WaWoR: Wasted Work Reduction during Snapshotting in EH-WSNs

Priyanka Singla, Smruti R. Sarangi, *Member IEEE*

***Abstract*—Energy harvesting wireless sensor networks (EH-WSNs) are useful for ambient monitoring, especially in hazardous and hard-to-reach environments, where sensor nodes sense and transmit data to a remote sink via multi-hop communication, enabling informed decision making. Accurate decisions require a *global snapshot* that comprises environmental parameters *simultaneously* sensed from *all* nodes. To realize an efficient snapshot collection scheme, all nodes should send an *equal* number of messages where each message contributes to the snapshot; excessive messages from some nodes are ineffective and in-turn increase the network traffic and contribute to the *wasteful* work done by nodes. Reducing such ineffectual messages is challenging due to the variable ambient energy supply at nodes, network congestion, limited information about the overall state of the system and varying node distances from the sink. To achieve this objective, we introduce *WaWoR*, a novel *distributed* system where nodes *dynamically* decide when to sense and transmit snapshot messages based on their physical locations, energy availability and perceived network congestion. *WaWoR* outperforms two theoretical hypothetical baselines and two state-of-the-art-systems. It captures $[90-175]\%$ of the snapshots compared to the theoretical baselines and $[1.02-4.26]\times$ more snapshots than state-of-the-art systems. Additionally, it reduces wasted messages, which further reduces the total energy consumption by $[1.6-9.9]\times$.**

***Index Terms*—fairness, energy harvesting wireless sensor networks, adaptive sensing, snapshots, effective throughput, energy.**

I. Introduction

Due to their nearly infinite lifespan and exceptionally low power consumption, energy harvesting devices (EHDs) are replacing traditional battery-powered Internet of Things (IoT) devices [1]. **E**nergy **h**arvesting **w**ireless **s**ensor **n**etworks (EH-WSNs) are predominantly used in applications that require accessing difficult-to-reach remote locations, such as monitoring submarine structures [2], forest fires, arctic oil pipelines, military and nuclear facilities [3], [4]. Nodes in these networks are typically arranged in a grid-like topology [5]–[7]. This establishes redundant paths that prevent network partitioning during periods of reduced ambient energy availability.

Data Collection at the Sink Node: Sensor nodes periodically sense ambient parameters such as the temperature, pressure and gas concentrations, and relay the readings through a multi-hop network to a distant sink node [8]. The sink node collects this data, analyzes it and then computes various spatio-temporal statistics to make informed decisions. For instance, consider an Arctic oil pipeline monitoring system, where the sensor data consists of oil pressure readings. When these readings are collected at the sink, they help detect and quantify leaks, and prompt necessary actions. To make accurate decisions, the sink node requires high-quality data, i.e., it should receive sensor values from ***all the*** nodes that are sensed at ***approximately*** the ***same time***. Specifically, a ***global snapshot*** of the entire network state concerning the monitored parameter (e.g., pressure) is required [9]. In our running example, if data about the oil pressure is not received from all the nodes (collected at roughly the same time), the sink cannot quantify the intensity and nature of the leakage. Furthermore, if sensed data is not timely and is stale, the sink may make incorrect decisions. For many such reasons, an accurate global snapshot is often required.

Need for a New Data Collection Approach for EH-WSNs: Existing data and snapshot collection techniques for WSNs [10], [11] cannot be readily applied to EH-WSNs due to limitations imposed by the available ambient energy. Specifically, ensuring successful communication between two nodes is not always possible [12]. Consequently, considerable efforts have been directed towards devising strategies for efficient snapshot collection in EH-WSNs. These strategies primarily focus on maximizing the number of data messages collected at the sink per unit time [13], [14]. In these works, all nodes periodically sense, and then transmit data messages towards the sink. However, due to different node-to-sink distances, nodes closer to the sink often end up sending more messages, which is disadvantageous towards nodes farther away from the sink.

Significance of Snapshots: Although the sink collects a lot of messages from nearby nodes, the overall quality of data can be low if every message received from a nearby node is not paired with a message received from a node placed farther away whose reading was collected at roughly the same point of time. In fact, the surplus messages waste network resources and devices' energy, which is especially limited in the case of energy harvesting devices [1].

There is a body of work that proposes efficient snapshot collection in EH-WSNs [8], [9], [15], [16]. However, these approaches rely on several problematic assumptions (highlighted in Section II), rendering them practically unfeasible. To remedy such issues and address the problem of achieving efficient network snapshot collection, we propose *WaWoR*– a practical distributed solution designed to maximize the number of snapshots collected at the sink node. Before discussing *WaWoR*, let us first understand the challenges of snapshot

Priyanka Singla (corresponding author), PhD, from the School of Information Technology, IIT Delhi, New Delhi, India (email: priyanka@sit.iitd.ac.in, priyanka.singla6@gmail.com, postal address: Lab No. 306, SIT Department, IIT Delhi, Hauz Khas, 110016, New Delhi, India)

Smruti R. Sarangi is with the Department of Computer Science and Engineering, IIT Delhi, New Delhi, India (email: srsarangi@cse.iitd.ac.in)

collection in detail.

A. Why Snapshot Collection is Challenging?

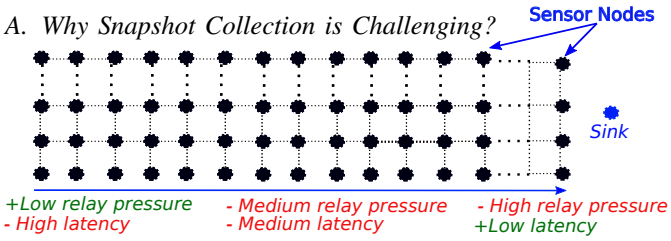

Fig. 1. Pictorial representation of an EH-WSN

Let's define the term "sensing period". Generally, if a node has a sensing period τ, it means it will send and transmit data every τ units of time. However, in practice, a node's buffer might get full and due to inadequate space, sensing and transmission may not be successful. Therefore, we interpret the sensing period as a node's aggressiveness or desire to send a message. A sensing period τ means that a node will try to sense and send data every τ milliseconds (the assumed unit of time).

Same Sensing Periods for All Nodes: Let us now begin by considering a basic snapshot collection scheme where all nodes have the same *sensing period*. Figure 1 shows a grid (mesh-based) network. We consider the sensing period of each node to be 150 ms. Although each node senses every 150 milliseconds, the number of messages reaching the sink varies between 60 and 380 over a 60-second duration (details in Section VII-A1). This variation is shown on the primary y-axis in Figure 2(a). For collecting a global snapshot, the sink needs to receive messages from all the nodes that were collected at roughly the same time; therefore, the minimum number of messages from any node determines the number of global snapshots that can be collected. Excess messages indicate wasted effort and contribute to unnecessary message transmission and network congestion. Additionally, oversensing leads to needless consumption of device energy – a critical resource in energy harvesting systems. Thus, we can conclude that this scheme does not capture snapshots efficiently.

Wasted Work: The disparity in the number of messages reaching the sink in the aforementioned scheme can be explained as follows. As indicated in Figure 1, nodes closer to the sink have low latency but along with their own messages, they need to *relay* messages from other upstream nodes. Conversely, nodes farther from the sink have high latency due to their larger distance from the sink and network congestion. Consequently, nodes closest to the sink benefit from low latency, while those farthest benefit from lower relay pressure. However, nodes in the middle suffer from both high latency and high relay pressure. These nodes in the middle determine the *effective rate of snapshot collection* (e.g., 60 snapshots per minute, in our example). The *oversensed* messages from the farthest and closest nodes represent *wasted work*.

Reducing Wasted Work: One way to reduce oversensing is to increase the sensing periods of nodes and make them much larger than $150ms$. However, this decrease in oversensing comes at the expense of a reduced number of snapshots. A more effective solution is to have ***different*** sensing periods for nodes depending upon their position relative to the sink. Please note, sensing periods cannot differ substantially, else they will fail to serve the purpose of collecting a global snapshot, i.e., to provide a global picture of the network. In practice, the ambient state does not change very rapidly and remains stable at the millisecond granularity [17]–[19]. We leverage such phenomena, and consider snapshots that are *approximately instantaneous* i.e., nodes sense at slightly different times, but within a close range ($< 600\ ms$ in our experiments). Using, different sensing periods allowed us to increase the total number of snapshots to 150 per minute (see Figure 2(b)), significantly reducing the wasted effort.

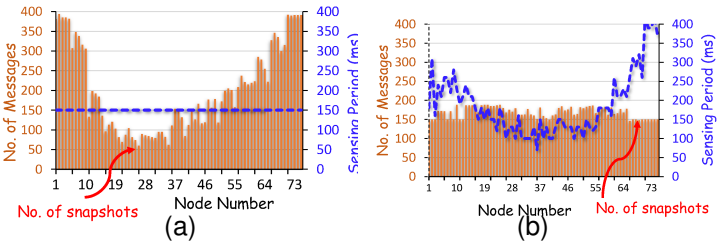

Fig. 2. Number of snapshots captured, when nodes have the **(a)** *same* sensing periods and **(b)** *different* sensing periods

Variable Sensing Periods: The last and most critical challenge is to handle the variability in ambient energy at nodes and the network dynamics (varying network congestion and nodes' states). Given the dynamic nature of the system, fixed sensing periods (even if it is different at different nodes) are insufficient. Therefore, *adaptive* sensing periods are required, influenced by factors such as the nodes' positions relative to the sink, energy availability and the instantaneous network state. To achieve this, we introduce *WaWoR–* an algorithm that enables nodes to autonomously and adaptively adjust their sensing periods. The goal is to optimize the sensing periods to *maximize the number of snapshots captured while minimizing the wasted work.*

B. Contributions

Our contributions in this work can be summarized as follows:
(1) We propose *WaWoR–* a distributed algorithm for adjusting nodes' sensing periods in accordance with their position, available ambient energy, and network dynamics (network congestion and nodes' states); these adjustments rely on periodic feedback received very infrequently from the sink.
(2) Unlike several existing works on EHD systems [6], [9], which propose a single-node solution and then analytically extend it to a multi-node system, abstracting out various details, we conducted detailed simulations of *WaWoR* for a comprehensive end-to-end system.
(3) We compared *WaWoR* against two theoretical baselines and two state-of-the-art systems. To the best of our knowledge, there does not exist a theoretical formulation that leads to an optimization problem that can be solved using standard solvers. The solution space is very complex given the size and scale of the system. We thus designed two hypothetical baselines: a static scheme where nodes have constant sensing periods throughout the simulation and a dynamic scheme with continuously changing sensing periods. Both of these schemes consider nodes with omniscient capabilities – they are aware of the ambient energy profiles at all nodes. Due to this, nodes are able to perceive the instantaneous state of the entire network, and choose optimal sensing periods.

