# Distributed Time-Slotted Channel Hopping in IoT Networks using Multi-Agent Bernoulli Bandits

Hrishikesh Dutta, *Member, IEEE,* Noel Crespi, *Senior Member, IEEE,* and Amit Kumar Bhuyan, *Student Member, IEEE*

***Abstract*—The proliferation of Internet-of-Things (IoT) devices has led to significant challenges in managing radio access and energy efficiency in resource-constrained wireless sensor networks (WSNs). The Time-Slotted Channel Hopping (TSCH) protocol, part of the IEEE 802.15.4e standard, offers a promising solution for addressing these challenges, particularly through its collision-free scheduling mechanism. However, existing TSCH scheduling schemes often rely on centralized coordination or extensive control message exchanges, leading to inefficiencies in terms of energy consumption and bandwidth usage. We propose a novel decentralized TSCH scheduling framework driven by Bernoulli Multi-Armed Bandit (MAB) learning. Our approach enables each node to independently learn and optimize its transmission schedule without requiring control information from a central server or neighboring nodes. We explore multiple bandit action selection policies, including $\epsilon$-greedy, Upper Confidence Bound (UCB), and Thompson Sampling, and highlight their limitations in low data-rate networks. To overcome these challenges, we introduce a Low-Rate Resilient Policy (LRRP) that enhances TSCH scheduling in sparse traffic conditions by synthesizing packets to compensate for sample deficiencies. Experimental results demonstrate that our framework achieves collision-free scheduling and substantial energy savings while ensuring scalability for large networks. The proposed method outperforms traditional policies, particularly in heterogeneous and low-traffic environments, making it highly suitable for resource-constrained IoT networks. The bandit-driven scheduling approach is shown to achieve upto $\approx 19\%$ increase in throughput for a 50-nodes IoT network, compared with existing scheduling mechanism.**

***Index Terms*—Decentralized Scheduling, Time-slotted Channel Hopping (TSCH), Mult-Armed Bandit, IoT Networks.**

## I. Introduction

The rapid proliferation of Internet-of-Things (IoT) devices and Wireless Sensor Networks (WSNs) has introduced critical challenges in managing network traffic, energy consumption, and communication overhead. These technologies are widely deployed across various domains such as smart cities, agriculture, health monitoring, and wearables. As the number of connected devices continues to grow, ensuring reliable and energy-efficient connectivity has become increasingly difficult, particularly in resource-constrained environments. Key challenges include maintaining stable radio connectivity while optimizing energy usage and minimizing bandwidth overhead.

To address these challenges, several standard protocols have been developed for IoT and sensor networks, with specific focus on connectivity and power efficiency. Protocols such as WirelessHART [1], ISA100.11a [2], and IEEE 802.15.4e [3] provide mechanisms for reducing energy consumption while maintaining reliable communication. The IEEE 802.15.4e standard introduces the Time-Slotted Channel Hopping (TSCH) protocol, designed to ensure high reliability and low power consumption in IoT applications. TSCH allows devices to communicate using scheduled time-slots and channel hopping, which mitigates interference and enhances the chances of achieving collision-free communication.

TSCH schedules can be computed either centrally or in a distributed manner. Centralized approaches, such as those discussed in [4]–[9], rely on a central controller to assign transmission schedules based on global network knowledge. While this guarantees optimal schedule allocation, it imposes significant communication overhead between the controller and IoT nodes, leading to increased bandwidth usage and energy expenditure. On the other hand, distributed approaches [10]–[12] aim to allow nodes to coordinate schedules locally, reducing the need for extensive communication. However, these methods still rely on control message exchanges, as seen in Orchestra [10] and DIS-TSCH [11], introducing ad-

### *Notations and Nomenclature*

| | |
|---|---|
| $\mathcal{N}$ | Set of nodes cells |
| $\tau$ | Network Topology |
| $\mathcal{S}$ | Set of TSCH cells |
| $C$ | Number of channels |
| $T$ | Number of timeslots |
| $\mathcal{A}$ | Global Action space |
| $R_i(t)$ | Reward for player $i$ in epoch $t$ |
| $\alpha$ | Learning rate |
| $V_t*(i)$ | Value of an arm for player $i$ at time $t$ |
| $t_r$ | Ramp up duration |
| $T_L$ | Sleep scheduling kickoff duration |
| $\epsilon$ | Exploration parameter in epsilon-greedy |
| $c$ | UCB parameter |
| $\alpha_s, \beta_s$ | Beta distribution parameters for Thompson Sampling |
| $\mathcal{R}_k(t)$ | Regret for player $k$ at time $t$ |
| $a_*$ | Optimal arm |
| $\hat{\mu}^*$ | Optimal arm value |
| $\lambda$ | Poisson data rate mean |
| $\mathcal{F}$ | Mapping Function |
| ASN | Absolute Slot Number |
| $CH_{offset}$ | Channel offset |
| $N_{Ch}$ | Number of available channels |
| TSCH | Time Slotted Channel Hopping |
| MAB | Multi-Armed Bandit |
| LRRP | Low-Rate Resilient Policy |
| UCB | Upper Confidence Bound |
| TS | Thompson Sampling |
| MAC | Medium Access Control |
| ppf | Packets per frame |

Hrishikesh Dutta and Noel Crespi are at the Data Intelligence and Communication Engineering Lab, Telecom SudParis, Institut Polytechnique de Paris, France (email: hrishikesh.dutta@telecom-sudparis.eu; noel.crespi@telecom-sudparis.eu)

Amit Kumar Bhuyan is at the Electrical and Computer Engineering Department, Michigan State University, USA (email: bhuyanam@msu.edu)

ditional energy and bandwidth costs—an undesirable effect in resource-constrained networks. While several recent works have applied machine learning to optimize TSCH scheduling [13]–[15], they too often rely on centralized control or incur high communication overhead. For instance, [13] uses Hierarchical Reinforcement Learning (RL) to optimize schedules but still depends on centralized training, making it unsuitable for resource-constrained networks. Distributed RL-based approaches, such as those in [12], [16], address some of these issues but still require control message exchanges, resulting in energy and bandwidth overhead. Additionally, techniques like QL-TSCH [16] assume high network traffic, which limits their effectiveness in low-traffic IoT deployments.

In this paper, we propose a fully decentralized scheduling mechanism for IoT networks using the TSCH access scheme, generalized for any traffic conditions. Unlike existing approaches, our method enables each node to compute its transmission schedule independently, without relying on control message exchanges with other nodes or a central controller. This decentralized approach is particularly advantageous for low-power and resource-constrained IoT networks, as it reduces energy consumption and communication overhead. By eliminating the need for network-wide coordination, the proposed scheme also scales efficiently, handles network heterogeneity and adapts to dynamic conditions.

Our proposed scheduling mechanism models each node as a Bernoulli Multi-Armed Bandit (MAB) that learns from its experienced collisions to allocate collision-free transmission slots. While the proposed scheduling mechanism is designed to operate across a wide range of traffic conditions, special attention is given to low-traffic scenarios due to the limitations of conventional learning-based scheduling algorithms in sparse networks. Standard learning-based TSCH scheduling methods struggle to converge in heterogeneous and low-data-rate environments due to sample insufficiency. Since real-world IoT deployments often exhibit a mix of high and low traffic conditions, our approach ensures reliable scheduling performance under varying network loads, making it applicable to both dense and sparse deployments.

Note that while sparse traffic conditions reduce the likelihood of collisions, they do not guarantee collision-free scheduling. In fact, as shown later in this paper, there is more than 25% loss in performance in the traditional scheduling methods for data rate of 0.6 packet per frame in a 30-nodes multi-point-to-point networks. Additionally, such scenarios often lead to idle listening and wasted energy, particularly in these approaches. The proposed learning-assisted framework ensures robust scheduling by enabling nodes to adapt dynamically to traffic patterns and network heterogeneity. By optimizing sleep and transmit schedules through a decentralized learning approach, the proposed mechanism minimizes energy expenditure while maintaining performance. It also addresses the sample insufficiency challenge inherent in sparse traffic scenarios, enabling nodes to learn collision-free schedules even with limited data transmission. The specific contributions of this paper are as follows:

- We develop a fully decentralized TSCH scheduling architecture that operates without a central controller or inter-node coordination. The scheduling problem is formulated as a multi-player Bernoulli bandit game, where each node independently learns to find an efficient schedule. This is motivated from the decentralized decision-making capabilities of MAB models, which rely solely on local observations, such as collision feedback. Unlike predefined scheduling schemes that require prior knowledge of network conditions, the proposed approach can adapt to network heterogeneity and changing traffic conditions.
- A new MAB arm selection policy, incorporating synthetic packet generation, is proposed to enhance the scheduling scheme's performance in low-traffic environments. The efficiency of this policy is compared with the existing baseline policies for different networking scenarios.
- An analytical model for determining regret definitions and bounds for these bandit policies is presented in the context of the TSCH resource allocation problem. This model theoretically validates the proposed policy's performance improvement in low-traffic network conditions.
- A comprehensive analysis of the proposed scheduling mechanism is performed under various network conditions, including scalability and adaptability to heterogeneous IoT networks.

## II. Related Work

### A. Resource Allocation in IoT Networks

Scheduling and resource allocation for IoT and sensor networks have been an interesting topic for researchers. In [17], the authors developed an adaptive access protocol for vehicular ad-hoc networks (VANETs), following a Time Division Multiple Access (TDMA) scheme. The mechanism developed in that work relied on the information up to the three-hop neighborhood of a vehicular IoT node to support high priority safety applications. Simulation results demonstrated the efficiency of the protocol in terms of a higher packet delivery ratio (PDR) than that of the state-of-the-art approaches. Similar TDMA scheduling protocols were developed for dynamic networks in [18], that allow nodes to find a collision-free slot in real time based on their neighbors' access scheme information. The performance of these MAC scheduling schemes, however, rely on the accuracy of information from the neighboring nodes. The authors in [19] introduced a prediction-enabled TDMA MAC protocol (PTMAC), in which each node continuously monitors for the occurrence of packet collisions in each slot in a TDMA frame. Upon collision detection, the node informs its neighbor to change its schedule to another time slot. Additionally, a node can detect potential collisions beyond its two-hop neighborhood by receiving notifications from its neighbors, allowing it to further adjust its time slot. However, when a node is reassigned to a free time slot, PTMAC may introduce new potential collisions, as it does not account for collisions within these previously unused slots. Another TDMA-based access control scheme (TCGMAC) for non-stationary topologies is proposed in [20], in which a game-theoretic approach for scheduling using a hybrid TDMA and CSMA access scheme is adapted. The paper demonstrates the ability of the protocol to reduce latency and packet loss, but does not guarantee a collision-free scheduling.

