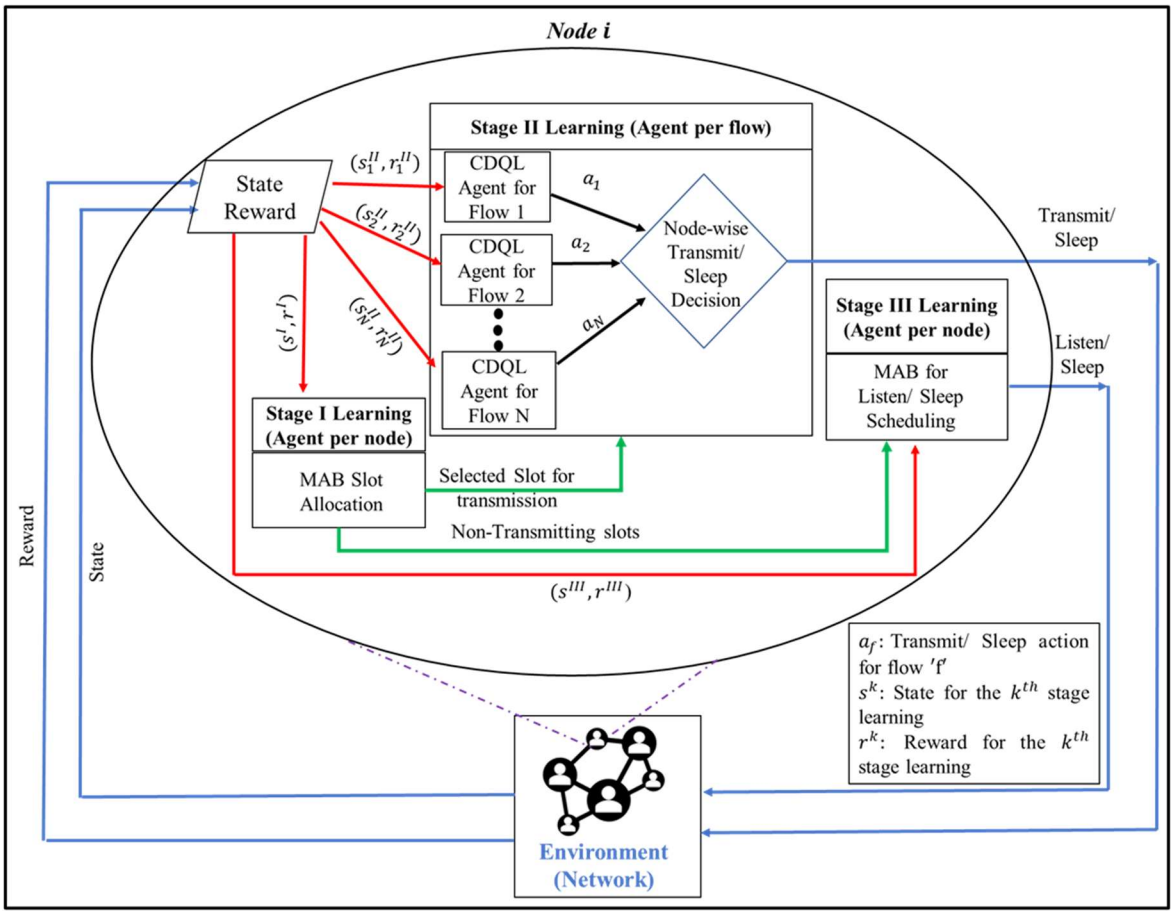

Figure 8.1: Different Stages of the learning modules from an individual node's perspective.

learning architecture implemented in each node, as depicted in Figure 8.1. The stage-I of the developed framework has a per-node Multi-Armed Bandit (MAB) agent that makes the node independently learn to find a collision-free TDMA transmission slot. The next task is to decide whether or not to transmit in that slot, which is controlled by a flow-level Reinforcement Learning agent in stage-II. The idea is to determine a transmit-sleep schedule so that the energy consumption is minimized, while maintaining a stable packet queue. Finally, the sleep-listen decisions on the transmission slots of the one-hop neighbor nodes are accomplished by a node-level MAB agent in stage-III. This agent makes the nodes learn an energy-efficient sleep schedule by keeping the throughput at an application-specific acceptable level. As the learning models in stages I and III are consistent with those discussed in Chapter 7, we will not delve into these

details here and will proceed directly to stage-III of the proposed architecture that uses a scalable contextual deep learning framework for making it adaptive to data flow rate.

Transmission Decisions using Flow-specific Contextual RL: Each node is equipped with a flow-specific RL agent to learn an efficient transmit-sleep schedule (stage II in Fig. 8.1). Each learning decision epoch is of T frames, which is a hyperparameter to be empirically explored. The epoch duration T should be large enough to capture a statistically meaningful RL state and reward information. However, a large T will also mean slow convergence. The parameter T has to be chosen emperically. In an epoch, the action of the agent is defined as the probability of packet transmission in a frame. The probabilities are discretized in m distinct values in the range [0, 1], where m is a hyperparameter. It will also be shown in Section 8.3 that the action space granularity m plays a significant role in the RL performance both in terms of energy efficiency and convergence speed.

The RL state space for a flow is defined by the congestion as perceived by an agent. The congestion state is coded as the change in length of the queue dedicated to the flow over the decision epoch of T frames. The following are the three possible states for the agent associated with a flow i at epoch t:

$$S_i(t) = \begin{cases} s_i^1, if\ \Delta Ql_i(t) > 0 \\ s_i^2, if\ \Delta Ql_i(t) = 0 \\ s_i^3, if\ \Delta Ql_i(t) < 0 \end{cases} \tag{8.1}$$

Here $\Delta Ql(t) = Ql(t) - Ql(t-1)$ is the change in the length of the queue for the i^{th} flow in the t^{th} epoch. A positive temporal gradient of queue length ΔQl indicates a high congestion in the network and vice versa.

In order to make the learning adaptive to time-varying flow rates, another dimension called "context" is added to the Q-table. In this scenario, the context is encoded by the agent-estimated flow rate $\hat{\lambda}$. The flow rate is estimated by an agent from its queue length and queue service rate derived from the RL action in that epoch. Formally, the estimated data rate perceived by the agent associated with flow i can be computed as: $\hat{\lambda}_i(t) = \mu_i(t) + \frac{\Delta Ql_i(t)}{s}$. Now, from this estimated flow rate $\hat{\lambda}(t)$, the context at epoch t can be determined as

$$C_i(t) = \frac{\widehat{\lambda_\iota}(t) + C_i(t-1)}{2} \tag{8.2}$$

It is to be noted that the time-varying flow rate is captured by the context definition of the framework, by giving more importance to the estimated data rate at time t as compared to that at $t-1$. This can be explained by expanding Eqn (8.2) as follows.

$$C_i(t) = \frac{\widehat{\lambda_\iota}(t) + C_i(t-1)}{2}$$

$$\Rightarrow C_i(t) = 0.5 \times (\widehat{\lambda_\iota}(t) + 0.5 \times (\widehat{\lambda_\iota}(t-1) + C_i(t-2))$$

$$\Rightarrow C_i(t) = 0.5 \times \widehat{\lambda_\iota}(t) + 0.5^2 \times \widehat{\lambda_\iota}(t-1) + 0.5^3 \times \widehat{\lambda_\iota}(t-2) + 0.5^4 \times \widehat{\lambda_\iota}(t-3) + \cdots$$

Thus, the framework is made adaptive to dynamic flow rates by giving more importance to the recent rate compared to the old estimated rates. Also, from Eqn. (8.2), it can be observed that, while computing the contexts, some importance is given to the old flow-rates as well, so that the variance resulting from random nature of the flow rate is taken care of.

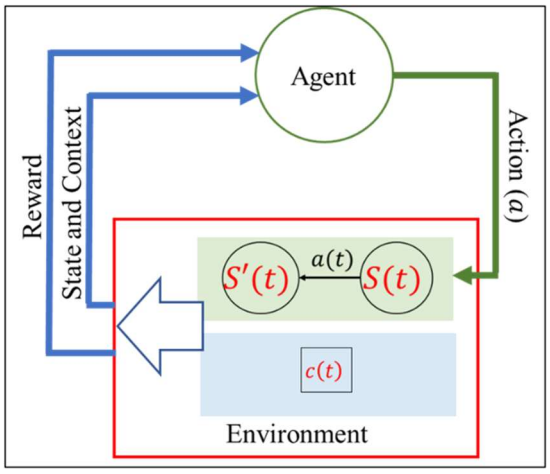

Figure 8.2: Contextual Reinforcement learning.

To be noted that the agents' actions here are decided based on both the context and state the environment is in. But unlike the states that change as a result of the agents' actions, these actions do not affect the change in context of the environment. In other words, unlike state transitions, the transition of contexts is oblivious to the agents' actions and is totally controlled by the environment (Fig. 8.2). These contexts are independent of each other, which means that the reward for an action chosen for a particular context follows an Independent and Identical Distribution (IID). If a tabular method such as Q-learning is used to solve such a Markov Decision Process (MDP) with 2-dimensional state-context space, it gives rise to Contextual Q-Learning (CQL), where the Q-table has a dimension of $\mathcal{S} \times \mathcal{C} \times \mathcal{A}$, where $\mathcal{S}, \mathcal{C}, \mathcal{A}$ represent the size of state space, context space and action space respectively. The Q-value is updated using Eqn. (8.3), where r is the reward received, α is a learning rate, γ is a discount factor, and s' is the next state, caused by action a.

$$Q(s,c,a) \leftarrow Q(s,c,a) + \alpha \left[r(s,c,a) + \gamma \times \max_{\forall a' \in A} Q(s',c,a') - Q(s,c,a) \right] \tag{8.3}$$

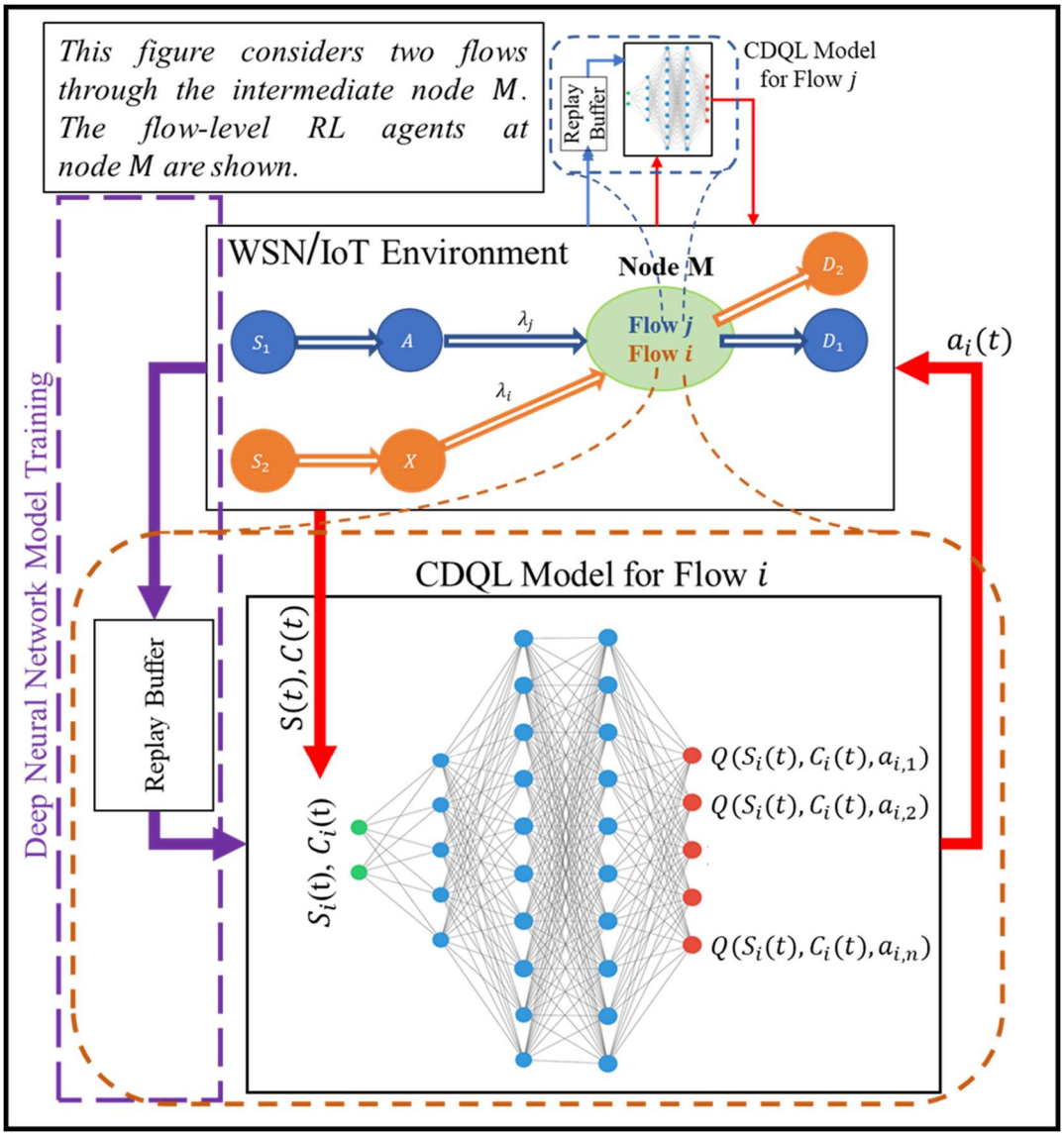

Figure 8.3: Contextual Deep Q-learning framework for Stage-II learning.

As can be seen from the definition of context in Eqn. (8.3) context can take continuous real values depending on a flow's data rate. This causes scalability problems if tabular RL mechanisms, like Q-learning, are used. The scalability issue will manifest as slow learning convergence for a large context space size. In the extreme case, for a continuous context variable, which is the case here, the three-dimensional Q-table would theoretically have infinite entries, which would make such tabular methods fail in such scenarios.

This problem is taken care of by using a deep Neural Network for estimating the Q-values for given <state, context> tuple. This is what is used in deep Q-learning [6] where a Neural Network is used to approximate the Q-function. In this case, the ANN model has two input neurons that take state, defined by change in queue length (Eqn. 8.1) and context, defined by estimated flow rate (Eqn. 8.2) as inputs. The output layer of the NN model has m Neurons, where m is the dimensionality of the action space $\mathcal{A}$. As discussed above, each of those m actions represents a probability of transmission in an epoch. The m output neurons provide the estimated Q-value of the m actions for the current queue status (state) and estimated flow rate (context) the NN received as input. A high-level system diagram of the Contextual Deep Q-learning framework for a flow i associated with an intermediate node M in a wireless network is shown in Fig. 8.3. As shown in the figure, the flow-specific learning agent i uses a Neural Network model to estimate the Q-values for the set of all the possible actions (transmit probability) for the corresponding $<S_i(t), C_i(t)>$ tuple, where $S_i(t), C_i(t)$ are defined by Eqns. 8.1 and 8.2 respectively. The agent then decides the best transmission strategy for the corresponding status of queue and flow-data rate based on the estimated Q-values and a learning policy (ϵ-greedy policy).

Training of the deep learning model is similar to traditional DQL framework [90]. The only difference here is that the experience tuples stored in the replay buffer has the additional context information defined by estimated flow rate. Thus, as learning progresses, the experience tuples for the CDQL agent for flow i defined by $<S_i(t), a_i(t), S_i'(t), C_i(t), R_i(t)>$ are stored in the replay buffer. The DNN model for the CDQL agent for that flow is trained using mini-batches of these tuples. These mini-batches are sampled from the replay buffer randomly following a uniform distribution. The CDQL model update for flow i at epoch t uses the following Mean

Square Error (MSE) loss function to update the parameters (θ_t) of the model:

$$L_{i,t}(\theta_{i,t}) =$$

$$\mathbb{E}_{(S_i,a_i,S_i',C_i,R_i)}[\left(R_i + \gamma \times \max_{\forall a' \in A} Q(S_i', C_i, a_i'; \theta_{i,t}^{-}) - Q(S_i, C_i, a_i, \theta_{i,t})\right)^2] \tag{8.4}$$

After learning converges, that is, once the deep Q network is trained, the learnt model can estimate the Q-values for any unseen state and context of the environment. In other words, the learnt model can find a suitable transmission policy for any flow rate and queue status.

In addition to handling the Q-table scalability issues caused due to continuous context space, Contextual Deep Q-Learning (CDQL) also plays an important role in managing the trade-off between network energy efficiency and learning convergence time. It also makes the MAC protocol scalable with network degree, as will be shown in Section 8.3. The details of the Neural Network architecture and other related hyperparameters is provided in Section 8.3.

Using the ANN-based function approximator and the learning policy, the CDQL agent takes an action for a given state/context. The action taken by the CDQL agent i is evaluated based on a numerical reward received after each decision epoch, which is computed from the reward function shown in Eqn. (8.5).

$$R_i(t) = \begin{cases} +1, \ if\ \widehat{\lambda_\iota}(t) < P_{tx}(t) < \widehat{\lambda_\iota}(t) + \frac{1}{m} + \delta \\ -1, \ otherwise \end{cases} \tag{8.5}$$

Here $\widehat{\lambda_\iota}(t)$ and $P_{tx}(t)$ represent the estimated flow rate and packet transmission probability in decision epoch t. The service rate μ is the expected value of the packet transmission probability P_{tx}: $\mu = \mathbb{E}[P_{tx}]$. The idea is to make the nodes learn a queue service rate $\mu(t)$, so that the queue length does not blow up and hence the end-to-end delay is maintained, while managing the energy

consumption. To summarize, the agent associated with the flow i learns a suitable service rate of the queue for flow i for achieving the above objectives.

Inter-flow Transmission Conflict Coordination: Each flow-level CDQL agent in stage-II operates independently in order to learn whether to sleep or transmit in its parent node's transmission slot that was selected using MAB learning in Stage-I. Since there can be multiple flows through a node, multiple such agents may learn to transmit in that slot, thus leading to inter-flow transmission conflicts. Such conflicts are handled using a node-wise transmit/sleep decision maker (shown in Fig. 8.1) that randomly assigns the transmission slot to one of those conflicting flows.

Table 8.1: Neural Network Model Details

Parameter	**Value**
Hidden Layer	4
Hidden Layer Depth	10× 20× 30 ×20
Loss	MSE
Optimizer	Adam
Hidden Layer Activation	Rectified Linear Unit

8.3 Experiments and Results

8.3.1 Experimentation Details

The experiments are performed to analyze the performance of the proposed CDE-MAC protocol using a MAC layer simulator with embedded learning components. The simulation kernel performs event scheduling in terms of packet generation, transmissions, and receptions. The developed CDQL-based MAC logic is implemented by embedding Deep Reinforcement Learning and Multi-Armed Bandits update equations within the MAC layer logic.

As mentioned in Section 8.2, the Neural Network used for CDQL implementation has two input neurons with state and context as inputs to the model and an output layer with m neurons, outputting the Q-values for m possible actions for the corresponding <state, context> tuple. The details on the Neural Network architecture is tabulated in Table 8.1.

The baseline parameters for experimentation (learning rate α, exploration ϵ, discount factor γ, action space dimensionality m, epoch duration T) are denoted in Table 8.2.

Table 8.2: Baseline Experimental Parameters

Parameter	**Value**
m	10
α (CDQL)	0.01
ϵ (Stage-II)	$e^{-t/10000}$
ϵ (Stage-III)	$0.01 + 0.99 \times e^{-t/3000}$
γ	0.9
w	100
δ	0.05
T	100
ϵ (ESS-MAC, Stage-II)	$e^{-t/2500}$
α (ESS-MAC)	0.1

Performance of the flow controlling agent is evaluated based on the transmission energy efficiency (η_{tx}) computed by the average queue utilization ratio ($\eta_{tx} = \frac{\lambda}{\text{Average } \mu}$). In addition, the same metrics employed to evaluate ESS-MAC in Chapter 7 are also utilized to assess CDE-MAC. For a given data rate λ, energy efficiency is high for a low service rate μ, as the node has to remain awake for low duration. However, the service rate should be always higher than the flow rate λ, otherwise the queues will not be stable causing the packet delay to blow up.

8.3.2 Results and Analysis

Using the framework detailed in Section 8.3.1 and the experimental parameters mentioned in Section 8.2, experiments were conducted on a mesh network shown in Fig. 8.4. The performance of the proposed CDQL-based CDE MAC logic is compared to that of the RL-based ESS-MAC protocol defined in chapter 7 for time-varying traffic conditions. Initial data rate for each flow in the network are $\lambda_1 = 0.25, \lambda_2 = 0.05, \lambda_3 = 0.15$ ppf. The change in the data rate λ over time for the 3 flows are shown in the figure. The following observations can be made from this set of experiments. First, in CDE-MAC, the nodes can adapt their transmission schedule instantaneously to the changing flow rate. On the other hand, in ESS-MAC, the nodes take longer

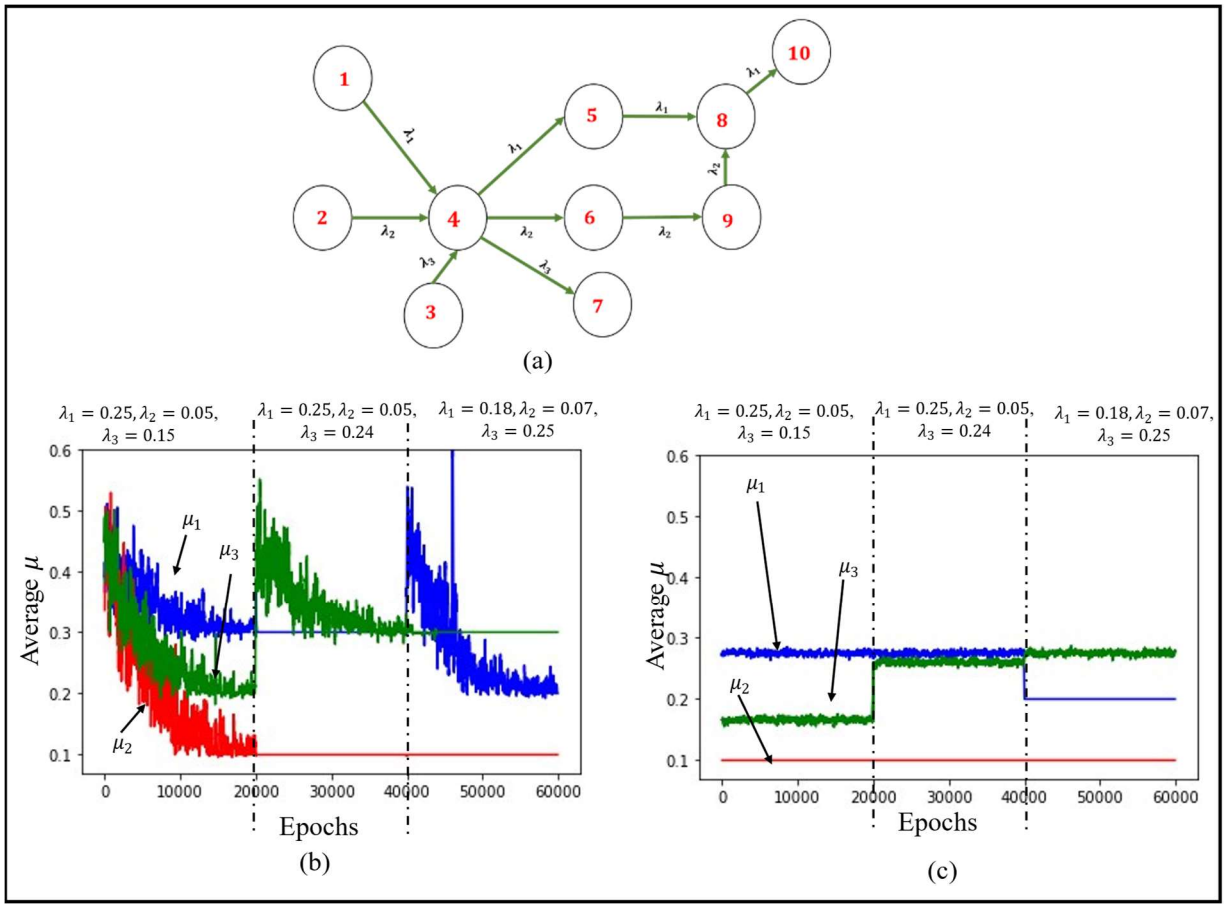

Figure 8.4: Performance comparison of CDE-MAC with ESS-MAC in dynamic traffic.

duration to adapt their transmission policies to the dynamic traffic conditions. This is because, in CDE-MAC, the pre-trained CDQL model can predict the action according to the estimated Q-function based on the current <context, state> computed from estimated data rate and queue length. However, for the RL-based ESS-MAC logic, the Q-table is not parameterized by flow-rate, and hence for any change in λ, the learning agent has to learn the Q-table from the scratch, thus resulting in high recovery time. This restricts the usage of ESS-MAC in applications with bursty traffic scenarios.

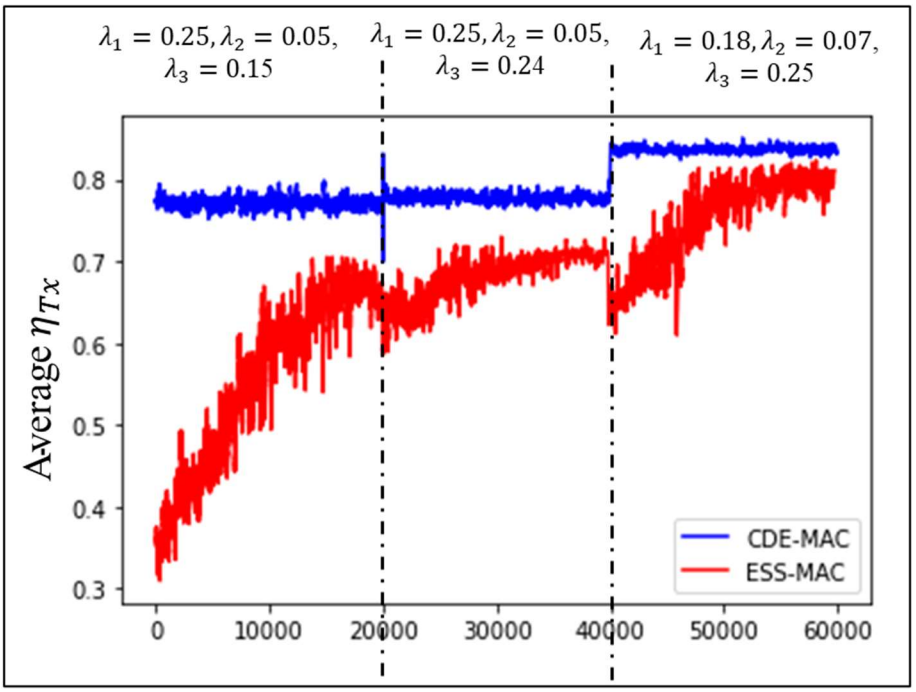

Figure 8.5: Efficiency of CDE-MAC vs ESS-MAC for a time-varying flow rate for topology in Fig. 8.4.

