ENERGY-NEUTRAL DATA DELIVERY IN ENVIRONMENTALLY-POWERED WIRELESS SENSOR NETWORKS

ALVIN CERDENA VALERA

(M.Sc., NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2015

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

[illegible]

ALVIN CERDENA VALERA

25 May 2015

Acknowledgments

This thesis is a culmination of several years of hard work that would not be possible without the guidance, help, support, and inspiration from many people. I am using this opportunity to express my appreciation to everyone who supported me throughout the course of my studies.

I wish to extend my gratitude to my supervisors, *Prof. Wee Seng Soh* and *Dr. Hwee Pink Tan*, for their encouragement, ideas, and support in bringing this work to completion. I would like to thank Prof. Soh for convincing me to pursue the doctorate degree, for helping me in the application process, for taking time to review my work and providing valuable feedback, and most of all, for giving me direction when I felt lost.

I would like to express my appreciation to Dr. Tan, who is also my reporting officer at the Institute for Infocomm Research (I^{2}R), for allowing me to pursue this degree in spite of the known challenges between balancing work and studies, for always having time to read my manuscripts and providing constructive criticisms that greatly improved my work, and most of all, for his support and encouragement in times of difficulties.

I am grateful to all of my friends and colleagues at I^{2}R, for their help and ideas throughout the course of my work. I would like to specifically mention *Hwee Xian*, *Yunye*, *Huiguang*, *Xiaoping*, *Brian*, *Wai Leong*, *Shaowei*, *Ido*, *Pengfei*, and the *Sense and Sense-abilities* team, who in one way or the other, helped me in my studies by attending my seminars, lightening my workload during examinations and paper submission deadlines, and for giving me all the friendly advice about Ph.D. and life in general.

I would like to thank my son, *Albert*, for always including my studies in his prayers, and for always being a good son in spite of my shortcomings. I will be forever grateful and thankful to my loving and understanding wife, *Karen*, for permitting me to embark on this difficult journey knowing that our other plans would be temporarily put on hold, for giving me time to work at home and even during our holidays, for staying up late at night with me during deadlines, for encouraging and pushing to move on whenever I encounter difficult roadblocks, and most of all, for for being there to comfort me in times of trouble.

Most especially, I thank the Lord Almighty for continuously and generously supplying me with the needed wisdom and perseverance to bring this work to full fruition.

Contents

List of Figures

List of Tables

List of Symbols

u, v, w	Variables to denote arbitrary nodes
$\mathbf{N}_v$	Single-hop neighbors of v
$\mathbf{P}_v$	Predecessor nodes of v for a given routing graph
S_{vt}	Successor node at v to sink node t
k	Variable used to denote epoch or cycle
T	Duration of a epoch
$\delta_v(k)$	Target duty cycle of v at epoch k
$\hat{\delta}_v(k)$	Measured duty cycle usage of v at epoch k
(u, v)	Link from u to v
P_{st}	Path from s to t
p_{uv}	Packet delivery rate of link (u, v)
W_{uv}	Sleep latency from u to v
$\mathsf{Pr}(A)$	Probability of A
$\mathsf{E}(X)$	Expectation of X
$\mathsf{Var}(X)$	Variance of X
$\mathcal{U}(a, b)$	Uniform distribution in $[a, b]$
S	Number of slots in an epoch (in synchronized schemes)
$\tau = \frac{T}{S}$	Slot duration (in synchronized schemes)
$n_v(k)$	Number of active slots in an epoch (in synchronized schemes)
$r_v(k)$	Number of receive slots in an epoch (in synchronized schemes)
$t_v(k)$	Number of transmit slots in an epoch (in synchronized schemes)
$\mathbf{\Gamma}_v(k)$	Wakeup schedule of v in an epoch (in synchronized schemes)

Summary

The use of *energy harvesting* to recharge energy stores can enable *environmentally-powered wireless sensor networks* (EPWSNs) to operate perpetually without the need for battery replacement which is not only laborious or expensive but outright infeasible in certain scenarios. Notwithstanding this advantage, EPWSNs face a difficult challenge: the amount of energy available for consumption can be unpredictable and variable over time. Thus, unlike battery-powered wireless sensor networks that aim to conserve energy, the key objective in EPWSNs is to attain *energy-neutrality*, a trajectory where the energy demand and supply are always balanced. In this thesis, we have developed schemes to enable energy-neutral data delivery in EPWSNs, addressing the problems posed by energy-neutrality including dynamic wakeup scheduling, low latency and reliable path selection, dynamic duty cycle allocation, and sustainable bulk transfer.