### B. TSCH Scheduling Mechanisms

There are existing works that deal with scheduling in TSCH networks. The work in [4] proposes a mobile scheduling protocol for TSCH-based IoT networks. That proposed MSU-TSCH algorithm enables cell allocation by establishing a virtualization connection between the sink and the IoT nodes. Such an approach reduces the collisions and energy expenditure in the network, but relies on a central server for updating the scheduling policies. In [5], the authors develop a TSCH cell allocation strategy for a niche low-latency application of in-vehicle sensor network. A cross layer mechanism is adopted for topology management and TSCH scheduling using an optimized graph isomorphism algorithm. The proposed approach is applicable for networks with limited sensor nodes that send data to a central sink. A polynomial-time algorithm for maximizing network throughput in a TSCH network with centralized connectivity is proposed in the work reported in [6]. In addition, an auction based scheduling policy is designed to assist the throughput maximizing algorithm that also ensures a fair bandwidth allocation across the network. The centralized TSCH algorithm proposed in [7] executes a sequential multi-hop scheduling by allocating cells to links following non-conflict and non-interference rules. A central controller does the allocation by assigning an optimized number of cells to each link depending on the network traffic conditions. In the work reported in [8], the authors develop an SDN-controlled scheduling scheme for industrial IoT applications. The centralized SDN controller does resource allocation using a link quality estimation algorithm while keeping latency at check. Similar centralized schedulers designed for wireless industrial networks are reported in [9]. Their technique of time-grouping different links allows the scheduler to achieve better reliability than the existing scheduling mechanisms.

### C. Reinforcement Learning-enabled Scheduling

Dynamic channel allocation for satellite IoT network using a centralized Deep Reinforcement Learning scheme is proposed in [21]. The proposed mechanism allows a central controller to learn scheduling decisions on-the-fly to reduce the average transmission latency. Similarly in [22], the authors exploit Phasic Policy Gradient Reinforcement Learning for resource allocation in TSCH network of sensors transmitting data to a central base station. A runtime resource management framework for TSCH wireless sensor networks in real time using Reinforcement Learning is proposed in [13]. The authors use a data-driven approach for self-adaptation of optimal sloframe length. In [14], the researchers consider the problem of energy management in TSCH networks and propose an RL algorithm for efficient radio scheduling to reduce energy expenditure. The reported performance shows the ability of the protocol to realize power savings while achieving similar networking performance. Search for an optimal channel hopping using MAB is reported in [15]. The proposed approach improves network reliability and energy efficiency, without guaranteeing a collision-free transmission. The paper [23] develops a strategy, where different RL agents address a multi-objective problem, optimizing throughput, power efficiency, and network delay based on predefined application requirements. All these centralized scheduling approaches ensure a high reliability in cell allocation, as the decisions are controlled by a single entity (the central controller/server). However, in addition to putting the entire computation burden on the central controller, these approaches also require additional energy and bandwidth overhead for communicating the policies and learning observables to and from the server.

### D. Decentralized Scheduling Approaches

There are a few recent attempts to formulate the decentralized implementation of the TSCH scheduling problem. In the work reported in [12], the authors break down the centralized implementation into several localized implementation with the computation at a parent node within a cluster. The cell allocation problem is coined as a Markov Decision Process (MDP) at each parent node and Deep Q-Learning is used for function approximation of the scheduling actuation. The CSMA-CA enabled RL policy improves the QoS, while trading it off with learning convergence. Similarly, multi-agent Q-learning is adopted for TSCH scheduling in a decentralized manner in [16]. The proposed mechanism reduces the network collision without eliminating contention. The authors in paper [10] investigate the use of TSCH protocol with Orchestra scheduling approach for decentralized IoT arrangement. The demonstrated performance reported high end-to-end latency for heterogeneous traffic distribution. The work in [11] presented a constant-time distributed scheduling algorithm for multi-hop sensor networks. The CSMA-CA based protocol reduces network delay and energy wastage, while still not allowing a contention-free allocation. A non-stationary policy for multi-user spectrum sharing in cognitive radio network is handled in a decentralized manner in [24]. The problem of channel allocation in IoT networks over unlicensed spectrum is modeled as a contextual multi-player MAB game in [25]. The paper aims to find an optimal channel allocation while finding a good balance between efficiency and scalability. A multi-player MAB approach for decentralized TSCH cell allocation in a heterogeneous network is presented in [26]. The learning-driven protocol increases network throughput but does not provide a collision-free schedule. In addition, the collisions increase with a rise in the number of users accessing the spectrum, leading to scalability issues. Although implemented in a decentralized setting, these approaches rely on signaling and control information sharing among neighbors, which lead to energy-bandwidth overhead. In addition, most approaches allow contention to exist, thus, limiting them for applications with no access delay concerns. Moreover, these protocols with decentralized learning are all developed for networks with high data rates and do not perform well in low traffic conditions, due to sampling problems. Our work considers all these aspects, while developing the decentralized TSCH scheduling framework using Bernoulli Bandit, making it suitable for a wide range of networks incorporating heterogeneity, scalability and low-traffic conditions.

Although Multi-Armed Bandit (MAB) models have been explored in other contexts, applying them to decentralized Time-Slotted Channel Hopping (TSCH) scheduling in

TABLE I: Summary of Related Work

| Study | Approach | Strengths | Limitations |
|---|---|---|---|
| [17]–[20] | TDMA-Based Scheduling | High PDR, Adaptive collision detection | Relies on accurate neighbor information |
| [4]–[9] | Centralized TSCH Scheduling | Optimized collision avoidance, Energy-efficient | High overhead, Not scalable |
| [13]–[15], [21]–[23] | RL for TSCH Scheduling | Adaptive to dynamic traffic | Centralized connectivity requirement, Scalability issues |
| [10]–[12], [16], [24]–[27] | Decentralized Scheduling | Throughput improvement | Limited to high traffic rate, Lacks collision-free guarantee |

resource-constrained IoT networks introduces several unique challenges. First, sparse traffic conditions result in sample deficiencies, hindering the learning process. To address this, the proposed framework generates synthetic packets to ensure sufficient training data. Second, achieving collision-free scheduling without explicit node coordination requires innovative approaches to decentralized learning. Our method enables nodes to infer optimal schedules based solely on local collision feedback, overcoming scalability issues associated with centralized systems. Lastly, balancing trade-offs between energy efficiency, throughput, and latency in dynamic and heterogeneous networks adds further complexity. The proposed framework addresses this by optimizing a long-term reward function, ensuring a balanced and adaptive scheduling strategy.

## III. Background

### A. Time Slotted Channel Hopping (TSCH)

Time Slotted Channel Hopping is one of the operating modes of the IEEE 802.15.4 standard [3]. It was primarily developed for reducing collisions by providing slotted access across different channels in a medium. It has also been shown to be effective in mitigating the effects of multipath fading and interference achieved using channel hopping mechanism.

In the TSCH protocol, the network is assumed to be time synchronized. A slotframe is discretized into certain timeslots, where a node transmits a packet and receives an acknowledgement. If a packet transmission fails, the node attempts retransmission in the next available time slot, following the TSCH collision avoidance protocol. The size of the slotframe determines the maximum number of schedulable timeslots and the timeslot cycle, which is preset based on network size and degree. At the beginning of each timeslot, a TSCH node selects a frequency channel according to a predefined hopping sequence. The channel hopping sequence is defined by the network designer and is used to determine the frequency channel for each timeslot. The channel selection ($Ch_i$) for node $i$ is executed using the following equation.

$$Ch_i = \mathcal{F}[(ASN + CH_{Offset})\%N_{Ch}] \tag{1}$$

Here, $\mathcal{F}$ is the mapping function that defines the hopping sequence, $ASN$ is the absolute slot number representing the number of timeslots elapsed since the start of the network, $CH_{Offset}$ is the channel offset and $N_{Ch}$ is the number of available channels.

The TSCH scheduler assigns specific timeslots for communication between nodes. In the traditional approach, a centralized arbitrator computes the schedule for each of the TSCH nodes, which then downloads and executes the scheduling protocol. There are decentralized implementations of the TSCH scheme as well, as proposed by some recent works [10]–[12]. Here the nodes cooperate with one another, by means of extra signaling, such as via a hash function, to compute a collision-free schedule. In addition to collision control, one additional requirement of an efficient schedule is energy management. In other words, the nodes are expected to be asleep in all the timeslots where they are not expected to transmit or receive packets. Thus, an effective channel hopping schedule would enable the resource-constrained nodes to save energy.

### B. Multi-Armed Bandits (MAB)

Multi-Armed Bandits [28] is a class of Reinforcement Learning (RL) Algorithms [29] used in non-associative settings. This kind of framework is applicable in scenarios where the system or the learning environment does not transition from one state to another. Essentially, an MAB algorithm does not possess the state-based concept found in traditional reinforcement learning. A well-studied variant within MAB is the '$k$-armed bandit' problem. Here, the learning agent, or bandit, has $k$ potential arms or actions to select from. Each arm is associated with a stochastic reward whose distribution is unknown to the agent. Upon choosing an arm or action, the agent receives a sample reward based on this unknown distribution, providing feedback on the selected action's outcome on the system performance. The agent's objective is to maximize the total cumulative reward over an infinite time horizon by learning to estimate the reward distributions of the available actions.

Formally, the value $V(a,t)$ of an arm $a$ at time $t$, in simplest form, is given by the discounted sample average of the reward associated with the arm, that is,

$$V(a,t) = V(a,t-1) + \alpha \times r_a(t) \tag{2}$$

The MAB agent or the bandit picks an arm $a$ from the set of possible arms $\mathcal{A}$, observes the reward and updates the arm value using Eqn. 2, with a learning rate $\alpha$. By the law of large numbers, the model converges when the number of reward samples collected is sufficient enough that the discounted sample average of the reward distribution becomes close to the expected reward for all the arms. This condition can be expressed as:

$$|\mathbb{E}[r_a] - \lim_{N\to\infty}\sum_{t=1}^{N}\alpha^t \times r_a(t)| < \epsilon; \forall a \in \mathcal{A} \tag{3}$$

Bernoulli Bandit [30], [31] is a class of Multi-Armed Bandit problem, where the reward distribution follows Bernoulli Distribution. This indicates that, for each arm, the reward is binary and occurs with probability $p$ and $(1-p)$ respectively. Many real world events that have binary outcomes can be modeled using a Bernoulli Bandit problem, for example, clinical trials, A/B testing etc. Although, theoretically, as shown in [32], it is possible to compute a deterministic optimal policy (OPT) for a Bernoulli reward. However, for anything beyond some simple scenarios, finding such policies is considered infeasible in current research [33] for which suboptimal policies are used in practice.