The second observation is that the transmission energy efficiency (η_{Tx}) is higher for CDE-MAC as compared to ESS-MAC (Fig. 8.5). This is because of the ability of the ANN model to do Q-function approximation for any unseen state-context tuple in CDQL-based CDE MAC protocol. In other words, for any flow-rate λ, the context and state are computed by the agent and the actions are taken such that the service rate (μ) is higher but close to the estimated flow rate ($\hat{\lambda}$).

This makes the node to learn a suitable sleep-transmit schedule that can save energy without blowing up the queue length. However, for ESS-MAC, because of the discrete nature of the Q-table, any unseen $\hat{\lambda}$ is mapped to a discrete state from the queue length dynamics, and only the higher notch of μ is learnt depending on the action space granularity m. As a result, although it learns to keep the queue length in check, but it remains awake for more than required, thus wasting precious energy resources.

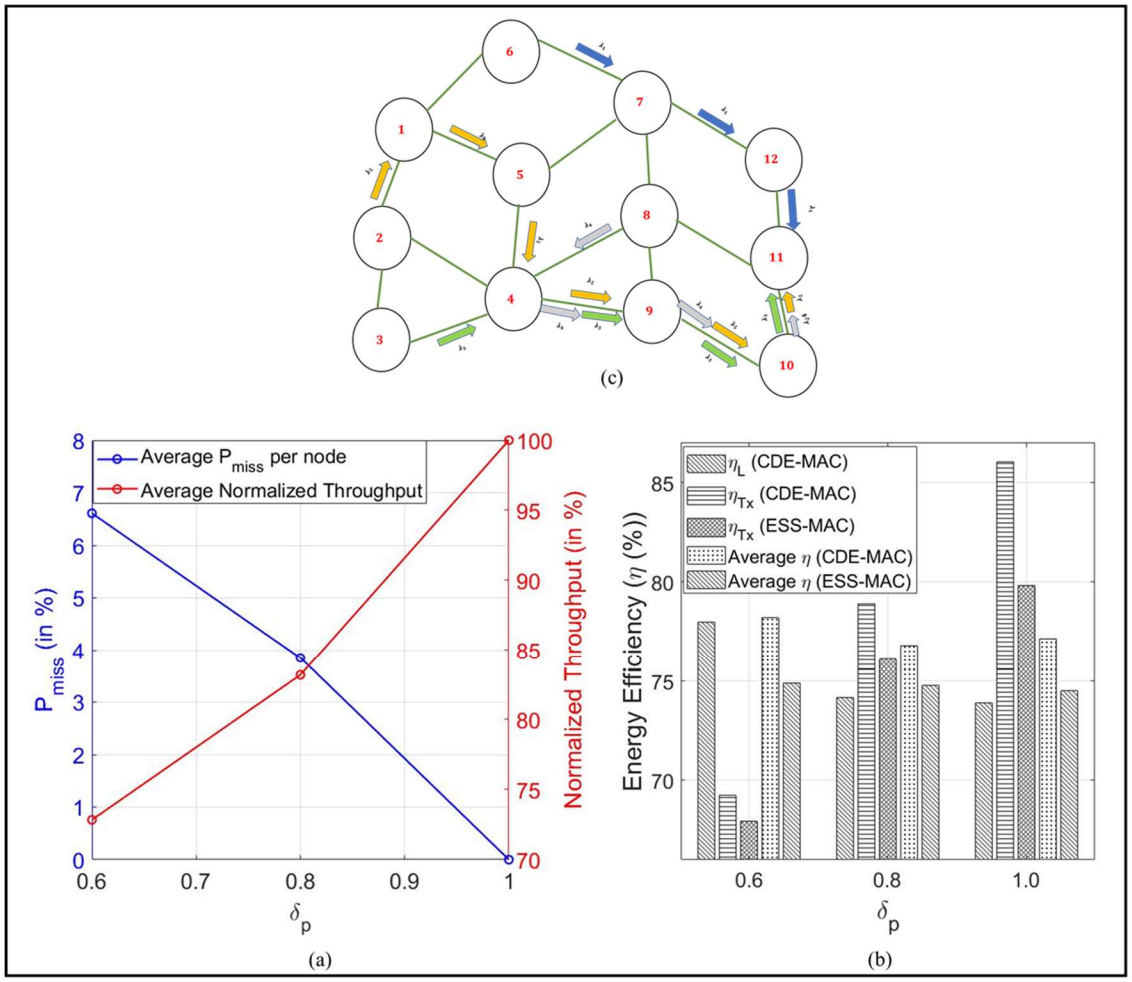

Figure 8.6: (a) Missed packet probability, throughput and (b) Energy efficiency for different values of δ_p for mesh topology in (c).

Performance of the proposed CDE-MAC logic is tested for different values of listening priority coefficient (δ_p) for the mesh network with 4 active flows shown in Fig. 8.6. The data rate of the

flows in the topology follows Poisson distribution with mean values of $0.22, 0.17, 0.08,$ and 0.13 packets per frame duration respectively. The key observations from the figure are as follows. First, with increase in δ_p, the missed reception probability (P_{miss}) resulting from bad sleep decisions goes down, and hence throughput increases (Fig. 8.6 (a)). This is because the node remains on with high probability for higher δ_p. With increase in δ_p, the node behaves in a more conservative manner in order to minimize missed packet reception resulting from oversleeping. To be noted that for $\delta_p = 1$, $P_{miss} \approx 0$ and throughput $\approx 100\%$. This is useful, especially in applications that cannot afford packet losses. However, reduction in P_{miss} for high value of listening priority coefficient (δ_p) comes with the price of reduced listening energy efficiency (η_L) (Fig. 8.6 (b)). This is because, for higher δ_p, the node learns to remain awake for more duration and hence consumes more energy. This reduces the energy efficiency. Another observation from the plot is that the transmission energy efficiency (η_{Tx}) is higher for CDQL-based CDE-MAC protocol as compared to RL-based ESS-MAC. This is because of the same reason explained above for Fig. 8.5, that is the ability of CDE-MAC to approximate the Q-function for any flow-rate. Since η_L and η_{Tx} both are higher for CDE-MAC, the average energy efficiency (η) is also higher for CDE-MAC as compared to ESS-MAC.

Note that η_{Tx} goes down with δ_p for both CDE-MAC and ESS-MAC (Fig. 8.6 (b)). The reason behind this behavior is that the effective data-rate (λ) goes down with increase in δ_p because of packet misses due to oversleeping. As a result, $\eta_{Tx} = \frac{\lambda}{\mu} = \frac{\lambda}{\lambda + c\prime}$ decreases with increase in λ. Also, the average end-to-end delay values in this experiment are 24.23, 7.10 and 16.98 for $\delta_p = 0.6, 0.8, 1.0$ respectively. This means that the end-to-end delays for the flows are contained and do not blow up. This is made sure by the Stage-II CDQL learning agent by making the agents

learn a transmission schedule such that $\mu > \lambda$ is maintained.

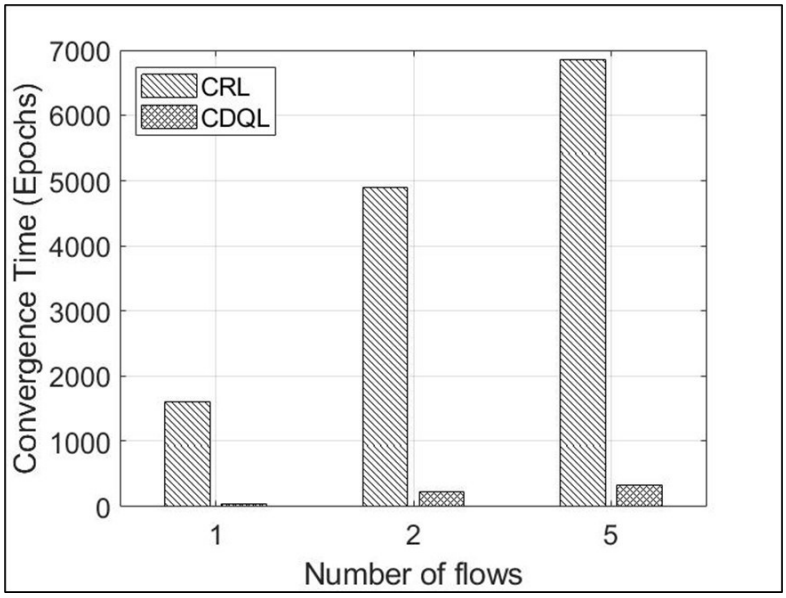

Figure 8.7: Learning convergence time comparison for CDQL and CRL.

An important characteristic of the proposed framework is that use of deep network makes the CDQL-based MAC protocol scalable with number of flows through a node as well as with action space granularity m. To demonstrate the scalability achieved as a result of using DQL, we compare the training convergence time for CDQL and CRL (Contextual Reinforcement Learning) with increase in the number of flows through a node. As shown in Fig. 8.7, unlike in CRL, training convergence time in CDQL scales well with an increase in the number of flows and is much lower (95-97%) than CRL. To be noted that for consistency with ESS-MAC, in CRL update equations, the maximum Q-value of the current state is considered as is used in [30] for fast convergence.

Another observation is that although increase in m increases the convergence time and energy efficiency for both CRL and CDQL, the increase in convergence time for a unit gain in energy efficiency (η_{Tx}) is higher using CRL compared to CDQL (Fig. 8.8). This is because for a specific value of action space granularity m, convergence is faster for CDQL as compared to CRL. In

addition, CDQL has the ability to do function approximation and hence allows higher maximum energy efficiency for sufficiently fine grain action space.

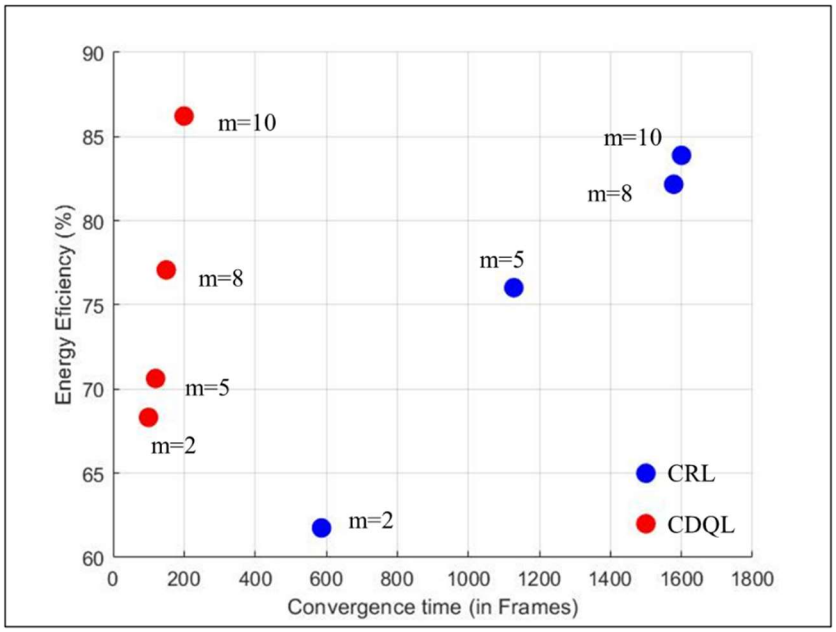

Figure 8.8: Convergence time- Efficiency trade-off for CDQL and CRL.

8.4 Summary

This chapter presents a novel Contextual Deep Q-Learning (CDQL) and MAB-based framework for data flow and energy management in wireless networks. It is shown how the proposed mechanism allows wireless nodes to learn policies for sleep-listen-transmit scheduling. The learnt policy can support throughput-sustainable flows while minimizing sleep-induced packet loss and idle awake duration. The use of contextual learning makes the developed protocol adaptive to time-varying and heterogeneous network data flow rate. Moreover, it is shown how deep Q-learning manages the trade-off between energy efficiency and learning convergence time and makes the protocol scalable to network degree.

An underlying assumption of the energy-management approaches outlined in chapter 7 and 8 is the presence of homogeneous energy profile across all wireless nodes. An intriguing progression

of this work would involve crafting learning mechanisms for networks with spatio-temporal energy profiles, commonly found in energy harvesting networks. In the next chapter, we broaden the framework established in this chapter to cater to such resource-constrained networks, where the energy availability is influenced by both temporal dynamics and geographical attributes.

Chapter 9: Protocol Synthesis for Energy Harvesting Networks using Cooperative Reinforcement Learning

The frameworks proposed in chapters 7 and 8 are centered around developing RL-driven solutions for judicious management of energy in resource-constrained networks. These strategies primarily target scenarios involving battery-powered networks characterized by a static and homogeneous energy distribution. In the subsequent phase of this study, we aim to make the protocol synthesis concept more generalized to accommodate networks with spatiotemporal energy profile. To achieve this goal, this chapter introduces a multi-agent RL-enabled architecture designed to strike a balance between performance and energy expenditure in energy-harvesting networks.

9.1 Motivation

Efficiency of transmit-sleep scheduling is important in energy-constrained wireless networks from the following two perspectives. First, to reduce network service disruptions due to energy shortage and second, to maintain reliable network performance in terms of throughput and delay. Traditionally, such schedules in the wireless nodes are often pre-programmed [63], and as such they often fail to deliver desired performance in a situation-specific manner. For example, sleep scheduling policies are often oblivious to network traffic patterns and heterogeneities, which can lead to wastage of precious networking resources, including energy. In addition, for networks with energy harvesting capabilities [91] [92], where the energy availability may change both temporally and geographically, these traditional scheduling methods do not naturally allow nodes to react according to the spatiotemporal energy profiles and variabilities. Shortcomings are often manifested in the form of not being able to maintain the desired balance between network

performance and network lifetime. In this chapter, a learning-enabled paradigm is proposed that allows the nodes to learn a joint transmit-sleep scheduling policy in order to overcome the above limitations.

There are existing reported works [55] [59] [70] [93] [94] [95] that deal with sleep scheduling and energy management in networks with energy constraints. The authors in [93] use a game-theoretic approach to find sleep-scheduling policy for solar powered sensor networks. In addition to the fact that these policies are static with respect to network traffic and topologies, they also do not consider transmission scheduling decisions. As will be shown later in this chapter, transmission scheduling plays an important role in maintaining network performance in energy-harvested networks. There are RL-based approaches [59] [56] adopted for sleep scheduling in energy harvested networks. Besides the above limitations arising from not considering policies for transmission strategies, these often rely on a centralized arbitrator for learning scheduling policies. Centralized learning, apart from being computationally inefficient and creating burden on a central server, comes with an additional bandwidth and energy costs for downloading learnt scheduling policies from the server to network nodes. Moreover, performance of the learnt policies heavily depend on the reliability of information collected from the sensor/IoT nodes over a possibly error-prone channel. Another limitation of the learning-driven scheduling approaches [58] [72] is that these are not suitable for networks with multiple hops. This is because of the fact that learning for multi-hops networks is challenging, since learning errors in a node can be carried over to all the downstream nodes in the topology it is part of.

In this chapter, a decentralized and cooperative Reinforcement Learning [96] [97] approach for joint sleep-transmission scheduling is presented to overcome the issues mentioned above. The scheduling problem is first modeled as a Markov Decision Process (MDP) and solved using

Reinforcement Learning as a temporal difference solution. The objective is to efficiently manage the ultra-thin energy budget of energy-harvesting sensor/IoT nodes, while maintaining acceptable network performance. This is realized by cooperative learning behavior achieved by two RL agents deployed per node. Each of these agents jointly learn scheduling policies for the node in order to achieve the above-mentioned objective. This learning architecture is supported by the online sharing of a learning confidence parameter by each node. Sharing of this confidence parameter assists decentralized learning of the scheduling policy for the downstream nodes in the network topology. In this way, each IoT/sensor node learns a transmit-sleep scheduling policy independently in a decentralized manner. This cooperative learning architecture is suitable for learning in networks with multiple hops between source and destination, where learning performance for a node is heavily dependent on the scheduling policies learnt by other nodes in the topology, which makes the proposed system scalable with network size.

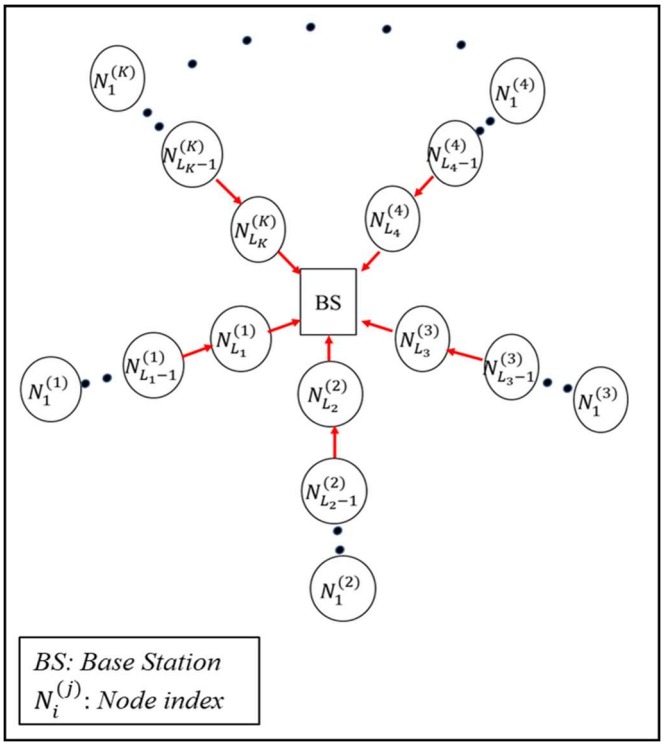

Figure 9.1: Example multi-point-to-point multi-hop network topology.

9.2 System Model

9.2.1 Network and Traffic Model

To maintain generality, we consider a multi-point-to-point and multi-hop network model. Fig. 9.1 shows a simplified star topology network in which the wireless sensor nodes/IoT devices send data to a base station using multi-hop routes. At the MAC layer, it is assumed that the network time is synchronized, slotted, and the MAC frames are of fixed size, which is dimensioned *a priori* based in the degree of the network topology. It is noteworthy that for networks without time synchronization ability, the concept developed in chapters 5 and 7 can be leveraged.

Application layer packet generation at the source nodes follow Poisson distribution with packet generation rate λ packet per frame (ppf). Each node maintains an M/G/1/K buffer/queue, where the Poisson distributed queue arrival rate is governed by λ, and the queue service rate is determined by the transmission-sleep policy actuated by the proposed learning mechanisms as presented in Section 9.3.

9.2.2 Energy Harvesting and Model

The wireless sensor/IoT nodes rely on energy harvesting for powering their communication subsystem. While the framework would scale for any source of energy harvesting, in this chapter, we have considered a solar energy harvesting model to demonstrate the efficacy of the proposed mechanisms. A 2-state Markov Model is used for simulating solar energy harvesting [93] [91]. The states of the Markov model are (1) low radiation state, in which sunlight is blocked by clouds and hence, radiation is not enough to charge the node batteries, and (2) high radiation state, in which there is direct sunlight and is sufficient to charge the battery. This is represented by the

transition probability matrix **R**, where state 1, 2 represent high and low radiation states respectively:

$$\mathbf{R} = \begin{bmatrix} R_{1,1} & R_{1,2} \\ R_{2,1} & R_{2,2} \end{bmatrix} \tag{9.1}$$

Assuming that the cloud size is exponentially distributed with mean 'C' m and the wind speed is w_s m/s, the elements of matrix **R** can be obtained using the analytical model in [2]:

$$R_{1,2} \approx \left(\frac{1}{\mu_s}\right) t \times \mathrm{e}^{\left(\frac{1}{\mu_s}\right)t},$$

$$R_{2,1} \approx \left(\frac{1}{\mu_c}\right) t \times \mathrm{e}^{\left(\frac{1}{\mu_c}\right)t},$$

$$R_{1,1} = 1 - R_{1,2} \text{ and } R_{2,2} = 1 - R_{2,1}$$

Here, $\mu_c = \frac{C}{w_s}$, $\mu_s = \frac{C \times (1-P_c)}{w_s(P_c)}$ are the average time for which the radiation is low and high respectively; P_c is the probability of solar radiation in low state and t is the length of a time frame. Each node has a battery capacity of B units, where a packet transmission consumes one unit of battery. Thus, the battery status at time t is given by $b \in \{0,1,2,\ldots,B\}$, where the battery is charged in high radiation state with probability P_{charg} . When $b = 0$, battery is completely depleted and needs recharging. If $b = B$, the battery is fully charged and, the recharging circuitry is turned off.

The energy consumption model in [98] is considered for the radio hardware energy dissipation. In this model, power consumed while transmitting and receiving a packet are given by Eqns. (9.2) and (9.3) as follows:

$$P_T = (1 + \alpha_T) \times P_{tx} + P_{ct} \tag{9.2}$$

$$P_R = P_{ct} \tag{9.3}$$

Here, P_{tx} represents the transmission power of the transmitter with amplifier inefficiency factor

α_T, and P_{ct} is the circuitry power consumption, which is a constant depending on a specific transmitter.

9.3 Cooperative Reinforcement Learning for Joint Transmit-Sleep Sleep Scheduling

The primary objective of the proposed learning architecture is to make the wireless nodes learn a suitable transmit-sleep schedule for efficient energy management. And that is while maintaining the network performance in terms of throughput and delay. This is achieved using a cooperative and decentralized Reinforcement Learning technique.

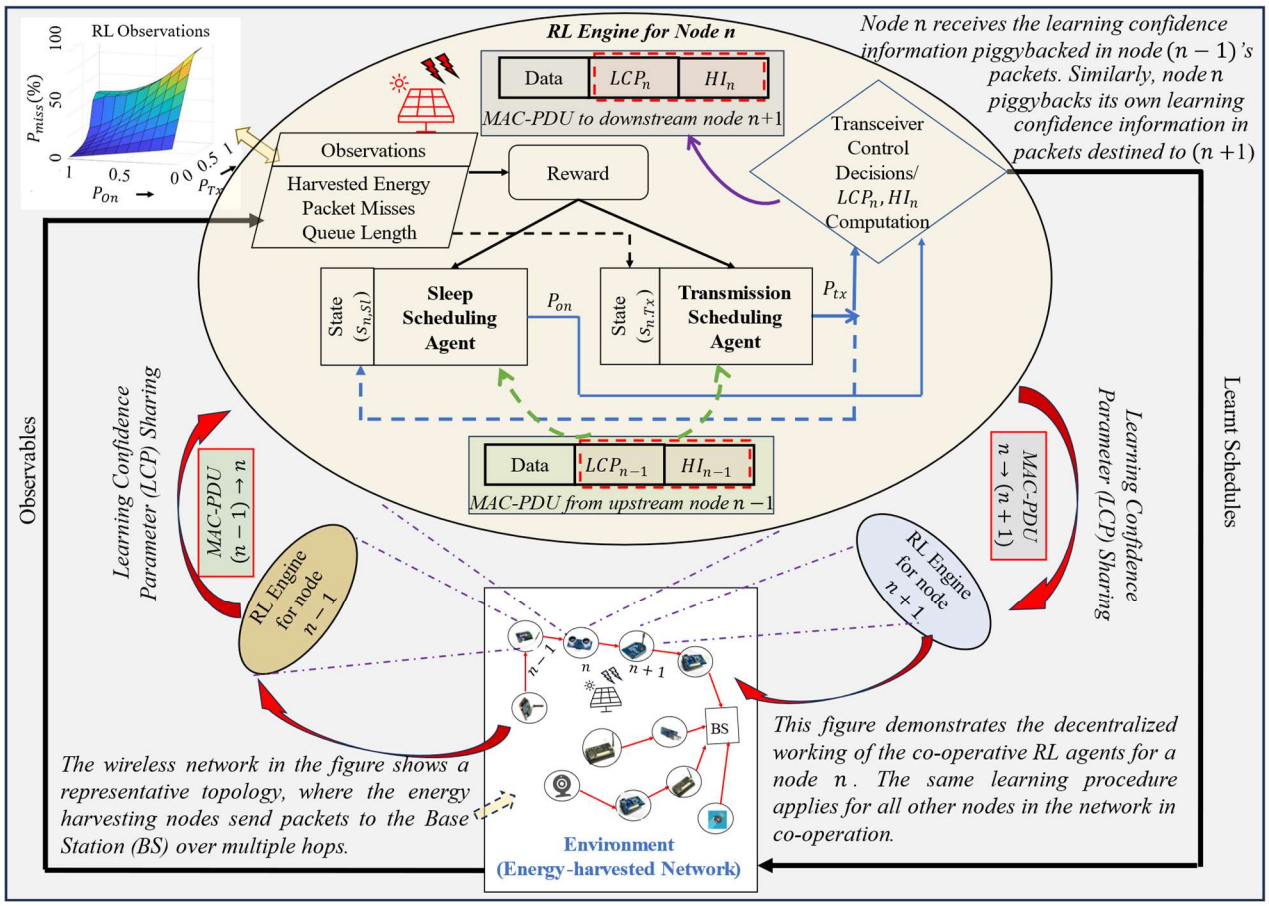

Figure 9.2: Proposed Cooperative Multi-Agent Reinforcement Learning Framework for Joint Transmission-Sleep Scheduling.

Fig. 9.2 shows a high-level working model of the RL-based architecture for a node 'n' in an energy-harvesting wireless network. There are two RL-agents deployed in each node: a

Transmission Scheduling Agent and a *Sleep Scheduling Agent*. For both, the RL environment is the network itself. Both the agents share the same reward function which is computed from the RL-related observable variables including harvested energy and network performance parameters. The state definition for those agents, however, are different and independent. As indicated by the dotted arrows, the state for the transmission scheduling agent is perceived directly from the network observables, whereas the state for the sleep scheduling agent is dependent on the RL-policies of the transmission scheduler. Using these independent notions of states, these two agents cooperatively learn the node's transmit and sleep schedules.

An important feature of the proposed learning architecture is the sharing of an online learning confidence parameter with the one-hop neighbors of each node. In the figure, this is depicted by the *Learning confidence Parameter (LCP) sharing*, indicated by the red arrows. This parameter, which is piggybacked in the MAC PDU, is used to prevent unreliable learning parameter update by the RL agents. Another parameter piggybacked in the PDU is the *Hop Index (HI)*, which is defined as the number of hops a node is away from the source of a passing flow. The figure shows examples of two incoming and outgoing MAC PDUs for node n. One received from its one-hop neighbor '$n-1$', and the other destined for node '$n+1$'. These two characteristics of the proposed framework, namely, 1) cooperatively learning the joint transmit and sleep schedules by the two RL agents, and 2) inter-node sharing of the confidence parameter to assist learning by the RL agents, give rise to cooperative learning behavior of the proposed approach.