The first hypothetical scheme, *OptStat*, computes the sensing periods by exhaustive experimentation statically (before execution commences). Specifically, it evaluates various sensing periods and selects the one yielding the most snapshots at the sink. The second scheme, *OptDyn* is an AI-based approach, wherein nodes *dynamically* change their sensing periods in response to the instantaneous energy available.

Please note that the energy profile has a very important bearing on the final results. So, we have considered real-world traces corresponding to the most commonly used ambient sources: solar, vibrational, and RF sources [20]–[22]. The considered profiles have different power intensities and different granularities of variation (from milliseconds to seconds).

Our experiments show that for all three sources, *WaWoR* could capture 1.8-75% more snapshots as compared to *OptStat*, and approximately $[-10, 1.8]\%$ more snapshots than *OptDyn*. In our experimental comparison with two state-of-the-art methods, *WaWoR* demonstrated the ability to capture $[1.03, 4.26]\times$, $[1.02, 3.64]\times$ and $[1.08, 3.41]\times$ more snapshots for solar, vibrational and RF sources, respectively. Beyond the snapshot count, we also assessed the number of wasted messages and the corresponding wasted energy for the three methodologies. The results show that the wastage in *WaWoR* was $[1.8 - 9.9]\times$ lower compared to the two state-of-the-art systems.

The paper is organized as follows: we discuss the related work in Section II and provide the necessary background in Section III. Section IV describes our nodes, the network topology, and the end-to-end application. Then we provide a mathematical formulation of the problem in Section V, followed by discussing our proposed algorithm *WaWoR* in Section VI. Subsequently, *WaWoR* is implemented and evaluated in Section VII, and the paper is finally concluded in Section VIII.

II. Related Work

Researchers have proposed various schemes to enhance data collection efficiency in EH-WSNs, typically focusing on either ❶ energy efficiency, ❷ high-throughput, or ❸ quality data delivery (fairness/latency) at the sink (explained and summarized in Figure 3). We compare these approaches to *WaWoR*, which offers all three benefits simultaneously. Additionally, most of the existing proposals are not always practically deployable as they rely either on ambient energy prediction, or lack low-level communication details (e.g., the MAC protocol employed), or use topologies without redundant paths risking network disconnection (summarized in Table I).

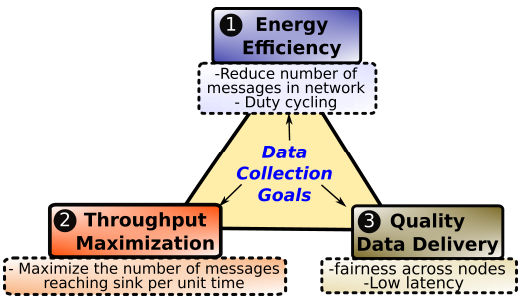

Fig. 3. Goals of data collection algorithms

❶ ***Energy Efficiency***: One approach to save energy is by reducing the number of messages each node sends. *Clustering-based routing* schemes, where sensor nodes form local sub-networks, are quite popular in this space. In these schemes, instead of each node sending messages individually to the sink, a cluster head – usually the node with the most energy – aggregates and transmits data to the sink via other cluster heads. For more details on clustering-based routing, please refer to the comprehensive survey by Sah et al. [23]. Although clustering provides some energy efficiency, the challenge of determining when cluster heads should transmit remains to be a serious problem. These points become congested and act as single points of failure. *WaWoR* addresses this problem and provides many redundant paths. It can in principle be combined with cluster-based schemes; however, during the development process, we did not feel a strong necessity for doing so.

Another energy-saving mechanism is running nodes in a duty-cycled mode [24], which can also be combined with *WaWoR* for greater efficiency. Note that the problem with this approach is that when a message is sent, the receiver node may not be awake or be able to receive the message.

Next, let us discuss other goals, where approaches focus on designing optimal sensing strategies, similar to *WaWoR*.

TABLE I
Summary of Related Work

Reference	Dual† Objective	Relies on Energy Prediction?	MAC Protocol	Topology with Redundant Paths?
Zhang et al. [13]	No	Yes**	B-MAC*	No
Jeong et al. [14]	No	Yes	Unspecified	No
Blondia et al. [6]	No	Yes**	Receiver-Initiated	Yes
Zhu et al. [9]	No	Yes	Unspecified	No
Zhang et al. [8]	No	Yes**	Unspecified	No
Quan at al. [16]	No	No	Unspecified	No
Liu et al. [15]	Yes	Yes**	Unspecified	No
WaWoR	Yes	No	Receiver-Initiated	Yes

†: Maximize throughput and quality; **: Uses a predefined harvesting function or energy harvesting rate; B-MAC*: Berkeley MAC [25], a type of sender-initiated protocol;

❷ ***Throughput Maximization***: This section discusses approaches aimed at increasing the message throughput, measured at the sink [13], [14]. Zhang et al. [13] present an algorithm to optimize node sensing periods to maximize the number of messages reaching the sink. However, unlike *WaWoR*, this approach does not consider fairness; it uses tree-based topologies, which do not provide redundant paths. It also employs the B-MAC protocol (Berkeley MAC) [25], where the sender initiates communication, which is less efficient than receiver-initiated communication. The latter is typically used in EH-WSNs for performance reasons [12].

Jeong et al. [14] propose a scheme, where nodes periodically forecast their energy levels and use two thresholds to decide whether to transmit, sense or receive data. If the predicted energy exceeds the capacitor's limit, the node transmits; if it falls below a threshold, the node turns off its radio to conserve energy for sensing. For energy levels in between, the node can sense or receive data. The approach's dependence on energy prediction limits its practicality as it is not always easy to predict ambient energy. Moreover, prediction also requires energy. Finally, the authors do not specify the MAC protocol employed, a key factor in assessing performance. Unlike this method, *WaWoR* does not rely on energy prediction and uses the well-established *receiver-initiated MAC protocol*.

Blondia et al. [6] use the receiver-initiated MAC protocol like *WaWoR*, but their approach has a few impractical assumptions. For instance, they assume a message can be received if the node's buffer has empty space, ignoring the node's energy level. However, without energy, in an EH-WSN, the node cannot function – this makes the approach infeasible. Additionally, they overlook the $CSMA$ (carrier-sense multiple access) mode before transmitting a beacon (a status message, refer to Section IV-B). This choice disregards the possibility of potential collisions, which can lead to errors.

❷ ***Quality Data Delivery***: From the data collected, the sink should be able to infer the entire network state. It thus clearly requires data from all the nodes. However, depending on the application, optimizations are possible where all the nodes need not send data. For example, considering the cluster-based routing scheme (discussed above) for a temperature monitoring application, an average value can be sent by the cluster head if neighboring nodes report the same temperature. However, the core problem still exists – data from *all the* cluster heads should reach the sink. Thus, determining an optimal transmission schedule is nevertheless of prime importance. References [8], [9], [16] propose algorithms to generate schedules but they rely on ambient energy prediction, which is not always feasible, and they do not specify the MAC protocol used[1].

Zhu et al. [9] present a two-phase algorithm for creating data collection schedules. In the first phase, nodes broadcast their energy status to neighbors. The issue is that uncertain ambient energy can cause nodes to miss these broadcasts, disrupting coordination. Hence, successful broadcasts cannot be assumed, which is the centerpiece of their approach.

Quan et al. [16] use a cluster-based routing protocol for data aggregation – adaptively building an aggregation tree based on nodes' transmission schedules and energy harvesting capabilities. They assume that all nodes know each other's schedules and energy levels, which is impractical in EH-WSNs. Their method divides the network into layers, allowing nodes in a layer to transmit only after all the nodes in the previous layer have completed. This is challenging due to the communication required within each layer. Furthermore, the authors do not specify the MAC protocol employed. Second, in any EH-WSN, excessive communication is not possible because of the paucity of energy.

Besides single-objective approaches, [15] proposes a solution addressing both throughput and fairness. It assumes nodes know the ambient energy profiles similar to our hypothetical omniscient baseline, which is seldom the case. This approach uses a tree topology, which unlike the grid topology [5], [6], [27], [28] does not provide redundant paths. Furthermore, authors do not address how fairness is assured when some nodes lack ambient energy, which affects communication.

Table I summarizes the related work highlighting the key features of each approach. For capturing snapshots in energy harvesting systems, a topology with redundant paths and an approach independent of ambient energy prediction are *crucial* – these are generic and practical requirements. The table summarizes approaches targeting both throughput maximization and good quality (low latency or high fairness or any other custom definition as defined in the corresponding reference). *WaWoR* complements all the known approaches that emphasize energy efficiency and excels in other aspects: feasibility, redundant paths and maximizing the rate of snapshot collection.

[1]Note that while low end-to-end latency is also crucial for quality, it is a separate research area with extensive literature (e.g., Age of Information, AoI [26]). Such approaches are orthogonal to our work. Therefore, they are not discussed here.

III. Background

A. Energy Harvesting Devices

EHDs are low-powered, battery-free IoT devices that rely on ambient energy. Due to the intermittent and unpredictable nature of ambient energy, EHDs are equipped with a small energy storage unit (typically a capacitor) to smooth out power variations. These devices operate in two modes: (i) *separate charge-execution* mode, where the capacitor is fully charged and then it drives the device till it is fully discharged, and (ii) *parallel charge-execution* mode, where the available ambient energy simultaneously charges the capacitor and powers the device. If no ambient energy is available, the capacitor drives the device [1]. However, in both the modes, if there is an extended period without ambient energy, leading to capacitor discharge, the device becomes inoperable and loses its state. To address this, checkpoint-restore mechanisms are employed for correct and efficient functioning [1], [20], [21]. In this paper, we opt for the parallel charge-execution mode as it can take advantage of the ambient energy's variable nature. If abundant ambient energy becomes available during execution, the device's capacitor can be charged.

B. Medium Access Control (MAC) Protocols

MAC protocols define how the sensor nodes use their radios, share the network, avoid collisions, etc. As network communication is an energy-intensive activity, EHD MAC protocols should be efficient. The existing MAC protocols for wireless sensor networks can be categorized as synchronous or asynchronous depending on whether the sensor nodes' active (ON) timings are synchronized or not. Synchronous MAC protocols are inefficient for EH-WSNs due to the uncertainty of the ambient energy; hence, such networks commonly use asynchronous MAC protocols. Depending on who initiates the communication, the two basic communication paradigms are sender-initiated and receiver-initiated. EH-WSNs typically use receiver-initiated MAC protocols to save energy [6], [12].