## IV. Decentralized TSCH Cell Allocation using Bernoulli Bandits

The proposed TSCH scheduling mechanism, as mentioned earlier, is developed considering completely decentralized implementation. That is, each TSCH node node is allowed to find its schedule independently without the control of a central arbitrator. This gives the advantage of accomplishing cell allocation with a reduced bandwidth and energy overhead, owing to the removal of the need of sharing and downloading control information and scheduling policies.

Let us consider a network with the set of nodes $\mathcal{N}$ and a given topology defined by a directed graph $\tau$. As shown in Fig. 1, each node is equipped with a Bernoulli Bandit that has $|\mathcal{S}|$ number of arms, where $\mathcal{S}$ represents the set of cells in a TSCH slotframe. Note that $|\mathcal{S}| = C \times T$, where $T$ and $C$ denote the number of timeslots and channels in a slotframe, respectively. The bandit action is to pick a cell from set $\mathcal{S}$ such that the collisions in the entire network are minimized. Thus, the entire network can be visualized as a multi-player Bernoulli Bandit, where each agent or bandit interacts via its arm or action. Each player independently and without information sharing, aims to cooperatively find a schedule that minimizes MAC packet collisions in the network.

Thus, we have a multi-player bandit scenario, where the set of players is $\mathcal{N} = 1, 2, ....., N$. Here, the action space of each player $i \in \mathcal{N}$ is the set of $|\mathcal{S}|$ arms $A_i = \mathcal{S}$. The global action space in this setting can be given as $\mathcal{A} = \cup_{i=1}^{N} A_i$. For a finite time-horizon $T$ , and the arm $a_i(t)$ chosen by player $i$ at time $t$, the action profile **a(t)** is defined as the vector of the actions taken by the players, that is, $\textbf{a(t)} = \{a_1(t), a_2(t), ....a_N(t)\}$. Because of the decentralized nature of the problem, any bandit (or player) can only know about its own arm and has no knowledge about the arms pulled by other bandits or players in the environment.

The Bernoulli reward for a player $i$, in this scenario, is dictated by the fate of the transmitted packet in the chosen TSCH cell. In other words, the player $i$ receives a reward of $0$ if the packet gets collided due to overlapped transmission by other nodes in the network; otherwise it receives a reward of $1$ for each successful packet transmission. Formally, reward for player $i$ in an MAB decision epoch $t$ is defined as:

$$R_i(t) = \begin{cases} 1, \text{for successful transmission by } i \text{ at epoch } t \\ 0, \text{for collision} \end{cases} \tag{4}$$

From the bandit point of view, if a set of players $\mathcal{P} \subset \mathcal{N}$ that are directly connected to each other, defined by graph $\tau$, pick the same arm $a_*(t)$, then each of these players will receive a reward drawn from a distribution with zero mean.

The objective of each player (that is, node in this context), in a global sense, is to pick an arm $a^* \in \mathcal{A}$ that maximizes the expected long term reward.

$$a^* = \text{arg max}_{a \in \mathcal{A}} \sum_{i=1}^{N} \mathbb{E}[r_i(a_i, \mathcal{P})] \tag{5}$$

From the perspective of TSCH, the concept used here is that each node, acting as a Bernoulli Bandit, will learn to independently find a TSCH transmission schedule ($\mathcal{S}_{Tx} \subset \mathcal{S}$) such that there is no packet collision. This is demonstrated by an example shown in 2 for a simple 4-node TSCH network. Each node selects a transmission cell and monitors whether the transmission results in success or collision. Depending on the transmission outcome, it updates its learning parameters using the specified reward function to maximize long-term reward. Initially, as illustrated in Fig. 2, each node randomly selects TSCH cells, causing overlapping transmissions and collisions. However, as learning advances, each node learns to choose cells that avoid collisions, achieving a collision-free state after convergence.

Once the transmission scheduling is done, the next course of action for a node is to determine the listening schedule, that is, to find out on which TSCH cells it should remain awake for successful reception of MAC packets from its neighbors. The problem here boils down to finding the TSCH schedules of the neighboring nodes. This is important because if the listening schedule is not efficient, there will be idle listening that would lead to energy wastage. Nevertheless, once the transmission schedule is determined using the Bernoulli Bandit framework detailed above, determining the listening schedule is straightforward. This is accomplished by the sleep-listen scheduler (indicated in Fig. 1) after the transmission scheduling bandit agents converge. The sleep-listen scheduler ensures that the wireless node it is part of, remains awake in all the TSCH cells for a kickoff duration, comprised of $T_L$ number of slotframes. Within this kickoff duration, the sleep-listen scheduler computes the TSCH cells on which the node should expect packet transmissions from its neighbors. If it receives $p(>0)$ number of packets in cell $s \in \mathcal{S}$, then the node remains on in cell $s$ for listening after the kickoff duration. For the set of cells $\mathcal{S}_L^C \subset \mathcal{S} - \mathcal{S}_{Tx}$ where $p = 0$ after $T_L$ slotframes, the sleep-listen scheduler makes the node to turn off the radio transceiver in those cells.

## V. Bandit Policies for TSCH Scheduling

As explained in Section IV, each TSCH node is equipped with a Bernoulli Bandit agent that independently executes a learning policy to achieve the common network objective of finding a collision-free transmission schedule. Transmission scheduling, thus, in this context, is simply finding a TSCH cell in the slotframe such that there is no overlap in packet transmission. To achieve this, each bandit picks an arm (that is, a TSCH cell) to interact with the environment (wireless

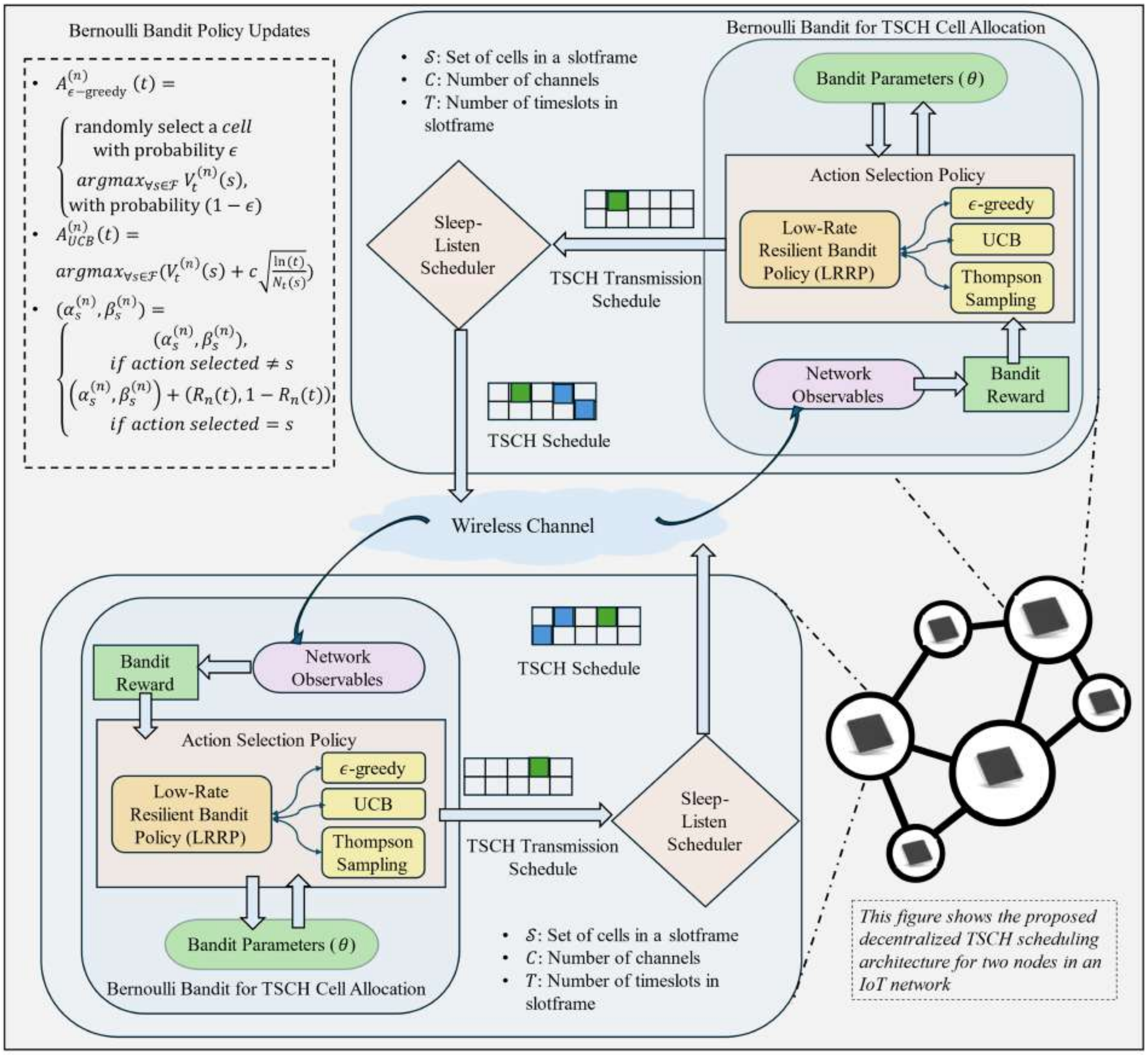

Bernoulli Bandit Policy Updates
$A^{(n)}_{\epsilon-\text{greedy}}(t) =$
$\begin{cases} \text{randomly select a } cell \text{ with probability } \epsilon \\ argmax_{\forall s \in \mathcal{F}} V_t^{(n)}(s), \text{ with probability } (1-\epsilon) \end{cases}$
$A^{(n)}_{UCB}(t) =$
$argmax_{\forall s \in \mathcal{F}}(V_t^{(n)}(s) + c\sqrt{\frac{\ln(t)}{N_t(s)}})$
$(\alpha_s^{(n)}, \beta_s^{(n)}) =$
$\begin{cases} (\alpha_s^{(n)}, \beta_s^{(n)}), \; if\ action\ selected \neq s \\ \left(\alpha_s^{(n)}, \beta_s^{(n)}\right) + (R_n(t), 1 - R_n(t)), \; if\ action\ selected = s \end{cases}$
Bernoulli Bandit for TSCH Cell Allocation
$\mathcal{S}$: Set of cells in a slotframe
$C$: Number of channels
$T$: Number of timeslots in slotframe
Bandit Parameters ($\theta$)
Action Selection Policy
Low-Rate Resilient Bandit Policy (LRRP)
$\epsilon$-greedy
UCB
Thompson Sampling
Sleep-Listen Scheduler
TSCH Transmission Schedule
TSCH Schedule
Network Observables
Bandit Reward
Wireless Channel
Network Observables
Bandit Reward
TSCH Schedule
Action Selection Policy
$\epsilon$-greedy
UCB
Low-Rate Resilient Bandit Policy (LRRP)
Thompson Sampling
TSCH Transmission Schedule
Sleep-Listen Scheduler
Bandit Parameters ($\theta$)
Bernoulli Bandit for TSCH Cell Allocation
$\mathcal{S}$: Set of cells in a slotframe
$C$: Number of channels
$T$: Number of timeslots in slotframe
This figure shows the proposed decentralized TSCH scheduling architecture for two nodes in an IoT network

Fig. 1: Bernoulli Bandit Framework for Decentralized TSCH Scheduling

network) and receives a Bernoulli reward that evaluates the selected arm. The goal of the bandit is to find an arm selection policy that maximizes the expected reward.