Using this framework, each node learns an energy-conserving schedule independently. And that is without a centralized arbitrator. The detailed working of the two RL agents and their cooperative nature are explained below.

9.3.1 Transmission Scheduling Agent

The function of this agent is to learn a suitable transmission schedule for its host node. Finding a suitable transmission schedule can be tricky in that a conservative transmission periodicity can lead to excessive queue/buffer growth even though it will be favorable for better energy conservation. Having an aggressive schedule, on the other hand, can lead to low packet delay and drops (i.e., smaller queues) at the expense of excessive energy consumption, which may lead to a shutdown of the node due to exhausted energy that was harvested. Therefore, the objective is to develop a transmission schedule that can balance those two effects and be adaptive with the spatiotemporal-temporal characteristics of energy harvesting.

The MDP action space ($\mathcal{A}_T$) for this agent is defined by the probability of transmitting a packet (P_{tx}) in the queue. To keep the action space discrete, this probability is quantized into $|\mathcal{A}_T|$ discrete values in the range $[0, 1]$, where $|\mathcal{A}_T|$ denotes the cardinality of the action space. These actions are selected using ϵ-greedy policy in an RL decision epoch which is set to a duration of h frames.

The state space ($\mathcal{S}_T$) as perceived by the agent is represented by the energy influx to the node resulting from harvesting. The RL state at a decision epoch is given by the energy harvested at that epoch. Similar to the packet transmission probability, the harvested energy is also discretized into $|\mathcal{S}_T|$ distinct ranges. Formally, the state as perceived by an agent for node n at an epoch t is defined as $s_{n,Tx}(t) = g(\frac{1}{h}\sum_{k=t-h}^{t} E_{in,n}(k))$. Here $E_{in,n}(k)$ is the energy harvested in frame k by node n and $g(x)$ is the function for quantization as defined by:

$$g(x) = \begin{cases} s_T, & if\ -0.5 \leq s_T - 0.5 \leq 10x < s_T + 0.5 \leq 8.5 \\ 9, & if\ x \geq\ 0.85 \end{cases} \tag{9.4}$$

Here $\mathcal{s}_T \in I$ and $\mathcal{s}_T \geq 0$. Tabular Q-learning is used for solving the MDP, owing to the limited computation, storage, and processing capacity of the resource-constrained IoT/sensor nodes. The reward for the agent in node n at an epoch t in this setting is given by Eqn. 9.5.

$$\mathcal{R}^{(n)}(t) = \begin{cases} \mathcal{r}_P^{(n)}\left[1 - \delta_{P_{loss}^{Rx}(t),0}\right] + \mathcal{r}_Q^{(n)}(t) \times \delta_{P_{loss}^{Rx}(t),0}, if\ n \in \Psi' \\ \mathcal{r}_Q^{(n)}(t), \qquad\qquad if\ n \in \Psi \end{cases} \tag{9.5}$$

Here, $P_{loss}^{rx}(t)$ denote the number of missed packet reception at epoch t; $\delta_{i,j}$ is the Kronecker delta function; Ψ is the set of source nodes in the network; and $\mathcal{r}_P^{(n)}, \mathcal{r}_Q^{(n)}$ are the reward function associated with reducing missed packet receptions and maintaining a sustainable MAC packet queue respectively, as defined in Eqns (9.6) and (9.7) respectively.

$$\mathcal{r}_P^{(n)}(t) = \frac{1}{P_{rx}^{loss}(t)} \tag{9.6}$$

$$\mathcal{r}_Q^{(n)}(t) = \begin{cases} \frac{1}{\Delta Ql(t)} + \left(1 - \frac{1}{\Delta Ql(t)}\right) u(\Delta Ql(t) + \Gamma) + \frac{1-\Delta Ql(t)}{2\Delta Ql(t)} \delta_{\Delta Ql(t),-\Gamma}, \\ \qquad\qquad if\ n \in \Psi' \\ sgn(\Delta Ql(t) + \Gamma) - \delta_{\Delta Ql(t),-\Gamma}, \\ \qquad\qquad if\ n \in \Psi \end{cases} \tag{9.7}$$

In Eqns (9.6) and (9.7), $u(.)$ and $sgn(.)$ represent unit step and signum functions respectively. $\Delta Ql(t) = Ql(t) - Ql(t-1)$ is the temporal gradient of the MAC packet queue length which needs to be negative for a stable queue. $\Gamma > 0$ is a hyperparameter to ensure that learning does not get stuck in a local optimal point. The selection of these hyperparameters is detailed in Section 9.4.

The physical interpretation of the reward function can be explained as follows. As can be seen from Eqn. (9.5), there are two primary components of reward. First, the reward associated with successful packet reception $\mathcal{r}_P^{(n)}$ is an inverse function of missed packet reception rate (Eqn 9.6). It means that for the high value of the missed packet reception rate, the agent receives a low value

of reward. Similarly, in order to reduce the queuing delay, a negative temporal gradient of queue length is rewarded. This is captured by the reward component $r_Q^{(n)}(t)$ and is responsible for maintaining a stable queue. Note that, for a source node that is generating packets and has nothing to do with packet reception, the RL agent's only task is to reduce the packet delay and hence its reward only consists of the component $r_Q^{(n)}(t)$ (Eqns. (9.5, 9.6)).

Using the RL framework detailed above, the transmission scheduling agent learns to find a packet transmission schedule to reduce the missed packet receptions because of energy shortage and also to reduce the packet delay. However, to be noted that the transmission strategy learnt by this agent is contingent upon the transceiver on/off policy of the node. To exemplify, if the packet transmission probability of a node is P_{tx} and the node off probability is P_{off}, then the effective transmission rate of the node is given by $P_{tx} \times (1 - P_{off})$. In other words, a node's effective transmission probability is given by:

$$P_{tx}^{eff} = P[\text{a node transmits in a frame} | \text{ the node is on in that frame}].$$

Thus, the node's sleep decisions indirectly affect the learning behavior of the transmission scheduling. In addition, efficient sleep decisions also have a role to play in reducing the packet missed receptions. This calls for the requirement of a sleep scheduling agent that can cooperate with the transmission scheduling agent to achieve the predefined goals.

9.3.2 Sleep Scheduling Agent

The role of this agent is to find a transceiver '*On/Off*' schedule for the energy harvesting node it is part of. The goal of the agent is to find a schedule so that the limited energy budget of the node can be efficiently managed while maintaining reliable communication by reducing packet loss and packet delay. Sleeping for the right duration is important, because sleeping more may lead

to unintended packet missed receptions, while sleeping less can lead to high energy expenditure. Thus, using the RL-framework, this agent learns a sleep schedule in order to achieve the goal mentioned above.

The action space ($\mathcal{A}_S$) of the sleep scheduling agent is given by the probability of keeping the radio transceiver on (P_{on}) in an RL decision epoch of h frames. This probability is discretized into $|\mathcal{A}_S|$ discrete values in the range $[0, 1]$, where $|\mathcal{A}_S|$ denotes the cardinality of the action space of the sleep scheduling agent. The actions are selected using ϵ-greedy policy to maximize the expected long-term expected reward using Q-learning.

The state space ($\mathcal{S}_S$) for this agent is defined by the probability of packet transmission (P_{tx}) by the node in a learning epoch. The probability P_{tx} and hence the state of the sleep scheduling agent is directly controlled by the learning policy of the transmission scheduling agent. The logic behind using P_{tx} as the state for this agent is that the packet transmission probability and the transceiver '*on*' probability jointly determine the energy expenditure and throughput of the node. As a result, for a given transmission strategy decided by the transmission scheduler, this agent has to act accordingly to find a suitable transceiver on probability (P_{on}). Formally, the state perceived by the sleep scheduling agent for node n at a decision epoch t is given by Eqn 9.8 ($\mathcal{s}_S \in I$ and $\mathcal{s}_S \geq 0$).

$$s_{n,Sl}(t) = f\big(P_{Tx,n}(t)\big) = \mathcal{s}_s \; if \; \mathcal{s}_s \leq \; 10 \times P_{Tx}(t) \leq \mathcal{s}_s + 1 \tag{9.8}$$

Here, $P_{Tx,n}(t)$ is the probability of transmission by node n at epoch t. The actions taken by the sleep scheduling agent are evaluated using the same reward function given in Eqn. 9.5. The same reward is used for both the agents since they share the common objective of reducing missed

packet receptions and delays. These two agents, thus, cooperate with each other to jointly learn a policy that can achieve the above objective.

9.3.3 Learning Confidence Parameter Sharing

For achieving a cooperative learning behavior, each node piggybacks a Learning Confidence Parameter (LCP) in the MAC layer PDU. This inter-node parameter sharing mechanism is demonstrated by the red arrows in Fig. 9.2. The figure also shows the incoming MAC-PDU (received from node '$n-1$') and outgoing MAC-PDU (destined for node '$n+1$') for the node n. This Learning Confidence Parameter represents the confidence level in the action selection policy for that node and is computed as given in Eqn. 9.9.

$$LCP(t) = \min\left(\zeta \times \left(\left(1 - \Delta_w^{P_{on}}(t)\right) - \eta\right)^{\zeta}, 1\right) \tag{9.9}$$

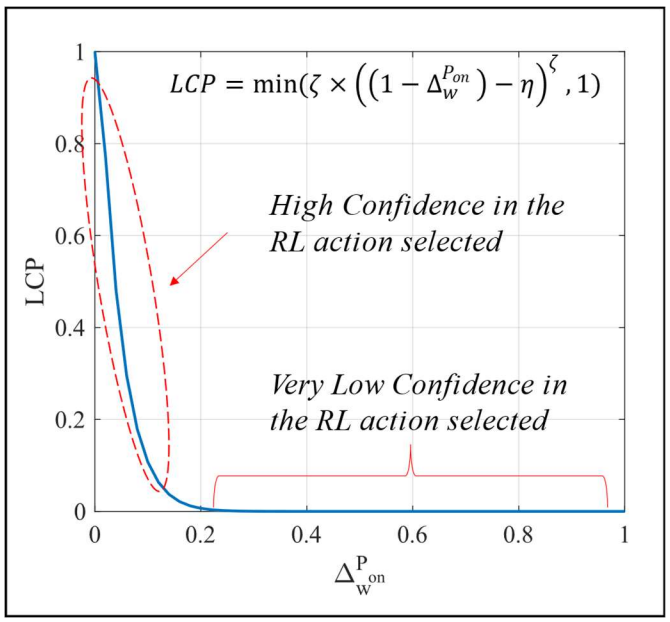

Figure 9.3: Computation of LCP from gradient of transceiver on probability.

Here, $\Delta_w^{P_{on}}(t) = P_{on}(t) - P_{on}(t-w)$ is the temporal gradient of transceiver on probability (P_{on}) over a window of w epochs. The learning confidence is low for a high value of this gradient $\Delta_w^{P_{on}}$ (more oscillatory actions) and vice-versa. A non-linear function is used to map $\Delta_w^{P_{on}}$ to Learning

Confidence Parameter, so that for the range of values of $\Delta_w^{P_{on}}$ for which the confidence in learning is low increases. This can be seen in Fig. 9.3, where LCP is plotted against $\Delta_w^{P_{on}}$ for $w = 1000$, $\zeta = 20$ and $\eta = 0.13$.

The significance of using *LCP* is that during the initial exploratory stage of learning, there may be bad action selection by a node. This leads to high missed packet receptions and thus affects the data rate of the flow it is part of. Hence, for the RL agents associated with the downstream nodes in the flow, it will lead to poor learning performance if their Q-values are updated when bad actions are chosen by for the upstream nodes' agents. As for example, in the network shown in Fig. 9.2, bad learning policies for node '$n-1$' would affect learning performance of node 'n' and '$n+1$', if nodes 'n' and '$n+1$' are not aware of confidence on the learning policies for node '$n-1$'. Sharing the *LCP* helps avoid such scenarios. This is because, while updating the Q-table, the temporal difference error is discounted by *LCP* for all nodes except the source. For the source node, the table update is done using the regular Bellman equation, since its learning is not dependent on the *'learning confidence'* of any other node in the network. The Q-value update equation is given by Eqn (9.10).

$$Q_n(s_n, a_n) \leftarrow \begin{cases} Q_n(s_n, a_n) + LCP \times \alpha \left[\mathcal{R}^{(n)} + \gamma \max_{\forall a'} Q\,(s_n', a_n') - Q(s_n, a_n)\right], \\ \qquad\qquad\qquad\qquad\qquad\qquad if\ n \in \Psi' \\ Q_n(s_n, a_n) + \alpha \left[\mathcal{R}^{(n)} + \gamma \max_{\forall a'} Q_n\,(s_n', a_n') - Q_n(s_n, a_n)\right], \\ \qquad\qquad\qquad\qquad\qquad\qquad if\ n \in \Psi \end{cases} \tag{9.10}$$

Here $s_n, s_n', a_n, \mathcal{R}^{(n)}$ denote the current state, next state, action, and reward respectively for node n. Note that the same update rule is used by both the Transmission Scheduling Agent and Sleep Scheduling agent. In addition to *LCP*, each node n also piggybacks the *Hop Index (HI_n)*, defined as the number of hops the node is away from the source (Fig. 9.2). This information is used in

determining the exponential decay rate in the ϵ-greedy exploratory behavior policy. Formally, the value of ϵ for a node n while following ϵ-greedy exploratory action selection is given by:

$$\epsilon_n(t) = \min\left(\epsilon_{max} e^{-\frac{t - HI_n \times \tau}{\epsilon_{dec}}}, 1\right) \tag{9.11}$$

In this equation, τ, ϵ_{dec} are learning delay and exponential decay rate respectively. The empirical selection of these hyperparameters is given in Section 9.4.

9.3.4 Cooperation among Learning Agents

As discussed in Section 9.3.1, 9.3.2 and 9.3.3, there are two RL agents deployed in each energy-harvesting sensor node that jointly learn the transmit-sleep schedules for that node. Moreover, each node shares a learning confidence parameter (LCP) with its one-hop neighbors in order to assist them with an informed-update of their own Q-tables. There are two levels of hierarchy in the Reinforcement Learning cooperation: intra-node cooperation and inter-node cooperation. The intra-node cooperation is between the two learning agents that jointly learn the transmit and sleep scheduling policies. The cooperation here is achieved by sharing the reward model and state information among the transmit and sleep scheduling learning agent towards the common objectives of minimizing packet loss and access delay. As has been discussed in Section 9.3.1 and 9.3.2, the state of the sleep-scheduling RL agent is controlled by the policies learnt by the transmission scheduling agent. The second level of cooperation is across the nodes. In this inter-node cooperation, two neighboring nodes cooperate by sharing learning control information, which is then used to update their respective Q-values. This ensures that the learning parameters are not updated for observables that are not reliable. The Q-table update is performed in each node in a decentralized manner, thus making the framework scalable with network size and degree.

It is to be noted that this decentralized setting is different from Federated Learning (FL) [99] approaches. In FL, the learning updates are performed locally at each device independently. The parameters or loss gradients are then periodically shared with a centralized arbitrator which then computes global model parameters using federated averaging. The updated global parameters are then shared back to the client devices. On the contrary, in the decentralized learning approach used here, the learning agents do not share model parameters with each other, and the table updates are done at each agent independently without relying on any central arbitrator.

Using the decentralized and cooperative multi-agent RL model detailed above, each IoT/sensor node learns a joint transmit-sleep policy in a situation-specific manner so as to manage available harvested energy efficiently to maintain a reliable communication in the resource-constrained network.

9.4 Experimentation

Experiments were performed to analyze the performance of the proposed scheduling protocol using a MAC layer simulator with embedded learning components. The baseline experimental parameters are tabulated in Table 9.1. The post-convergence performance of the learning-based MAC protocol is evaluated on the following metrics.

Missed Packet Reception Rate (P_{miss}) represents the rate of packet misses due to oversleeping and shortage of energy. Queue drop rate (Q_{drop}) indicates the rate of packet drops resulting from filled up MAC packet queue. Packet loss rate (P_{loss}) captures the combined packet losses resulting from both missed reception and full queue. Queueing delay is computed from queue length using Little's Law [100]. Packet Delivery Ratio (PDR) is the percentage of packets successfully received by the destination out of the total number of packets sent from the source.

The performance of the scheduling policy learnt using the proposed RL-framework is compared with the following scheduling approaches.

Table 9.1: Baseline Experimental Parameters

Parameter	**Value**
P_C	0.2
w_s	33.33 m/s
C	50 m
α	0.99
γ	0.1
h	200
B	150
P_{charge}	0.9
Ql_{max}	1000
$\lvert\mathcal{A}_T\rvert = \lvert\mathcal{A}_S\rvert$	10
τ_1	100
τ_2	1
ν	0.90
δ	0.001
α_T	2.4
P_{tx}	$0.353\ mW$
P_{ct}	$50\mu W$

Naïve Scheduling Policy: This is the baseline policy where a node is '*On'* in a frame with probability P_{on} and it transmits with probability P_{tx}. These probabilities are manually preset to fixed values and experiments are performed with different combinations of these probabilities to understand the variation of the performance variables (P_{miss} and packet delay) as response to these sleep-transmit decisions. We then analyze where the learning-based solutions lie in the space of those performance variables.

Battery State-based policy: This is an existing scheduling approach proposed in [93]. Here the node sleep decisions are based on the amount of available battery power in the sensor node. The node goes from active to sleep mode if the normalized battery capacity is below a preset threshold b_{low} and wakes up again when the battery state is above the threshold b_{high}. Mathematically, the transition probabilities from active to sleep states and vice-versa are formulated by Eqn. (9.12):

$$P_{a\to s} = \begin{cases} 1, & if\ \frac{b}{B} \le b_{low} \\ 0, & if\ \frac{b}{B} > b_{low} \end{cases} \text{ and } P_{s\to a} = \begin{cases} 1, & if\ \frac{b}{B} \ge b_{high} \\ 0, & if\ \frac{b}{B} < b_{high} \end{cases} \tag{9.12}$$

Queue length-based policy: This scheduling strategy decides sleep schedules based on the MAC queue length. The node sleeps with high probability if the queue length is higher than a predefined threshold Ql_{th} and vice versa.

$$P_{a\to s} = \begin{cases} \delta_1, if Ql \le Ql_{th} \\ \delta_2, if Ql > Ql_{th} \end{cases} and\ P_{s\to a} = \begin{cases} \delta_3, if Ql \le Ql_{th} \\ \delta_4, if Ql \le Ql_{th} \end{cases} \tag{9.13}$$

Solar radiation-based policy: This scheduling logic makes the node to go to sleep with probability ϕ_s and to wake up with probability 1 and ϕ_w, when the solar radiation state (r) is low and high respectively. Formally, this can be expressed as

$$P_{a\to s} = \begin{cases} \phi_s, if\ r = 0 \\ 0, if\ r > 0 \end{cases} and\ P_{s\to a} = \begin{cases} \phi_w, if\ r = 0 \\ 1, if\ r > 0 \end{cases} \tag{9.14}$$

Hybrid policy: The hybrid scheduling policy takes both queue length and battery power into consideration for making scheduling decisions. This hybrid policy is given by Eqn. (9.15):

$$P_{a\to s} = \begin{cases} 1, if\ \frac{b}{B} \le b_{low}\ or\ Ql = 0 \\ 0, \qquad\qquad if\ \frac{b}{B} > b_{low} \end{cases}, P_{s\to a} = \begin{cases} 1, if\ \frac{b}{B} \ge b_{high} \\ 0, if\ \frac{b}{B} < b_{high} \end{cases} \tag{9.15}$$

Note that the existing scheduling strategies mentioned above: *Battery State-based, Queue length-based, Solar radiation-based and hybrid policies* are proposed in [93]. The hyperparameters used for these policies are chosen based on what is suggested by the authors in [93]: $b_{low} =$

$0.04. b_{high} = 0.2, Ql_{th} = 3, \delta_1 = 0.8, \delta_2 = 0.2, \delta_3 = 0.3, \delta_4 = 0.7, \phi_s = 0.56, \phi_w = 0.44.$

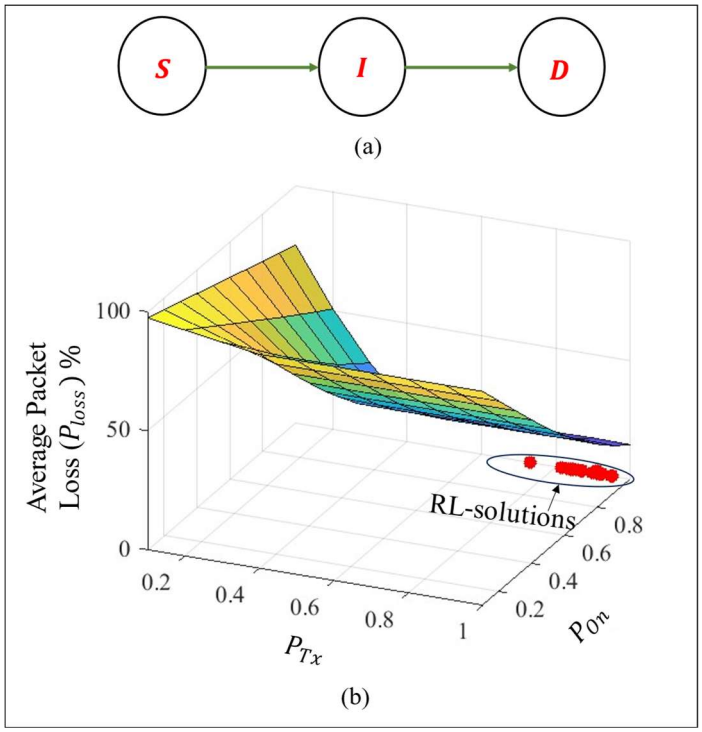

Figure 9.4: Average packet loss with naïve policy and proposed RL-based policy.

9.5 Results and Analysis

With the aim of understanding the working of the proposed framework, we first experiment with a simple linear topology and then with more complex topologies. For the liner topology case, a source node transmits packets at a rate of $\lambda = 0.4$ packet per frame (ppf). In this scenario, intermediate node I is the energy harvesting node (Fig. 9.4 (a)). First, to get an insight into the baseline performance, we experiment with the naïve scheduling policy, where the node is kept 'On' in a frame with probability P_{on} and it transmits with probability P_{tx}. The surface plot in Fig. 9.4 (b) shows the variation of packet loss rate P_{loss} for different values of P_{on} and P_{tx}. It is observed that with an increase in 'node on' probability P_{on}, there is a decrease in P_{loss}. This is because, higher the value of P_{on}, more likely the node is to remain On in a frame and hence there is less probability of missed packet reception. Another point to note in the plot is that, for low

values of P_{tx}, the gradient of decrease in P_{loss} with P_{on} decreases. This is because, for low probability of packet transmission, queues build up and get full, resulting in packet drops. As a result, for high probability of the node being On, even if the missed packet reception is low, still there is high P_{loss} because of packet drops resulting from full MAC packet queues. Fig. 9.5 demonstrates the variation of queuing delay with the scheduling probabilities P_{tx} and P_{on}. As seen from the figure, the queuing delay decreases with increase in P_{tx}. This is because, with an increase in P_{tx}, missed packet receptions increase and there are less packets received. Also, the queue service rate increases with an increase in P_{tx}. This causes the queue length, and hence queuing delay, to decrease with P_{tx}. Note that there is a range of values of P_{tx} for which the gradient in decrease in packet delay is very high. This range represents the scenario when the queue service rate μ gets close to the effective flow data rate λ_{eff}. In other words, when $\mu < \lambda_{eff}$, the queue is stable and packet delay is low; and when $\mu > \lambda_{eff}$, it leads to unstable queue and packet delay is very high. However, the effects of P_{on} on queuing delay is not significant.

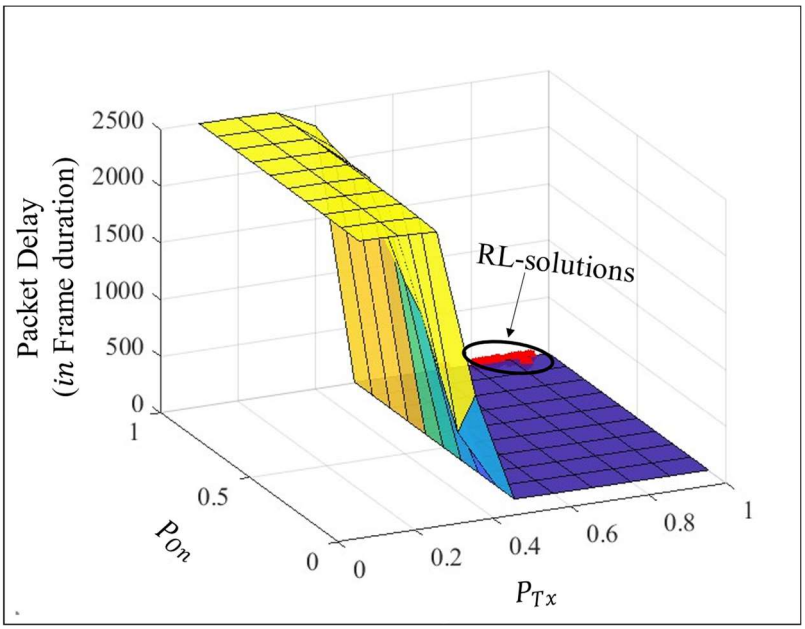

Figure 9.5: Average packet delay with naïve policy and proposed RL-based policy.

This is because, the expected queue length ($\mathbb{E}[N_Q]$) as defined by Eqn. 9.16 does not get affected by P_{on}, since the flow rate and service rate both are affected by P_{on} by the same amount.

$$\mathbb{E}[N_Q] \propto \frac{\rho^2}{1-\rho}, \text{ where } \rho = \frac{p_{on} \times \lambda}{p_{on} \times \mu} = \frac{\lambda}{\mu} \tag{9.16}$$

Thus, the desired scheduling policy should be such that the packet drops resulting from missed receptions and queue drops should be reduced while still maintaining a stable queue. To be noted that the variation of P_{loss} with p_{tx} and p_{on} is dependent on the energy harvesting parameters and network traffic, and hence the static policies cannot find the right balance among all the above-mentioned performance metrics.