We have shown that the expected sleep latency of a dynamic wakeup schedule is affected by the variance of the intervals between receive wakeup slots, *i.e.*, when the variance of the intervals is low (high), the expected latency is low (high). We designed a scheduling scheme using the *bit-reversal permutation sequence* (BRPS), with worst-case sleep latency slightly worse than the ideal scheme but is robust to duty cycle changes than the latter. BRPS has a lower computational complexity compared to Energy Synchronized Communication (ESC), a state-of-the-art scheme for EPWSNs, but can closely match the latter's latency performance and exceed its packet delivery ratio.

To enable the selection of reliable and low latency paths, we have formulated *expected transmission delay* (ETD), a metric which simultaneously considers sleep latency and packet loss. ETD is left-monotonic and left-isotonic and is therefore guaranteed to yield consistent, loop-free and optimal paths. Simulations show that compared with hop count and the state-of-the-art routing metric Expected Transmission Count (ETX), ETD provides the best performance in terms of packet delivery ratio and end-to-end delay.

In receive-centric wakeup scheduling schemes, the wakeup slots are meant solely for reception. Hence, nodes that need to relay packets must address the *duty cycle allocation problem*: how to apportion the duty cycle between packet reception and transmission. Using the packet arrival probability and the expected service time models that we have derived, we have formulated the duty cycle allocation problem as a constrained non-linear optimization problem that seeks to minimize the two-hop service time. We have developed LSLOTALLOC, a distributed low-complexity algorithm that uses linear search to find an optimal solution to the problem. Trace-driven simulations show the significant performance gain of LSLOTALLOC over a static allocation scheme in terms of end-to-end delay.

Finally, we have introduced PUMP-AND-NAP, a packet train forwarding technique for bulk transfer, that maximizes throughput while simultaneously enforcing compliance to (dynamic) duty cycle limitations. PUMP-AND-NAP uses an *adaptive controller* to periodically compute the maximum number of packets a node can receive and transmit in a train, given its duty cycle constraint. Experimental results show that PUMP-AND-NAP enables sustainable bulk transfer at high throughput compared to state-of-the-art schemes that greedily maximize throughput at the expense of downtime due to energy depletion.

Chapter 1

Introduction

Wireless sensor networks (WSNs) are becoming ubiquitous because of their diverse applications in areas such as agriculture, environmental monitoring, industrial and home automation, military, and structural health monitoring, to name a few [12]. A critical issue that plagues many deployments, however, is the *limited lifetime problem* due to the finite battery capacity of sensor nodes [107, 110]. Fortunately, advances in energy harvesting and storage technologies are enabling the deployment of *environmentally-powered wireless sensor networks* (EPWSNs), wherein the sensor nodes harvest energy from the environment to recharge their batteries or energy stores [65, 94, 107, 110]. By powering nodes with renewable energy, EPWSNs can operate perpetually without the need for battery replacement which is not only laborious or expensive but also infeasible in certain scenarios.

A typical EPWSN deployment consists of a set of sensor nodes placed around a particular region of interest. A gateway node links up the EPWSN to a backend server through a backhaul network. This setup is illustrated in Figure 1.1. Similar to battery-powered WSNs, the operation of EPWSNs can be decomposed into three main parts: (*i*) *sensing* of physical phenomenon, including analog-to-digital conversion and in some cases, signal processing; (*ii*) *storage* of digitized readings in a local storage such as the sensor node's random access memory or flash memory; and finally (*iii*) *transmission* of readings to a gateway node, for eventual transmission to the backend server for further analysis and storage. In this thesis, we focus