We first explore three standard state-of-the-art bandit policies: $\epsilon$-greedy, Upper Confidence Bound (UCB) and Thompson Sampling. We briefly summarize these policies in this section, talk about their limitations in the context of this problem of decentralized TSCH cell allocation and then propose a Low Rate Resilient Policy (LRRP) that is built on top of these standard policies to overcome these limitations.

***Baseline Policies***: Three standard action selection policies are explored as baselines in this paper: $\epsilon$-greedy, UCB and Thompson Sampling [28], [34], [35]. For $\epsilon$-greedy policy, the exploration-exploitation trade-off can be manually tuned by a user-defined parameter $\epsilon$ that dictates the probability of arm-selection stochasticity. In other words, the arm with the maximum value is selected with probability $(1-\epsilon)$ and all other arms are selected at random with probability $(\epsilon)$, which decays (exponentially, or linearly, based on the application requirements) as learning progresses. Formally, the arm selection logic for a node $i$, at a learning decision epoch $t$, can be expressed as

$$a_i^{\epsilon\text{- greedy}}(t) = \begin{cases} \text{randomly select a cell } s \text{ with probability } \epsilon \\ \text{arg max}_{\forall s \in \mathcal{S}} V_t^{(i)}(s) \text{with probability } 1-\epsilon \end{cases} \tag{6}$$

Here $V_t^{(i)}(s)$ denotes the value of the arm (or TSCH cell) $s$ selected by bandit (or node) $i$ at time $t$; which is updated as

$$V_t^{(i)}(s) = V_{t-1}^{(i)}(s) + \alpha(R_i(t) - V_{t-1}^{(i)}(s)) \tag{7}$$

In Eqn. 7, $\alpha$ is the learning rate and $R_i(t)$ is the reward received by bandit $i$ at time $t$.

On the other hand, in UCB arm selection policy, the exploration is adaptive and data-driven. MAB regret minimization in UCB is handled in a more elegant manner by favoring less explored arms that are likely to yield high rewards. This dynamic, confidence-interval-based exploration-exploitation balance is executed as:

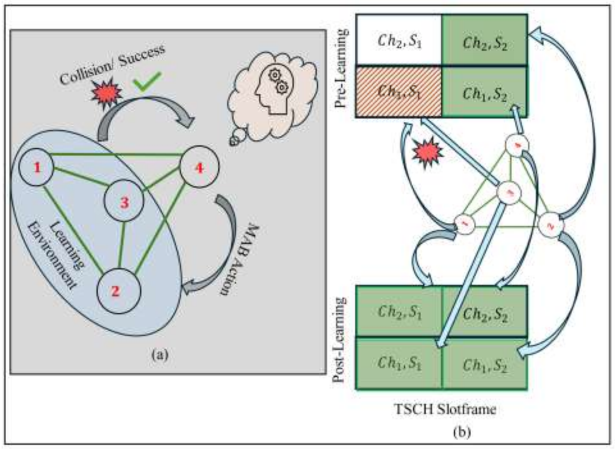


Fig. 2: An example scenario of learning TSCH transmission schedule in a four-nodes network.

$$a_i^{\text{UCB}}(t) = \text{arg max}_{\forall s \in \mathcal{S}}(V_t^{(i)}(s) + c\sqrt{\frac{ln(t)}{N_t(s)}}) \tag{8}$$

Here, $N_t(s)$ denotes the number of times cell $s$ has been picked as the bandit arm for transmission till epoch $t$ and hyperparameter $c$ controls the exploration-exploitation trade off. The value update for $V_t^{(i)}(s)$ is executed using Eqn. 7.

Note that for the $\epsilon$-greedy and UCB policies mentioned above, the bandit executes exploratory behavior using hyperparameters $\epsilon$ and $c$, to ensure that the bandit doe not get stuck to a sub-optimal arm. Using such 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 in this case.

Another baseline MAB action selection policy considered in this work is Thompson Sampling, which takes a Bayesian approach to navigate the exploration-exploitation dilemma in arm selection. In this approach, a prior probabilistic reward distribution is associated with each arm of a bandit. In this paper, $\beta$-distribution is considered 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 $(\alpha_s^{(i)}, \beta_s^{(i)})$ of node $i$'s distribution associated with arm (cell) '$s$' are updated at epoch $t$ using the following equation:

$$(\alpha_s^{(i)}(t), \beta_s^{(i)}(t)) = \begin{cases} (\alpha_s^{(i)}(t), \beta_s^{(i)}(t)) \\ \qquad\qquad , \text{ if arm selected} \neq \text{s} \\ (\alpha_s^{(i)}(t), \beta_s^{(i)}(t)) + (R_i(t), 1 - R_i(t)) \\ \qquad\qquad , \text{ if arm selected = s} \end{cases} \tag{9}$$

At the end of each learning epoch, samples are drawn from each arm's respective $\beta$- distribution and the arm with the highest sample value is picked. As the learning progresses, these distributions converge to the true reward distributions of the corresponding bandit arms which ensures that the arm with the highest expected reward value is chosen.

**Algorithm 1** Low Rate Resilient Policy (LRRP)

**Input:**

- Slotframe $\mathcal{S}$, TSCH cells $\{s_1, s_2, \ldots, s_k\}$
- Set node data packet rate $\lambda$ (packets per frame)
- Baseline bandit action selection policy (e.g., Thompson Sampling, $\epsilon$-greedy, UCB)
- Hyperparameters: $t_r$ (ramp-up duration), $\alpha$ (learning rate), $\epsilon$ (for $\epsilon$-greedy), $c$ (for UCB), prior distributions $\beta_s^{(i)}(\alpha_s, \beta_s)$ (for Thompson Sampling)

**Output:**

- Optimal TSCH cell $s^*$ for collision-free transmission

**Initialization:**

- Set value estimates $V_0^{(i)}(s)$ for each cell $s \in \mathcal{S}$
- Set prior parameters for $(\alpha_s^{(i)}, \beta_s^{(i)})$ for Thompson Sampling
- Initialize the slotframe and prepare nodes for transmission scheduling

```
for each node $i$ do
    for each learning epoch $t$ do
        if $t \leq t_r$ then
            if MAC queue of node $i$ is empty then
                Generate a synthetic noise packet
            end if
        end if
        Execute action selection policy:
        if $\epsilon$-greedy policy then
            Randomly select a cell $s$ with probability $\epsilon$,
             else choose $s = \arg\max_{\forall s \in \mathcal{S}} V_t^{(i)}(s)$
        else if UCB policy then
            $s = \arg\max_{\forall s \in \mathcal{S}} \left(V_t^{(i)}(s) + c\sqrt{\frac{\ln(t)}{N_t(s)}}\right)$
        else if Thompson Sampling policy then
            Sample $\mathcal{X}_t^{(i)}(s)$ from $\beta_s^{(i)}(\alpha_s, \beta_s)$,
             $s = \arg\max_{\forall s \in \mathcal{S}} \mathcal{X}_t^{(i)}(s)$
            $(\alpha_s^{(i)}(t), \beta_s^{(i)}(t)) =$
                $\begin{cases} (\alpha_s^{(i)}(t), \beta_s^{(i)}(t)), \\ \text{ if arm selected} \neq s \\ (\alpha_s^{(i)}(t), \beta_s^{(i)}(t)) + (R_i(t), 1 - R_i(t)), \\ \text{ if arm selected} = s \end{cases}$
        end if
        Transmit packet in selected cell $s$
        Compute Bernoulli reward $R_i(t)$
        Update value estimate for selected cell $s$:
        $V_t^{(i)}(s) = V_{t-1}^{(i)}(s) + \alpha(R_i(t) - V_{t-1}^{(i)}(s))$
    end for
end for
```

One common characteristic of these policies is that the bandit's model updates depend on the rate at which it is being able to sample the reward distribution. In other words, the model weight update frequency has the bottleneck of the rate of generation of the observables that can contribute to reward computations. In this context, since the reward is computed from the fate of a packet transmission (success/collision), the bandit update frequency is bounded by the packet generation rate of the TSCH nodes. To exemplify, for a homogeneous network data rate of $\lambda$ packets per frame (ppf), the nodes on the average would have to wait for $1/\lambda$ number of slotframes to update the learning parameters of the bandit it is associated with. This would, in turn, effect the learning performance, convergence speed and eventually would impact on the algorithm's adaptability to network dynamism as well as its key performance indices. Given that the learning convergence duration should be sufficiently smaller than the network time constant for an efficient and adaptive protocol, this slow convergence would lead to an inability to adapt to dynamic network conditions with a low data rate. Additionally, when the network size (that is the number of interacting bandits) increases, the learning convergence would become even slower. This explanation can be mathematically validated using the regret analysis presented in Section VI.

In order to ameliorate the above limitations, a strategy for handling the low data rate scenario is developed. In this proposed Low Rate Resilient Policy (LRRP), a *learning ramp-up* phase is introduced while training the bandit, following an action selection policy. During this phase, comprising of $t_r$ epochs, each node generates MAC packets with the payload containing encoded noise samples generated using a stochastic distribution, if the packet queue is empty because of low data traffic. The generated packets are transmitted following the actuated transmission schedule driven by the bandit arm selection policy.