The performance achieved by the RL-based scheduling approach is shown by the red points on the surface plot in Fig. 9.4 and 9.5. It can be observed that the proposed RL-based mechanism finds sleep and transmission schedules that give lower packet loss rate and packet delay than the static naïve scheduling policy. Note that the schedules learnt by the cooperative RL approach have lower value of P_{loss} for the same P_{tx}, P_{on} values used in the naïve scheduling approach. This is because the RL approach allows the nodes to learn dynamic sleep-transmit scheduling. In other words, the learnt scheduling probabilities oscillate epoch by epoch. To exemplify, a transmit probability of 0.6 can indicate refraining from transmission deterministically for four

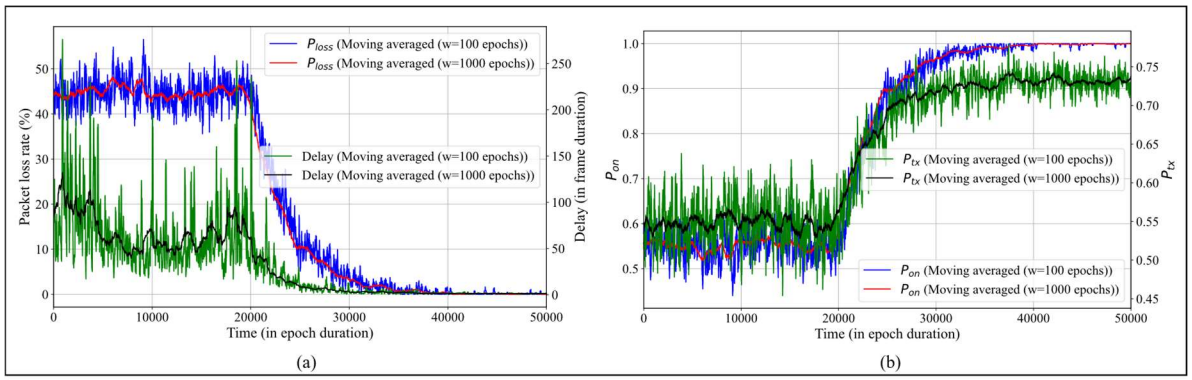

Figure 9.6: RL convergence behavior with respect to (a) Packet loss rate and delay and (b) sleep and transmit probabilities.

consecutive epochs (800 frames), thus recharging its battery, and then transmitting for next six epochs (1200 frames), thus, utilizing its recharged battery. It was observed that such dynamic policies learnt by the RL agents helped the node to obtain a packet loss (P_{loss}) rate less than the static naïve policies for the same $< P_{tx}, P_{on} >$ tuple.

The learning convergence behavior of the proposed architecture in terms of packet loss rate and packet delay is shown in Fig. 9.6 (a) and (b). It is observed that initially, at the start of the learning, P_{loss} and packet delay is high; but over time, the nodes learn to find transmission and sleep scheduling policies, so that both packet loss rate and packet delay go down. Another observation is that on an average, the learning convergence time remains in the vicinity of 10^4 frame durations. For a typical MAC frame duration of 3-4 ms [101], the RL convergence can happen within 30-40 sec. This makes the approach highly practical in that it can cope with network

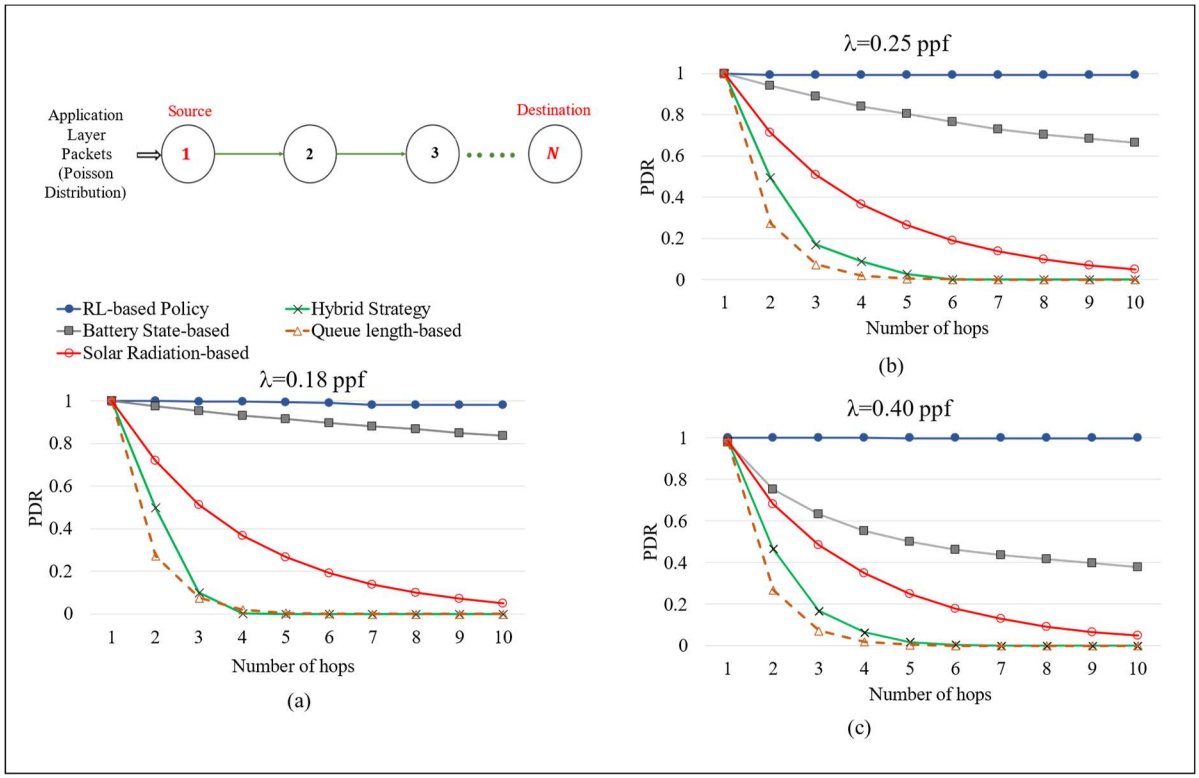

Figure 9.7: Performance comparison of proposed RL-based protocol with existing approaches for a linear topology with multiple hops.

condition changes with a time constants that are order of 10s to 100s of minutes. For many static (i.e., non-mobile) sensor networks, the time constants for network topology and traffic condition changes can be even larger, often up to hours or days.

The performance of the proposed learning-driven scheduling logic for topologies with multiple hops between the source and destination is shown in Fig. 9.7. Comparison is done with the existing sleep-scheduling protocols: *Battery State-based, Queue length-based, Solar radiation-based and hybrid policies.* Experiments are performed for three different data rates: $\lambda = 0.18, 0.25$ and 0.40 ppf. It is observed that with increase in the number of hops between source and destination node, the packet delivery ratio (PDR) falls more drastically for the other scheduling logics as compared to that of the proposed RL-based scheduling approach. The reason behind this behavior is that the proposed scheduling mechanism allows the nodes to cooperatively learn the transmit and sleep scheduling strategy by sharing the online Learning Confidence Parameter (LCP) within the localized neighborhood. This piggybacked information is used by the learning agents to avoid bad updates of Q-table (as explained in Section 9.3) and helps in maintaining performance for networks where the source and destination nodes are separated by many numbers of hops. As seen from the figure, with the increase in flow data rate, the benefit of the proposed framework in terms of PDR becomes more significant. This is

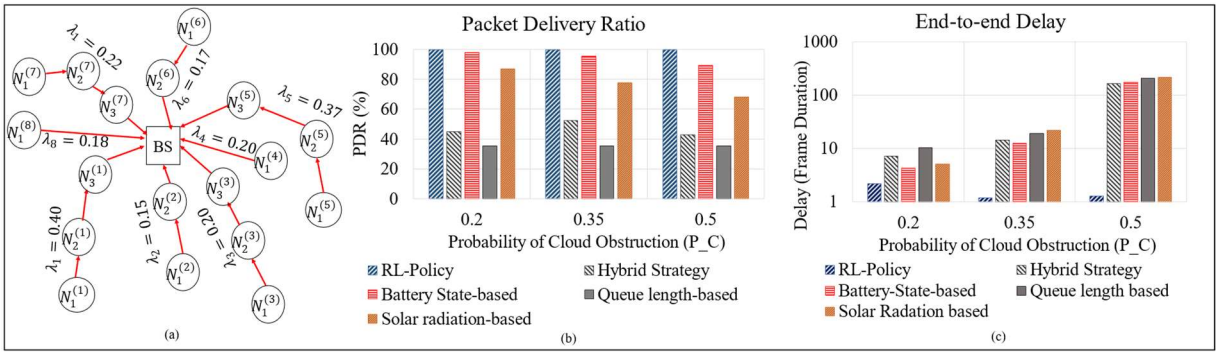

Figure 9.8: Performance comparison of proposed RL-based protocol with existing approaches for different solar radiation conditions in a heterogeneous topology with 8 flows.

because, for high data rate, there is a high packet loss rate due to energy shortage when the existing scheduling approaches are used. But, using the proposed learning-based scheme, the transmit and sleep scheduling agents learn cooperatively to find a joint transmit-sleep schedule to reduce the packet losses and hence to achieve a high packet delivery ratio.

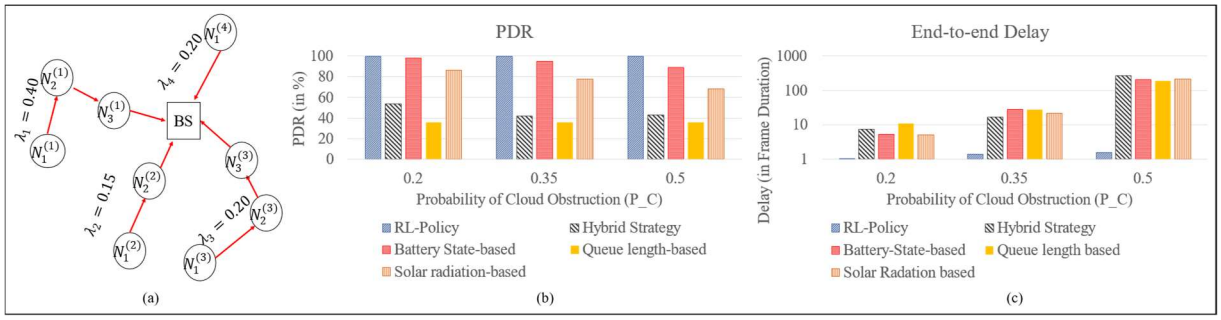

Figure 9.9: Performance comparison of proposed RL-based protocol with existing approaches for different solar radiation conditions in a heterogeneous topology with 4 flows.

To test the generalizability of the proposed framework, we experimented with heterogeneous networks consisting of multiple flows with different flow lengths and different Poisson data-rates in each flow (Fig. 9.8 (a) and 9.9 (a)). Two networks (with 8 and 4 flows) are considered with multi-point to point traffic, where the flow rates are shown in the figures. The plots in Fig. 9.8 and 9.9 compare the performance of the proposed RL-based scheduling policy with the other baseline approaches mentioned above. It is observed that for different values of cloud obstruction probability (P_C), the packet delivery ratio is high and end-to-end packet delay is less for the proposed RL-based scheduling logic. With increase in cloud obstruction probability, packet drops due to energy shortage increases for the *Battery State-based, Queue length-based, Solar radiation-based, and hybrid sleep scheduling policies.* However, the learning-driven transmit-sleep scheduling strategy makes the nodes learn to manage the energy budget so that the packet delivery ratio remains high and packet delay remains low. This makes it suitable for energy-constrained wireless sensor and IoT networks to efficiently utilize resources to maintain sustainable communication.

9.6 Summary

In this chapter, a cooperative Reinforcement Learning architecture is developed for energy management in energy-harvesting wireless sensor networks. This decentralized framework allows the nodes to learn a joint transmit-sleep scheduling policy in order to efficiently utilize the lean energy budget of the sensor nodes. And it is done while maintaining reliable communication in terms of low packet loss rate and packet delay. A cooperative two-agents RL model is developed. Using that model, the nodes learn a situation-specific transmit-sleep schedule. An online learning confidence parameter sharing module allows the nodes to learn scheduling policies by ignoring unreliable RL observations. This makes the system scalable for long flows with many hops between the source and destination. With simulation studies, the developed RL-based scheduling policy is shown to outperform existing MAC sleep schedulers, especially for networks with multiple hops between source and destination. The framework is experimented with scenarios of heterogeneous topologies, traffic scenarios and energy harvesting conditions. The simulation results demonstrate that the proposed learning-driven scheduling approach achieves a packet delivery ratio that is $\approx 60\%$ higher than the existing mechanisms for a network with 10-hops between source and destination. Moreover, for adverse energy harvesting conditions, the developed framework is shown to achieve $\approx 10\%$ increase in packet delivery ratio and two-orders of magnitude lower end-to-end delay than the existing scheduling mechanisms.

The learning mechanisms developed so far operate under the assumption of full cooperation among the RL agents. That is, it is assumed that all nodes aim to learn policies that can boost performance of the entire network. However, real-world scenarios may involve malicious nodes attempting to disrupt the performance of other nodes (learning agents). The subsequent chapters

in this thesis consider the presence of such malicious nodes in the network and performance of the learning-synthesized protocols under such circumstances is investigated.

Chapter 10: Slot Allocation using Multi-Armed Bandits in the Presence of Malicious Nodes

The frameworks developed till this point in the thesis rely on the foundational assumption that all the learning agents cooperate with one another to learn policies that enhance the overall performance of the system. However, implementation of these approaches in real world environments introduces a new layer of complexity, as there may exist malicious nodes intending to degrade the performance of rest of the network. In this chapter, we delve into the implications of these malicious nodes on network performance. Our focus shifts towards examining the behavior and performance of learning-synthesized protocols when faced with such adversarial circumstances.

10.1 Motivation

Malicious nodes attempting to disrupt TDMA slot arrangements can lead to reduced MAC-layer throughput, delay, and other performance. Situations can be aggravated in a distributed TDMA-based approach, in which there is no central arbitrator to allocate slots, and as such, the detection of malicious nodes in the network is difficult. This chapter proposes a learning based TDMA slot allocation scheme that enables participating wireless nodes, in a decentralized manner, to learn how to detect such adverse conditions caused by allocation attacks and change transmission strategies in real-time for minimizing the effects of such attacks.

We consider an attack model in which multiple malicious nodes attempt to access wireless TDMA slots, with the goal of disrupting slot allocation of TDMA policy-complying non-malicious nodes. The malicious nodes follow various quasi-random slot allocation policies in order to achieve the above objective towards reducing throughput experienced by the non-

malicious nodes. Note that this attack model does not assume collusions among the malicious nodes. As a defense mechanism to such attacks, a Multi-Armed Bandit (MAB) learning-driven approach is proposed as the slot-allocation policy for the non-malicious nodes. The objectives of these nodes are to: (i) minimize throughput reduction caused by the malicious nodes, (ii) reduce the effective throughput experienced by the malicious nodes themselves. It is assumed that the non-malicious nodes are not aware of the specific attack policies used by the malicious nodes. They only react to the network level effects of malicious activities.

One key characteristic of the proposed framework is that the slot allocation schemes by the nodes are decentralized in nature. In other words, each wireless node independently selects TDMA slots without explicit information sharing with its peers. This decentralized setup makes slot-allocation more challenging in the presence of malicious nodes. This is because the inference made by a non-malicious node about the current and past channel access status is deliberately corrupted by the malicious nodes' non-complying, and thus harmful access behavior. In order to accomplish successful slot allocation in the presence of such unreliable information, two MAB-based schemes are proposed for non-malicious nodes' slot allocation policies: *reactive policy* and *robust policy*. While the goal of the reactive policy is to maximize the non-malicious nodes' own throughput, the robust policy aims at restricting the malicious nodes' share of wireless bandwidth. The performance of these policies and their trade-offs are analyzed for different networking scenarios and application-specific requirements.

The learning-driven slot allocation scheme achieves several benefits as compared with the existing scheduling mechanisms [51] [53] [58] [27] [102] [103] [104] [105] [106]. First, as mentioned earlier, the traditional TDMA slot scheduling approaches are pre-programmed, and as a result, these are inefficient in handling heterogeneities and dynamics of the network traffic

and topological conditions. Second, most of the existing protocols do not consider the presence of malicious nodes in the network. As pointed out earlier, decentralized learning in the presence of malicious nodes is challenging, which is dealt with in this work. Moreover, these existing allocation logics mostly cannot adapt to dynamic and stochastic slot allocation behavior of the malicious nodes. This leads to low network throughput in addition to wastage of precious energy resources as a result of inefficient slot allocation policies.

10.2 Network and Traffic Model

In order to maintain generality, multi-point to multi-point networks with mesh topologies are considered. In a network of N nodes, K number of nodes are assumed to be malicious, where $K < N$. Fig. 10.1 shows a representative partially connected mesh topology with 13 nodes, out of which 2 are malicious. At the MAC layer, it is assumed that the network time is synchronized. Time is slotted and the MAC frames are of fixed size, which is dimensioned *a priori* based on the average degree of the network graph.

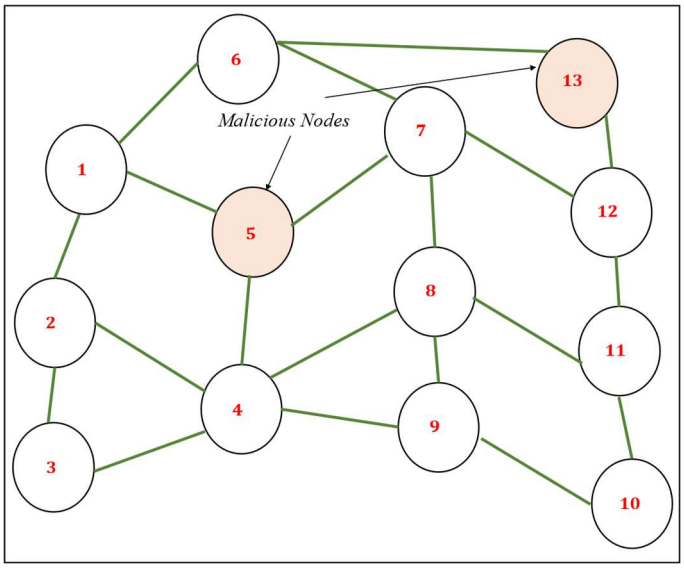

Figure 10.1: A representative network topology with 2 malicious nodes.

Two different traffic models are used for application layer packet generation at the source nodes: (i) Constant Bit Rate (CBR), and (ii) Poisson distributed. For both, the mean packet generation rate is denoted by λ packet per MAC frame (ppf).

As an attack policy, the malicious nodes select transmission slots in a frame using various quasi-random slot allocation policies as detailed in Section 10.4. The goal of the malicious nodes is to select TDMA slots so as to minimize the throughput as experienced by the non-malicious nodes. The non-malicious nodes use a Multi-Armed Bandit Learning-based approach for selecting MAC slots so as to minimize the damage caused by the malicious nodes. The details of the learning approach for slot-selection are furnished in Section 10.6.

10.3 Slot Allocation Policies of Non-Malicious Nodes

In the absence of malicious nodes, the TDMA slot allocation problem boils down to each node independently finding a non-overlapping slot. This is achieved using a Multi-Armed Bandit (MAB) Learning-based framework, where each wireless node acts as an F-Armed Bandit agent. Here F is the frame size in number of slots, which is preset based on the network size/degree. The MAB-enabled slot allocation in the absence of malicious nodes is detailed in chapter 5. Where the MABs arm or action is defined by the selection of a transmission slot (out of F slots) in the MAC frame. After transmitting a packet in the selected slot, the bandit or the node receives feedback from the environment whether the transmission was a success or collision. Based on this feedback, the bandit computes a numerical reward for evaluating the taken action.

In this setting, the reward is modeled such that for a collision-free transmission, the agent receives a reward of $+1$. A reward of 0 is received if the selected slot overlaps with transmission from other nodes. Formally, the reward function $R_i(t)$ for node i at a learning epoch t can be denoted

as:

$$R_i(t) = \begin{cases} 1, & success \\ 0, & collision \end{cases} \tag{10.1}$$

The goal of the agent is to find arm/action that maximizes its expected long-term reward.

10.4 Threat Models and Performance Objectives

The threat model used in this work assumes the presence of K number of malicious nodes in an N-node network, where $K < N$. It is assumed that these nodes are unaware of the MAC slot scheduling strategies followed by the non-malicious nodes. In this chapter, we assume that malicious nodes work independently without collusions. That is, each malicious node selects TDMA slots independently and without any information sharing with each other. The primary goal of the malicious nodes is to reduce throughput of the non-malicious nodes by disrupting their slot allocation. This is done while trying to maximize their own throughput.

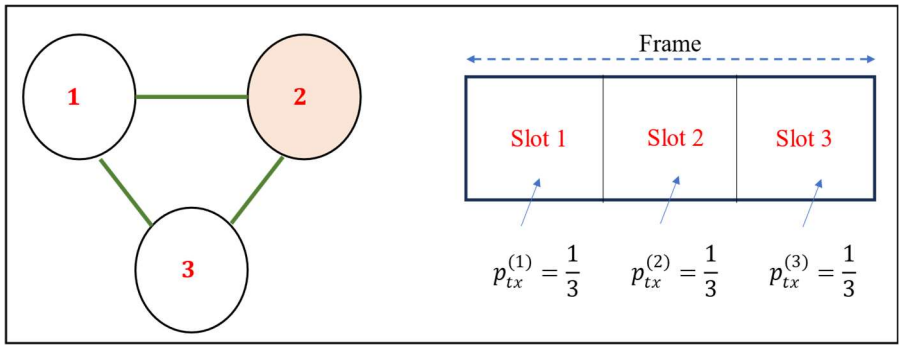

Figure 10.2: An example scenario of slot allocation policy of malicious nodes.

In order to achieve the above objective, each of the malicious nodes randomly selects a slot in the frame of F slots. Although different distributions can be followed by the malicious nodes for selecting slots, here we consider uniform $\mathcal{U}(1, F)$ and truncated normal distribution [107] [108] for validating the concept. The idea behind using a uniform distribution is that uniform distribution $\mathcal{U}(a, b)$ has the maximum entropy among all continuous distributions supported in the interval $[a, b]$. Using a high entropy distribution by the malicious nodes makes it challenging

for the non-malicious nodes to find a collision-free slot allocation policy. This can be explained using a scenario of three-nodes network as shown in Fig. 10.2, where node 2 is malicious. It selects a slot in the MAC frame following a uniform distribution $\mathcal{U}(1,3)$. This means that the probability of selecting each slot s $(1 \leq s \leq 3)$ by the malicious node is equal to $1/3$. Now, from the non-malicious nodes' perspective, selection of each slot in the frame has equal probability of collision. Now, when a non-malicious node selects one of those slots and experiences a collision, it will change its slot in the next frame in order to avoid the collision. It will also update its MAB learning parameters, assuming that the collision is the result of some other non-malicious node using that slot for transmission. However, even in the next frame, it will have the same probability of collision, because of the uniform distributed scheduling policy of malicious node. In addition, whenever a non-malicious node encounters a collision, it cannot figure out whether the collision is because of an overlapping transmission from other non-malicious nodes or from the malicious ones.

The stochasticity of the slot allocation mechanism by the malicious nodes can be varied in order to intensify the disruption of the non-malicious nodes' slot scheduling. This is accomplished by adding an additional degree of freedom in malicious nodes' lot selection policy, known as 'hop duration'. The hop duration defines the number of frames a malicious node waits before changing its currently selected slot. This change in slot is done again following a uniform random distribution $\mathcal{U}(1, F)$. As an example, hop duration 1 indicates that the malicious node would select a random slot in every frame for transmitting its packet. Similarly, hop duration of 'h' indicates that after a slot 's' is selected for transmission, it will continue its transmission in that slot 's' for h frames before randomly selecting another slot '$\hat{s}$'. This makes it challenging for the non-malicious nodes to estimate or learn the slot allocation mechanism of the malicious nodes.

This is because, for a large hop duration, the non-malicious nodes would assume that the malicious nodes slot allocation does not change over time. When the malicious nodes change their transmission slots after hop duration of h frames, it would not only lead to collisions in the system, but also disrupt the knowledge of non-malicious nodes regarding the slot allocation of other nodes. This is because, when the malicious nodes change transmission slot after h frames, the resulting collisions will change the estimated MAB reward value for that slot. In the absence of an efficient learning mechanism, the non-malicious nodes would have to start over again to find a collision-free slot schedule.

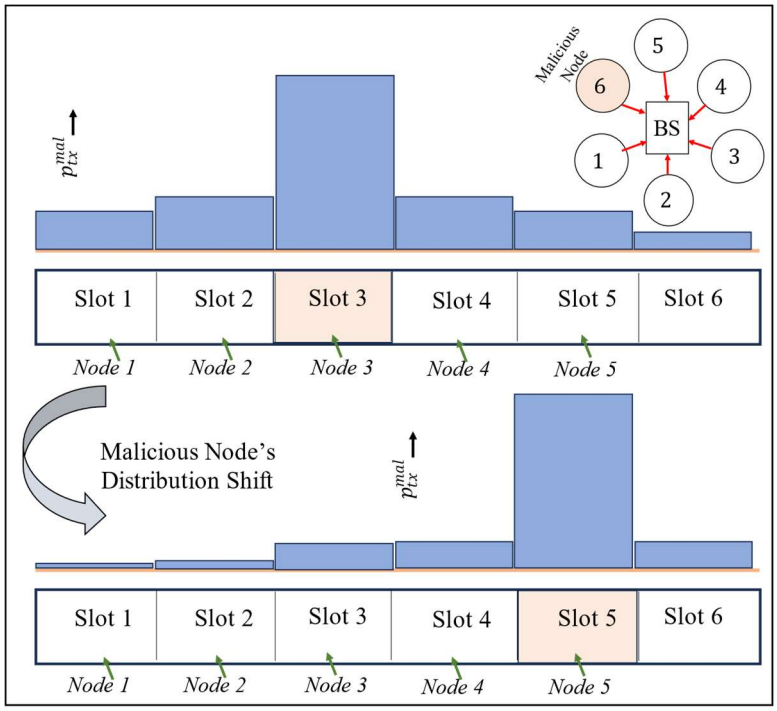

Figure 10.3: Malicious nodes changing distribution to attack targeted nodes.

This chapter also considers the scenario when the malicious nodes select slots randomly following a truncated normal distribution $\psi(\bar{\mu}, \bar{\sigma}, a, b)$, where $\bar{\mu}$, $\bar{\sigma}$, are the truncated normal mean and standard deviation; and a, b $(0 < a, b \leq F)$ are the truncation bounds. In the absence of any defense mechanisms by the non-malicious nodes, using such a bell-shaped distribution

allows the malicious nodes to lower throughput of certain targeted nodes more than others. This is because, when the malicious nodes select slots in a frame using a normal distribution, it can modulate the likelihood of collisions for each slot in the frame by tuning $\bar{\mu}$ and $\bar{\sigma}$. This is demonstrated in an example scenario shown in Fig. 10.3, where, by varying the mean values of the distribution, the malicious node can change its target from node 3 to node 5.

In the presence of this threat model, the non-malicious nodes have two primary objectives: (i) to minimize the disruption in network performance (in terms of throughput) caused by the malicious nodes and (ii) to reduce the bandwidth share of malicious nodes so as to prevent them from sending any malicious data. The non-malicious nodes achieve the above objectives by using a Multi-Armed Bandit-driven learning framework as detailed in Section 10.6.