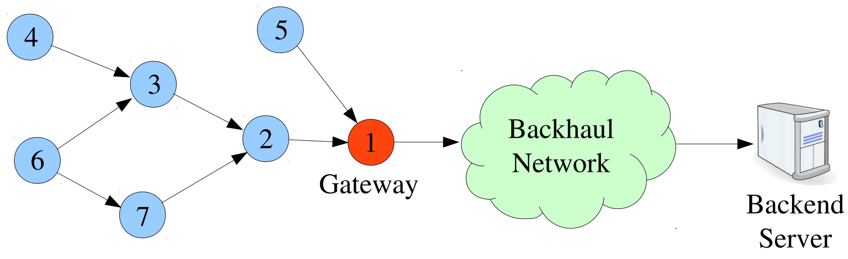


Figure 1.1: An example EPWSN deployment, with node 1 as the gateway and nodes 2–7 acting as sensor nodes. The gateway links up the sensor network to the backend server through the backhaul network. This thesis focuses on the delivery or transfer of sensor readings from the sensor nodes to the gateway.

on the third aspect, that is, the delivery of sensor data from every sensor node in the network to a gateway node.

1.1 Energy Harvesting in Sensor Networks

To identify and understand the relevant issues that can affect the performance of data delivery in EPWSNs, we briefly study energy harvesting and how it is utilized in the context of wireless sensor networks. We describe the various components needed to assemble an environmentally-powered wireless sensor node, followed by a presentation of the current state in energy harvesting and storage technologies.

1.1.1 Energy Harvesting

Energy harvesting, also referred to as "*energy scavenging*" in the literature, is the process of converting ambient energy from the environment into electrical energy to power devices such as sensor nodes and mobile electronics [94]. Figure 1.2 shows the various components of an EPWSN node: (*i*) *energy harvester* for converting ambient energy to electrical energy; (*ii*) *energy storage* for storing harvested energy; and (*iii*) *sensor load* which essentially consists of the sensor node electronics (mainboard, microcontroller, radio, sensors and other peripherals). Because ambient energy is readily available, energy harvesting could enable perpetual operation without the need for battery replacement [107, 110].

There are numerous sources of ambient energy and they can be grouped into

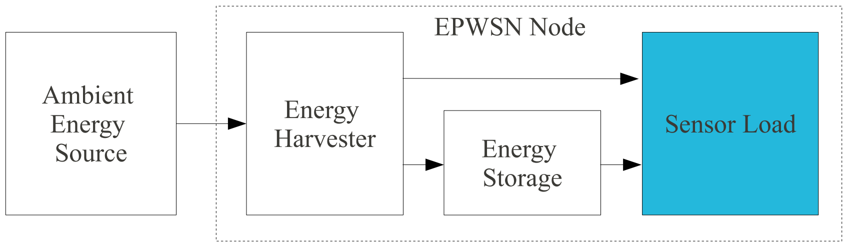


Figure 1.2: Components of an EPWSN node.

several classes according to their underlying physical process [94]:

- **Mechanical:** from sources such as wind, vibration, mechanical stress and strain and human body movement;
- **Light:** from sunlight or room (artificial) light;
- **Thermal:** waste energy from engines, furnaces, heaters and friction sources;
- **Electromagnetic:** from inductors, coils, transformers and radio frequency sources; and
- **Others:** from chemical and biological sources.

The conversion of ambient energy to electrical energy requires the use of an energy harvester or transducer. Table 1.1 provides a summary of achievable energy harvesting rates of several state-of-the-art energy harvesting technologies [56, 74, 94, 110]. Solar energy, which is one of the most abundant and readily available energy, can be harvested using photo-voltaic (PV) cells which can have 25% efficiency [74]. When such a PV cell is directly exposed to sunlight which has an irradiance of 1000 W/m^2 (this is a typical value of direct solar irradiance [3]), it can potentially generate 250 W/m^2 or 25 mW/cm^2.