The physical interpretation of this strategy is that in the event of low data traffic, the learning model is assisted using samples synthetically generated using the same policy, as would have been followed if there were any packets in the transmission queue. Although the payload in the packets contain noise that is irrelevant to the receiver, the transmission scheduling of these packets is done by following the same bandit model. In other words, at any given point of time $t$, the bandit arms are drawn from the distribution $\mathcal{D}(\theta; t)$ and its parameters are updated (depending on the bandit action selection policy), irrespective of the payload. This helps the node to estimate the reward distribution, and consequently the TSCH scheduling of its directly connected neighbors, so as to find a collision-free schedule in a decentralized manner. While this approach is effective in both high and low traffic conditions, sparse networks introduce a unique challenge: standard learning-based scheduling policies suffer from sample insufficiency, leading to suboptimal convergence. The proposed framework enhances learning in these scenarios by ensuring a stable scheduling even when network conditions are not ideal. This design choice allows the proposed framework to operate efficiently across different IoT deployments, regardless of data traffic volume.

Note that the Low Rate Resilient Policy (LRRP) encapsulates a baseline bandit policy which each player follows to draw its arm from. This is the same policy that the bandits use for sample generation when there is sample deficiency due to low network data traffic. Thus, its regret bound and the convergence behavior would be dictated strongly by the baseline bandit arm selection logic used. In this work, we focus on Thompson Sampling-enabled LRRP mechanism, owing to its strong theoretical regret bound and its robust behavior in estimation of the reward distribution [28]. The steps of execution of Low Rate Resilient Policy (LRRP) are formally enumerated in Algorithm 1. The bandit regret definition in the context of TSCH scheduling and the bounds for different policies are formulated in the following section.

It is worth emphasizing here that in the proposed decentralized scheduling framework, no explicit control information exchange occurs between nodes. Instead, implicit coordination is achieved through local observations of collisions. Each node independently learns its optimal schedule using a Bernoulli Multi-Armed Bandit (MAB) approach, where decisions are biased by a long-term reward function. This design choice reduces communication overhead and energy consumption, particularly in large-scale IoT networks. Experimental results (see Section VII) demonstrate that the proposed framework achieves comparable throughput and low collision rates without requiring explicit node interaction. Additionally, the Low-Rate Resilient Policy (LRRP) addresses potential sample insufficiency in sparse traffic conditions, ensuring robust schedule optimization and maintaining network performance.

Notably, the proposed Bernoulli MAB learning framework differs from existing MAB-based scheduling approaches in several ways. First, the Low-Rate Resilient Policy (LRRP) addresses the challenge of sample insufficiency in sparse traffic scenarios by generating synthetic packets during idle periods, ensuring reliable learning even under low-data-rate conditions. Second, unlike traditional methods that rely on explicit coordination between nodes, the proposed framework eliminates communication overhead by enabling fully decentralized learning based solely on local collision feedback. Third, the framework simultaneously optimizes multiple network objectives, including throughput, and energy efficiency, making it particularly suitable for resource-constrained IoT networks.

## VI. Convergence and Complexity Analysis

### A. Regret Bounds

In this section we define the Bandit regret in the context of TSCH cell allocation and present an analysis on the theoretical regret bounds that we obtain from different arm selection policies mentioned earlier in section V. In general, MAB regret for agent $k$ at epoch $T$ can be expressed by the following equation.

$$\mathcal{R}_k(T) = T\mu^*(k) - \sum_{t=1}^{T} \mathbb{E}[\mu(a_k(t))] \tag{10}$$

In Eqn. 10, $\mu^*(k))$ is the expected reward for the optimal arm, and $\mu(a_k(t))$ is the reward for arm $a_k(t)$ selected by $k$ in epoch $t$. The regret for the entire system with $N$ players (IoT nodes) can be expressed from Eqn. 10 as follows.

$$\hat{\mathcal{R}}(T) = T\hat{\mu}^* - \sum_{t=1}^{T} \mathbb{E}[\hat{\mu}(\mathbf{a}(\mathbf{t}))] \tag{11}$$

In the scenario of TSCH cell allocation with the bandit reward given by Eqn. 4, the optimal expected reward is achieved the case when the nodes find a cell that does not overlap in timeslot or in channel in the slotframe. In that case, the value of the optimal expected reward $\hat{\mu}^* = N$. Before moving forward, one generalization that is required for Eqn. 11 in the context of this paper is the inclusion of packet generation rate ($\lambda_k$). Additionally, the optimal arm value becomes $\hat{\mu}^* = \sum_{k=1}^{N}\lambda_k$. Then the regret expression can be defined as:

$$\hat{\mathcal{R}}(T;\boldsymbol{\lambda}) = T\sum_{k=1}^{N}\lambda_k - \sum_{t=1}^{T}\mathbb{E}[\hat{\mu}(\mathbf{X}(\mathbf{t})^T.\mathbf{a}(\mathbf{t}))] \tag{12}$$

Here $\mathbf{X}(\mathbf{t})$ is an $N$-D vector with each element comprised of an indicator function $\mathbb{1}_k(t), 1 \le k \le N$, which is defined as:

$$\mathbb{1}_k(t) = \begin{cases} 1, \text{ if there is a packet to be transmitted by } k \text{ at } t \\ 0, \text{ otherwise} \end{cases} \tag{13}$$

The regret bounds for the baseline bandit policies referred in section V can be derived in the context of this problem using the formulation given in [36]–[40]. Following the same approach, the regret bounds for the proposed LRRP with these baseline policies can be derived as follows. Since LRRP allows the nodes to generate synthetic packets in the event of low data rate, the effective sample generation rate becomes $\lambda_k \to 1$. Thus the regret bounds can be obtained as:

$$\hat{\mathcal{R}}(T;\boldsymbol{\lambda})_{\text{LRRP-}\epsilon\text{-greedy}} \le \mathcal{O}(T^{-1/3}(|\mathcal{S}| \times logT)^{1/3}) \tag{14}$$

$$\hat{\mathcal{R}}(T;\boldsymbol{\lambda})_{\text{LRRP-UCB}} \le 8\sum_{i:\mu_i<\mu^*}\frac{lnT}{\mu^*-\mu_i} + (1+\frac{\pi^2}{3})\sum_{i=1}^{|\mathcal{S}|}(\mu^*-\mu_i) \tag{15}$$

$$\hat{\mathcal{R}}(T;\boldsymbol{\lambda})_{\text{LRRP-TS}} \le \mathcal{O}(\sqrt{NT \times lnT}) \tag{16}$$

In the above bounds, $\mathcal{S}$ denotes the set of cells in a TSCH slotframe and $N = |\mathcal{N}|$ is the number of nodes contending for cells in that frame, as has been explained in section IV.

### B. Complexity Analysis

The computational complexity of the proposed decentralized TSCH scheduling framework is analyzed based on its major components: Multi-Armed Bandit (MAB) parameter update, reward calculation, and schedule optimization. This analysis considers both per-node and network-level complexities to highlight the efficiency and scalability of the proposed method.

Each node maintains $|\mathcal{S}|$ arms, representing time slots, and updates the reward for the selected arm based on observed collisions. For each selected arm, the reward is updated in $\mathcal{O}(1)$ time. The selection of an arm involves computing or retrieving the action policy (e.g., $\epsilon$-greedy, UCB, or Thompson Sampling). For $\epsilon$-greedy, the complexity is $\mathcal{O}(|\mathcal{S}|)$ as it scans all $|\mathcal{S}|$ arms. For UCB, the complexity is also $\mathcal{O}(|\mathcal{S}|)$ due to the computation of confidence bounds for all arms. For Thompson Sampling with a simple Beta distribution update, the sampling step is $\mathcal{O}(|\mathcal{S}|)$. Per Time Step Complexity: Thus, the total complexity per time step for MAB learning is $\mathcal{O}(|\mathcal{S}|)$.

With respect to the Low-Rate Resilient Policy (LRRP), the issue of parameter update in sparse traffic is addressed by synthesizing packets during idle periods. This involves generating a synthetic packet, which is a constant-time operation ($\mathcal{O}(1)$). In other words, adding LRRP does not increase the overall time complexity of MAB learning, as it only adds a constant overhead.

Thus, for a single node operating over $T$ time steps, the total computational complexity is: $\mathcal{O}(T.K)$. This includes $T$ arm selection and reward update operations over $K$ arms. In order to compute network-level complexity, for a network with $N$ nodes, where each node operates independently, the total complexity scales linearly with the number of nodes: $\mathcal{O}(N.T.K)$. Since no explicit control message exchange occurs, the algorithm avoids the quadratic or higher complexities typically seen in centralized scheduling methods.

Memory Requirements: Each node stores the state of $|\mathcal{S}|$ arms (e.g., reward values, confidence bounds, or distribution parameters). The memory requirement per node is $\mathcal{O}(|\mathcal{S}|)$. At the network level, the total memory requirement is: $\mathcal{O}(N.K)$.

Comparison to Centralized Methods: Centralized scheduling algorithms often require global coordination, resulting in $\mathcal{O}(N^2)$ or higher complexity due to the need for network-wide conflict resolution and control message processing. In contrast, the proposed decentralized framework scales efficiently, making it suitable for large networks with minimal computational and communication overhead.