10.5 Benchmark Throughput in the Presence of Malicious Nodes

In this section, we derive the benchmark throughput that can be achieved in the presence of the threat model stated above. Let us consider a wireless network, where $\mathcal{N}$ is the set of non-malicious nodes and $\mathcal{M}$ is the set of malicious nodes. Without loss of generality, let us consider a multi-point-to-point network, where all the nodes send fixed-size packets to a base-station. Let us define the frame size as F slots, where $F = |\mathcal{N}| + |\mathcal{M}|$. Let $\mathcal{F}$ denotes the set of all slots in the frame.

As discussed earlier, the malicious nodes use uniform or truncated normal distribution to select slots in the frame. Now, the best slot allocation policy for the non-malicious nodes is to co-operate among themselves to avoid collisions. In spite of such collaboration, the non-malicious nodes cannot avoid the collisions that are caused by the malicious nodes' transmissions.

In this situation, the probability that a packet transmitted by a non-malicious node gets collided

is given by the conditional probability:

$$P_{coll,\mathcal{N}} = P[Node\ m \in \mathcal{M}\ selects\ slot\ 's'|\ Node\ n \in \mathcal{N}\ selects\ a\ slot\ 's']$$

$$= P[m_1(s) \cup m_2(s) \cup \dots \dots \cup m_{|\mathcal{M}|}(s)]$$

$$= P[\bigcup_{i \in \mathcal{M}} m_i(s)] \tag{10.2}$$

Here, $m_i(s)$ denotes the event that malicious node $m_i \in \mathcal{M}$ selects slot s. Therefore, the throughput of the non-malicious nodes is given by:

$$S_{\mathcal{N}} = 1 - P_{coll,\mathcal{N}}$$

$$= 1 - P[\bigcup_{i \in \mathcal{M}} m_i(s)] \tag{10.3}$$

First, we will derive the benchmark throughput for the case when the malicious nodes follow uniform distribution for slot allocation. In this case, the above Eqn. (10.3) can be expanded as:

$$S_{\mathcal{N},U} = 1 - \sum_{i=1}^{\left\lceil \frac{|\mathcal{M}|}{2} \right\rceil} \binom{|\mathcal{M}|}{2i-1} \times \frac{1}{F^{(2i-1)}} + \sum_{i=1}^{\left\lfloor \frac{|\mathcal{M}|}{2} \right\rfloor} \binom{|\mathcal{M}|}{2i} \times \frac{1}{F^{(2i)}} \tag{10.4}$$

Now, the probability that a packet transmitted by a malicious node m_k gets collided is given by the conditional probability $P_{coll,\mathcal{M}}$:

$$P_{coll,\mathcal{M}} = P[\{Node\ n \in \mathcal{N}\ selects\ slot\ 's' \cup Node\ m'$$

$$\in \{\mathcal{M} - m_k\}\ selects\ slot\ 's'\}|\ Node\ m_k \in \mathcal{M}\ selects\ a\ slot\ 's']$$

$$= P[Node\ n \in \mathcal{N}\ selects\ slot\ 's'] + P[Node\ m' \in \{\mathcal{M} - m_k\}\ selects\ slot\ 's'] - P[\{Node\ n$$

$$\in \mathcal{N}\ selects\ slot\ 's'\} \cap \{Node\ m' \in \{\mathcal{M} - m_k\}\ selects\ slot\ 's'\}]$$

$$= \frac{|\mathcal{N}|}{F} + P\left[\bigcup_{i \in \{\mathcal{M} - m_k\}} m_i(s)\right] - \frac{|\mathcal{N}|}{F} \times P\left[\bigcup_{i \in \{\mathcal{M} - m_k\}} m_i(s)\right] \tag{10.5}$$

The throughput of the malicious nodes is given by:

$$S_{\mathcal{M},U} = 1 - P_{coll,\mathcal{M}}$$

$$= 1 - \left(\frac{|\mathcal{N}|}{F} + P\left[\bigcup_{i \in \{\mathcal{M} - m_k\}} m_i(s) \right] - \frac{|\mathcal{N}|}{F} \times P\left[\bigcup_{i \in \{\mathcal{M} - m_k\}} m_i(s) \right] \right) \tag{10.6}$$

$$= 1 - \Bigg(\frac{|\mathcal{N}|}{F} + \left(1 - \frac{|\mathcal{N}|}{F}\right) \times \left(\sum_{i=1}^{\left\lceil \frac{|\mathcal{M}|-1}{2} \right\rceil} \binom{|\mathcal{M}| - 1}{2i-1} \times \frac{1}{F^{(2i-1)}} - \sum_{i=1}^{\left\lfloor \frac{|\mathcal{M}|-1}{2} \right\rfloor} \binom{|\mathcal{M}| - 1}{2i} \times \frac{1}{F^{(2i)}} \right) \Bigg) \tag{10.7}$$

The network throughput in this case is given by:

$$S = \frac{|\mathcal{M}|}{|\mathcal{M}| + |\mathcal{N}|} S_{\mathcal{M},U} + \frac{|\mathcal{N}|}{|\mathcal{M}| + |\mathcal{N}|} S_{\mathcal{N},U}$$

Now, instead of transmitting in every frame, if a malicious node m_i transmits packets at the rate of λ_i packets per frame (ppf), then the throughput of non-malicious and malicious nodes can be derived from Eqns. (10.3) and (10.6) respectively as follows:

$$S_{\mathcal{N},U} = 1 - P[\bigcup_{i \in \mathcal{M}} m_i(s; \lambda_i)] \tag{10.8}$$

$$S_{\mathcal{M},U} = 1 - \left(\frac{|\mathcal{N}|}{F} + P\left[\bigcup_{i \in \{\mathcal{M} - m_k\}} m_i(s; \lambda_i) \right] - \frac{|\mathcal{N}|}{F} \times P\left[\bigcup_{i \in \{\mathcal{M} - m_k\}} m_i(s; \lambda_i) \right] \right) \tag{10.9}$$

Here, $m_i(s; \lambda_i)$ denotes the event that malicious node $m_i \in \mathcal{M}$ transmits packet in slot s with a rate of λ_i ppf. When all the malicious nodes transmit with the same transmission rate (that is, $\lambda_i = \lambda, \forall i \in \mathcal{M}$), then Eqn. (10.8) and (10.9) simplifies to:

$$S_{\mathcal{N},U} = 1 - \sum_{i=1}^{\left\lceil \frac{|\mathcal{M}|}{2} \right\rceil} \binom{|\mathcal{M}|}{2i-1} \times \left(\frac{\lambda}{F}\right)^{(2i-1)} + \sum_{i=1}^{\left\lfloor \frac{|\mathcal{M}|}{2} \right\rfloor} \binom{|\mathcal{M}|}{2i} \times \left(\frac{\lambda}{F}\right)^{(2i)} \tag{10.10}$$

$$S_{\mathcal{M},U} = 1 - \left(\frac{|\mathcal{N}|}{F} + (1 - \frac{|\mathcal{N}|}{F}) \times P\left[\bigcup_{i \in \{\mathcal{M} - m_k\}} m_i(s; \lambda_i) \right] \right)$$

$$= 1 - \left(\frac{|\mathcal{N}|}{F} + \left(1 - \frac{|\mathcal{N}|}{F}\right) \times \left(\sum_{i=1}^{\left\lceil \frac{|\mathcal{M}|-1}{2} \right\rceil} \binom{|\mathcal{M}|-1}{2i-1} \times \left(\frac{\lambda}{F}\right)^{(2i-1)} - \sum_{i=1}^{\left\lfloor \frac{|\mathcal{M}|-1}{2} \right\rfloor} \binom{|\mathcal{M}|-1}{2i} \times \left(\frac{\lambda}{F}\right)^{2i} \right) \right) \tag{10.11}$$

Next, let us consider the scenario when a malicious node follows a truncated normal distribution $\psi_{mal}(\bar{\mu}, \bar{\sigma}, a, b; x)$ for slot allocation ($0 < a, b \leq F$). From the perspective of the non-malicious nodes, the ideal scenario is when they learn to avoid transmission in the slots where the likelihood of malicious transmission is the highest. The benchmark throughput for the non-malicious nodes, in this case, then becomes:

$$S_{\mathcal{N},N} = 1 - P_{coll,\mathcal{N}}$$

$P_{coll,\mathcal{N}}$ is the probability of collision experienced by the non-malicious nodes given by:

$$P_{coll,\mathcal{N}} = \frac{1}{F-1} \mathcal{L}(\psi_{mal}) \tag{10.12}$$

$$S_{\mathcal{N},N} = 1 - \frac{1}{F-1} \mathcal{L}(\psi_{mal}) \tag{10.13}$$

Here, $\mathcal{L}(.)$ is a function for computing the likelihood of malicious node selecting a slot other than slot s^*, where $s^*$is defined as:

$$s^* = argmax_{\forall x \in \mathcal{F}}(\psi_{mal}(\bar{\mu}, \bar{\sigma}, a, b; x))$$

Then the function $\mathcal{L}(\psi_{mal})$ can be expressed as:

$$\mathcal{L}(\psi_{mal}) = \sum_{k \in \mathcal{F} - s^*} \int_{x=k-1.5}^{k-0.5} \psi_{mal}(\bar{\mu}, \bar{\sigma}, a, b; x) dx \tag{10.14}$$

The malicious throughput in this case becomes:

$$S_{\mathcal{M},N} = max_{\forall x \in \mathcal{F}}(\psi_{mal}(\bar{\mu}, \bar{\sigma}, a, b; x)) \tag{10.15}$$

In the presence of multiple malicious nodes with the slot selection probability distribution $\psi_{mal}^{(i)}$, the benchmark throughput values can be computed simply by considering the joint distribution followed by the malicious nodes, $\psi_{mal}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\sigma}}, \mathbf{a}, \mathbf{b}; \mathbf{x})$ in Eqns. 10.13-10.15. The above equations define the benchmark throughput for the non-malicious nodes, and the corresponding throughputs for the malicious nodes for the given threat model. Now, the objective of the learning-driven MAC protocol is to make the non-malicious nodes learn slot scheduling policies in a decentralized manner such that the above benchmark throughput can be achieved.

10.6 Slot Allocation in the Presence of Malicious Nodes

Transmission scheduling for a non-malicious node, in this context, is to find a slot in the MAC frame such that its throughput can be maximized. It should be noted that in this scenario, in which the malicious nodes use random distribution to allocate slots, it is not possible to obtain a completely collision-free system. The primary objective of the learning mechanism here is to make the nodes learn schedules that give throughput close to the benchmark throughput derived in Section 10.5.

As explained earlier in section 10.3, each node acts as a Multi-Armed Bandit agent that interacts with the wireless network (learning environment) via its arms/actions. For each arm selected, the agent receives feedback in terms of a numerical reward. The goal of each bandit is to find

arm/action that maximizes its expected long-term reward.

Two different action selection policies are used to accomplish the learning objectives in the presence of malicious nodes: (a) Reactive Policies, and (b) Robust Policies. The details of these policies are given below.

Reactive Policies: The idea of reactive slot-scheduling policies is to make the non-malicious nodes change their slot allocation based on network dynamism caused due to quasi-random slot-scheduling followed by the malicious nodes. The goal of these policies is to make the non-malicious nodes maintain high throughput in scenarios in which the malicious nodes change their allocated slot after a hop duration of h frames.

Two different action selection policies are used as reactive policies: ϵ-greedy and Upper Confidence Bound (UCB) [12]. The action selection logic for each of these policies for a node n, at a learning decision epoch t, can be formulated as follows:

$$A^{(n)}_{\epsilon-\text{greedy}}(t) = \begin{cases} \text{randomly select a transmission slot } s \text{ with probability } \epsilon \\ argmax_{\forall s \in \mathcal{F}} V_t^{(n)}(s), \text{ with probability } (1-\epsilon) \end{cases} \quad (10.16)$$

$$A^{(n)}_{UCB}(t) = argmax_{\forall s \in \mathcal{F}}(V_t^{(n)}(s) + c\sqrt{\frac{\ln(t)}{N_t(s)}}) \quad (10.17)$$

In Eqns. (10.16) and (10.17), $V_t^{(n)}(s)$ is the value of the arm s (selecting slot s) at epoch t for node n; $N_t(s)$ defines the number of times slot s has been selected for transmission till epoch t; $\mathcal{F}$ is the set of all the slots in the MAC frame; and c, ϵ are learning hyperparameters for controlling the exploration-exploitation trade-off. The value update equation for both these policies is given by:

$$V_t^{(n)}(s) = V_t^{(n)}(s) + \alpha(r_n(t) - V_t^{(n)}(s)) \quad (10.18)$$

Here, α is the learning rate and $r_n(t)$ is the reward received by bandit n at epoch t.

Note that for the ϵ-greedy policy, the agent constantly explores with probability ϵ, so that the algorithm does not get stuck to a sub-optimal action. Similar behavior is achieved using the coefficient c in the UCB action selection policy (Eqn. 10.17). Using this exploratory behavior, these policies help the agent to update its value function to account for any dynamism in the environment, which is the wireless network.

The ability of these policies to react to network dynamics makes these applicable in situations where the malicious nodes follow a quasi-random slot allocation policy by not changing the slot in every frame. That is, when the malicious nodes wait for a hop duration of h frames to stochastically change its transmission slot. In such scenarios, these reactive policies make the non-malicious nodes adapt their slot scheduling policies, when the malicious nodes switch to a different slot after a hop duration of h frames.

It will be shown later in Section 10.7 that these reactive policies are useful when the objective is to maximize the throughput of the non-malicious nodes without caring for the malicious nodes' throughput. However, in addition to maximizing non-malicious nodes' throughput when the application requires restricting the throughput of the malicious nodes, then these policies cannot provide the desired performance.

Robust Policies: In order to address the limitations of the reactive policies mentioned above, we have developed a robust slot-scheduling approach. The idea behind this is to make the learning policies of non-malicious nodes less reactive to the hop duration in slot selection by malicious nodes. This would prevent the malicious nodes from getting more wireless bandwidth by using a large hop duration while switching slots in the frame.

For implementing robust slot allocation approach, Thompson Sampling [109] [110] is used as

MAB action selection policy. Thompson Sampling is a Bayesian approach to deal with the exploration-exploitation dilemma in MAB action selection. In this approach, a prior probabilistic reward distribution is associated with each arm of a bandit. Here, we use β- distribution as the prior. As learning progresses, the bandit sees more and more samples of the reward for the arm selected, and thus, update the distribution associated with the arm. The parameters $\left(\alpha_s^{(n)}, \beta_s^{(n)}\right)$ of node n's β- distribution associated with arm (slot) 's' are updated at epoch t using the following equation:

$$(\alpha_s^{(n)}, \beta_s^{(n)}) = \begin{cases} (\alpha_s^{(n)}, \beta_s^{(n)}), \ if\ action\ selected \neq s \\ \left(\alpha_s^{(n)}, \beta_s^{(n)}\right) + (r_n(t), 1 - r_n(t)), if\ action\ selected = s \end{cases} \quad (10.19)$$

After each learning epoch, samples are picked following each arm's β- distribution and the arm with the highest sample value is selected. As learning progresses, these distributions converge to the true reward distributions of the bandit arms. This makes sure that the arm with the highest expected reward value is selected. Note that as compared with the ϵ-greedy and UCB action selection policies, Thompson Sampling has slow convergence in general, since the entire distribution parameters are updated during learning. However, the likelihood of selecting optimal action is high.

Using the learning framework detailed above, each non-malicious node learns to find a TDMA slot schedule such that the throughput reduction caused by the malicious nodes' non-cooperative strategies is minimized and the bandwidth share of malicious nodes is reduced. This is done by each node in a completely decentralized manner and without explicit share of learning policies among them.

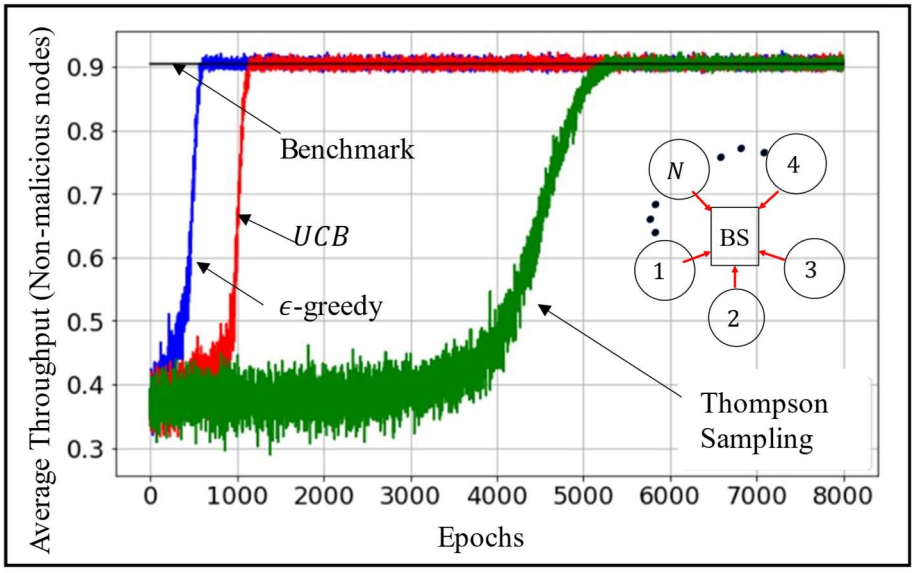

Figure 10.4: Convergence behavior of MAB-driven slot allocation.

10.7 Experiments and Results

To implement the proposed protocol, the MAB model is embedded on top of the MAC layer functions. The baseline learning related parameters in the experiments are $\epsilon = e^{-t/50}, c = 0.15, \alpha = 0.01$. To demonstrate the working of the proposed framework, we first experimented with a multi-point-to-point network with 50 wireless nodes (i.e., $N = 50$) sending data to a base station (Fig. 10.4). Initially we consider the scenario with one malicious node in the network (i.e., $|\mathcal{M}| = 1$) that follows uniform distribution to choose slot with hop duration $h = 1$. The learning

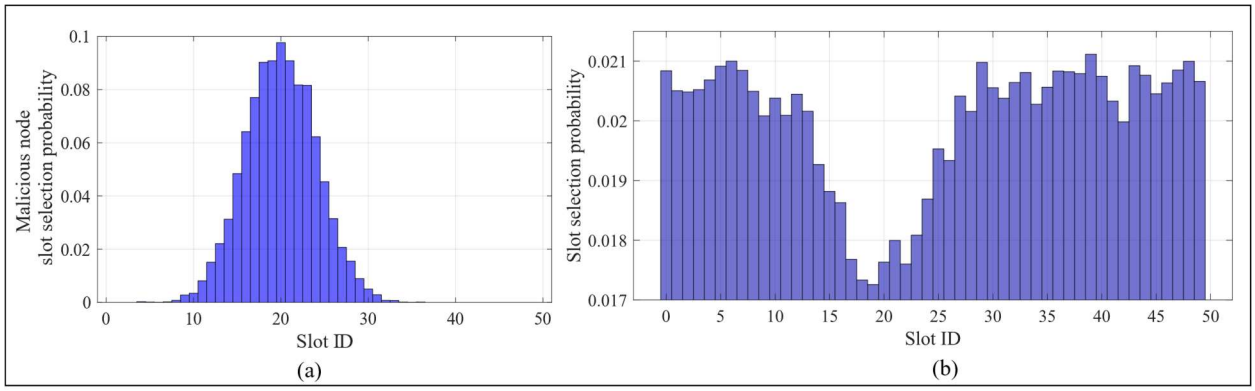

Figure 10.5: (a) Truncated Normal distribution policy for malicious nodes' slot allocation (b) non-malicious nodes' learnt slot selection policy as a response to malicious nodes' policy.

convergence behavior of the non-malicious nodes is shown in Fig. 10.4. It can be observed that at the initial stage of learning, the average throughput of the malicious nodes is low. This is because of the collisions experienced due to overlapping transmissions by other nodes. Note that both malicious and non-malicious nodes are responsible for these collisions. However, as learning progresses, each of the non-malicious nodes independently learns to find a transmission schedule that does not overlap with each other. The average throughput after learning convergence equals the theoretical benchmark derived in Eqn. (10.4) and is shown by the black horizontal line in the figure. This observation remains valid for all the three different MAB action selection policies. It is to be noted that because of the non-compliant slot schedule followed by the malicious nodes, there are still collisions that remains after the learning convergence, which restricts the throughput from reaching to hundred percent.

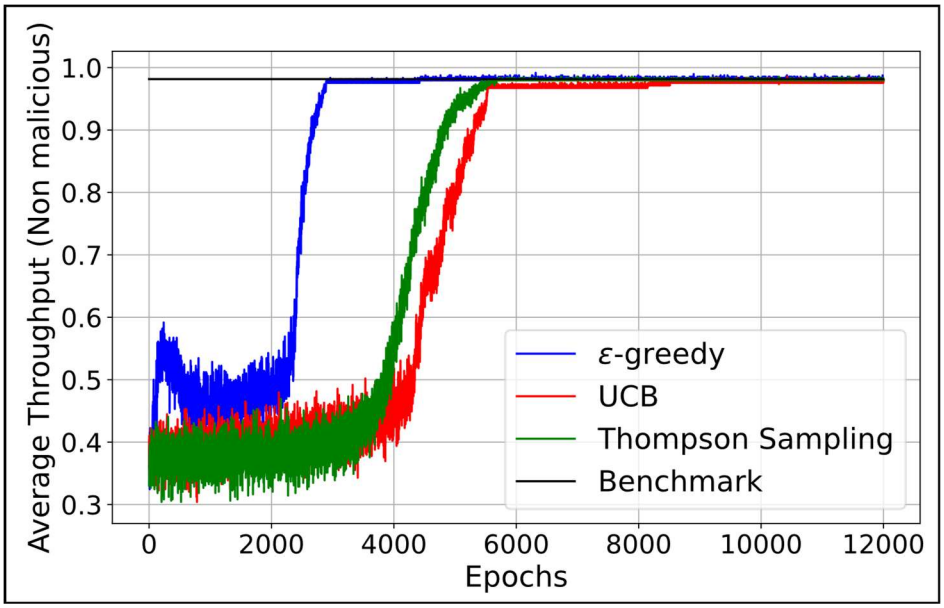

Figure 10.6: Learning convergence behavior of non-malicious nodes for truncated Normal distribution policy followed by malicious nodes.

Next, we consider the case when the malicious node follows truncated Normal distribution $\psi(\bar{\mu}, \bar{\sigma}, a, b)$, with $\bar{\mu} = 20$, $\bar{\sigma} = 4$, $a = 0$, $b = 40$. The slot section probability for the malicious nodes is represented by the distribution shown in Fig. 10.5 (a). Now, when the non-malicious

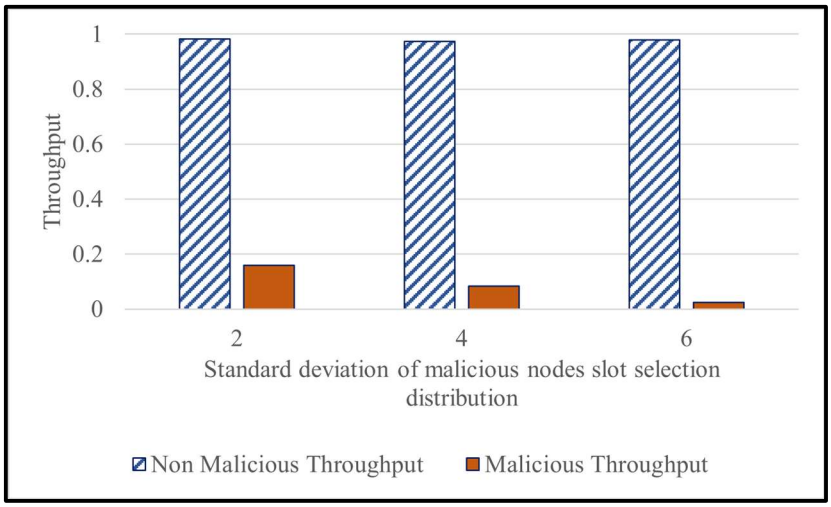

Figure 10.7: Effect of standard deviation of malicious slot allocation on throughput.

nodes use MAB-driven slot allocation policy, the non-malicious nodes learn to avoid those slots that have highest transmission probabilities by the malicious nodes. This can be observed from Fig. 10.5 (b), which depicts the slot selection probability distribution of non-malicious nodes. It is observed that there is a dip in the probability of slot selection around the mean ($\bar{\mu} = 20$) of truncated Normal distribution followed by the malicious nodes. In this way, the non-malicious nodes learn to minimize the collisions resulting from the malicious nodes' slot selection policies. Note that this is learnt without any prior knowledge of the malicious nodes' slot selection policies. The learning convergence behavior in this case is shown in Fig. 10.6. The takeaway from the figure is that the non-malicious nodes learn policies to reach the theoretically derived benchmark throughput.

Fig. 10.7 shows the effect of standard deviation of the distribution used by malicious nodes on network performance. It is observed that the throughput achieved by non-malicious nodes is hardly sensitive to the standard deviation of malicious nodes' policies. However, from malicious nodes' perspective, throughput reduces with increase in standard deviation, as it increases the probability of collision with other network nodes, because of heavy tail in the distribution of

malicious nodes' slot selection.

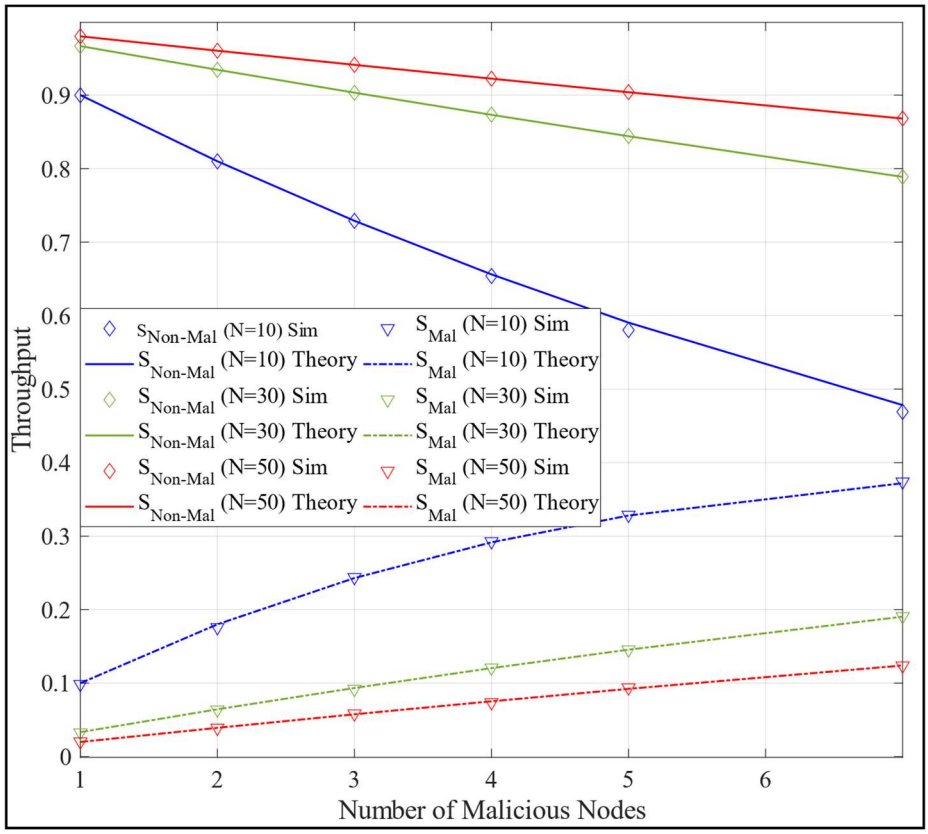

Figure 10.8: Effects of network size on performance.