IV. Energy Harvesting-Based Sensor Network

A. Network Topology

The nodes in a wireless sensor network (WSN) can be arranged as per a variety of topologies: star, tree, and grid (mesh). However, energy harvesting wireless sensor networks (EH-WSNs) predominantly adopt the grid topology [5], [6], [27], [28]. Star and tree topologies are less commonly employed in EH-WSNs since they do not provide redundant paths between nodes. Redundancy is necessary because EHDs rely on unpredictable ambient energy; nodes can randomly turn off, disconnecting the network. So, applications in EH-WSNs

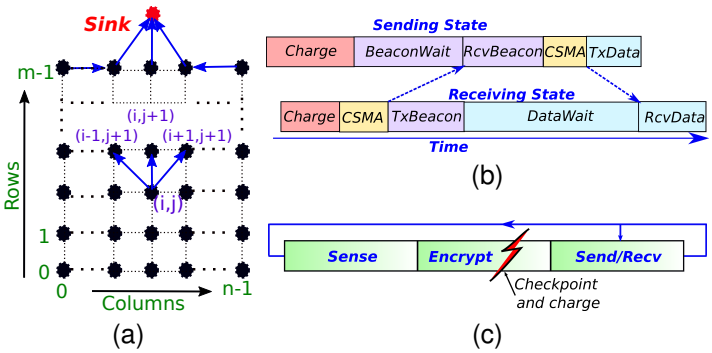

Fig. 4. **(a)** The wireless sensor network **(b)** The receiver-initiated MAC protocol **(e)** End-to-end application

mostly use the *grid* topology since it provides diverse paths. Grid topologies also cover the target area much better [29].

We considered an energy harvesting wireless sensor network with a grid topology made up of m rows and n columns (see Figure 4(a)). Similar to [6], we postulate that sensor nodes situated at the intersections of rows and columns have a limited communication range; hence, messages are sent to the sink node (continuously powered) by performing hop-by-hop communication via neighboring nodes. Particularly, a node with coordinates (i, j) can send data to its immediate neighbors (towards the sink node), that is, to nodes with coordinates $(i-1, j+1)$, $(i, j+1)$, or $(i+1, j+1)$. Similar to [6], we also assume that the nodes within the same row do not communicate with one another. However, we consider that nodes at both ends of the last row, i.e., $(m-1)^{th}$ row, forward data to their neighboring nodes (in the same row) that are closer to the sink. This is due to the nodes' restricted communication range, which prohibits them from sending messages directly to the sink.

B. Receiver-Initiated MAC Protocol

Similar to recent works on EH-WSNs [6], [12], we employ a *receiver-initiated* MAC protocol for communication. Instead of a time-slot-based protocol, we adopt a byte-based protocol, as the former is inefficient in EH-WSNs due to nodes' variable energy. Below, we provide a brief overview; for a more detailed discussion, please refer to the survey by Sherazi et al. [12].

In this protocol, a node can be in the ❶*Receiving* state or ❷*Sending* state, depending on its current task (Figure 4(b)). Each state comprises several modes: *CSMA, Charge, BeaconWait, TxBeacon, RcvBeacon, DataWait, TxData, and RcvData*.

A node first monitors the network in *CSMA* (Carrier Sense Multiple Access) mode for a time period before transmitting. If the network is clear throughout the period, the node sends the message; if not, it backs off and retries later. Additionally, before any operation (sending/receiving), the node checks its energy levels. If the energy is insufficient for completing the operation, the node switches to *charge* mode.

❶*Receiving State*: Because this protocol is receiver-initiated, the receiving node initiates communication by broadcasting a *beacon* to its neighbors (indicated by the dotted arrow in Figure 4(b)). During the broadcast, the node is in *TxBeacon* mode. Following that, the node enters the *DataWait* mode and awaits data from a sending node.

Beacon Structure: The beacon structure follows the design used in the state-of-the-art MAC protocols. Particularly, a beacon comprises a receiverId – the ID of the receiver initiating the communication, which the sender uses to transmit data, and beaconId – a unique identifier for the beacon, which can be a sequence number or a timestamp.

❷*Sending State*: A node with data to send initially waits for a beacon in the *BeaconWait* mode. Upon receiving a beacon, it transitions to the *RcvBeacon* mode. The node then monitors the channel (in *CSMA* mode). If the channel is free, it switches to the *TxData* mode and transmits the data (illustrated by the dotted arrow in Figure 4(b)).

Data Packet Structure: Each data packet consists of (i) msgId – a unique identifier for the data message, (ii) senderId – the ID of the transmitting node, (iii) beaconId – the associated beacon identifier, (iv) payload length – the size (in bytes) of the transmitted data, and (v) payload – the actual data being sent.

Please note, there are no packet losses in the system because the sender transmits data only upon receiving the beacon, thus knowing that the receiver would be awake to receive the data. The beacon acts as a synchronization mechanism. Additionally, to prevent buffer overflow, the receiver notifies the sender of its available buffer space, which can be incorporated within the beacon. Further details are discussed in Section IV-C.

C. Node Design

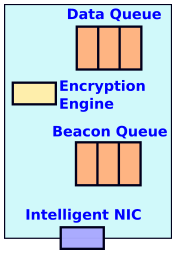

Fig. 5. Primary components in a sensor node

As shown in Figure 5, each sensor node has four primary components: an encryption engine, which encrypts the sensed data; a data queue, which stores the sensed data as well as data received from neighbors to forward towards the sink; a beacon queue, which stores the received beacons; and an intelligent network interface card (NIC) that adds the received packets to the corresponding queue depending on whether they are beacons or data. For ease of readability, we omit the sink node and the energy harvesting components in the figure.

Instead of using a very elaborate and energy-hungry MAC protocol [12], we used a standard EHD NIC, and then implemented the following modifications on it to enhance the operational performance of sensor nodes.

- *Intelligent NIC*: A basic NIC adds all the received packets (both beacons and data) to a single queue. Due to the limited queue size, beacons can clog the queue, preventing incoming data packets from being queued and thus being dropped. To avoid this packet loss, we use separate queues – our intelligent NIC adds packets to the queues as needed.
- *Selective beacon multicast*: Broadcasting beacons to all the neighbors may lead to an excessive number of data

packets. This may exceed the receiver's data queue capacity. Thus, we follow a round-robin policy, i.e., the beacon is multicast only to κ neighbors, where κ is the number of empty slots in the receiver's data queue.
Apart from this, a receiver node also checks if it has received data from all of the neighbors to whom it sent beacons. If data is not received from any neighbor for multiple consecutive beacon periods, the node assumes that the neighbor's energy has been exhausted, and it stops sending beacons to that neighbor for some time (η ms). In our experiments, we used $\eta = 12ms$.
Due to this selective multicast, we augment the beacon structure with the transmitting node's identifier ($SenderId$) and the number of slots reserved for that sender ($Available\_Entries$). Thus, the updated beacon structure is: <ReceiverId, BeaconId, SenderId, Available_Entries>.

- *Removal of old beacons*: It is possible for a node to receive multiple beacons while performing other tasks (e.g., sensing). Among these, only the latest beacon is considered, and the rest are discarded.

D. End-to-end Application

We consider a scenario that is common in a variety of applications – military monitoring, environmental sensing, and structural health monitoring [4]. In this, nodes sense their surroundings, encrypt the sensed data, and send it to the sink. Please note, similar to prior work on energy harvesting [1], we checkpoint the node's state if its energy is about to exhaust. The checkpointed state is restored, and the execution continues when the device is recharged. Thus, the entire application flow can be summarized as shown in Figure 4(c), wherein nodes *periodically* perform the sensing operation after a time interval known as the *sensing period* ($\mathcal{SP}$). Then, after performing local encryption (and possibly checkpointing), a node enters either the *receiving* or the *sending* state. For a comprehensive understanding, the entire process is illustrated using a finite state machine in Section 1 of the Appendix.

E. Objective Function

We consider a system with N nodes, where the ***per-node throughput***, denoted by T_i (here, $i \in 1, 2, \ldots, N$), signifies the number of messages arriving at the sink from a node i within a specific time interval. Additionally, we use T_{min} and T_{max} to represent the minimum and maximum number of messages received at the sink from any node, i.e., $T_{min} = \min_{\forall i \in N} T_i$ and $T_{max} = \max_{\forall i \in N} T_i$. Please note that T_{min} also represents the number of *snapshots* collected at the sink node. We now define define two system-wide metrics:

❶ ***Effective Throughput*** ($\mathcal{ET}$): It represents the total number of useful messages received at the sink from all the nodes and is computed as $\mathcal{ET} = T_{min} \times N$. Please note, *effective throughput* is not a novel metric; rather, it represents the real snapshot throughput after discarding oversensed messages. $\mathcal{ET}$ serves as a measure of the number of snapshots collected.

❷ ***Fairness*** ($\mathcal{F}$): We define fairness as the ratio of the minimum to the maximum throughput ($\mathcal{F} = T_{min}/T_{max}$). The fairness metric assesses the divergence in nodes' throughputs and is a measure of oversensing. A high value of the fairness metric ($\mathcal{F}$) indicates that the throughputs of all the nodes are similar, signifying a low level of oversensing. Therefore, high fairness corresponds to fewer wasted messages and less energy wastage. Please note, this metric is well known for capturing divergence and has been considered in previous works [30], [31].

Our objective is to determine the per-node sensing periods (SP_i, $\forall i \in [1, N]$) that maximize the *effective throughput* achieved at the sink while staying within some application-specific fairness threshold ($0 < F_{th} \leq 1$). Formally,

$$\begin{aligned} maximize \quad & \mathcal{ET} \\ s.t. \quad & \mathcal{F} >= F_{th} \end{aligned} \tag{1}$$

Please note that the fairness constraint is important because it is possible to have a scenario where the nodes' per-node throughputs are extremely skewed and thus the number of oversensed messages is quite high in addition to a high T_{max}. The fairness constraint is a measure of the wasted work ($wasted\ work$ = 0 if $fairness = 1$).

Stable and Unstable System: Depending upon the value of $\mathcal{F}$, we classify the system as stable or unstable. If $\mathcal{F} < F_{th}$, the system is *unstable*; otherwise, it is *stable*.

V. Mathematical Formulation

We formulate this as a *control-theoretic problem* where the system evolves iteratively by dynamically adjusting the sensing periods of nodes to achieve fairness ($\mathcal{F} \geq F_{th}$) while maximizing the effective throughput ($\mathcal{ET}$). At each iteration k, the sensing periods of nodes are updated based on their observed throughputs at the sink node from the previous iteration. These updates are designed to steer the system towards the objective, eventually leading to a stable and fair system. To demonstrate system stability and convergence, we employ a *Lyapunov function* [32]. The notations used in the formulations are summarized in Table II.