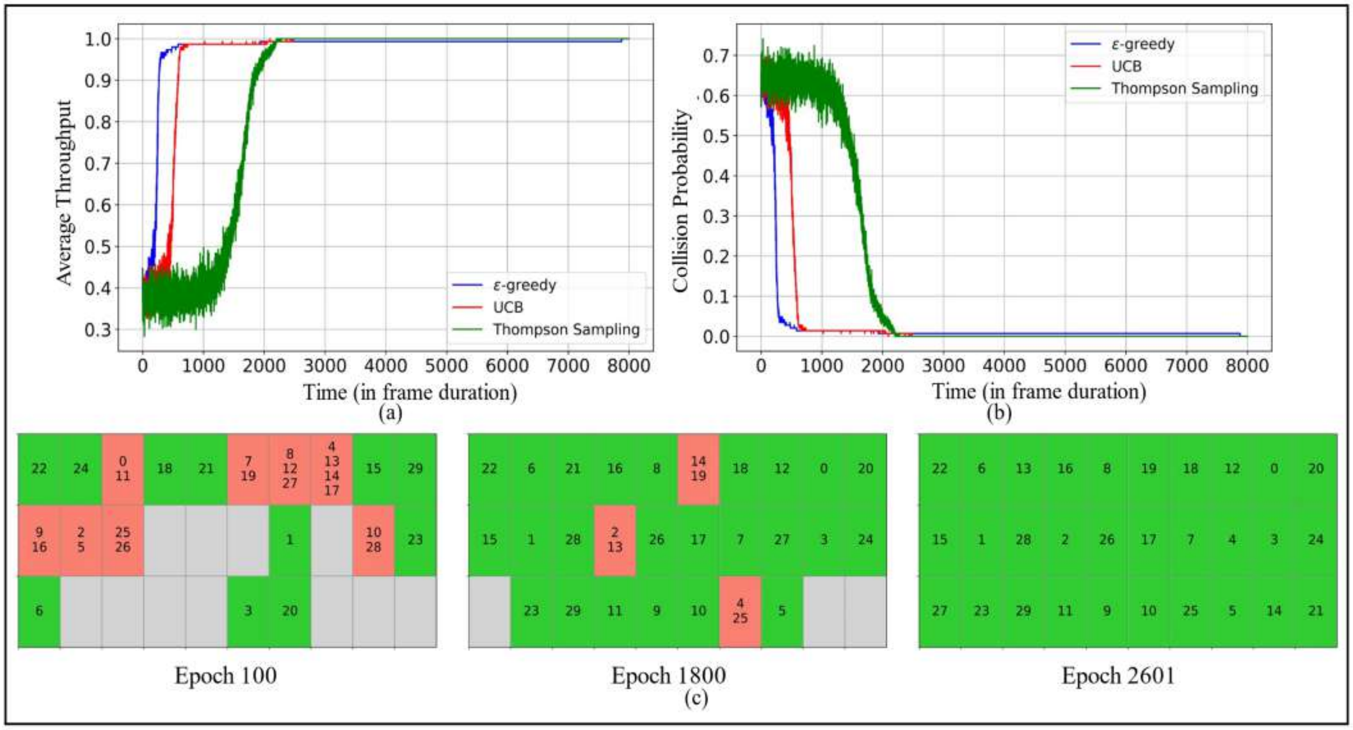
TABLE II: Default Simulation Parameters

| Parameter | Value |
|---|---|
| $\alpha$ | 0.01 |
| $c$ | 0.15 |
| $\epsilon$ | $e^{-t/50}$ |
| $t_r$ | 5000 frame duration |
| $T_L$ | 100 |
| $V_0^{(i)}(s), \forall i, s$ | Uniform random |
| $(\alpha_s^{(i)}(t), \beta_s^{(i)}(t)), \forall i, s$ | (0,0) |

## VII. Experiments and Results

### A. Experimental Setup

The performance evaluation of the proposed TSCH scheduling framework driven by Bernoulli Bandits was conducted through experiments using a MAC network simulator. The simulations were executed on a system equipped with an Intel(R) Core(TM) i7 (10th gen) processor running a 64-bit Windows 10 (v22H2) operating system. The developed time-driven MAC layer simulation software is designed in Python 3.7. The simulation kernel with embedded learning components performs event scheduling in terms of packet generation, transmissions and receptions. We consider a generalized network model with multi-point-to-multi-point, partially-connected topologies. Specifically, two broad categories of IoT networks are explored. First, a multi-point-to-point connectivity with $N$ nodes sending MAC packets to a central base



Epoch 100

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 22 | 24 | 0<br>11 | 18 | 21 | 7<br>19 | 8<br>12<br>27 | 4<br>13<br>14<br>17 | 15 | 29 |
| 9<br>16 | 2<br>5 | 25<br>26 | | | | 1 | | 10<br>28 | 23 |
| 6 | | | | | 3 | 20 | | | |

Epoch 1800

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 22 | 6 | 21 | 16 | 8 | 14<br>19 | 18 | 12 | 0 | 20 |
| 15 | 1 | 28 | 2<br>13 | 26 | 17 | 7 | 27 | 3 | 24 |
| | 23 | 29 | 11 | 9 | 10 | 4<br>25 | 5 | | |

Epoch 2601

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 22 | 6 | 13 | 16 | 8 | 19 | 18 | 12 | 0 | 20 |
| 15 | 1 | 28 | 2 | 26 | 17 | 7 | 4 | 3 | 24 |
| 27 | 23 | 29 | 11 | 9 | 10 | 25 | 5 | 14 | 21 |

Fig. 3: Convergence behavior of different bandit policies with respect to (a) throughput, (b) collision probability (c) TSCH frame status (the number within cell indicates the node transmitting packet in that cell)

station and second, a mesh network with bidirectional traffic flow between the one-hop neighbors. At the MAC layer, it is assumed that the network time is synchronized. Time is slotted and there exists multiple channels that the node can use for transmission, as is the case with general Time Slotted Channel Hopping (TSCH) networks mentioned earlier. The MAC slotframes are of fixed size, which is dimensioned a priori based on the average degree of the network graph.

The application layer traffic generation at the source is stochastic and follows a Poisson distribution with mean data rate of $\lambda$ packets per frame. Note that the traffic pattern follows an Independent Identical Distribution (IID) across the network. Each node maintains an M/G/1/K buffer/queue, where the Poisson distributed queue arrival rate is governed by $\lambda$, and the queue service rate is determined by the TSCH scheduling policy actuated by the proposed learning mechanisms.

The default experimental parameters are presented in Table II. The performance metrics used for evaluation of the proposed approach were: (i) throughput, measured as the fraction of successfully transmitted packets relative to the total generated packets; (ii) collision probability, defined as the proportion of packets that experienced transmission collisions; (iii) energy efficiency, defined as the fraction of time nodes remained active versus in sleep mode and (iv) learning convergence, evaluated in terms of the time required for nodes to attain a stable, collision-free scheduling policy.

### B. Performance Analysis

To understand the performance achieved with the proposed TSCH scheduling framework driven by Bernoulli Bandits, we

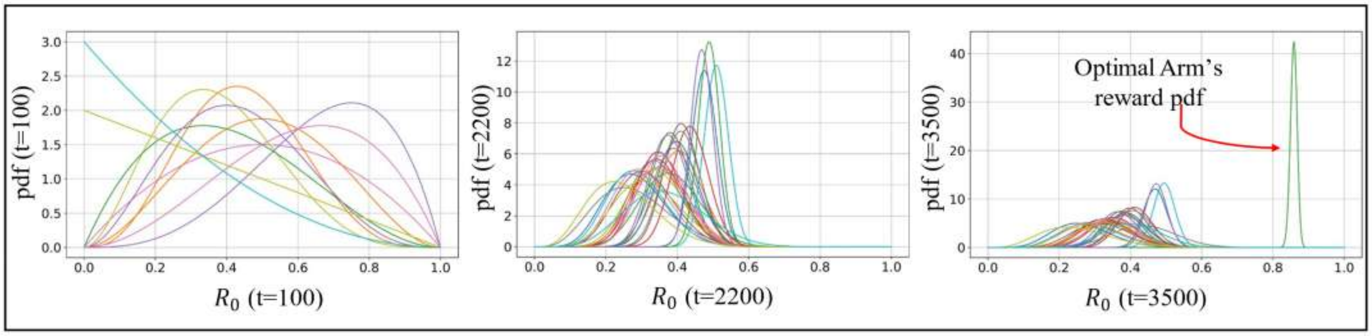


Fig. 4: $\beta$-distribution $pdf$ associated with each arm for three different stages of learning ($t = 100, 2200, 3500$)

first experiment with a multi point-to-point network, with 30 nodes sending data to a base station. The initial experiments are conducted with the baseline action-selection policies detailed in section V. The learning convergence behavior of these policies, viz, $\epsilon$-greedy, UCB and Thompson Sampling, for this network is demonstrated by throughput and collision probability in Fig. 3 (a) and (b). It can be observed that the convergence speed of Thompson Sampling (TS) is comparatively slow compared to that of $\epsilon$-greedy and UCB. This is because when the nodes use TS for learning the arm selection policy, then it updates its parameters of the prior distribution, which is $\beta$-distribution in our case, and samples are drawn from the updated parametric model. At the start of learning, the distribution is close to a uniform distribution and with an increase in the number of samples received (collision information in this case), the distribution gets closer and closer to a Gaussian distribution, with its statistical mode at the arm with maximum expected reward. On the other hand, the policies $\epsilon$-greedy and UCB are inherently greedy in nature and favor the action that gives short term high reward. This process helps them to attain convergence faster, but at the cost of settling in a sub-optimal arm selection, as can be seen from the figure, where $\epsilon$-greedy learns a sub-optimal policy resulting in a non-unitary network throughput.

The TSCH transmission schedule of all the nodes at different learning stages is shown in Fig. 3 (c). The TSCH cell allocation status is presented at three stages: at the beginning, in an intermediate stage, and at the end of learning convergence. Figure 3 (c) shows that initially, the nodes select the TSCH slots randomly, resulting in many collisions (shown in red) due to overlapped transmission in time. However, as learning progresses, the nodes learn to select TSCH cells, independently and in a decentralized manner, to find a TSCH schedule. such that the number of collisions goes down. The figure clearly shows that after learning converges, there are no collisions in the TSCH frame. From the Bernoulli Bandit perspective, the progression of the *probability density function (pdf)* of the reward associated with each arm (TSCH cell) of a single node, while using Thompson sampling, is depicted in Fig. 4. The figure shows the snapshots at three stages of learning, initial, intermediate and post-convergence stage. The figure shows that sampling probability of each arm, based on reward distribution, initially remain in the same ballpark; but with time the *pdf* of the optimal arm gets narrower and becomes associated with a high reward value, with a high sampling probability.

The convergence behavior shown in Figs. 3 and 4 are for the situation when the Poisson data rate is saturated at $\lambda = 1$ ppf. However, that assumption cannot be generalized to realistic network conditions. When the traffic data rate goes down, the problem of sample deficiency arises for bandit parameter update, as mentioned earlier in section V. This is experimentally shown by the throughput progression plot in Fig. 5, for the same 30-nodes multi-point-to-point network. It is observed that all the three baseline bandit policies, that is, $\epsilon$-greedy, UCB and Thompson Sampling, are unable to attain a collision-free TSCH transmission schedule. Note that $\epsilon$-greedy performs slightly better than UCB and Thompson Sampling due to its greedy behavior in arm selection. In fact, the more greedy a policy is, the less it is affected by the lack of samples. In other words, due to its greedy nature of reward maximization, the true nature of the reward distribution is ignored, and as a result the throughput degradation is less affected by the lack of training samples. On the other hand, in the absence of sample unavailability, Thompson Sampling cannot make the approximation of Gaussian distribution from the initial Uniform distribution, which manifests as collisions in the TSCH slotframe. Another key observation from Fig. 5 is that the proposed Low Rate Resilient Policy (LRRP) policy allows the nodes to learn a collision-free transmission schedule. This is accomplished by transmitting packets with payloads of noisy samples, in the scenarios of sample deficiency for parameter updates. It is worth noting that the initial throughput of the network, in the scenario of LRRP update rule, is lower than when the standard bandit policies are used. This is because, owing to synthesized packet generation, the network load increases which substantially increases the collisions, resulting in low throughput.