Simulations were subsequently performed to understand the scalability of the proposed architecture with increased network size. The change in average throughput for both malicious and non-malicious nodes with increase in network size is shown in Fig. 10.8. The following observations can be made. First, the learning-driven slot allocation scheme is able to find a schedule that gives a throughput that is equal to the theoretically computed benchmarks (Eqn. (10.4) and Eqn. (10.7)). As seen from the figure, this observation holds for different number of malicious and non-malicious nodes. Second, for a given network size, the increase in malicious nodes reduces the throughput of the non-malicious nodes. This is because of the increase in collisions resulting from non-complying malicious behavior. The effects of addition of malicious

nodes decreases with increase in network size.

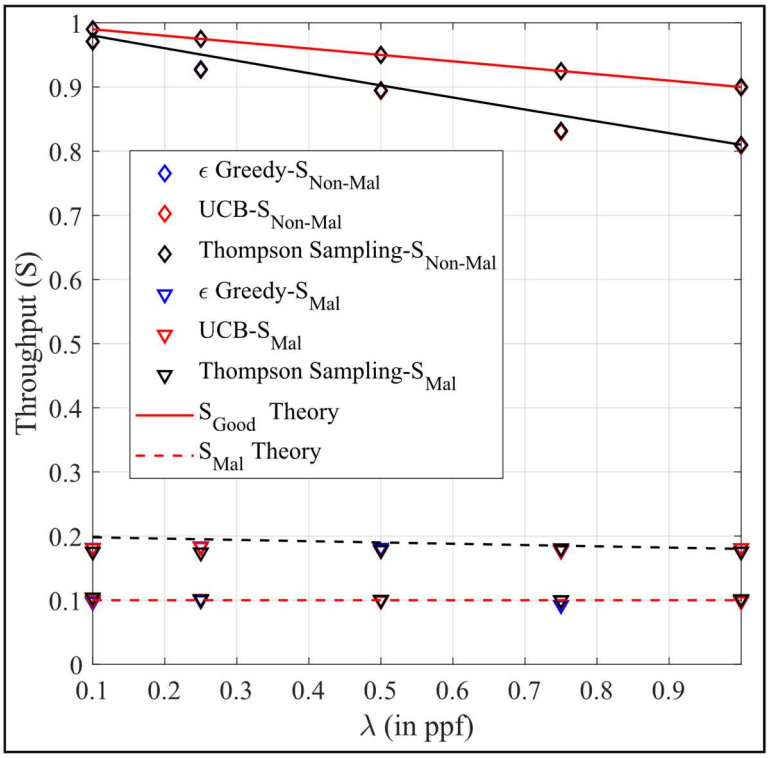

Figure 10.9: Effect of transmission rate (λ) of malicious nodes on performance.

Fig. 10.9 shows the variation of throughput due to change in the packet transmission rate λ by the malicious nodes. It is observed that with an increase in the rate of malicious nodes packet transmissions, the throughput of the non-malicious nodes goes down. This is due to the increase in the probability of collisions with an increase in the packet transmission rate by the malicious nodes. However, the throughput variation of the malicious nodes with increase in the packet transmission rate is very low. This is because, even when the packet transmission rate is low, all the slots are selected with equal probability, and whenever a packet is transmitted, the chance of collision is close to the case with high transmission rate. Note that the ability of the learning framework to make the non-malicious nodes learn slot scheduling that can give performance equal to theoretical benchmark holds for different values of packet transmission rate.

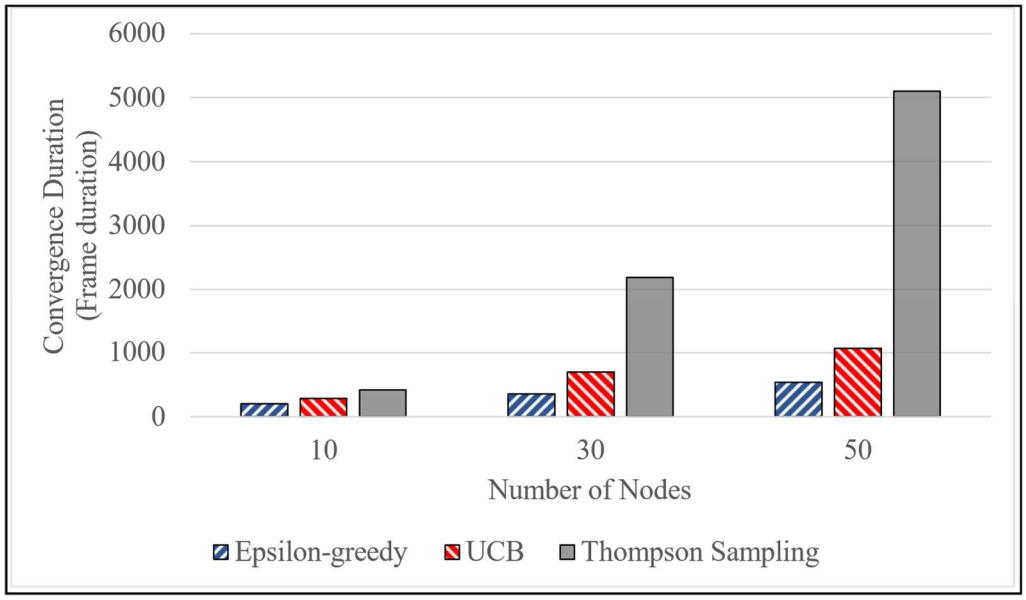

Figure 10.10: Convergence speed of different action selection policies.

While all the three MAB action selection policies are able to obtain the corresponding benchmark throughputs, it is observed in Fig. 10.10 that the convergence is slow for the robust policy as compared to the reactive ones. The reason is that the robust policies use Thompson Sampling as the action selection mechanism. In Thompson Sampling, learning is relatively slow since the entire distribution parameters need to be updated during learning. Also, among the reactive policies, ϵ-greedy converges faster compared to UCB. Since all the slots in the frame has an equal probability of being the optimal MAB arm, exploring with equal probability is suitable in this case. This provides faster convergence for the ϵ-greedy-based reactive slot scheduling policy. To summarize, for this scenario with the malicious node changing slot every frame ($h = 1$), the reactive policy (ϵ-greedy) performs better in terms of fast learning convergence.

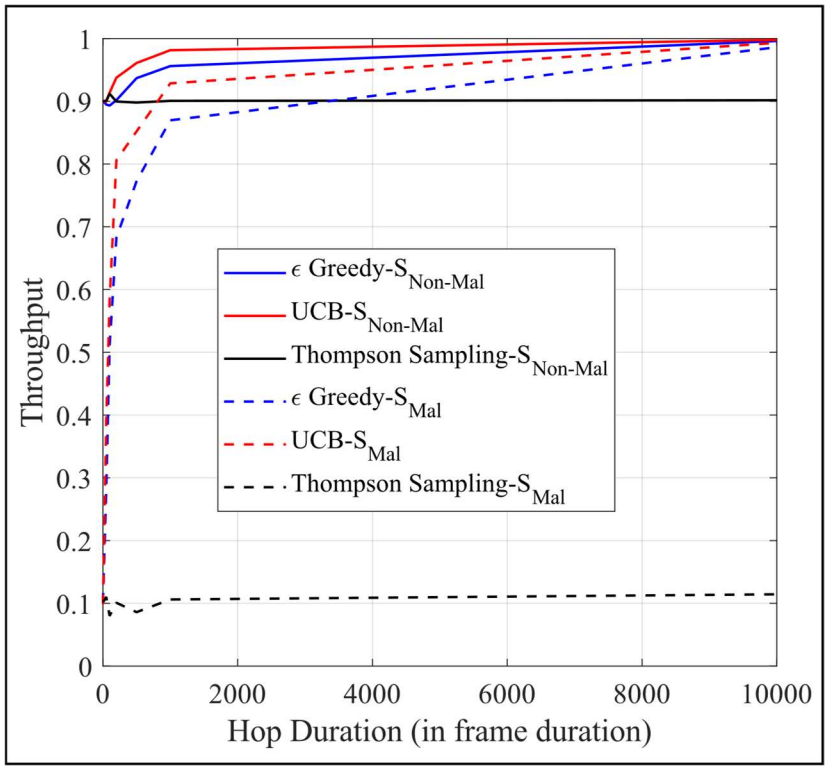

Figure 10.11: Performance of different policies with different hop durations adopted by the malicious nodes.

The plots shown in Fig. 10.11 demonstrate the throughput variation caused due to different hop durations followed by two malicious nodes in a 10-nodes network. When the non-malicious nodes run reactive policies, with such an increase in hop duration, the share of throughput that the malicious nodes can siphon out increases. At the same time, the throughputs of the non-malicious nodes also go up. This is because, when the hop duration is large, non-malicious nodes, using reactive policies, have enough time to update their Q-tables as per the malicious nodes' latest transmission slot. As a consequence, the collisions suffered by non-malicious nodes go down. As an extreme case, when the hop duration is sufficiently large ($h \gg c$, where c is the convergence duration), the non-malicious nodes find collision-free slots and throughput $S_{\mathcal{N}} \to 1$. Note that since the overall collisions in the network are reduced, this appears to be beneficial for both malicious and non-malicious nodes. However, the robust scheduling policy prevents the malicious nodes from receiving a higher share of bandwidth by using a large hop duration for

slot scheduling. This is because the action selection policy in Thompson Sampling is probabilistic, and parameters of the β-distribution in Thompson Sampling is based on the past globally observed samples. This makes the scheduling policy robust to the scheduling dynamics followed by the malicious nodes. This makes the robust policies suitable for scenarios where the goal is to reduce the bandwidth share of the malicious nodes.

Finally, as shown in Fig. 10.12, the proposed architecture is tested in a mesh network with two

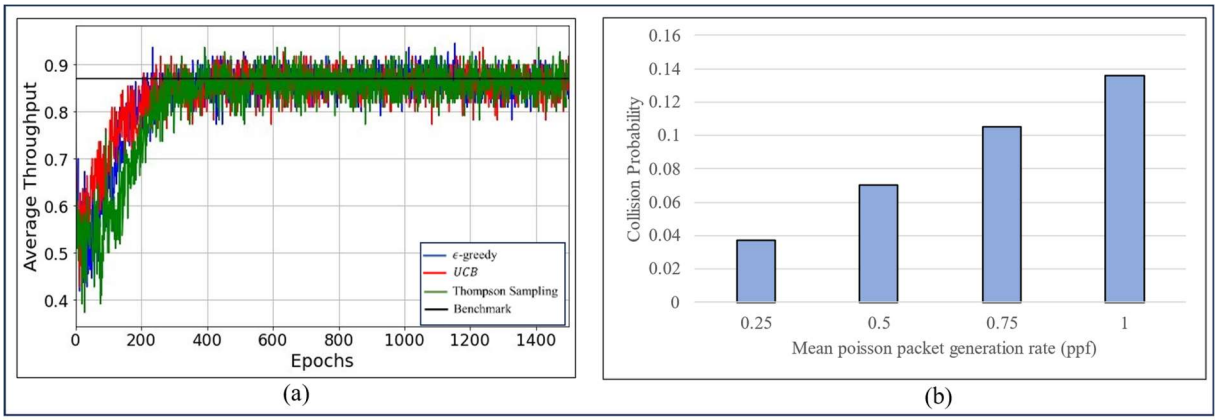

Figure 10.12: Performance of the framework for a mesh network shown in Fig. 10.1

malicious nodes (Fig. 10.1). The packet generation follows a Poisson distribution with the mean rate of λ packet per frame. The key observation here is that in a decentralized manner, the non-malicious nodes are able to find a schedule that gives a throughput equal to the theoretical benchmark. Another observation here is that the packet collision rate increases with an increase in the Poisson packet generation rate. This is because of an increase in the probability of selection of the same slot by the malicious and non-malicious nodes.

10.8 Summary

In this chapter, we analyze the concept of learning-enabled protocol synthesis in the presence of MAC layer attacks in wireless networks. Specifically, we consider an attack model in which multiple malicious nodes attempt to access wireless TDMA slots, with the goal of disrupting slot

allocation of TDMA policy-complying non-malicious nodes. An online learning mechanism for the non-malicious nodes is proposed with a two-dimensional goal: first, minimizing the throughput reduction caused by the malicious nodes and second, reducing the effective throughput experienced by the malicious nodes themselves. Two different learning policies, *robust* and *reactive* policies are proposed for the slot scheduling problem for different application-specific requirements. The reactive policies aim at maximizing the throughput of the non-malicious nodes without caring for the malicious nodes' throughput. On the other hand, the objective of the robust policies is to keep the throughput share of the malicious nodes at check. An analytical model is developed to find the benchmark network throughput in the presence of malicious nodes. It is shown that the proposed learning-based technique allows the nodes to learn policies to achieve that benchmark throughput. The proposed framework is validated for various slot-allocation policies adopted by the malicious nodes, and for different network topologies and traffic conditions. Building on the insights developed in this chapter, we consider a more adversarial threat model in the next chapter, where the malicious nodes collude to amplify the throughput reduction of the non-malicious nodes. The next chapter will focus on evaluating the performance of the MAB-enabled slot scheduling approach under such challenging conditions posed by the adversaries.

Chapter 11: Slot Allocation in the Presence of Colluding Malicious Nodes

11.1 Motivation

In the previous chapter, it was assumed that the malicious nodes independently select TDMA slots without any communication among them. In this chapter, we relax that assumption to allow the malicious nodes to collude and share information to increase the damage to the non-malicious nodes. The overall goal of the malicious nodes is to avoid collisions among themselves, and to increase collisions with the non-malicious nodes. This would result in reduction of throughput of the non-malicious nodes.

The high-level motivation behind the research in this chapter is still the same as the previous one. That is, to demonstrate the efficacy of the learning-driven slot allocation strategy in the presence of malicious nodes. Specifically, the framework addresses the disruptive impact of malicious nodes on TDMA slot arrangements, which can significantly reduce MAC-layer throughput and increase delay in wireless networks. The learning mechanism is developed for a generalized scenario of decentralized TDMA-based systems, lacking a central server, where detecting and mitigating such attacks become challenging. The proposed framework aims to enable wireless nodes to autonomously identify and respond to malicious slot allocation attacks in real-time. This is achieved by minimizing the throughput reduction caused by malicious nodes and improving network performance under dynamic and stochastic slot allocation scenarios.

A few works have been developed for investigating security issues and handling malicious behavior in wireless access control [111] [112] [113] [114]. The papers [115] [116] [117] [118] use machine learning algorithms to detect the presence of malicious nodes and other cyber

attacks. The work in [112] propose a centralized TDMA scheme for minimizing wireless bandwidth share in vehicular networks. This cannot be directly applied to many applications due to inherent limitations arising from the centralized TDMA approach. The work in [113] propose mechanisms to deal with spectrum sensing data falsification attacks in cognitive radio networks. This does not consider non-compliance of malicious nodes in MAC slot allocation. The works in [114], [111] aim at detecting attacks for resource allocation using machine learning. These papers do not aim at approaches to handle those attacks after successful detection. In addition, the mechanisms in papers [102] [103] [104] [105] rely on centralized or pre-programmed slot allocation for TDMA slot allocation.

In this research, a Multi-Armed Bandit-driven slot allocation scheme is developed which can detect and adjust to situations when malicious nodes attempt to destroy the TDMA-complying

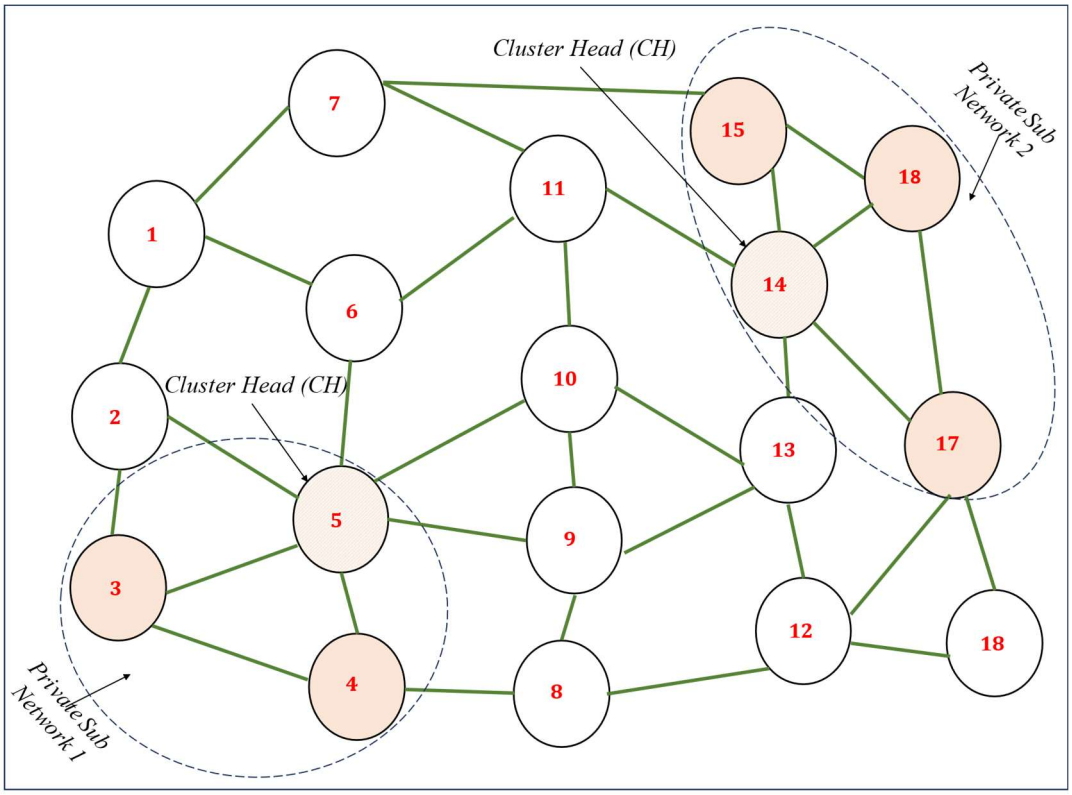

Figure 11.1: Collusion among malicious nodes.

slot allocation. This is accomplished on-the-fly and without the requirement of any central controller, thus making it suitable for use in distributed TDMA settings. This property also makes the learning-driven system scalable for large networks and mesh topologies.

11.2 Threat Models and Performance Objectives

The threat model assumes the presence of K number of malicious nodes in an N-node network, where $K < N$. It is assumed that these nodes are unaware of the MAC slot scheduling strategies followed by the non-malicious nodes.

The malicious nodes that are directly connected form a private subnetwork among themselves. One of the nodes in each subnetwork selects the slots for all other malicious nodes in that subnetwork using a uniform random distribution. This is demonstrated in Fig. 11.1, where malicious nodes 3, 4 and 5 collude to form a private subnetwork. Similarly, nodes 14, 15, 17 and 18 form another subnetwork. To avoid collisions within a private subnetwork, the transmission slots for the nodes in it are chosen to be individually unique. This, however, increases the likelihood of collisions with the non-malicious nodes, which are desirable from the standpoint of the malicious nodes. This can be explained further using an example of a simple three-nodes network (Fig. 11.2), in which nodes 2 and 3 are malicious. Let us consider a scenario when node 1 (non-malicious) selects slot 1 in the frame. When malicious nodes do not collude, then the probability of nodes 2 and 3 selecting slot 1 is given by:

$$P_2(s = 1) + P_3(s = 1) - P_2(s = 1) \times P_3(s = 1)$$

$$= \frac{1}{3} + \frac{1}{3} - \frac{1}{9} = \frac{5}{9}$$

On the other hand, when malicious nodes 2 and 3 collude so that they do not select the same slot in any frame, the probability of nodes 2 and 3 selecting slot 1 becomes:

$$P_2^{coll}(s=1) + P_3^{coll}(s=1) = \frac{2}{3}$$

Thus, the collision probability of the packets from node 1 increases. As a result, the average throughput of non-malicious nodes goes down, while that of the malicious nodes increases. The derivation of the benchmark throughput for this scenario is furnished in Section 11.3.

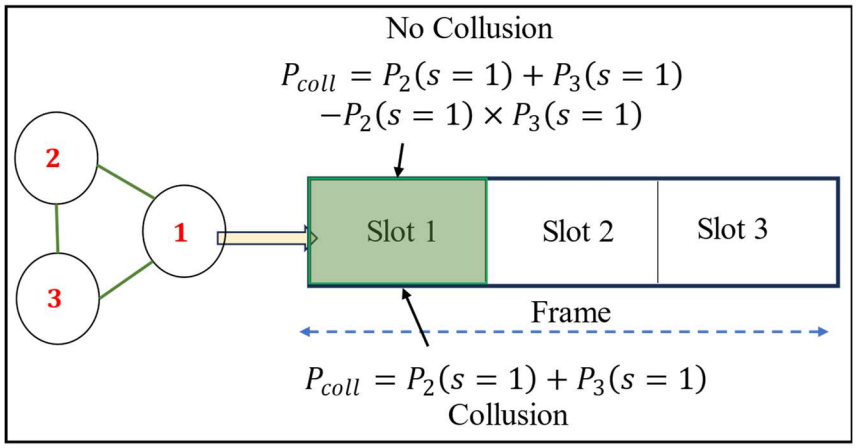

Figure 11.2: An example of the effect of collusion in a three-nodes network.

One important aspect here is the mechanism for sharing information between the colluding malicious nodes. As mentioned earlier, within each private subnetwork, one node acts as a cluster head and does slot allocation for the other nodes in that subnetwork. In Fig. 11.1, nodes 5 and 14, denoted as cluster heads, select slots for all malicious nodes in their respective subnetworks. We consider two different approaches for sharing information about the allocated slots among the malicious nodes in a subnetwork. First, we consider the availability of a separate communication channel. The second approach is by piggybacking the slot allocation information in the MAC layer PDU. Note that when the malicious nodes collude using piggybacking, the information sharing becomes stochastic due to the MAC packet collisions. In such scenarios, all the malicious nodes would select slots stochastically following a uniform distribution $\mathcal{U}(1, F)$, like the case of no-collusion as explained in chapter 10.

11.3 Benchmark Throughput in the Presence of Malicious Nodes

In this section, we derive the benchmark throughput for the scenario where the malicious nodes collude in order to avoid collisions among themselves. For deriving the equations, we consider a multi point-to-point wireless network comprising a set of non-malicious nodes $\mathcal{N}$ and set of malicious nodes $\mathcal{M}$ (same setup as the one used for deriving the benchmarks for non-collusion case in chapter 10). Frame size is F slots, where $F = |\mathcal{N}| + |\mathcal{M}|$. First, let us consider the case where the malicious nodes use a separate channel for collusion. In this case, the probability that a packet transmitted by a non-malicious node gets collided is given by:

$$P_{coll,\mathcal{N}}^{collusion\_sc} = P[Node\ m \in \mathcal{M}\ selects\ slot\ 's'|\ Node\ n \in \mathcal{N}\ selects\ a\ slot\ 's']$$

$$= P[m_1(s) \cup m_2(s) \cup \ldots\ldots \cup m_{|\mathcal{M}|}(s)]$$

$$= \sum_{i=1}^{|\mathcal{M}|} P[m_i(s)] \quad (\because P[m_i(s) \cap m_j(s)] = 0, \forall i \neq j) \tag{11.1}$$

Similarly, the probability of collision of packets transmitted by the malicious nodes is given by:

$$P_{coll,\mathcal{M}}^{collusion\_sc} = P[\{Node\ n \in \mathcal{N}\ selects\ slot\ 's'|\ Node\ m_k \in \mathcal{M}\ selects\ a\ slot\ 's']$$

$$= \sum_{i=1}^{|\mathcal{N}|} P[m_i(s)] \tag{11.2}$$

Therefore, the throughputs for the non-malicious and malicious nodes can be obtained from Eqn. 11.1 and 11.2 as follows:

$$S_{\mathcal{N}}^{collusion\_sc} = 1 - P_{coll,\mathcal{N}}^{collusion_{sc}} = 1 - \sum_{i=1}^{|\mathcal{M}|} P[m_i(s)]$$

$$= 1 - \frac{|\mathcal{M}|}{F} \tag{11.3}$$

$$S_{\mathcal{M}}^{collusion\_sc} = 1 - P_{coll,\mathcal{M}}^{collusion_{sc}} = 1 - \sum_{i=1}^{|\mathcal{N}|} P[m_i(s)]$$

$$= 1 - \frac{|\mathcal{N}|}{F} \tag{11.4}$$

Next, let us consider the scenario, where the malicious nodes piggyback the slot allocation information in MAC layer PDU. In this case, the malicious nodes will not be able to deterministically share slot allocation information, owing to MAC packet collisions experienced by malicious nodes. Then the probability of collision of packets transmitted by the malicious nodes is given by:

$$P_{coll,\mathcal{M}}^{collusion\_pb} = P[\{Node\ n \in \mathcal{N}\ selects\ slot\ 's' \cup Node\ m'$$

$$\in \{\mathcal{M} - m_k\}\ selects\ slot\ 's'\} |\ Node\ m_k \in \mathcal{M}\ selects\ a\ slot\ 's']$$

$$= \frac{|\mathcal{N}|}{F} + P_{coll,\mathcal{M}}^{collusion\_pb} \times P\left[\bigcup_{i\in\{\mathcal{M}-m_k\}} m_i(s)\right] - \frac{|\mathcal{N}|}{F} \times P_{coll,\mathcal{M}}^{collusion\_pb} \times P\left[\bigcup_{i\in\{\mathcal{M}-m_k\}} m_i(s)\right]$$

$$\Rightarrow P_{coll,\mathcal{M}}^{collusion\_pb} = \frac{\frac{|\mathcal{N}|}{F}}{\left(1 - P\left[\bigcup_{i\in\{\mathcal{M}-m_k\}} m_i(s)\right] + \frac{|\mathcal{N}|}{F} \times P\left[\bigcup_{i\in\{\mathcal{M}-m_k\}} m_i(s)\right]\right)} \tag{11.5}$$

The malicious nodes' throughput then becomes:

$$S_{\mathcal{M}}^{collusion\_pb} = 1 - \frac{\frac{|\mathcal{N}|}{F}}{\left(1 - P\left[\bigcup_{i\in\{\mathcal{M}-m_k\}} m_i(s)\right] + \frac{|\mathcal{N}|}{F} \times P\left[\bigcup_{i\in\{\mathcal{M}-m_k\}} m_i(s)\right]\right)} \tag{11.6}$$

Similarly, the throughput of the non-malicious nodes becomes:

$$S_{\mathcal{N}}^{collusion\_pb} = 1 - P[m_1(s) \cup m_2(s) \cup \ldots\ldots \cup m_{|\mathcal{M}|}(s)]$$

$$= 1 - \frac{|\mathcal{M}|}{F} - P_{coll,\mathcal{M}}^{collusion\_pb} \times \sum_{i=1}^{\left\lceil \frac{|\mathcal{M}|}{2} \right\rceil} \binom{|\mathcal{M}|}{2i-1} \times \frac{1}{F^{(2i-1)}} + P_{coll,\mathcal{M}}^{collusion\_pb} \times \sum_{i=1}^{\left\lfloor \frac{|\mathcal{M}|}{2} \right\rfloor} \binom{|\mathcal{M}|}{2i} \times \frac{1}{F^{(2i)}} \quad (11.7)$$

The above equations define the benchmark throughput for the non-malicious nodes, and the corresponding throughputs for the malicious nodes for the given threat model. Now, the objective of the learning-driven MAC protocol is to make the non-malicious nodes learn slot scheduling policies in a decentralized manner such that the above benchmark throughput can be achieved.