TABLE II
Glossary of Terms

Notations	Definition
N	Total number of nodes in the network
$i, T_i^\dagger$	ID and throughput of node i at the sink node
$\mathcal{SP}_i^\dagger$	Sensing period of node i
$a_i^\dagger$	Node specific parameter
$\mathcal{F}^\dagger$	System fairness
F_{th}	Application-specific fairness threshold
$T_{min}^\dagger, T_{max}^\dagger$	Min. and the max. throughput values in T_{list}
$\mathcal{G}\dagger$,$\mathcal{B}\dagger$,$\mathcal{N}\dagger$	Set of good, bad, and neutral nodes
$\mathcal{G}^*\dagger$,$\mathcal{B}^*\dagger$,$\mathcal{N}^*\dagger$	Subsets of good, bad, and neutral nodes whose sensing periods have to be updated
θ	Step size to update the nodes' SP
$V(k)$	Lyapunov function
$T_{list}^\dagger$	Array of $\langle i, T_i \rangle$, $\forall i \in [1, N]$
$G_{list}^\dagger, B_{list}^\dagger$	List of good and bad nodes, respectively
$N_{list}^\dagger$	List of all the nodes (sorted in ascending order of T)
$M_i^\dagger$	Number of messages sent by node i
p	Probability of updating a node's sensing period
m, n	Number of rows and columns in the network
R_row_i	Relative row number of node i

†: These values are computed at the end of each iteration

A. Sensing Period Update

The throughput of node i (T_i) is inversely proportional to its sensing period ($\mathcal{SP}_i$). At the end of each iteration, the sink nodes performs three tasks:

❶ ***System Stability Assessment***: The sink computes $T_{\min}(k)$ and $T_{\max}(k)$ to determine system's stability, based on which, it adjusts sensing periods. If the system is **unstable** ($\mathcal{F}(k) < F_{th}$), sensing periods are adjusted to stabilize the system – some nodes' sensing periods are increased while some are decreased. If they system is **stable**, the sensing period of certain nodes are reduced to enhance overall system's effective throughput ($\mathcal{ET}(k)$).

❷ ***Node Classification***: Nodes are categorized into three sets based on their throughput: *good* nodes ($\mathcal{G}$) – throughput close to $T_{max}(k)$, *bad* nodes ($\mathcal{B}$) – throughput close to $T_{min}(k)$, and *neutral* nodes ($\mathcal{N}$) – the remaining nodes. The throughput range $[T_{min}(k), T_{max}(k)]$ is divided into three equal parts to classify nodes. A node i is classified as follows:

$$i \in \begin{cases} \mathcal{G}, & \text{if } T_i(k) \geq T_{max}(k) - \frac{T_{max}(k)-T_{min}(k)}{3} \\ \mathcal{B}, & \text{if } T_i(k) < T_{min}(k) + \frac{T_{max}(k)-T_{min}(k)}{3} \\ \mathcal{N}, & \text{otherwise} \end{cases} \quad (2)$$

Let $|\mathcal{G}|, |\mathcal{B}|, |\mathcal{N}|$ denote the number of nodes in these categories.

❸ ***Sensing Period Update***: Based on the system stability, sensing periods are updated as follows:

Case 1: Unstable System ($\mathcal{F}(k) < F_{th}$) : To improve fairness, the sensing periods of nodes in $\mathcal{G}$ are increased (to reduce their throughput), while those in $\mathcal{B}$ are decreased (to increase their throughput). However, to ensure minimal perturbation, only $min(|\mathcal{G}|, |\mathcal{B}|)$ nodes from each set undergo updates. $\mathcal{B}^*$ consists of the **first** $min(|\mathcal{G}|, |\mathcal{B}|)$ elements of B, sorted in **ascending** order. $\mathcal{G}^*$ consists of the **first** $min(|\mathcal{G}|, |\mathcal{B}|)$ elements of G, sorted in **descending** order. Sorting helps in updating the worse and the best nodes. The updated sensing periods for the next iteration $(k+1)$ follow:

$$\mathcal{SP}_i(k+1) = \begin{cases} \mathcal{SP}_i(k) + \theta, & \text{if } i \in \mathcal{B}^* \\ \mathcal{SP}_i(k) - \theta, & \text{if } i \in \mathcal{G}^* \\ \mathcal{SP}_i(k), & \text{otherwise} \end{cases} \quad (3)$$

where $\theta > 0$ is the step size.

Case 2: Stable System ($\mathcal{F}(k) \geq F_{th}$): A subset $\mathcal{N}^* \subset (\mathcal{G} \cup \mathcal{B} \cup \mathcal{N})$ is identified, consisting of the 50% of all the nodes with lowest throughput. Their sensing periods are reduced to increase the system's effective throughput ($\mathcal{ET}$):

$$\mathcal{SP}_i(k+1) = \begin{cases} \mathcal{SP}_i(k) - \theta, & \text{if } i \in \mathcal{N}^* \\ \mathcal{SP}_i(k), & \text{otherwise} \end{cases} \quad (4)$$

Only a subset of nodes is updated to prevent system instability.

B. Convergence Analysis

In our update mechanism, poorly performing nodes (B) increase their sensing rate, while well-performing nodes (G) reduce theirs. Intuitively, this should shrink the range $[T_{min}, T_{max}]$, leading to fairness. However, intuition alone is insufficient, as second-order effects introduce complexities.

One such complexity arises from the non-linear interactions among nodes. The updates dynamically alter the number of nodes in each category – Good ($\mathcal{G}$), Bad ($\mathcal{B}$), and Neutral ($\mathcal{N}$) – leading to varying adjustment magnitudes across nodes. Consequently, the system does not undergo a simple linear transformation in each iteration. Additionally, since message forwarding occurs in a multi-hop manner, a change in the sensing period of one node influences downstream nodes. This interdependence means that local sensing period changes propagate in a complex manner, affecting message flow across the network in a non-trivial way.

To formally establish convergence, we employ a **Lyapunov function**, which satisfies the following properties:

1) **Non-negativity**: $V(k) \geq 0$ at all times.
2) **Strict decrease over iterations**: $V(k+1) < V(k)$, ensuring progression towards stability.
3) **Equilibrium condition**: $V(k) = 0$ only when the system reaches the fair state.

We define our following Lyapunov function as follows:

$$V(k) = (F_{th} - \mathcal{F})^2 + \gamma |T_{max}(k) - T_{min}(k)|^2 \quad (5)$$

where $\gamma > 0$ is a weighting parameter. The first term ensures $\mathcal{F}(k)$ remains close to F_{th}, while the second ensures the range $T_{max}(k) - T_{min}(k)$ contracts over time. Both terms are necessary: if only the first were considered, large variations in sensing periods could persist, leading to instability. Similarly, if only the second term were considered, the fairness ($\mathcal{F}(k)$) could still deviate from F_{th}, violating the fairness constraint.

Now, we show that our considered function satisfies all the three conditions of Lyapunov's function. Since, both terms are squared, so $V(k) \geq 0$. To show $V(k)$ is non-increasing (i.e., $V(k+1) < V(k)$), we analyze both cases:

Case 1: $\mathcal{F}(k) < F_{th}$ The sorted selection mechanism ensures that updates target the most extreme nodes, minimizing oscillations. Since equal number of nodes from sets $\mathcal{B}$ and $\mathcal{G}$ are updated, the gap $T_{\max}(k) - T_{\min}(k)$ decreases. This increases $\mathcal{F}(k)$, moving it towards F_{th}, ensuring $V(k+1) \leq V(k)$.

Case 2: $\mathcal{F}(k) \geq F_{th}$ Since 50% of nodes reduce their sensing period, $T_{\min}(k)$ increases, further shrinking $T_{\max}(k) - T_{\min}(k)$, leading to $V(k+1) \leq V(k)$.

Since $V(k) \geq 0$ and non-increasing, it converges to a steady state where $\mathcal{F}(k) \approx F_{th}$ and $T_{\min}(k)$ is maximized.

VI. Simulated Algorithms

The above formulation describes a simplified, non-deterministic system, as it does not consider certain complex aspects such as the nodes' energy harvesting characteristics and the MAC protocol. However, to implement a fully functional system capturing all the real-world complexities and interactions within the system, it is essential to simulate an entire network of nodes. We accomplish this step-by-step through two algorithms derived from the above mathematical formulation.

Overview: In the first algorithm, which is a hypothetical baseline, we incorporate the energy harvesting characteristics of the nodes and propose a *centralized* approach where the sink node computes and assigns updated sensing periods to all network nodes. We assume that the sink has complete and instantaneous knowledge of the entire network, including each node's current energy level and buffer size. Leveraging

this global visibility, the sink determines the optimal sensing periods and instantly communicates them to the nodes without considering any communication overhead. As a result, this serves as a *hypothetical, omniscient* solution, which we use as an *optimal* baseline.

We then introduce a more *practical* version of this algorithm that accounts for communication overheads and incorporates the complexity of the MAC protocol. Since computing and communicating updated sensing periods for all nodes would impose significant overhead on the sink, we propose a *distributed* algorithm in which each node determines its own sensing period locally instead of relying on the sink for direct assignments. These decisions are guided by minimal and infrequent feedback from the sink, specifically the maximum and minimum throughput values. This distributed approach improves scalability while maintaining effectiveness.

A. Centralized Algorithm (Hypothetical and Omniscient)

The centralized algorithm operates at the sink, which initially assigns sensing periods to all nodes. These values are chosen randomly but remain within a reasonable range. The sink then periodically updates the sensing periods after every *epoch* (a sufficiently long interval) by executing the function $UpdateAll\_SP$ (Algorithm 1). The notations used in the algorithm are summarized in Table II. After multiple iterations of this function, the sensing period configuration that yields the highest effective throughput ($\mathcal{ET}$) is selected.

Algorithm 1 Function $UpdateAll\_SP$

```
1: procedure UpdateAll_SP(T_list)
2:     F ← T_min/T_max;
3:     if F < F_th then                                   ▷ unstable system
4:         G ← goodNodes(T_list)
5:         B ← badNodes(T_list)
6:         G_list ← sortDescending(G)
7:         B_list ← sortAscending(B)
8:         n ← min(|G_list|, |B_list|)
9:         for itr ← 1 to n do        ▷ Update SP of n good and bad nodes
10:            i ← G_list[itr]
11:            SP_i ← SP_i + θ          ▷ Increase the SP of the good node
12:            i ← B_list[itr]
13:            SP_i ← SP_i − θ          ▷ Decrease the SP of the good node
14:        end for
15:    else                                                ▷ stable system
16:        N_list ← sortAscending(T_list)
17:        for itr ← 1 to |N_list|/2 do          ▷ Update 50% of the nodes
18:            i ← N_list[itr]
19:            SP_i ← SP_i − θ
20:        end for
21:    end if
22: end procedure
```

Algorithm 1 - Function UpdateAll_SP: The function takes as input T_{list}, a list containing the per-node throughput values (Line 1). Each entry in the list is a tuple $\langle i, T_i \rangle$, representing a node's identifier and its corresponding throughput. The sink first determines whether the system is in ***stable*** or ***unstable*** state (Line 2,3).