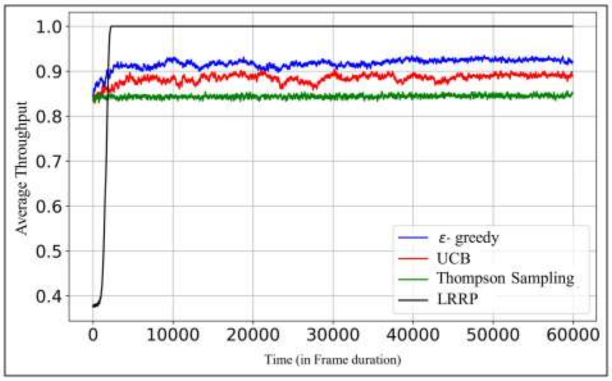


Fig. 5: Convergence behavior of bandit policies for low data rate ($\lambda = 0.45$ ppf)

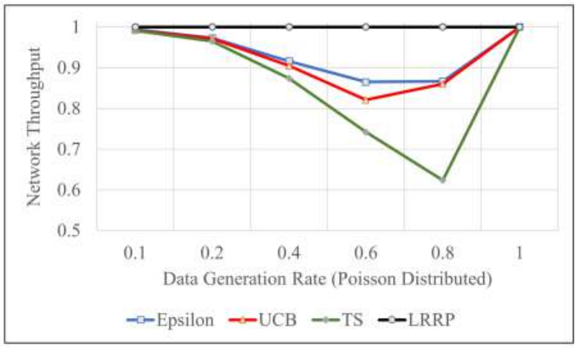


Fig. 6: Network throughput variation with traffic for different action selection policies for a 30-nodes multi-point-to-point network

The variation of network throughput for different network loading conditions is shown in Fig. 6. There is a non-monotonic behavior in the throughput variation with data traffic for the bandit policies of $\epsilon$-greedy, UCB and Thompson Sampling. This is because, with decrease in network load,

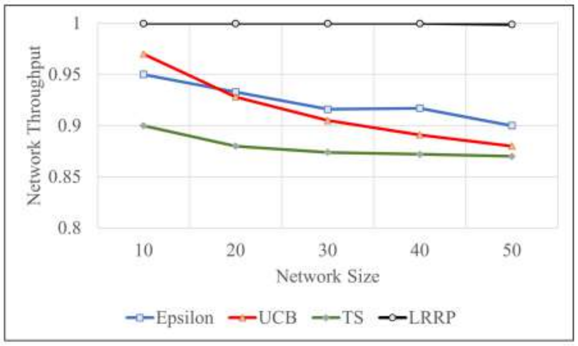


Fig. 7: Throughput variation with network size for different action selection policies for a given loading condition

the learning policy update is affected, resulting in a reduced throughput. Beyond a certain reduction in network load, the collision probability decreases due to low packet generation rate, thus reducing the network throughput degradation. Nevertheless, the unitary throughput is maintained when the nodes use LRRP as the Bernoulli Bandit policy. This is achieved due to the collision-free transmission-schedule accomplished using the robust behavior of LRRP for low traffic conditions.

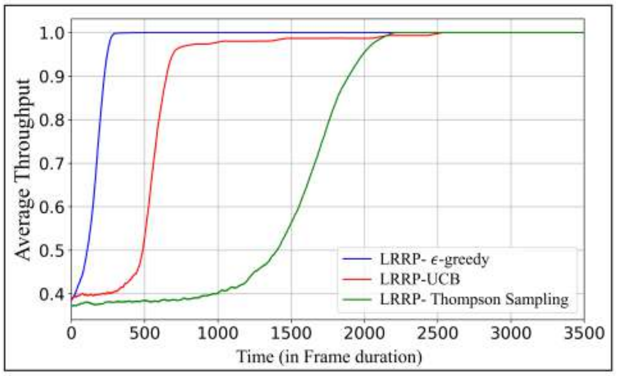


Fig. 8: LRRP learning convergence behavior with different baseline action selection policies

Another challenge faced by the standard learning algorithms is the scalability of network performance with network size. As depicted in Fig. 7, where the network traffic is chosen as $\lambda = 0.4$ ppf, with an increase in network size, the throughput goes down monotonically, due to high collision probability. On the contrary, the proposed LRRP logic makes the TSCH scheduling protocol scalable with network size, maintaining a collision-free transmission throughout. Notably, the fact that throughput reduction is affected most for the least greedy-policy still holds for the scenarios explored in Fig. 5.

TABLE III: Heterogeneous network loading

| Heterogeneous Load ($\lambda$ vector) | Average Load |
|---|---|
| 0.5,0.9,0.7,0.4,0.6,0.8,0.6,0.75 | 0.65625 |
| 0.1,0.2,0.15,0.25,0.3,0.25,0.1,0.15 | 0.1875 |
| 0.75,0.80,0.65,0.75,0.85,0.65,0.7,0.8 | 0.74375 |

The proposed LRRP action selection strategy can be implemented using any of the baseline bandit policies detailed earlier in section V. As shown in Fig. 8, using any of the three policies, viz., $\epsilon$-greedy, UCB and Thompson Sampling, as the baseline, the LRRP logic allows the nodes to find a collision-free TSCH schedule. It is also observed that the greedy-behavior of $\epsilon$-greedy and UCB allows them to achieve faster convergence as compared to Thompson Sampling, when used as baseline policies. However, that speed comes at the cost of settling in a sub-optimal solution, as is observed previously in Fig. 3. On account of this, in this work, we have primarily explored Thompson Sampling for baseline arm selection using LRRP mechanism.

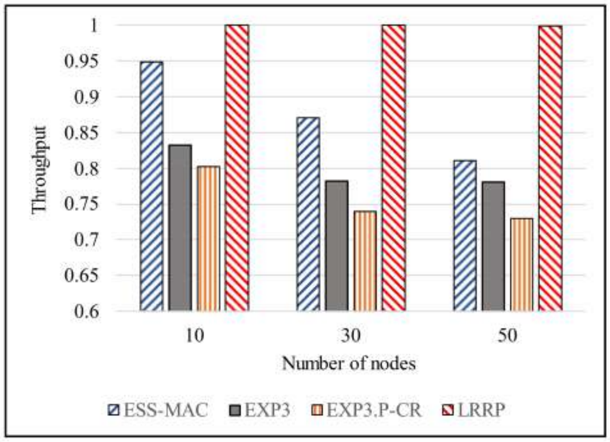


Fig. 9: Performance comparison of LRRP with existing Bandit-driven scheduling approach

Fig. 9 compares the performance of LRRP with an MAB-driven state-of-the-art scheduling strategies used in ESS-MAC [27], EXP3 [41] and EXP3.P with collision resolution (EXP3.P-CR) [42]. Experiments are performed for a multi point-to-point network topology for stochastic (Poisson distributed) traffic with mean data rate of $\lambda = 0.6$ ppf. It is observed that for all the networks, LRRP outperforms the benchmark slot scheduling policies in terms of network throughput. Moreover, in contrary to what is observed with ESS-MAC, EXP3 and EXP3.P-CR, the scheduling policy of LRRP is scalable with network size. With increase in network size, throughput achieved in the ESS-MAC scheduler decreases by 5, 13 and 19% as compared to LRRP for 10, 30 and 50-nodes network respectively. In other words, the collision probability in the access layer for a lower data rate significantly reduces for the proposed LRRP scheme as compared with ESS-MAC's slot scheduling policy. Similar observation holds for EXP3 and EXP3.P-CR as well. This also, in turn, leads to a better energy efficiency for LRRP due to decrease in the number of re-transmissions of access layer packets.

The proposed LRRP-driven TSCH schedule learning is explored in a mesh network as shown in Fig. 10 (a). The collision probability goes down as learning progresses, as indicated in the convergence behavior presented in Fig. 10 (b). Additionally, by experimenting with different heterogeneous traffic conditions listed in Table. III, the energy savings accomplished by informed turning-off of the node transceiver,

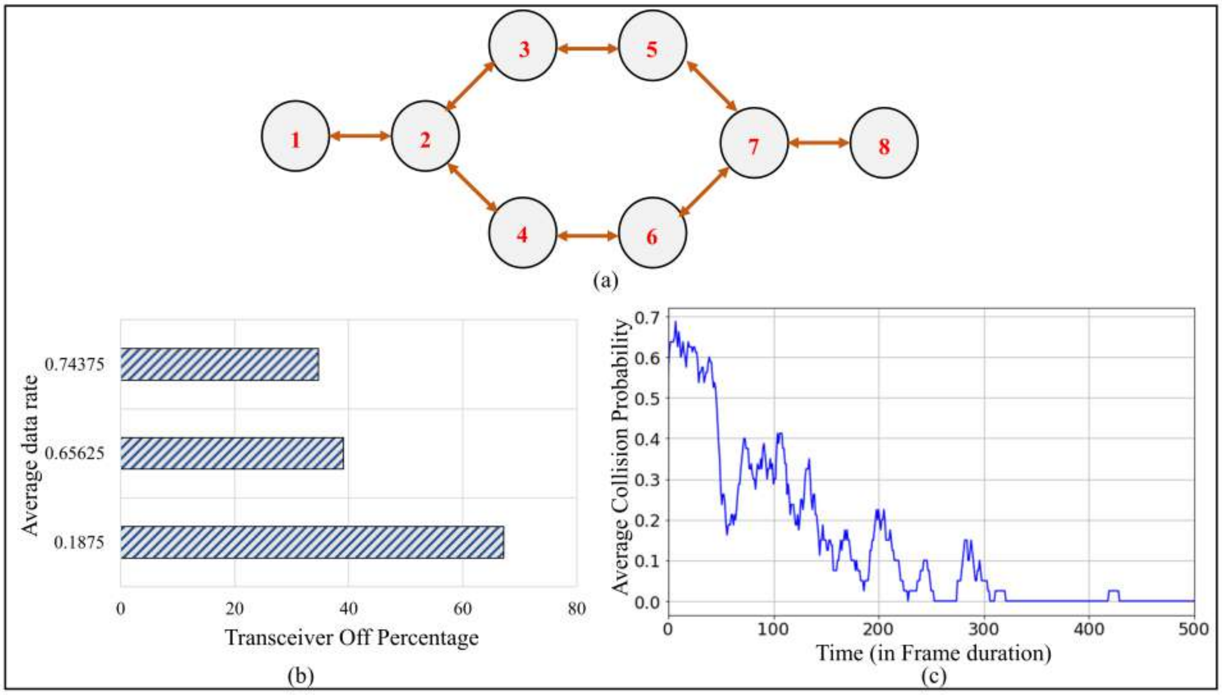


Fig. 10: Performance of bandit-driven TSCH schedule learning in a mesh network

using the MAB-assisted TSCH scheduling, are presented in Fig. 10 (b). The prima-facie take away from the figure is that, for low data-traffic conditions, nodes learn to find an energy-efficient transmit-listen schedule by keeping the transceiver in a TSCH cell turned off when it is not supposed to transmit or receive packets from its neighbors.