11.4 Slot Allocation in the Presence of Malicious Nodes

The non-malicious nodes use Multi-Armed Bandit for slot scheduling in this scenario. The learning framework is the same as the one developed in chapter 10. Each node is equipped with an MAB agent whose action is to select a slot for transmission such that the long term expected reward is maximized. The reward is defined such that a positive reward is assigned for a successful slot selection (that is no collision) and a zero reward when the packet transmitted in a selected slot gets collided. All the different kinds of action selection policies (that is reactive and robust) defined in previous chapter apply in this setting as well. However, while experimentation, we use Thompson Sampling for demonstrating the performance of the learning mechanism in this setting.

11.5 Experiments and Results

In this section, we present the results for the scenario when the malicious nodes collude to reduce collisions among themselves. As detailed earlier, the objectives of the malicious nodes are to reduce throughput of the non-malicious nodes and to increase their own throughput. Two cases

are considered for sharing information among the colluding malicious nodes: (i) availability of a separate communication channel; (ii) by piggybacking the information in the MAC layer PDU. First, we demonstrate this for a 10-nodes fully connected topology. Fig. 11.3 compares the throughput of both malicious and non-malicious nodes for three cases: (i) no-collusion (ii) collusion using a separate channel (collusion-sc) and (iii) collusion using piggybacking (collusion-pb). The following observations can be made. First, throughput reduction for non-malicious nodes is high when the malicious nodes collude. At the same time, average throughput of malicious nodes increases. Second, the effect of collusion reduces when malicious nodes use piggybacking as compared to the scenario when a separate channel is available for malicious nodes' information sharing. This is because of malicious nodes' packet loss due to collision when they piggyback information on MAC PDU. Third, throughput reduction of non-malicious nodes and throughput increase of malicious nodes, with increase in the number of malicious nodes, is consistent irrespective of the case when the malicious nodes collude or not. However, the effect of the number of malicious nodes on non-malicious throughput increases when there is collusion

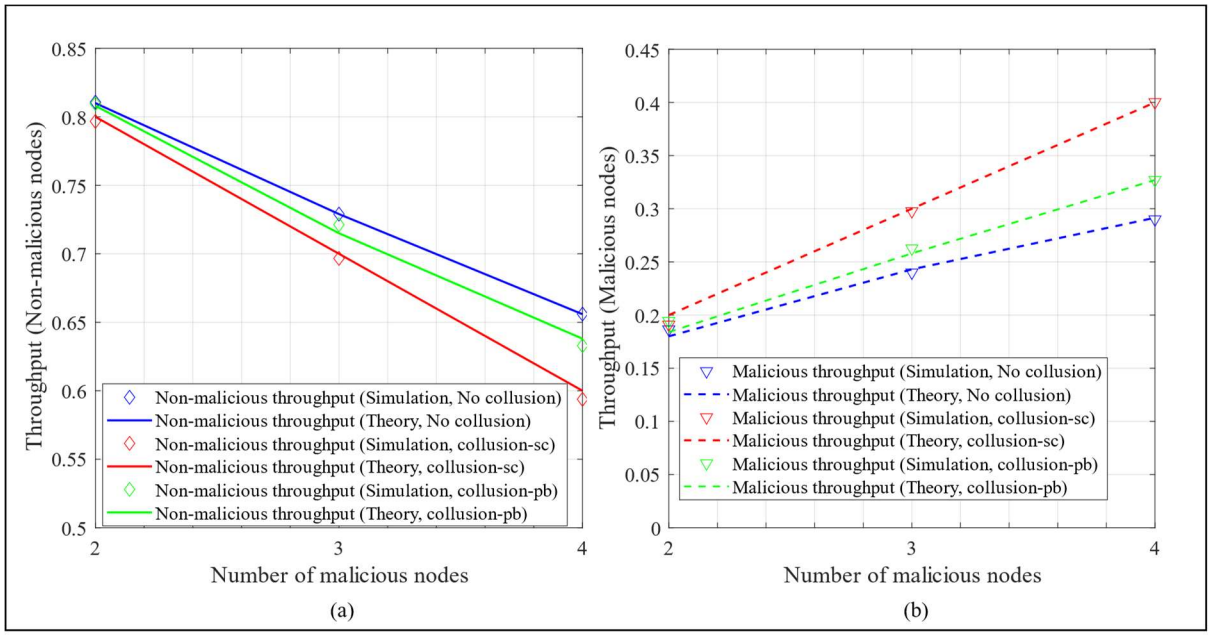

Figure 11.3: Effect of collusion on (a) non-malicious and (b) malicious nodes' throughput.

among the malicious nodes. Moreover, the figure also demonstrates the ability of MAB-driven slot allocation scheme to make the non-malicious nodes achieve the theoretically-derived benchmark throughput for all the above-mentioned scenarios.

The effect of collusion on the network performance is studied for mesh networks with sparse connectivity. As shown in Fig. 11.4, there are 5 malicious nodes forming two-separate private sub-networks, with node ID 4 and 12 acting as the respective Cluster Head. The following observations hold for the mesh networks as well. First, malicious nodes throughput increases, and non-malicious nodes throughput reduces in the presence of collusion. Second, the effect of malicious nodes on the rest of the network decreases when they collude using piggybacking as compared to using a separate channel.

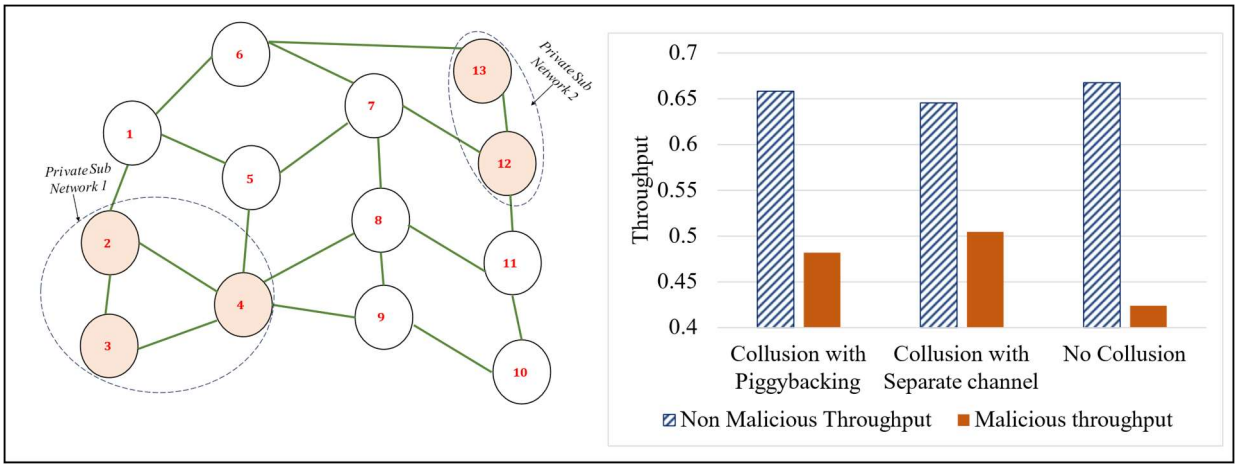

Figure 11.4: Effect of collusion in partially connected mesh networks.

11.6 Summary

In this chapter, we consider a scenario where the malicious nodes collude and share information so as to increase the throughput reduction of the rest of the network. The malicious nodes form private sub-networks among themselves to collude. Two information sharing models are considered for the malicious nodes: (i) availability of a separate channel and (ii) piggybacking

control information in MAC PDU. Performance of the MAB-driven slot allocation strategy is then evaluated in such adversarial scenarios. The theoretical benchmark throughputs are computed for different malicious nodes' slot allocation strategies and it is shown that the non-malicious nodes are able to achieve these benchmarks by learning suitable slot scheduling strategy.

Chapter 12: Conclusions and Future Works

In this thesis, we have formulated and developed the concept of network protocol synthesis using multi-agent Reinforcement Learning. The core idea of this paradigm is that each node, equipped with an RL engine, learns to find situation-specific protocol logic for network performance improvements. The developed framework is specifically targeted to resource-constrained networks with thin energy budget and limited underlying hardware support. We particularly focus on developing architectures for synthesizing access control protocols that deal with network performance improvement from multiple perspectives, viz., network throughput, access delay, energy efficiency, and wireless bandwidth usage.

The learning architectures aimed at finding policies for specific access control scenarios, throughout the thesis, are developed with a decentralized implementation. Each wireless node learns its own access layer logic independently and without the arbitration of a central server. This is done to reduce the energy expenditure and bandwidth overhead required for control information sharing to and from the server. This decentralized orchestration is achieved by a cooperative behavior followed by the learning agents while training. Towards the later part of the thesis, we analyze the performance of the synthesized protocols in adversities, when there are malicious nodes in the network, acting non-cooperatively, with an attempt to degrade the network performance.

The concept of learning-enabled network protocol synthesis has been explored for a diverse set of MAC arrangements. First, the idea is demonstrated in random access MAC settings, where the learning-driven logic is shown to be able to minimize collisions with a fair share of wireless bandwidth in the network. Next, the concept of RL-based protocol synthesis is used for TDMA-

based MAC arrangement for slot scheduling and transmit-sleep-listen decision making. It is shown in the thesis how learning can make the nodes take transmission/sleep decisions in a judicious manner in order to save energy while maintaining network performance. This is achieved using a multi-tier learning framework that manages the trade-off between energy expenditure, throughput, and delay. A contextual learning model is developed in order to make the system adaptive to network traffic dynamics and scalable with network size. The energy management framework is then extended for applications with energy-harvesting networks that have a spatiotemporal energy variation. Finally, performance of the learning driven protocols is studied under the influence of unreliable information caused by the presence of adversarial agents in the network.

12.1 Key Findings and Design Guidelines

The following are the key takeaway points that can be learnt from the results presented in this thesis.

1. In the absence of time-slotting, time synchronization, carrier sensing, the RL-enabled protocol can achieve the benchmark ALOHA throughput and sustain it for higher network traffic. The performance of the synthesized protocol depends on the granularity of the RL state-action space, which needs to be adjusted according to the application specific requirements. It is noteworthy that learning convergence time needs to be traded with network performance for these scenarios. Depending on the application requisites, the throughput can be distributed fairly, or node-level access priority can be assigned by tuning the RL reward function.

2. For time-slotted networks, in the absence of time synchronization, MAB can be used for TDMA slot scheduling in a decentralized approach for heterogeneous and mesh networks. While the MAB-driven protocol allows slot scheduling such that there is no packet collision, it comes

with a bandwidth redundancy because of absence of time synchronization. This bandwidth redundancy can be kept to a minimum by using a slot defragmentation mechanism after learning convergence. It was observed that there exists a trade-off between learning convergence time and bandwidth redundancy. Thus, a design guideline based on this study would be to first use MAB for slot scheduling with a much larger frame size and then use slot defragmentation to reduce the bandwidth overhead.

3. A multi-tier, hierarchical learning module driven by RL and MAB can be used for efficient energy management in networks. While performing a thorough characterization of the system, an inverse relationship between energy efficiency and network throughput was observed. From the perspective of network design and management, a user-tunable learning parameter (*packet miss reduction parameter*) was incorporated into the framework. By suitable application-specific tuning of the parameter, adjustment can be made to find a right balance in the throughput-energy consumption space.

4. The multi-tier energy management architecture can be made adaptive to network traffic and scalable with network size by following a context-specific learning and using a function approximator such as a Neural Network. Nonetheless, training a deep NN model is computationally complex considering the processing capability of sensor nodes. Computing elaborate back propagation and gradient descent can also consume a significant amount of energy, which cannot be afforded in resource-constrained sensor/ IoT nodes. This needs to be handled by offline training of the deep NN models pre-deployment. The learnt synaptic weights are required to be stored in the sensor nodes for estimating the appropriate actions for a given context/state.

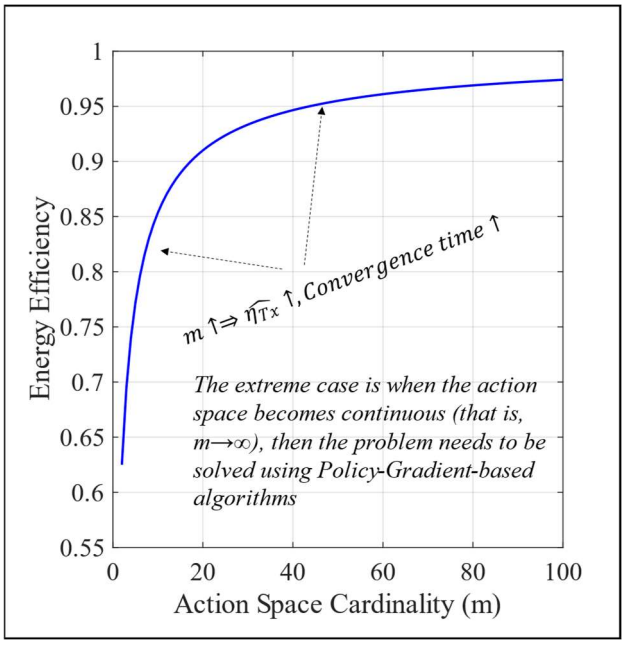

Figure 12.1: Dependency of Action Space size on network performance.

5. The action space cardinality $|\mathcal{A}|$, has to be chosen judiciously based on the application requirement and the computation capability of the sensor nodes. A granular action space increases the energy efficiency of the transmit-sleep scheduling agent, although at the expense of heavy computation load and convergence speed. For the CDQL framework, the dependency of action space size (controlled by parameter m) on energy efficiency ($\widehat{\eta_{Tx}}$) is shown in Fig. 12.1. The figure depicts an asymptotic increase of efficiency with increase in the value of m. The extreme case is when the action space becomes continuous (i.e., $m \to \infty$). In that situation, the problem needs to be solved using Policy-Gradient-based algorithms, which can be a future extension of this research. Note that the existing policy gradient algorithms in general show convergence limitations created by the restricted policy spaces defined by a multi-layer neural network, as proved theoretically in [119].

6. Energy management in an energy-harvesting network can be accomplished by learning geo-temporal specific, joint transmit-sleep scheduling logic. While developing the learning

paradigm for topologies with long flow, special consideration must be given to learning errors that can propagate along the flow. An approach to deal with such situations has been proposed in the thesis by allowing the nodes to share learning confidence with each other.

7. Decentralized TDMA slot allocation in the presence of malicious nodes can be achieved using Multi-Armed Bandits. Although collisions cannot be got rid of in such scenario, given the stochastic and non-compliance behavior of the adversaries, the collisions can be kept to a minimum. Based on the system-level defense requirements of minimizing the non-malicious throughput reduction or malicious bandwidth share, *reactive* or *robust* slot allocation schemes can be adopted.

12.2 Future Research Directions

This thesis has laid the foundation for conceptual development and numerical demonstrations of the paradigm of network protocol synthesis using multi-agent Reinforcement Learning and its variants. There are several areas that can be branched from this key concept as future research directions. Some of these research directions worth exploring are enumerated in this section.

12.2.1 Cross-Layer Learning for Energy Management

The protocol synthesis architectures for network energy management as proposed in chapters 7-9 is based on transceiver sleep-awake decision making, which is a MAC layer approach. In that scenario, it is assumed that the flows are fixed, meaning that the packet routes do not change over time. Now, there may be scenarios where the intermediate nodes in a flow do not have sufficient energy for transmission. In that case, the service is disrupted till the intermediate nodes are recharged with sufficient energy. However, there may exist an alternate route for the flow where the nodes have sufficient energy to ensure sustainability of the flow (Fig. 12.1).

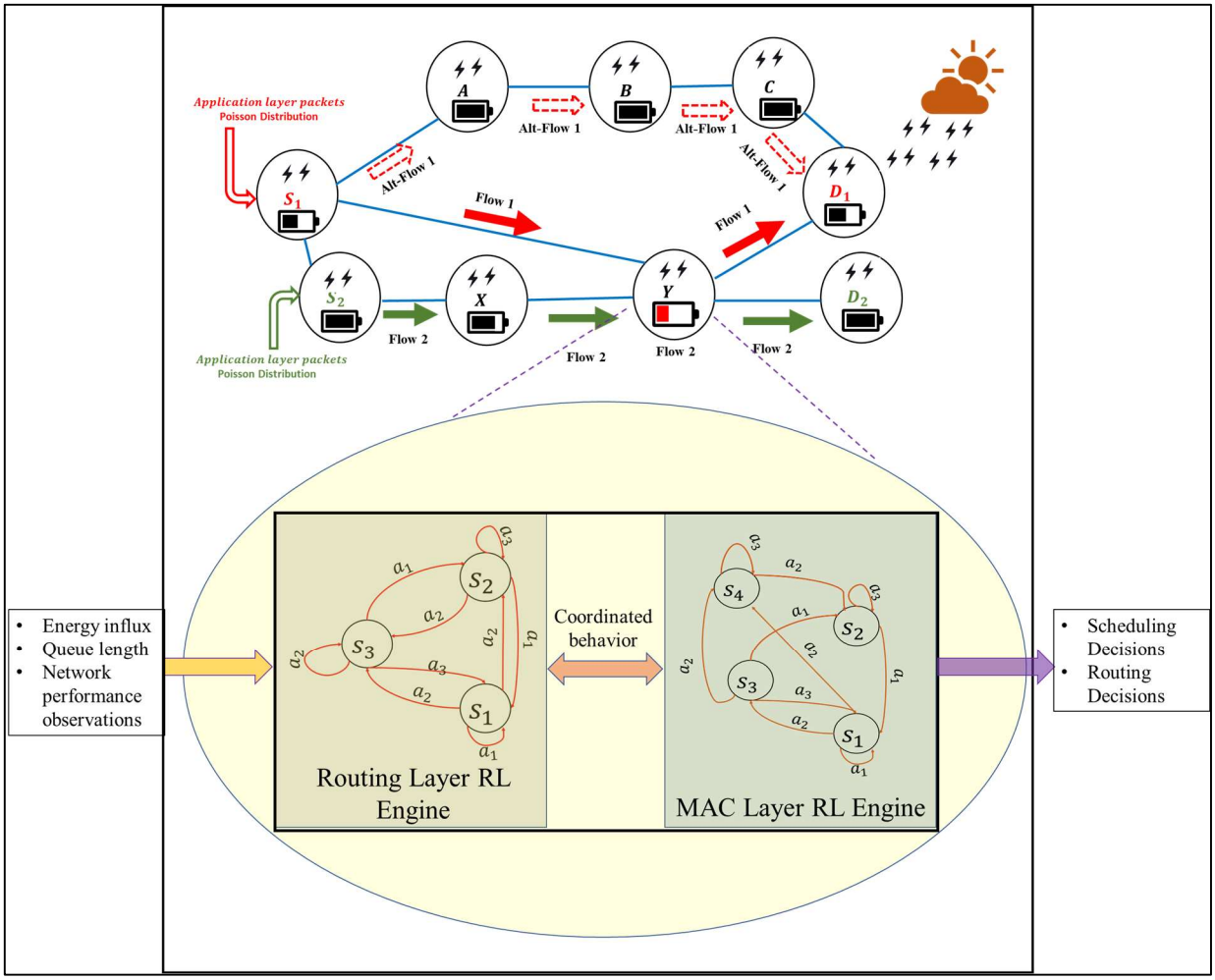

Figure 12.2: Cross-layer learning for Flow-Energy Management in energy-harvesting networks.

The idea here is to design a learning-based approach that can change the route based on the energy profiles of the nodes so as to minimize the duration of service disruption. This is demonstrated in Fig. 12.1, where node 'Y' has shortage of energy, that may lead to service disruption for both the existing flows through it. Now, there is an alternate route ($S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow D_1$) for sending packets from source S_1 to destination D_1. Although that is a longer route in terms of number of hop counts, but choosing that route will reduce the energy consumption at node 'Y', since it has to only forward packets for flow 2. This is beneficial for the sustainability of both the flows. This can be achieved by launching an RL engine at the routing layer of each node to decide, based on its own and its one-hop neighbor's energy profile, on the best possible

next hop to forward the packet in order to adaptively reduce the service disruption because of energy shortage.

Now, we have two different approaches for flow-energy management in a network with spatiotemporal energy profile: MAC layer scheduler (chapters 7-9) and routing layer flow manager. The high-level goal is to integrate to integrate these RL engines to design a cross-layer learning framework that can minimize the duration of service disruption because of energy shortage, while keeping delay and sleep-induced packet drops at an acceptable minimum. These agents cooperate with each other to achieve the desired objectives, viz., to minimize the duration of flow service disruption as well as the sleep-induced packet drops and end-to-end delay. The agent at routing layer decides the best route based on the energy profiles of the nodes and the one at the MAC layer decides the best sleep-awake scheduling policy for that specific route. An actor-critic architecture could also be used here, where the policies learnt by the actor in the MAC layer would be evaluated and supported by the critic logically located in the routing layer. The expectation is that this cross-layer design would be able to handle the trade-off more efficiently than the MAC layer design proposed in this thesis.

12.2.2 Integration of multiple learning agents for design-complexity reduction

The multi-tier learning frameworks for sleep-transmission scheduling in chapters 7 and 8 rely on the policies learnt by multiple learning agents. First, there is an MAB-based slot allocating agent per node for collision-free transmission in tier-I. Second, an RL-based transmit/sleep scheduling agent per flow in tier-II manages the trade-off between packet delay and energy consumption. There is an MAB-based listen/sleep scheduling agent per node for managing the trade-off between sleep-induced packet drops and energy consumption in tier-III. Thus, for a network with N nodes and N_F flows through each node, we would have $2 \times N$ MAB agents and $N_F \times N$ RL

agents. The design complexity in this implementation is high. Moreover, in this multi-tier setup, the learning policies of one agent is dependent on the learning behavior of others. So, inaccurate learning policies learnt by an agent affect the policies of others.

The future research idea here is to integrate functionalities of multiple learning agents into one so that the design complexity is reduced. A single model can be trained for achieving slot-allocation, transmit-sleep scheduling simultaneously, so as to improve network performance while reducing energy expenditure. In addition, a detailed complexity analysis of the access protocol synthesis frameworks developed in this thesis needs to be studied in terms of space complexity, computational cost and convergence time. The goal here would be to build a system that can achieve the same performance as is achieved by the developed architectures but with a less complex design and low computation cost.

There are certain challenges that need to be addressed in order to incorporate all the functionalities into a single learning agent. First, learning in the multi-tier framework is sequential in nature. That is, learning policies by the transmit/sleep scheduling is dependent on the slot scheduling agent. The question that needs to be addressed is how to accomplish such sequential/hierarchical behavior using a single learning agent. The second challenge is the handling of large state-action space created by integrating the learning modules. The functionalities of policy gradient approaches could be leveraged to address this. The computation load of these algorithms need to be evaluated from the perspective of resource-constrained nodes.

12.2.3 Protocol Synthesis in the Absence of Hardware Constraints

The protocol synthesis architectures developed in this thesis are mainly focused on resource-constrained networks with a thin energy budget. The random-access logics developed in chapters 3 and 4 are centered around low-complexity networks with limited underlying hardware support.

In such a constrained environment, the benchmark throughput performance is restricted to that achieved by ALOHA class of protocols. It would be interesting to explore the protocol synthesis framework for more advanced networked systems in the absence of such hardware limitations. A research direction along this line would be to leverage the ability of the protocol synthesis approach for finding situation-specific optimal networking solutions for WiFi [8] class of protocols. Enhancing WiFi throughput in the presence of large number of users has been of interest to researchers [120] [121] [122]. The learning-driven solution frameworks developed in this thesis and the corresponding findings can be extended for finding access control protocols in such scenarios. Moreover, managing WiFi throughput in a heterogeneous networking condition [123] [124] can be developed building on the techniques proposed in chapters 3 and 4 for assigning fairness and node-level access priorities in random access MAC arrangements.

While extending the protocol synthesis architecture to encompass networks without hardware constraints, it would also be useful to consider dynamic and error-prone channel conditions. In such situations, the learning observables would be affected by channel noise. The objective of the learning mechanism would be to find policies for improving network performance in such complex channel characteristics. A solution approach for this problem would be to develop a channel model, including the noise characterization, based on the RL observables. This can then be embedded into the state space for making the policies robust to channel errors and uncertainties.

12.2.4 Evaluating the Protocol Synthesis Framework under Advanced MAC Attacks

In chapters 10 and 11, we analyzed performance of the learning-driven slot allocation approach in the presence of adversaries trying to disrupt the network performance. Nevertheless, in practical networks, the MAC attacks can be much more complicated. One such scenario can be

extended from what is presented in chapter 11, where the malicious nodes form private sub networks to collude with the goal of reducing the throughput of the network. In a more advanced threat model, the malicious nodes can rely on the routing layer to share information among each other irrespective of their connections. Finding the benchmark network throughput and developing learning policies for such scenarios are worth exploring.

In addition, evaluation of performance of the energy management solutions presented in chapters 7-9 in the presence unreliable RL observations because of non-cooperative agents is a future direction of research. The situation would become more adverse here, because of the multi-tier learning architecture used. Malicious behavior in one of these tiers would affect the learning behavior at all the tiers.