Unstable system:

- The function $goodNodes()$ identifies the set $\mathcal{G}$ (refer to the mathematical formulation in Section V), which is then sorted in *descending* order of throughout to form G_{list}. Similarly, $badNodes()$ identifies the set $\mathcal{B}$, which is then sorted in *ascending* order of throughput to form B_{list} (Lines 4-7).
- As outlined in the formulation, the sensing periods of an equal number of nodes from both G_{list} and B_{list} are updated to improve fairness $\mathcal{F}$ and stabilize the system (Lines 9-14).
- A very low value of θ can significantly slow down system stabilization, whereas an excessively high θ may cause oscillations, hindering convergence. Empirical evaluation suggests that θ = 50 ms provides a reasonable balance.

Stable system:

- The nodes are sorted in ascending order of throughput (Line 16) to form N_{list}.
- The sensing periods of the first 50% of nodes in N_{list} are reduced, allowing them to transmit more messages and thereby increasing the system's effective throughput (Lines 17-20).

At the end of Algorithm 1, the updated sensing periods are instantaneously transmitted to the nodes (i.e., updated functionally). After 40 iterations, the sensing period configuration that maximizes $mathcalET$, while satisfying the fairness constraint, is chosen. This marks the completion of the *centralized algorithm*.

Note: All of this is happening in a software simulation.

Time Complexity: Since the algorithm involves sorting the throughput values of N nodes, its runtime complexity is $O(N \log N)$.

Challenges in Practical Deployment The *centralized* solution is impractical for real-world deployment due to the absence of real-time communication and integration with the MAC protocol. Introducing real-time communication trivially presents the following challenges:

1) Broadcasting the entire sensing period ($\mathcal{SP}$) vector to all nodes instead of performing functional updates would incur significant network overhead due to the large message size ($32N$ bits, assuming each sensing period is represented by 32 bits). As N increases, encapsulating all this information in a single message becomes infeasible.
2) Variations in node distances from the sink cause delays in the timely reception of updated sensing periods. Additionally, fluctuations in node energy availability may hinder reliable reception of updates, particularly for distant nodes.

B. WaWoR - A Distributed Algorithm

To address these challenges, we propose *WaWoR*– a novel best-effort distributed solution where nodes locally compute and incrementally update their sensing periods based on their proximity to the sink. This computation logic requires minimal feedback from the sink, which is sent only after an epoch. The feedback consists of a 4-tuple $\langle T_{min}, T_{max}, |G_{list}|, |B_{list}| \rangle$, where $|...|$ denotes set cardinality. The sink efficiently transmits this feedback to all nodes by *piggybacking* it onto beacon messages already used in the MAC protocol, avoiding additional network overhead of broadcasting the feedback explicitly.

Concept: The core idea aligns with Algorithm 1, with key differences:

1) Unlike Algorithm 1, where the sink had instant system-wide visibility, in *WaWoR*, nodes update their sensing periods probabilistically, due to the absence of a complete set $\mathcal{G}$ and $\mathcal{B}$.
2) Determining whether the node itself belongs to the set $\mathcal{G}$, $\mathcal{B}$, or $\mathcal{N}$ is more complex, as nodes lack direct throughput visibility at the sink. Instead, each node tracks the number of its sensed messages (M_i) that have been transmitted into the network, and use it as an estimate for throughput (T_i). Although some messages may still be in transit, M_i won't be much smaller than T_i because if messages are unable to pass through the network, a back pressure builds, stopping further transmissions. Thus, given sufficient time and assuming no message losses (Section IV-B), M_i serves as a reasonable approximation for T_i.

Algorithm 2 - WaWoR Algorithm: Each node periodically executes $update\_SP_i()$ locally, based on the received feedback $\langle T_{min}, T_{max}, |G_{list}|, |B_{list}|\rangle$ (Line 1). The fairness metric $\mathcal{F}$ is computed (Line 2), and its comparison with the application-specific fairness threshold (F_{th}) determines stability.

Algorithm 2 Distributed Algorithm

```
1:  procedure Update_SP_i(⟨T_min, T_max, |G_list|, |B_list|⟩, M_i, R_row_i)
2:      F ← T_min/T_max;
3:      if F < F_th then                                          ▷ unstable system
4:          if M_i ≥ T_max − ((T_max − T_min)/3) then             ▷ good node
5:              if |G_list| ≤ |B_list| then   ▷ definitely update all good nodes
6:                  p ← 1
7:              else
8:                  p ← 0.5 + 0.5 · R_row_i
9:              end if
10:             SP_i ← SP_i + θ, with probability p
11:         else if M_i ≤ T_min + ((T_max − T_min)/3) then        ▷ bad node
12:             if |B_list| ≤ |G_list| then   ▷ definitely update all bad nodes
13:                 p ← 1
14:             else
15:                 p ← 0.5 + 0.5 · (1 − R_row_i)
16:             end if
17:             SP_i ← SP_i − θ, with probability p
18:         end if
19:     else                                                      ▷ stable system
20:         p ← 0.5
21:         SP_i ← SP_i − θ, with probability p
22:     end if
23: end procedure
```

Unstable system:

- If the node is a *good* node (Line 4), it *increases* its sensing period, while it *decreases* its sensing period if it is a *bad* node (Line 11). The classification is based on M_i (the estimate for T_i) and the received T_{min} and T_{max} values (as described in Section VI-A). Similar to Algorithm 1, *WaWoR* intends to update the sensing periods of $min(|G_{list}|, |B_{list}|)$ nodes, to avoid excessive perturbations.
- However, unlike the centralized algorithm, here we do not know the relative goodness or badness of a node. Therefore, we devise a heuristic approach to probabilistically update the sensing periods (with probability p).
- A good node increases its sensing period with probability 1 if $|G_{list}| < |B_{list}|$ (Lines 5,6), and a bad node decreases its sensing period with probability 1 if $|B_{list}| < |G_{list}|$ (Lines 12,13). This ensures that at least $min(|G_{list}|, |B_{list}|)$ nodes from one of the sets (good/bad nodes) are updated.
- For other nodes, update probability p is based on the following intuition. A node *farther* from the sink will experience a higher message latency compared to the *closer* nodes. So a *good* node that is *closer* to the sink should increase its sensing period. However, this increase in the sensing period should be done with a low likelihood, as otherwise, it may start contributing to system instability by becoming a *bad* node in the next epoch (its throughput might get reduced drastically). We take this effect into account using the *relative row number*. For node i, its *relative row number* (R_row_i) is computed as $R\_row_i = row_i/m$, where row_i is its row number and m is the total number of rows in the network (refer to Figure 4(a) - row number 0 is the row farthest from the sink). If nodes far away from the sink are good nodes, their sensing periods will be updated with a low probability as compared to the good nodes closer to the sink.
- Line 8 ensures that update probability is always > 0.5; otherwise, it could be very small for nodes far away from the sink, given their small relative row number (particularly for large networks). In contrast to good nodes, the probability of a bad node decreasing its sensing period should be high for far away nodes. Particularly, for node i, its p is proportional to $1 - R\_row_i$ (Line 15). Once p is calculated, a good node probabilistically increments, while a bad node probabilistically decrements its sensing period by a step size θ (Lines 10 and 17).

Stable system:

- A node decreases its sensing period with 50% probability (Lines 19-22), aiming to increase the message transmission and hence the effective throughput ($\mathcal{ET}$).

Time Complexity: Unlike Algorithm 1, which requires sorting operations at the sink, *WaWoR* relies on simple arithmetic computations performed locally by individual nodes, thereby reducing the runtime complexity to $O(1)$.

C. Overheads of WaWoR

WaWoR has a negligible additional overhead as compared to a standard receiver-initiated MAC protocol's overheads. *WaWoR*'s overhead is primarily due to the periodic feedback broadcasts from the sink node. The frequency of these broadcasts depends on the variability of the energy of the ambient source. Furthermore, as mentioned before, we piggyback the *broadcast values* on the beacon messages rather than send any additional messages. In addition, the feedback information is quite small: we add four fields to the beacon: $T_{min}, T_{max}, |G_{list}|$, and $|B_{list}|$. Among these, both T_{min} and T_{max} can be represented as 32-bit integers, while the list sizes are bounded by the number of nodes (N) in the system and thus, each size can be represented using $log_2 N$ bits. As a result, the size overhead is $2 * (32 + log_2 N)$ bits. Please note that in comparison to the total number of message transmissions (both beacons and

sensed data) in the network, these feedback messages are quite infrequent and are sent after every epoch.

Impact of* WaWoR *on EH-WSNs: The proposed approach will enhance existing EH-WSNs without interfering with their current functionality. It will make global snapshot collection more efficient, sending fewer but more effective messages to the sink, thereby conserving energy. This enhancement requires only a minor hardware upgrade to the NIC, which can be easily implemented in existing EHDs.

VII. Evaluation

A. Description of the Simulation Environment

We used NetSim [33], a widely-adopted commercial event-driven simulator known for its accuracy. NetSim was chosen for its built-in energy model, which facilitates precise power consumption analyses. It also has a modular and extensible architecture that allows the incorporation of energy harvesting and custom MAC protocols. We enhanced NetSim by integrating the following two key components: ❶ energy harvesting support, enabling nodes to adapt their operation based on harvested energy, and ❷ a receiver-initiated MAC protocol, designed to enhance communication efficiency [6], [12].

NetSim was selected over other simulators (e.g., NS-3 [34], OMNET++ [35] and OPNET [36]), due to its ease of extensibility, built-in support for wireless networks, and capability to model energy-aware protocols. These enhancements ensured that our evaluation framework effectively captures the challenges and trade-offs in energy harvesting wireless sensor networks (EH-WSNs).

Figure 6 shows the high-level structure of the sensor nodes (both the sender and receiver) and the network (simulated as a queue of outstanding packets).

We assume that all the nodes have synchronized clocks and operate at the same frequency. The network is represented by a global 1D circular array (referred to as the *network queue*) indexed by the global clock. When a node has to transmit a message over the network, it checks if the network is *free* (modeled by a single bit to indicate the network status) and adds the message to the network queue. The index at which the message is added is computed based on the transmission latency. We have used the latency values corresponding to TI's CC1310 device [37], which has been used as a sensor node in recent energy harvesting-based monitoring applications [38].