## VIII. Conclusion

In this paper, we introduced a decentralized TSCH scheduling framework driven by Bernoulli Multi-Armed Bandit (MAB) learning for IoT networks. Unlike existing approaches that rely on centralized coordination or the exchange of control messages, the proposed method enables each node to learn its own transmission schedule independently, thereby reducing energy and bandwidth overhead. Through extensive experiments, we evaluated the performance of three standard MAB-based action selection policies—$\epsilon$-greedy, UCB, and Thompson Sampling—and demonstrated their limitations in scenarios with low traffic data rates, where sample insufficiency hinders convergence to optimal scheduling. To address these limitations, we propose the Low-Rate Resilient Policy (LRRP), which allows nodes to maintain collision-free schedules even under sparse traffic conditions by generating synthesized packets.

Our experimental results highlight several key findings. First, LRRP achieves superior performance in low-traffic networks by overcoming the sample deficiency problem, where traditional bandit policies struggle. Second, the proposed method ensures scalability, as demonstrated by its ability to maintain collision-free TSCH scheduling across varying network sizes. Third, this approach provides energy savings by making nodes aware of when they should power down their transceivers due to periods of inactivity. Finally, the results demonstrate that while greedy bandit action selection policies offer faster convergence, they often settle for suboptimal solutions, whereas Thompson Sampling, coupled with LRRP, balances exploration and exploitation for long-term performance gains.

Overall, the decentralized nature and low overhead of the proposed LRRP-driven TSCH scheduling framework make it a robust and scalable solution for resource-constrained IoT networks, offering collision-free access, enhanced energy efficiency, and scalability with network size and heterogeneity. Nevertheless, there are several open research problems in this context which can be explored as a future extension of this work. To begin with, this work requires the nodes to be able to reliably observe the collisions encountered by its transmitted packets. Extending the proposed framework to handle higher levels of interference and packet loss in challenging network environments is a future research direction. Another extension of this work is to develop decentralized mechanisms for assigning access priority for transmitting emergency information while maintaining other performance measures.

## References

[1] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, “Wirelesshart: Applying wireless technology in real-time industrial process control,” in *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2008, pp. 377–386.

[2] F. P. Rezha and S. Y. Shin, “Performance analysis of isa100. 11a under interference from an ieee 802.11 b wireless network,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 919–927, 2014.

[3] N. Choudhury, R. Matam, M. Mukherjee, and J. Lloret, “A performance-to-cost analysis of ieee 802.15. 4 mac with 802.15. 4e mac modes,” *IEEE Access*, vol. 8, pp. 41 936–41 950, 2020.

[4] W. Jerbi, O. Cheikhrouhou, A. Guermazi, and H. Trabelsi, “Msu-tsch: A mobile scheduling updated algorithm for tsch in the internet of things,” *IEEE Transactions on Industrial Informatics*, vol. 19, no. 7, pp. 7978–7985, 2022.

[5] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, “Topology management and tsch scheduling for low-latency convergecast in in-vehicle wsns,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1082–1093, 2018.

[6] M. O. Ojo, S. Giordano, D. Adami, and M. Pagano, “Throughput maximizing and fair scheduling algorithms in industrial internet of things networks,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3400–3410, 2018.

[7] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, “A centralized scheduling algorithm for ieee 802.15. 4e tsch based industrial low power wireless networks,” in *2016 IEEE Wireless Communications and Networking Conference*. IEEE, 2016, pp. 1–6.

[8] F. Veisi, J. Montavont, and F. Theoleyre, “Enabling centralized scheduling using software defined networking in industrial wireless sensor networks,” *IEEE Internet of Things Journal*, 2023.

[9] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, “Whitelisting without collisions for centralized scheduling in wireless industrial networks,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5713–5721, 2019.

[10] S. Rekik, N. Baccour, M. Jmaiel, and K. Drira, “A performance analysis of orchestra scheduling for time-slotted channel hopping networks,” *Internet Technology Letters*, vol. 1, no. 3, p. e4, 2018.

[11] R.-H. Hwang, C.-C. Wang, and W.-B. Wang, “A distributed scheduling algorithm for ieee 802.15. 4e wireless sensor networks,” *Computer Standards & Interfaces*, vol. 52, pp. 63–70, 2017.

[12] H. Hajizadeh, M. Nabi, and K. Goossens, “Decentralized configuration of tsch-based iot networks for distinctive qos: A deep reinforcement learning approach,” *IEEE Internet of Things Journal*, vol. 10, no. 19, pp. 16 869–16 880, 2023.

[13] F. F. Jurado-Lasso, M. Barzegaran, J. Jurado, and X. Fafoutis, “Elise: A reinforcement learning framework to optimize the slotframe size of the tsch protocol in iot networks,” *IEEE Systems Journal*, 2024.

[14] H. Nguyen-Duy, T. Ngo-Quynh, F. Kojima, T. Pham-Van, T. Nguyen-Duc, and S. Luongoudon, “Rl-tsch: A reinforcement learning algorithm for radio scheduling in tsch 802.15. 4e,” in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2019, pp. 227–231.

[15] H. Dakdouk, E. Tarazona, R. Alami, R. Féraud, G. Z. Papadopoulos, and P. Maillé, “Reinforcement learning techniques for optimized channel hopping in ieee 802.15. 4-tsch networks,” in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2018, pp. 99–107.

[16] 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. 139 727–139 736, 2020.

[17] 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. 21 712–21 728, 2022.

[18] F. Lyu, H. Zhu, H. Zhou, L. Qian, W. Xu, M. Li, and X. Shen, “Momac: Mobility-aware and collision-avoidance mac for safety applications in vanets,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 590–10 602, 2018.

[19] X. Jiang and D. H. Du, “Ptmac: A prediction-based tdma mac protocol for reducing packet collisions in vanet,” *IEEE transactions on vehicular technology*, vol. 65, no. 11, pp. 9209–9223, 2016.

[20] 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, p. e3730, 2019.

[21] J. Liu, B. Zhao, Q. Xin, and H. Liu, “Dynamic channel allocation for satellite internet of things via deep reinforcement learning,” in *2020 International Conference on Information Networking (ICOIN)*. IEEE, 2020, pp. 465–470.

[22] L. Bommisetty and T. Venkatesh, “Resource allocation in time slotted channel hopping (tsch) networks based on phasic policy gradient reinforcement learning,” *Internet of Things*, vol. 19, p. 100522, 2022.

[23] F. F. Jurado-Lasso, C. Orfanidis, J. Jurado, and X. Fafoutis, “Hrl-tsch: A hierarchical reinforcement learning-based tsch scheduler for iiot,” *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[24] J. Lu, L. Li, D. Shen, G. Chen, B. Jia, E. Blasch, and K. Pham, “Dynamic multi-arm bandit game based multi-agents spectrum sharing strategy design,” in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. IEEE, 2017, pp. 1–6.

[25] W. Wang, A. Leshem, D. Niyato, and Z. Han, “Decentralized learning for channel allocation in iot networks over unlicensed bandwidth as a contextual multi-player multi-armed bandit game,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 5, pp. 3162–3178, 2021.

[26] S. J. Darak and M. K. Hanawal, “Multi-player multi-armed bandits for stable allocation in heterogeneous ad-hoc networks,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2350–2363, 2019.

[27] H. Dutta, A. K. Bhuyan, and S. Biswas, “Reinforcement learning based flow and energy management in resource-constrained wireless networks,” *Computer Communications*, vol. 202, pp. 73–86, 2023.

[28] R. S. Sutton, “Reinforcement learning: An introduction,” 2018.

[29] I. Halperin, *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions: by Warren B. Powell (ed.), Wiley (2022). Hardback. ISBN 9781119815051.* Taylor & Francis, 2022, vol. 22, no. 12.

[30] S. Agrawal and N. Goyal, “Analysis of thompson sampling for the multi-armed bandit problem,” in *Conference on learning theory*. JMLR Workshop and Conference Proceedings, 2012, pp. 39–1.

[31] S. Pilarski, S. Pilarski, and D. Varró, “Delayed reward bernoulli bandits: Optimal policy and predictive meta-algorithm pardi,” *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 152–163, 2021.

[32] D. A. Berry and B. Fristedt, “Bandit problems: sequential allocation of experiments (monographs on statistics and applied probability),” *London: Chapman and Hall*, vol. 5, no. 71-87, pp. 7–7, 1985.

[33] S. Pilarski, S. Pilarski, and D. Varró, “Optimal policy for bernoulli bandits: Computation and algorithm gauge,” *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 1, pp. 2–17, 2021.

[34] C. Riou and J. Honda, “Bandit algorithms based on thompson sampling for bounded reward distributions,” in *Algorithmic Learning Theory*. PMLR, 2020, pp. 777–826.

[35] I. Osband, Z. Wen, S. M. Asghari, V. Dwaracherla, M. Ibrahimi, X. Lu, and B. Van Roy, “Approximate thompson sampling via epistemic neural networks,” in *Uncertainty in Artificial Intelligence*. PMLR, 2023, pp. 1586–1595.

[36] L. Wei and V. Srivastava, “Nonstationary stochastic bandits: Ucb policies and minimax regret,” *IEEE Open Journal of Control Systems*, 2024.

[37] P. Auer and R. Ortner, “Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem,” *Periodica Mathematica Hungarica*, vol. 61, no. 1-2, pp. 55–65, 2010.

[38] C. Kang, “So many decisions, so little time: A brief introduction to stochastic multi-armed bandits,” 2021.

[39] V. Kuleshov and D. Precup, “Algorithms for multi-armed bandit problems,” *arXiv preprint arXiv:1402.6028*, 2014.

[40] S. Agrawal and N. Goyal, “Near-optimal regret bounds for thompson sampling,” *Journal of the ACM (JACM)*, vol. 64, no. 5, pp. 1–24, 2017.

[41] D. Lee, Y. Zhao, J.-B. Seo, and J. Lee, “Multi-agent reinforcement learning for a random access game,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 9119–9124, 2022.

[42] M. Bande and V. V. Veeravalli, “Adversarial multi-user bandits for uncoordinated spectrum access,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 4514–4518.