The goal here would be to develop solutions that can handle such attacks and find policies so that performance degradation caused by malicious nodes is reduced. The concept of game-theoretic approaches for solving non cooperative games could be leveraged to find policies in the presence of such advanced attack models. Nash conditions for these scenarios can be formulated to derive theoretical benchmarks, where no agent can improve its performance by changing its strategy alone, assuming other nodes' strategies remain unchanged.

12.2.5 Leveraging Generative Learning Models for Protocol Synthesis

Recent developments of the deep learning architectures, such as, Generative Adversarial Networks (GANs) [125], Transformers [126], Variational Autoencoders (VAEs) [127], or language models like OpenAI's GPT (Generative Pre-trained Transformer) [128] have inspired researchers to explore the benefits of Generative AI in multiple applications. Transformer model, which is core to many of these learning frameworks, can capture long range dependencies of an input sequence, because of its self-attention mechanism. This makes these frameworks well-

suited for tasks that require understanding context across different parts of the input sequence. This property of these learning models can be leveraged for developing policies and protocols for wireless networks. Network conditions, such as congestion status, energy budget, throughput, channel conditions etc., can serve as the context for these models which can learn suitable and context-specific policies for maximizing network performance. In addition, pre-trained transformer models, such as GPT, can be fine-tuned on specific tasks with relatively small amounts of task-specific data. This transfer learning approach enables quick adaptation of these models to new tasks. This property of these models can be exploited for learning network protocols that can adapt to time-varying network conditions. However, one key research question here would be to find out how to make these large and complex models suitable for wireless nodes with limited processing capacity, energy budget and memory constraints. Developing a generative AI-driven framework for network protocol synthesis considering the above constraints can be a future direction of this thesis. To understand how these approaches would compare with the findings of this thesis would be worth exploring.

BIBLIOGRAPHY

[1] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wireless personal communications,* vol. 58, no. 1, pp. 49-69, 2011.

[2] Z. Sheng, S. Yang, Y. Y. A. V. Vasilakos, J. A. McCann and K. K. Leung, "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities," *IEEE wireless communications,* vol. 20, no. 6, pp. 91-98., 2013.

[3] O. Hahm, E. Baccelli, H. Petersen and N. Tsiftes, "Operating systems for low-end devices in the internet of things: a survey," *IEEE Internet of Things Journal,* vol. 3, no. 5, pp. 720-734, 2015.

[4] N. Abramson, "The ALOHA system: Another alternative for computer communications," in *Fall Joint Computer Conference*, 1970..

[5] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part I-carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE transactions on Communications,* vol. 23, no. 12, pp. 1400-1416, 1975.

[6] C. Bisdikian, "An overview of the Bluetooth wireless technology," *IEEE Communications magazine,* vol. 39, no. 12, pp. 86-94, 2001.

[7] S. C. Ergen, "ZigBee/IEEE 802.15. 4 Summary," UC Berkeley, 2014.

[8] "WiFi (Wireless Fidelity)," [Online]. Available: https://www.wi-fi.org/. [Accessed 12 December 2022].

[9] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," 2000.

[10] G. Wang, D. Turgut, L. Bölöni, Y. Ji and D. C. Marinescu, "Improving routing performance through m-limited forwarding in power-constrained wireless ad hoc networks," *Journal of Parallel and Distributed Computing,* vol. 68, no. 4, 2008.

[11] P. Sok and K. Kim, "Distance-based PRoPHET routing protocol in disruption tolerant network," in *International conference on ICT convergence (ICTC), IEEE*, 2013.

[12] R. S. Sutton and A. G. Barto., Reinforcement learning: An introduction, MIT Press, 2018.

[13] C. Watkins and P. Dayan, "Q-learning," *Machine learning,* vol. 8, no. 3, pp. 279-292., 1992.

[14] L. Matignon, G. J. Laurent and N. L. Fort-Piat, "Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[15] S. Qu, T. Chu, J. Wang, J. Leckie and W. Jian, "A centralized reinforcement learning approach for proactive scheduling in manufacturing," in *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2015.

[16] M. Moradi, "A centralized reinforcement learning method for multi-agent job scheduling in Grid," in *6th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2016.

[17] J. K. Gupta, M. Egorov and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *In International conference on autonomous agents and multiagent systems*, Cham, 2017.

[18] C. Zhang and V. Lesser, "Coordinating multi-agent reinforcement learning with limited communication," in *International Conference on Autonomous agents and multi-agent systems*, 2013.

[19] P. Xuan, V. Lesser and S. Zilberstein, "Communication decisions in multi-agent cooperation: Model and experiments," in *Proceedings of the fifth international conference on Autonomous agents*, 2001.

[20] K. Zhang, Z. Yang, H. Liu, T. Zhang and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *International Conference on Machine Learning,*, 2018.

[21] D. Upadhyay, A. K. Dubey and P. S. Thilagam, "Time synchronization problem of wireless sensor network using maximum probability theory," *International Journal of System Assurance Engineering and Management,* vol. 9.2, pp. 517-524, 2018.

[22] W. Ye, J. Heidemann and D Estrin "An energy-efficient MAC protocol for wireless sensor networks," in *Twenty-first annual joint conference of the IEEE computer and communications societies*, 2002.

[23] W. Ye, J. Heidemann and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor network," *IEEE/ACM Transactions on networking,* vol. 12.3, pp. 493-506, 2004.

[24] H. Dutta and S. Biswas, "Towards Multi-agent Reinforcement Learning for Wireless Network Protocol Synthesis," in *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2021.

[25] H. Dutta and S. Biswas, "Medium access using Distributed Reinforcement Learning for IoTs with low-complexity wireless transceivers," in *IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021.

[26] H. Dutta and S. Biswas, "Distributed Reinforcement Learning for scalable wireless medium access in IoTs and sensor networks," *Computer Networks,* vol. 202, p. 108662, 2022.

[27] H. Dutta, A. K. Bhuyan and S. Biswas, "Wireless MAC Slot Allocation Using Distributed Multi-Armed Bandit Learning and Slot Defragmentation," in *International Wireless Communications and Mobile Computing (IWCMC)*, 2022.

[28] H. Dutta, A. K. Bhuyan and S. Biswas, "Reinforcement Learning for Protocol Synthesis in Resource-Constrained Wireless Sensor and IoT Networks," in *International Symposium on Ubiquitous Networking*, 2022.

[29] H. Dutta, A. K. Bhuyan and S. Biswas, "Multi-armed Bandit Learning for TDMA Transmission Slot Scheduling and Defragmentation for Improved Bandwidth Usage," in *International Conference on Information Networking (ICOIN)*, 2023.

[30] H. Dutta, A. K. Bhuyan and S. Biswas, "Reinforcement Learning for Flow and Energy Management in Resource-Constrained WIreless Networks," *Computer Communications, Elsevier,* vol. 202, pp. 73-86, 2023.

[31] H. Dutta, A. K. Bhuyan and S. Biswas, "Contextual Deep Reinforcement Learning for Flow and Energy Management in Wireless Sensor and IoT Networks," *IEEE Transactions on Green Communications and Networking,* 2024.

[32] H. Dutta, A. K. Bhuyan and S. Biswas, "Contextual Deep Reinforcement Learning for Flow and Energy Management in Wireless Sensor and IoT Networks," in *International Conference on Information Networking (ICOIN)*, 2024.

[33] H. Dutta, A. K. Bhuyan and S. Biswas, "Cooperative Reinforcement Learning for Energy Management in Multi-Hop Networks with Energy Harvesting," *(submitted to) IEEE Transactions on Green Communications and Networking,* 2024.

[34] H. Dutta, A. K. Bhuyan and S. Biswas, "Using Multi-arm Bandit Learning for Thwarting MAC Layer Attacks in Wireless Networks".

[35] Y. Chu, P. D. Mitchell and D. Grace, "ALOHA and Q-Learning based medium access control for Wireless Sensor Networks," in *International Symposium on Wireless Communication Systems (ISWCS)*, Paris, 2012.

[36] Z. Lan, H. Jiang and X. Wu, "Decentralized cognitive MAC protocol design based on POMDP and Q-Learning," in *7th International Conference on Communications and Networking*, China, Kun Ming, 2012.

[37] S. Galzarano, A. Liotta and G. Fortino, "QL-MAC: A Q-Learning Based MAC for Wireless Sensor Networks," in *Algorithms and Architectures for Parallel Processing. ICA3PP, Lecture Notes in Computer Science*, Cham, 2013.

[38] S. Galzarano, G. Fortino and A Liotta, "A Learning-Based MAC for Energy Efficient Wireless Sensor Networks," in *Internet and Distributed Computing Systems, IDCS, Lecture Notes in Computer Science*, Cham, 2014.

[39] A. P. Renold and S. Chandrakala, "MRL-SCSO: multi-agent reinforcement learning-based self-configuration and self-optimization protocol for unattended wireless sensor networks," *Wireless Personal Communications,* vol. 96, no. 4, pp. 5061-5079, 2017.

[40] H. Bayat-Yeganeh, V. S.-M. and H. Kebriaei, "A multi-state Q-learning based CSMA MAC protocol for wireless networks," *Wireless Networks,* vol. 24, no. 4, pp. 1251-1264, 2018.

[41] R. Ali, N. Shahin, Y. B. Zikria, B.-S. Kim and S. W. Kim, "Deep reinforcement learning paradigm for performance optimization of channel observation–based MAC protocols in dense WLANs," *IEEE Access,* vol. 7, pp. 3500-3511, 2018.

[42] Y. Chu, P. D. Mitchell and D. Grace, "ALOHA and q-learning based medium access control for wireless sensor networks," in *International Symposium on Wireless Communication Systems (ISWCS), IEEE*, 2012.

[43] Y. Chu, S. Kosunalp, P. D. Mitchell, D. Grace and T. Clarke, "Application of reinforcement learning to medium access control for wireless sensor networks," *Engineering Applications of Artificial Intelligence,* vol. 46, pp. 23-32, 2015.

[44] S. H. Park, P. D. Mitchell and D. Grace, "Reinforcement Learning Based MAC Protocol (UW-ALOHA-Q) for Underwater Acoustic Sensor Networks," *IEEE Access,* vol. 7, pp. 165531-165542, 2019.

[45] S. H. Park, P. D. Mitchell and D. Grace, "Performance of the ALOHA-Q MAC Protocol for Underwater Acoustic Networks," in *International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, Southend, United Kingdom, 2018.

[46] Y. Yu, T. Wang and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *Journal of Special Areas in COmmunication,* vol. 37, no. 6, pp. 1277-1290, 2019.

[47] T. Lee and O. J. Shin, "CoRL: Collaborative Reinforcement Learning-Based MAC Protocol for IoT Networks," *Electronics,* vol. 9, no. 1, 2019.

[48] J. Niu and Z. Deng, "Distributed self-learning scheduling approach for wireless sensor network," *Ad Hoc Networks,* pp. 1276-1286, 2013.

[49] J. Lee, H. Yoon and I. Yeom, "Distributed fair scheduling for wireless mesh networks using IEEE 802.11," *IEEE transactions on vehicular technology,* vol. 59, no. 9, pp. 4467-4475, 2010.

[50] M. R. Lenka, A. R. Swain and M. N. Sahoo, "Distributed slot scheduling algorithm for hybrid CSMA/TDMA MAC in wireless sensor networks," in *International Conference on Networking and Advanced Systems (ICNAS), IEEE*, 2016.

[51] L. Bommisetty and T. G. Venkatesh, "Resource Allocation in Time Slotted Channel Hopping (TSCH) Networks Based on Phasic Policy Gradient Reinforcement Learning," *Internet of Things,* vol. 19, 2022.

[52] H. Park, H. Kim, S.-T. Kim and P. Mah, "Multi-agent reinforcement-learning-based time-slotted channel hopping medium access control scheduling scheme," *IEEE Access,* vol. 8, pp. 139727-139736., 2020.

[53] J. Liu, B. Zhao, Q. Xin and H. Liu, "Dynamic channel allocation for satellite internet of things via deep reinforcement learning," in *International Conference on Information Networking (ICOIN)*, 2020.

[54] F. Ahmed and H.-S. Cho, "A time-slotted data gathering medium access control protocol using Q-learning for underwater acoustic sensor networks," *IEEE Access,* vol. 9, pp. 48742-48752, 2021.

[55] R. Mohammadi and Z. Shirmohammadi, "RLS2: An energy efficient reinforcement learning-based sleep scheduling for energy harvesting WBANs."," *Computer Networks,* vol. 229, 2023.

[56] X. Cao, W. Xu, X. Liu, J. Peng and T. Liu, "A deep reinforcement learning-based on-demand charging algorithm for wireless rechargeable sensor networks," *Ad Hoc Networks,* vol. 110, p. 102278, 2021.

[57] Y.-H. Xu, G. Yu and Y.-T. Yong, "Deep reinforcement learning-based resource scheduling strategy for reliability-oriented wireless body area networks," *IEEE Sensors Letters,* vol. 5, no. 1, pp. 1-4, 2020.

[58] Y.-H. Xu, J.-W. Xie, Y.-G. Zhang, M. Hua and W. Zhou, "Reinforcement Learning (RL)-based energy efficient resource allocation for energy harvesting-powered wireless body area network," *Sensors,* vol. 20.1, no. 44, 2020.

[59] G. Chen, Y. Zhan, G. Sheng, L. Xiao and Y. Wang, "Reinforcement learning-based sensor access control for WBANs," *IEEE Access,* vol. 7, pp. 8483-8494, 2018.

[60] M. Roy, D. Biswas, N. Aslam and C. Chowdhury, "Reinforcement learning based effective communication strategies for energy harvested WBAN," *Ad Hoc Networks,* vol. 132, 2022.

[61] C. Savaglio, P. Pace, G. Aloi, A. Liotta and G. Fortino, "Lightweight reinforcement learning for energy efficient communications in wireless sensor networks," *IEEE Access,* vol. 7, pp. 29355-29364, 2019.

[62] Z. Liu and I. Elhanany, "RL-MAC: a reinforcement learning based MAC protocol for wireless sensor networks," *International Journal of Sensor Networks,* vol. 1.3, pp. 117-124, 2006.

[63] W. Ye, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Twenty-first annual joint conference of the IEEE computer and communications societies*, 2002.

[64] T. Van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor network," in *1st international conference on Embedded networked sensor systems*, 2003.

[65] F. Ahmed and H.-S. Cho, "A time-slotted data gathering medium access control protocol using Q-learning for underwater acoustic sensor networks," *IEEE Access,* vol. 9, pp. 48742-48752, 2021.

[66] Y. Li, K. K. Chai, Y. Chen and J. Loo, "Smart duty cycle control with reinforcement learning for machine to machine communications," in *IEEE International Conference on Communication Workshop (ICCW)*, 2015.

[67] R. de Paz Alberola and D. Pesch, "Duty cycle learning algorithm (DCLA) for IEEE 802.15. 4 beacon-enabled wireless sensor networks," *Ad Hoc Networks,* vol. 10.4, pp. 664-679, 2012.

[68] Z. Guo and H. Chen, "A reinforcement learning-based sleep scheduling algorithm for cooperative computing in event-driven wireless sensor networks," *Ad Hoc Networks, Elsevier,* vol. 130, 2022.

[69] N. Zhu, X. Xu, S. Han and S. Lv, "Sleep-Scheduling and Joint Computation-Communication Resource Allocation in MEC Networks for 5G IIoT," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Nanjing, China, 2021.

[70] X. Wang, H. Chen and S. Li, "A reinforcement learning-based sleep scheduling algorithm for compressive data gathering in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking,* vol. 1, 2023.

[71] S. N. Das, S. Misra, B. E. Wolfinger and M. S. Obaidat, "Temporal-correlation-aware dynamic self-management of wireless sensor networks," *IEEE Transactions on Industrial Informatics,* vol. 12, no. 6, pp. 2127-2138, 2016.

[72] L. Wang, G. Zhang, J. Li and G. Lin, "Joint optimization of power control and time slot allocation for wireless body area networks via deep reinforcement learning," *Wireless Networks,* vol. 2020, pp. 4507-4516, 2020.

[73] Z. Lan, H. Jiang and X. Wu, "Decentralized cognitive MAC protocol design based on POMDP and Q-Learning," in *7th International Conference on Communications and Networking in China, Kun Ming*, 2012.

[74] A. Leon-Garcia, Probability, statistics, and random processes for electrical engineering, 2017.

[75] V. Otterlo, Martijn and M. Wiering, "Reinforcement learning and markov decision processes," in *Reinforcement Learning*, Berlin, Heidelberg, 2012.

[76] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science 2,* pp. 331-434, 1990.

[77] X. Pi, Y. Cai and G. Yao, "An energy-efficient cooperative MAC protocol with piggybacking for wireless sensor networks," in *International Conference on Wireless Communications and Signal Processing (WCSP)*, 2011.

[78] F. Yu, T. Wu and S. Biswas, "Toward in-band self-organization in energy-efficient MAC protocols for sensor networks," *IEEE Transactions on mobile computing,* vol. 7, no. 2, 2007.

[79] A. R. Cassandra, L. P. Kaelbling and M. L. Littman, "Acting optimally in partially observable stochastic domains," in *Aaai*, 1994.

[80] L. P. Kaelbling, M. L. Littman and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence,* vol. 101, pp. 99-134, 1998.

[81] L. G. Roberts, "ALOHA packet system with and without slots and capture," *ACM SIGCOMM Computer Communication Review,* vol. 5.2, pp. 28-42, (1975).

[82] N. Abramson, "The ALOHA system: Another alternative for computer communications," in *Fall joint computer conference*, 1970.

[83] N. Morozs, P. Mitchell and Y. V. Zakharov, "TDA-MAC: TDMA without clock synchronization in underwater acoustic networks," *IEEE Access,* vol. 6, pp. 1091-1108, 2017.

[84] N. Morozs, P. Mitchell and Y. Zakharov, "Unsynchronized dual-hop scheduling for practical data gathering in underwater sensor networks," in *Fourth Underwater Communications and Networking Conference (UComms). IEEE*, 2018.

[85] A. Slivkins, "Introduction to multi-armed bandits," *Foundations and Trends in Machine Learning ,* Vols. 12, no. 1-2 , pp. 1-286, 2019.

[86] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *arXiv preprint arXiv:1402.6028 ,* 2014.

[87] D.D. Falconer, F. Adachi and B. Gudmundson, "Time division multiple access methods for wireless personal communications," *IEEE Communications Magazine ,* vol. 33.1, pp. 50-57, 1995.

[88] G. Tesauro, "Extending Q-learning to general adaptive multi-agent systems," *Advances in neural information processing systems,* vol. 16, pp. 871-878, 2003.

[89] C. Savaglio, P. Pace, G. Aloi, A. Liotta and G. Fortino, "Lightweight reinforcement learning for energy efficient communications in wireless sensor networks," *IEEE Access,* vol. 7, pp. 29355-29364, 2019.

[90] V. Mnih, et. al., "Human-level control through deep reinforcement learning," *nature,* vol. 518, no. 7540, pp. 529-533, 2015.

[91] D. Niyato, E. Hossain and A. Fallahi, "Analysis of different sleep and wakeup strategies in solar powered wireless sensor networks," in *IEEE International Conference on Communications*, 2006.

[92] E. Ibarra, A. Antonopoulos, E. Kartsakli, J. J. Rodrigues and C. Verikoukis, "QoS-aware energy management in body sensor nodes powered by human energy harvesting," *IEEE Sensors Journal,* vol. 16, no. 2, pp. 542-549, 2015.

[93] D. Niyato, E. Hossain and A. Fallahi, "Sleep and wakeup strategies in solar-powered wireless sensor/mesh networks: Performance analysis and optimization," *IEEE Transactions on Mobile Computing,* vol. 6.2, pp. 221-236, 2006.

[94] C. Guihong, Y. Zhan, Y. Chen, L. Xiao, Y. Wang and N. An, "Reinforcement learning based power control for in-body sensors in WBANs against jamming," *IEEE Access,* 2018.

[95] A. K. Sangaiah, A. Javadpour, F. Ja'fari, H. Zavieh and S. M. Khaniabadi, "SALA-IoT: Self-reduced internet of things with learning automaton sleep scheduling algorithm," *IEEE Sensors,* 2023.

[96] W. B. Powel, Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions, Wiley, 2022.

[97] L. Tianxu, K. Zhu, N. C. Luong, D. Niyato, Q. Wu, Y. Zhang and B. Chen, "Applications of multi-agent reinforcement learning in future Internet: A comprehensive survey," *IEEE Communications Surveys and Tutorials,* vol. 24, no. 2, pp. 1240-1279, 2022.

[98] Z. Liu, B. Liu and C. W. Chen, "Joint power-rate-slot resource allocation in energy harvesting-powered wireless body area networks," *IEEE Transactions on Vehicular Technology,* vol. 67, no. 12, pp. 12152-12164, 2018.

[99] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li and Y. Gao, "A survey on federated learning," *Knowledge Based Systems,* vol. 216, 2021.

[100] J. D. Little and S. C. Graves, "Little's law," *Building intuition: insights from basic operations management models and principles,* pp. 81-100, 2008.

[101] X. Zhang, X. Jiang and M. Zhang, "A black-burst based time slot acquisition scheme for the hybrid TDMA/CSMA multichannel MAC in VANETs," *IEEE Communications Letters,* vol. 8, no. 1, pp. 137-140, 2018.

[102] J. Wu, H. Lu, Y. Xiang, F. Wang and H. Li, "SATMAC: Self-adaptive TDMA-based MAC protocol for VANETs," *IEEE Transactions on Intelligent Transportation Systems,* vol. 23, no. 11, pp. 21712-21728, 2022.

[103] G. Mavros, V. Asteriou, K. Kantelis, P. Nicopolitidis and G. Papadimitriou, "Node Classification-Based Bandwidth Allocation: A New Scheduling Algorithm for TDMA-Based LANs," *IEEE Access,* vol. 10, pp. 96212-96223, 022.

[104] J. Xu, Y. Zhang, Y. Zhao, J. Kan and L. Lin, "A TDMA protocol based on data priority for in-vivo wireless nanosensor networks," in *INFOCOM*, 2020.

[105] I. Jabandžić, S. Giannoulis, R. Mennes, F. De Figueiredo, M. Claeys and I. Moerman, "A dynamic distributed multi-channel TDMA slot management protocol for ad hoc networks," *IEEE Access,* vol. 9, pp. 61864-61886, 2021.

[106] S. Li, Y. Liu, J. Wang, Y. Ge, L. Deng and W. Deng, "TCGMAC: A TDMA-based MAC protocol with collision alleviation based on slot declaration and game theory in VANETS," *Transactions on Emerging Telecommunications Technologies,* vol. 30, no. 12, 2019.

[107] J. Burkardt, "The truncated normal distribution," Department of Scientific Computing Website, Florida State University, 2014.

[108] W. C. Horrace, "Moments of the truncated normal distribution," *Journal of Productivity Analysis,* vol. 43, pp. 133-138, 2015.

[109] C. Riou and J. Honda., "Bandit algorithms based on thompson sampling for bounded reward distributions," *Algorithmic Learning Theory, PMLR,* 2020.

[110] I. Osband, Z. Wen, S. Asghari, V. Dwaracherla, M. Ibrahimi, X. Lu and B. Van Roy, "Approximate thompson sampling via epistemic neural networks," *Uncertainty in Artificial Intelligence, PMLR,* pp. 1586-1595, 2023.

[111] M. A. Javed and S. Zeadally, "AI-empowered content caching in vehicular edge computing: Opportunities and challenges," *IEEE Network,* vol. 35, no. 3, pp. 109-115, 2021.

[112] M. Hadded, K. Toumi, A. Laouiti and P. Muhlethaler, "A trust framework for centralized tdma scheduling mechanism in vehicular ad hoc networks," *International Journal of Interdisciplinary Telecommunications and Networking (IJITN),* vol. 12, no. 4, pp. 74-87, 2020.

[113] M. S. Abdalzaher and O. Muta., "Employing game theory and TDMA protocol to enhance security and manage power consumption in WSNs-based cognitive radio," *IEEE Access,* vol. 7, pp. 132923-132936, 2019.

[114] A. Al-issa, M. Al-Akhras, M. ALsahli and M. Alawairdhi, "Using machine learning to detect DoS attacks in wireless sensor networks," in *IEEE Jordan international joint conference on electrical engineering and information technology*, 2019.

[115] F. Alsubaie, M. Al-Akhras and H. A. Alzahrani, "Using machine learning for intrusion detection system in wireless body area network," in *First International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, 2020.

[116] M. Nouman, U. Qasim, H. Nasir, A. Almasoud, M. Imran and N. Javaid, "Malicious node detection using machine learning and distributed data storage using blockchain in WSNs," *IEEE Access,* vol. 11, pp. 6106-6121, 2023.

[117] S. Ismail, T. Khoei, R. Marsh and N. Kaabouch, "A comparative study of machine learning models for cyber-attacks detection in wireless sensor networks," in *IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication*, 2021 .

[118] S. Ismail, Z. E. Mrabet and H. Reza, "An ensemble-based machine learning approach for cyber-attacks detection in wireless sensor networks," *Applied Sciences,* vol. 13, no. 1, 2022.

[119] R. Yuan, R. M. Gower and A. Lazaric, "A general sample complexity analysis of vanilla policy gradient," in *International Conference on Artificial Intelligence and Statistics. PMLR*, 2022.

[120] K. Shin, I. Park, J. Hong, D. Har and D. Cho, "Per-node throughput enhancement in Wi-Fi densenets," *IEEE Communications Magazine,* vol. 53, no. 1, pp. 118-125, 2015.

[121] A. Aldalbahi, M. Rahaim, A. Khreishah, M. Ayyash and T. Little, "Visible light communication module: An open source extension to the ns3 network simulator with real system validation," *IEEE Access,* vol. 5, pp. 22144-22158, 2017.

[122] K. Nguyen, Y. Ji and S. Yamada, "A cross-layer approach for improving WiFi performance," in *International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE*, 2014.

[123] B. Jung, N. Song and D. Sung, "A network-assisted user-centric WiFi-offloading model for maximizing per-user throughput in a heterogeneous network," *IEEE Transactions on Vehicular Technology,* vol. 63, no. 4, pp. 1940-1945, 2013.

[124] A. Baiocchi, "Maximizing the stable throughput of heterogeneous nodes under airtime fairness in a CSMA environment," *Computer Communications,* vol. 210, pp. 229-242, 2023.

[125] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial networks," *Communications of the ACM,* vol. 63, no. 11, pp. 139-144, 2020.

[126] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017.

[127] D. Kingma and M. Welling, "An introduction to variational autoencoders," in *Foundations and Trends in Machine Learning*, 2019, pp. 307-392.

[128] G. Yenduri, G. Srivastava, P.K.R. Maddikunta, R.H. Jhaveri, W. Wang, A.V. Vasilakos, and T.R. Gadekallu, "Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions," *arXiv preprint arXiv:2305.10435,* 2023.