We assume that all the nodes have static routes configured, which means that each node knows its position in the network and is aware of its neighbors.

1) Simulation Details: Each node primarily performs three tasks: sensing, sending and receiving messages (Section IV-B). To carry out the sensing task, the node senses its environment, encrypts the sensed data using the encryption engine, and adds the encrypted packet to the node's data queue (shown by steps (1.1) and (1.2) in Figure 6).

To perform the sending and receiving tasks, the following sequence of steps is followed. Since a receiver-initiated protocol is used, the receiver first multicasts a beacon to its neighboring nodes by adding the beacon messages to the network queue

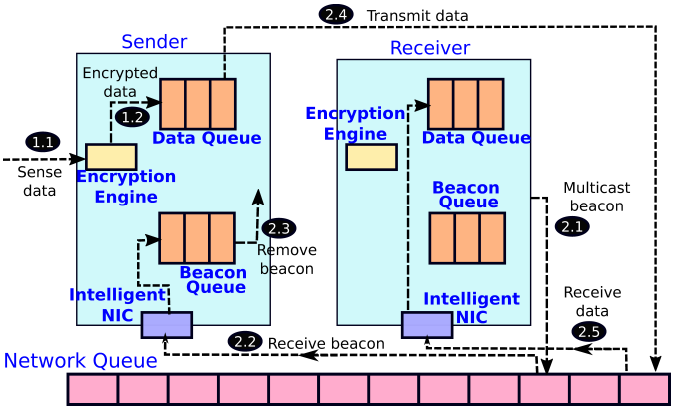

Fig. 6. Brief overview of the simulator

(step 2.1). Upon receiving the beacon (step 2.2), the sender node processes it by transmitting a data packet from its data queue, which is then added to the network queue (steps 2.3 and 2.4). This completes the sending process. The data packet is then removed from the network queue and delivered to the receiver based on the transmission and reception latencies (step 2.5), thus completing the receiving task.

B. Network configuration

After integrating *WaWoR* into our network simulator, we conducted experiments using the following network configuration. We examined networks of various sizes and skews (a skew of η means the number of rows in the network grid is η times the number of columns, as shown in Figure 4(a)). Specifically, we considered networks with $50, 75, 100$, and 150 nodes, with a skew set to 6 (other skew values yielded similar results). Similar to previous studies, nodes were assumed to be stationary and aware of their immediate neighbors on the path to the sink [6], so node discovery was not included in our experiments. The simulation was run for two minutes to allow the system to stabilize completely. A fairness threshold (F_{th}) of 0.75 (adjustable based on application requirements) was employed in all our experiments. Each node sensed according to its sensing period as described in Section IV-E.

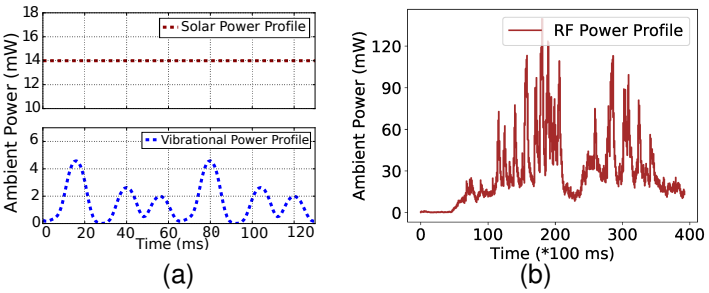

Fig. 7. Ambient power profiles of three sources: (a) solar and vibrational (b) RF (adapted from [20], [21])

C. Ambient Energy Profiles

We conducted experiments using commonly used ambient power profiles (Figure 7): solar, RF (radio frequency) and vibrational [20], [21], [38]. These profiles were chosen because the corresponding sources are easily accessible in various environments. The profiles differ in intensity and variability. The

solar profile provides a constant power of $14mW$ [21], [22], while the vibrational and RF profiles offer variable power. The vibrational profile exhibits a roughly sinusoidal pattern [21], and the RF power fluctuates significantly, representing a mix of power from multiple sources, as modeled by an RF trace from Mementos [20].

D. Baselines Considered for Comparison

As outlined in Section I, *WaWoR* is initially compared against two theoretical baselines, consisting of two *hypothetical omniscient* algorithms. Subsequently, we compare *WaWoR* with two state-of-the-art EH-WSN algorithms.

(1) Hypothetical Baselines The two *hypothetical omniscient* algorithms are (i) *OptDyn* and (ii) *OptStat*. *OptDyn* is the same as the *Centralized Algorithm* discussed in Section VI-A, where sensing periods of nodes are *dynamically* changed based on the instantaneous energy available. *OptStat*, on the other hand, *statically* experiments with a very large number of sensing periods and chooses the one that maximizes the effective throughput at the sink. Both *OptDyn* and *OptStat* are omniscient – fully aware of the ambient energy profiles at all nodes – and hypothetical, meaning they are practically infeasible. In contrast, *WaWoR* operates with each node only aware of its own state, without any knowledge of the network dynamics (network congestion and the states of nodes' queues) or the energy profiles at other nodes.

In *OptStat*, we consider a set of sensing periods (referred to as SP_{set}). For our experiments, we considered $SP_{set} = \{50, 100, 150, 200, \ldots, 3000\}$ milliseconds, covering the entire spectrum of practically possible sensing periods. From this set, we choose two values X and Y s.t. $X < Y$, and *fit a function* between them. Similar to Sah et al.'s [7], we have considered linear and exponential functions. The sensing periods of the nodes are set according to this function (row-wise). Specifically, all the nodes in the first row (farthest from the sink) sense after every X time units and nodes in the last row (closest to the sink) sense after every Y time units. The sensing period of the intermediate nodes varies linearly (or exponentially) in the interval $[X, Y]$. We sweep through SP_{set}, experimenting with every possible combination of X and Y from SP_{set}, and then choose the sensing periods that result in the maximum effective throughput at the sink. In Section 2 of the Appendix, we have shown the significance of OptSat, i.e., why is it required to perform extensive experimentation.

Please note that the results of *OptStat* depend upon the granularity of SP_{set}; the finer the granularity, the greater is the probability of achieving a higher effective throughput. Despite their impracticality, these two omniscient algorithms provide a good baseline for evaluating *WaWoR*.

(2) State-of-the-art Algorithms Among the existing works discussed in Section II, the most recent state-of-the-art contributions are by Jeong et al. [14] and Quan et al. [16], which have dynamic sensing periods similar to *WaWoR*. Therefore, we decided to compare *WaWoR* against these approaches. However, as noted in Section II, these approaches lack comprehensive implementation details and rely on some impractical assumptions. In our evaluation, we have accounted for practical considerations, implementing these approaches using a receiver-initiated MAC protocol to establish a common baseline for a fair comparison. Additionally, we used the same application flow for all three approaches, where each node sequentially performs sensing, encryption and reception/transmission tasks (see Figure 4(c)).

We primarily compare the effective throughput ($\mathcal{ET}$), which reflects the number of snapshots captured (i.e., $\mathcal{ET} = no\_of\_snapshots \times no\_of\_network\_nodes$ within a given time interval)[2]. Similar to prior work [13]–[16], our focus is on maximizing throughput at the sink rather than minimizing the end-to-end delay since the snapshotting throughput is a more relevant metric in EH-WSNs. It is important to note that throughput and delay are distinct and orthogonal concepts, and the choice of metric depends on the specific use case.

E. Experiments

1) WaWoR*'s efficiency compared to theoretical baselines:* Next, we demonstrate the efficiency of *WaWoR* by comparing it with *OptDyn* and *OptStat*. To illustrate *WaWoR*'s scalability[3], we present results for networks with varying node counts: 50, 100, and 150.

(1) Network Size - 50: Figure 8 compares the effective throughputs (messages received in 2 seconds in the steady state) and the corresponding sensing periods (SP) of the three systems. It shows that *WaWoR* achieves an effective throughput similar to and sometimes better than both the hypothetical systems across all the energy sources. Specifically, *WaWoR*'s throughput is within 1% of *OptDyn* for vibrational and RF sources and 1.8% higher for the solar source. *WaWoR*'s advantage over $OptDyn$ arises because *OptDyn instantly* adjusts sensing periods, while *WaWoR* updates them only after receiving feedback from the sink causing updates to be *delayed*, which paradoxically prove to also be timely. This positively impacts $\mathcal{ET}$ as nodes closer to the sink receive feedback earlier, adjust their sensing periods, and thus improve network dynamics (congestion level), which in turn helps farther nodes make better decisions. This is a stable observation.

Compared to *OptStat*, *WaWoR* performs better by 1.8%, 19% and 75% for solar, vibrational and RF sources, respectively. The greater gains for sources with more variability highlight the benefits of *WaWoR*'s adaptive nature.

Figures 8(d,e,f) compare the sensing periods across all the three systems. The graphs demonstrate that the per-node throughputs align with the per-node sensing periods: nodes with smaller sensing periods tend to have higher throughputs. Note that if a node's sensing period is set to τ ms, the actual duration between two consecutive sensing events may differ from τ due to the node potentially being off at the scheduled sensing time (due to variable and uncertain ambient energy). Therefore, we plot the sensing periods *actually* observed by the nodes.

(2) Network Size - 100: Figure 9 compares the three systems in a 100-node network. Its results are similar to the nodes in a 50-node network. For the solar source, all the three

[2]Henceforth, we interchangeably use $\mathcal{ET}$ and $number\_of\_snapshots$

[3]Due to space limitations, the sensing periods for larger networks have been moved to Section 3 of the Appendix.

designs achieve $2-3\times$ the throughput compared to the other two sources. This is due to the high power availability when the solar source (constant $14mW$) is used. This increases the likelihood of successful communication since nodes have sufficient energy.

WaWoR outperforms $OptDyn$ by $2-26\%$ for vibrational and RF sources, respectively, and is 90% as effective for the solar source. *WaWoR*'s advantage with fluctuating-energy sources comes from its delayed updates, leading to better decision making. This delay helps distant nodes adapt better to network changes and choose more appropriate sensing periods. In contrast, $OptDyn$'s instantaneous updates make it *excessively reactive*, causing nodes to increase sensing and transmission *immediately* (based on the sink's decision) when a burst of energy is available at these nodes. This can lead to network congestion without significantly improving the sink throughput. Thus, there is a non-linear relationship between sensing periods and the throughput. Delayed updates favor *WaWoR* as they let the network to stabilize allowing distant nodes to adapt. However, this advantage is only for variable sources like RF and vibrational. For the constant solar source, where there are no bursts, $OptDyn$ performs better as the sink's decisions remain optimal. In contrast, *WaWoR*'s delayed updates cause distant nodes to continue with suboptimal sensing periods for a longer time, reducing performance, especially as network size increases.

When compared to *OptStat*, *WaWoR*'s dynamic nature enabled it to achieve 3% higher throughput for the solar source and $11-57\%$ higher throughput for the other two sources.

(3) Network Size - 150: Figure 10 shows that for larger networks (with 150 nodes), all the three systems show a similar $\mathcal{ET}$ across the three energy sources. *WaWoR* is within $95-98\%$ of $OptDyn$ and $2-8\%$ better than $OptStat$. *WaWoR*'s slight underperformance compared to $OptDyn$ is due to a larger delay in feedback reaching distant nodes, leading them to continue sensing at outdated and suboptimal periods for extended durations.

The experiments show that *WaWoR* excels in small and medium-sized networks. In larger networks, it performs comparably to other systems. It is also important to note that $OptStat$ may not always perform very well as its performance depends on the granularity of SP_{set} and the choice of the function for setting the sensing periods. A single function may not be optimal for all the sources and network sizes – for example, $OptStat$ performed well with a linear function in large networks yet struggled with an RF source in smaller networks.

2) Throughput Efficiency of WaWoR *compared to State-of-the-art Systems:* Table III compares the effective throughputs ($\mathcal{ET}$) of *WaWoR* with that of Quan et al.'s [16] and Jeong et al.'s [14] works. The results show that *WaWoR* consistently outperforms Jeong et al.'s work [14] across all the ambient sources and network sizes with performance improvements ranging from $[1.14-4.26]\times$. This notable performance difference is due to Jeong et al.'s conservative transmission strategy, which only allows transmission when the device's estimated energy exceeds the capacitor's limit. In contrast, *WaWoR* allows transmission whenever there is sufficient energy to send the stored data packets. In the case of *WaWoR*, this strategy does not lead to congestion in the network or dropped messages.

Compared with Quan et al.'s [16] approach, *WaWoR* shows a $[1.02-1.35]\times$ increase in $\mathcal{ET}$. While Quan et al.'s approach reduces traffic by sending packets layer by layer (at any time, only one layer sends and the next layer receives), it incurs substantial overheads. Nodes coordinate within each layer to ensure that the previous layer has completed its transmissions.

3) Energy Efficiency of WaWoR *compared to State-of-the-art Systems:* Table III compares the three approaches in terms of wasted messages and energy consumption. Jeong et al.'s [14] approach resulted in $[0.8-9.9]\times$ more wasted messages and

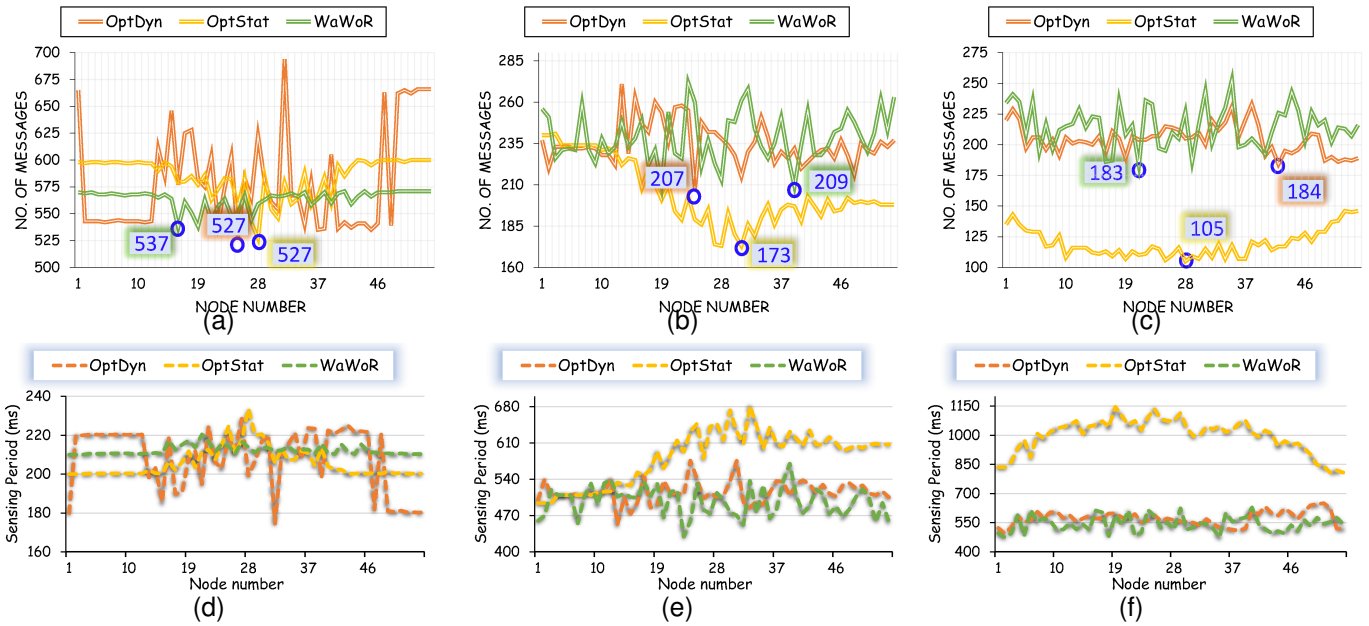

Fig. 8. **Network size:** 50 **(a-c)** Effective Throughput ($\mathcal{ET}$) comparison for different ambient power profiles: **(a)** solar, **(b)** vibrational, **(c)** RF and **(d-f)** Sensing periods (SP) comparison for different ambient power profiles: **(d)** solar, **(e)** vibrational, **(f)** RF

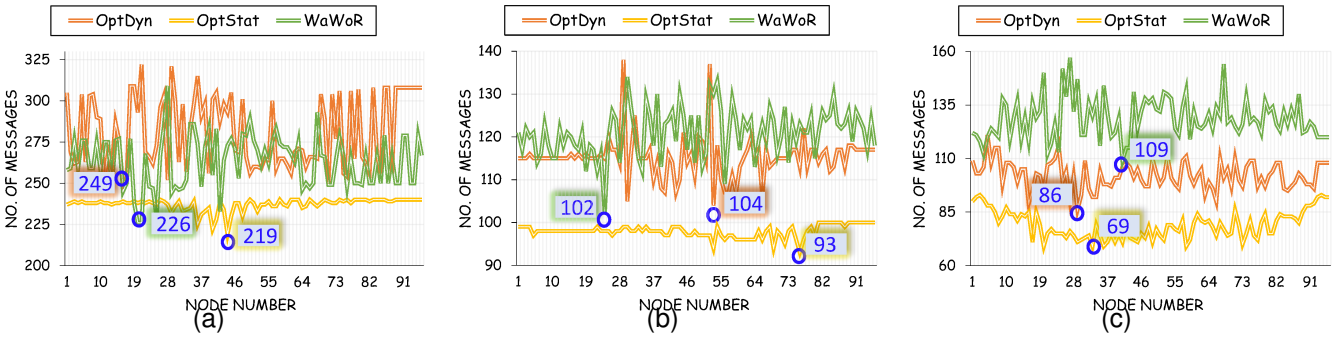

Fig. 9. **Network size:** 100 **(a-c)** Effective Throughput ($\mathcal{ET}$) comparison for different ambient power profiles: **(a)** solar, **(b)** vibrational, **(c)** RF

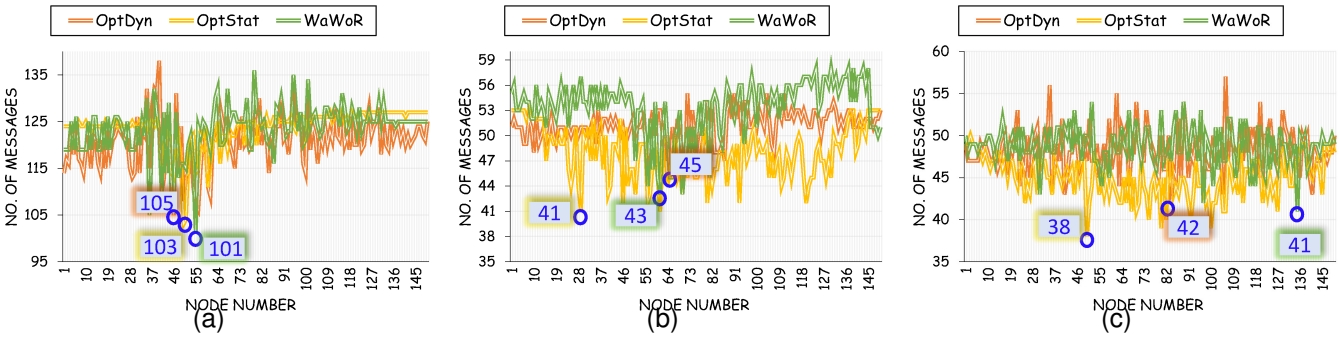

Fig. 10. **Network size:** 150 **(a-c)** Effective Throughput ($\mathcal{ET}$) comparison for different ambient power profiles: **(a)** solar, **(b)** vibrational, **(c)** RF

TABLE III
EFFECTIVE THROUGHPUT AND THE WASTEFUL WORK COMPARISON OF THE THREE SYSTEMS

#Nodes		**50 Nodes**			**100 Nodes**			**150 Nodes**		
Source		***WaWoR***	**Jeong**	**Quan**	***WaWoR***	**Jeong**	**Quan**	***WaWoR***	**Jeong**	**Quan**
Solar	$\mathcal{ET}$	537	126	520	226	60	202	101	30	93
	Msgs(K)	1.5	14.5	5.7	3.6	8.4	16.2	3.4	8.3	11.6
	En(mJ)	2.1	20.9	4.1	5.2	12.1	11.7	4.9	11.9	8.4
Vib.	$\mathcal{ET}$	209	139	200	102	28	100	43	34	40
	Msgs(K)	1.6	4.4	9	1.8	3.9	8	1.6	1.1	5
	En(mJ)	2.3	6.3	6.5	2.6	5.6	5.8	2.3	1.5	3.6
RF	$\mathcal{ET}$	183	90	169	109	32	81	41	36	33
	Msgs(K)	1.7	3.2	7.6	1.9	3.6	6.5	1.2	0.9	4.1
	En(mJ)	2.5	4.6	5.5	2.7	5.2	4.7	1.7	1.3	3
Msgs(K): Number of wasted messages ($\times 1000$); **En(mJ): Energy (in mJ) consumed in wasteful work**										

energy consumption than *WaWoR* with $[1.14 - 4.26]\times$ lower $\mathcal{ET}$. Similarly, Quan et al.'s [16] approach had $[3.4 - 5.6]\times$ more wasted messages and $[1.6-2.8]\times$ higher energy wastage. Despite using a layer-by-layer communication strategy, Quan et al.'s approach requires additional intra-layer communication, which *WaWoR* avoids.

To summarize, *WaWoR* captures $[1.02 - 4.26]\times$ more snapshots than state-of-the-art approaches, while also achieving energy savings of up to $9.9\times$ compared to Jeong et al. and $2.8\times$ compared to Quan et al.

VIII. Conclusion And Future Work

A. Conclusion

In this paper, we propose *WaWoR*– a distributed system in which the nodes locally update their sensing periods so that the effective throughput (number of snapshots) at the sink is maximized while satisfying the fairness constraints. The nodes can adapt their sensing periods to the variation in ambient energy. This proposal does not substantially impact the functionality of existing EH-WSNs, but will increase their efficiency by reducing the redundant messages and the overall energy usage. We show the efficiency of *WaWoR* by comparing it with four different systems, which include two theoretical baselines (both are hypothetical omniscient) and two state-of-the-art systems. Our system achieves a throughput ranging from $[90 - 126]\%$ of *OptDyn* and is within $[101.8 - 175]\%$ of *OptStat*. Compared to two state-of-the-art systems, *WaWoR* has an effective throughput that is $[1.02-4.26]\times$ higher. Along with this, *WaWoR* wastes around $[1.8 - 9.9]\times$ fewer messages and consumes approximately $[1.6 - 9.9]$ lesser energy as compared to two state-of-the-art algorithms.

B. Future Work

The proposed work can be further optimized in several ways. First, nodes can adjust the sensing periods based on the characteristics of the sensed signal. For instance, if the signal fluctuates only once a week (on average), frequent sensing isn't necessary, regardless of energy availability or network traffic. Furthermore, akin to cluster-based routing, nodes can be "locally grouped" (limited clustering) to reduce the number of redundant connections. We can have a few redundant cluster heads such that there is no single point of failure. The spatio-temporal similarities in the sensed values can be further leveraged to reduce the amount of sensing. This means that if a node detects that a neighboring node has already sensed and transmitted similar data, it need not send a message itself. This will require some coordination among nodes, which

incurs an energy cost, but it could be advantageous in certain environments.

REFERENCES

[1] P. Singla and S. R. Sarangi, "A Survey and Experimental Analysis of Checkpointing Techniques for Energy Harvesting Devices," *Journal of Systems Architecture*, p. 102464, 2022.

[2] O. Alamu, T. O. Olwal, and K. Djouani, "Energy Harvesting Techniques For Sustainable Underwater Wireless Communication Networks: A Review," *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, p. 100265, 2023.

[3] F. K. Shaikh and S. Zeadally, "Energy harvesting in wireless sensor networks: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 1041–1054, 2016.

[4] Z. Cai, Q. Chen, T. Shi, T. Zhu, K. Chen, and Y. Li, "Battery-Free Wireless Sensor Networks: A Comprehensive Survey," *IEEE Internet of Things Journal*, 2022.

[5] D. Crescini, F. Touati, and A. Galli, "Multiparametric Sensor Node for Environmental Monitoring Based on Energy Harvesting," *Atmosphere*, vol. 13, no. 2, p. 321, 2022.

[6] C. Blondia, "Evaluation of the end-to-end response times in an energy harvesting wireless sensor network using a receiver-initiated mac protocol," *Ad Hoc Networks*, p. 102971, 2022.

[7] D. K. Sah, K. Cengiz, P. K. Donta, V. N. Inukollu, and T. Amgoth, "EDGF: Empirical Dataset Generation Framework for Wireless Sensor Networks," *Computer Communications*, vol. 180, pp. 48–56, 2021.

[8] J. Zhang, H. Gao, K. Zhang, Q. Chen, and J. Li, "A Distributed Framework for Low-Latency Data Collection in Battery-Free Wireless Sensor Networks," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8438–8453, 2021.

[9] T. Zhu, J. Li, H. Gao, and Y. Li, "Latency-efficient Data Collection Scheduling in Battery-free Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 3, pp. 1–21, 2020.

[10] O. Gurewitz, M. Shifrin, and E. Dvir, "Data Gathering Techniques in WSN: A Cross-Layer View," *Sensors*, vol. 22, no. 7, p. 2650, 2022.

[11] J. Li, S. Cheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Approximate Holistic Aggregation in Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 13, no. 2, pp. 1–24, 2017.

[12] H. H. R. Sherazi, L. A. Grieco, and G. Boggia, "A comprehensive review on energy harvesting MAC protocols in WSNs: Challenges and tradeoffs," *Ad Hoc Networks*, vol. 71, pp. 117–134, 2018.

[13] B. Zhang, R. Simon, and H. Aydin, "Maximal Utility Rate Allocation for Energy Harvesting Wireless Sensor Networks," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, 2011, pp. 7–16.

[14] S. Jeong, H. Kim, D. K. Noh, and I. Yoon, "Energy-Aware Data Aggregation Scheme for Energy-Harvesting Wireless Sensor Networks," in *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*. IEEE, 2016, pp. 140–143.

[15] R.-S. Liu, K.-W. Fan, Z. Zheng, and P. Sinha, "Perpetual and Fair Data Collection for Environmental Energy Harvesting Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 4, pp. 947–960, 2010.

[16] Q. Chen, Z. Cai, L. Cheng, H. Gao, and J. Li, "Energy-collision-aware Minimum Latency Aggregation Scheduling for Energy-harvesting Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 17, no. 4, pp. 1–34, 2021.

[17] X. Bai, Z. Wang, L. Sheng, and Z. Wang, "Reliable Data Fusion of Hierarchical Wireless Sensor Networks with Asynchronous Measurement for Greenhouse Monitoring," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 3, pp. 1036–1046, 2018.

[18] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 254–263.

[19] A. Mouapi, N. Hakem, and G. Y. Delisle, "A new approach to design of RF energy harvesting system to enslave wireless sensor networks," *ICT Express*, vol. 4, no. 4, pp. 228–233, 2018.

[20] B. Ransford, J. Sorber, and K. Fu, "Mementos: System Support for Long-Running Computation on RFID-Scale Devices," in *ASPLOS*, 2011.

[21] P. Singla, S. Singh, and S. R. Sarangi, "Flexicheck: An adaptive Checkpointing Architecture for Energy Harvesting Devices," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019.

[22] S. J. Roundy, *Energy Scavenging for Wireless Sensor Nodes with a Focus on Vibration to Electricity Conversion*. University of California, Berkeley, 2003.

[23] D. K. Sah and T. Amgoth, "Renewable Energy Harvesting Schemes in Wireless Sensor Networks: A Survey," *Information Fusion*, vol. 63, pp. 223–247, 2020.

[24] D. Zhao, D. Liu, and X. Cao, "Distributed Optimization Framework for Mobile Data Collection With Energy Harvesting in Duty-Cycle WSNs," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 6456–6473, 2022.

[25] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 95–107.

[26] H. Wang, Q. Sun, and S. Wang, "A Survey on the Optimisation of Age of Information in Wireless Networks," *International Journal of Web and Grid Services*, vol. 19, no. 1, pp. 1–33, 2023.

[27] A. Basabaa and E. S. Elmallah, "Bounds on Path Exposure in Energy Harvesting Wireless Sensor Networks," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–7.

[28] K.-W. Lim, K. Kapusta, G. Memmi, and W.-S. Jung, "Multi-hop Data Fragmentation in Energy Harvesting Wireless Sensor Networks," in *2020 Global Internet of Things Summit (GIoTS)*. IEEE, 2020, pp. 1–6.

[29] J. Mao, X. Jiang, and X. Zhang, "Analysis of node deployment in wireless sensor networks in warehouse environment monitoring systems," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–15, 2019.

[30] L. Dai, W. Chen, L. J. Cimini, and K. B. Letaief, "Fairness Improves Throughput in Energy-Constrained Cooperative Ad-Hoc Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 7, pp. 3679–3691, 2009.

[31] H. Yousefi'zadeh, H. Jafarkhani, and A. Habibi, "Layered Media Multicast Control (LMMC): Rate Allocation and Partitioning," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 540–553, 2005.

[32] Lyapunov function, "Estimation lemma – Wikipedia, the free encyclopedia," 2024, [Online; accessed 4-March-2025]. [Online]. Available: https://en.wikipedia.org/wiki/Lyapunov_function

[33] Tectos. (visited on 07/04/2023) NetSim – Network Simulator. https://www.tetcos.com/. Tectos.

[34] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network Simulations with the ns-3 Simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[35] A. Varga, "OMNeT++," *Modeling and tools for network simulation*, pp. 35–59, 2010.

[36] J. Prokkola, "Opnet-Network Simulator," *URL http://www. telecomlab. oulu. fi/kurssit/521365A tietoliikennetekniikan simuloinnit ja tyokalut/Opnet esittely*, vol. 7, 2006.

[37] Texas Instruments. (2018) CC1310-SimpleLink Ultra-Low-Power Sub-1 GHz Wireless MCU. https://www.ti.com/product/CC1310.

[38] P. Loreti, A. Catini, M. De Luca, L. Bracciale, G. Gentile, and C. Di Natale, "The Design of an Energy Harvesting Wireless Sensor Node for Tracking Pink Iguanas," *Sensors*, vol. 19, no. 5, p. 985, 2019.

Priyanka Singla received her Bachelor's degree in Computer Science and Engineering from Thapar University, Patiala, Punjab, India, and her Master's degree (M.Sc. Engineering) from the Indian Institute of Science, Bangalore, Karnataka, India. She earned her Ph.D. from the Indian Institute of Technology Delhi, India, and subsequently worked at Intel, Bangalore. She is currently working with Baya Systems, Bangalore. Her research interests include architectural design for energy harvesting systems.

Smruti R. Sarangi received the B.Tech. degree in computer science from the Indian Institute of Technology Kharagpur, Kharagpur, India, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana Champaign, Champaign, IL, USA. He is currently an Usha Hasteer Chair Professor with the Computer Science and Engineering Department, Indian Institute of Technology Delhi, New Delhi, India, where he holds a joint appointment with the Department of ELectrical Engineering and the School of Information Technology. He has published extensively in peer reviewed conferences and journals, holds 5 U.S. patents, and has filed 3 Indian patents. He has authored the popular undergraduate textbook on computer architecture entitled Computer Organisation and Architecture (McGraw Hill). His current research interests include processor reliability, architectural support for operating systems, and processors for the Internet of Things. Dr. Sarangi is a member of the ACM and IEEE.