1.1.2 Energy Storage Technologies

Energy storage or buffer is an important component of an EPWSN node. It serves two important functions [68]: (*i*) to act as storage for unused or excess harvested energy; and (*ii*) to act as additional energy supply when load consumption is not met by harvested energy. It is possible to power a sensor node directly from an

Table 1.1: Efficiency of State-of-the-Art Energy Harvesting Technologies

Energy source	Harvesting device / transducer	Efficiency	Achievable harvesting rate
Solar	Photo-voltaic cells	25%	25 mW/cm^2
Indoor light	Photo-voltaic cells	25%	25 μW/cm^2
Thermal	Thermoelectric generator	-	60 μW/cm^2
Wind	Anemometer	-	1,200 mWh/day
Electromagnetic	RF antenna	-	10^{-5} – 0.1 mW/cm^2
Indoor vibrations	EM induction	-	0.2 mW/cm^2
Vibrations (walking)	EM generator	-	0.95 mW
Vibrations (running)	EM generator	-	2.46 mW
Heel strike	Piezoelectric	7.5%	5 W

energy harvester without any energy buffer but its operation will be severely constrained. In particular, such a node can only operate when the amount of harvested power is greater than or equal to the required node consumption. When the amount of harvested power is not sufficient, the node will not operate and the harvested power will be wasted. In cases where the amount of harvested power exceeds the node consumption, the excess will likewise be wasted.

Currently, there are two dominant energy storage technologies that can be utilized in EPWSN [65, 68, 74, 109, 110, 137]: (*i*) secondary or rechargeable batteries; and (*ii*) supercapacitors, also known as ultracapacitors or electrochemical double layer capacitors. Although there are many types of rechargeable batteries available in the market, nickel metal hydride (NiMH) and lithium ion (Li-ion) are considered to be more suitable for sensor nodes [109, 110].

As far as EPWSNs are concerned, the most important characteristics of an energy storage technology are energy storage capacity, number of full recharge cycles, and self-discharge rate or leakage. Table 1.2 provides a comparison of several energy storage devices in terms of the three characteristics [109]. In gen-

Table 1.2: Comparison of Energy Storage Devices

Device	Type	Capacity (mAh)	Recharge cycles	Self-discharge rate
Maxwell BCAP350	350F super-capacitor	243	500,000	<30%/month
Maxwell PC10	10F super-capacitor	6.9	500,000	<30%/month
Panasonic HHR210AA/B	NiMH	2,000	300	<30%/month
Panasonic CGR17500	Li-ion	830	500	<10%/month

eral, rechargeable batteries provide high energy capacity while supercapacitors can provide low to moderate energy capacity. In terms of self-discharge rate, Li-ion batteries are slightly better than supercapacitors. One major advantage of supercapacitors is the number of full recharge cycles which is three orders of magnitude higher than that of rechargeable batteries. This has significant impact on the lifetime of the storage device, enabling supercapacitors to last for 10-20 years compared to a maximum of 5 and 3 years for Li-ion and NiMH, respectively [109].

1.2 Energy-Neutrality and Its Challenges

As enumerated by Akyildiz, *et al.* [12], sensor networks face numerous challenges including highly dynamic network topology due to failure-prone nodes and wireless links, limited memory and processing power and most importantly, limited network lifetime due to battery capacity limitations. Energy harvesting has the potential to eliminate the problem of limited network lifetime but it poses a major constraint on the amount and consistency of energy that can be supplied to the sensor node. Unlike a battery-powered WSN node where the energy supply is guaranteed (while its battery is not exhausted), the energy supply of an EPWSN node can be unpredictable and varies over time [51,61,68].

In battery-powered WSNs, network protocols are designed to conserve as much energy as possible, knowing that the energy supply is finite and will eventually be depleted. Network lifetime can be maximized by minimizing the energy consumption of individual nodes while at the same time balancing the energy consumption across nodes [15]. In EPWSNs where the energy supply can be replenished, the notion of network lifetime is inappropriate and this renders energy conservation as an unsuitable design objective.

1.2.1 Definition of Energy-Neutrality

The new guiding principle in the design of EPWSN protocols is *energy-neutrality* or *energy neutral operation* which consists of two simultaneous goals: (*i*) optimizing the network performance while (*ii*) ensuring that energy supply and energy demand are balanced [51, 61, 68, 120, 137]. Several authors proposed essentially the same idea using different terms, namely, "energy-neutrality" [68], "energy-synchronized" [51], and "energetic sustainability" [80]. The key intuition behind energy-neutrality is that by letting a node's energy consumption to be equal to the *sustainable* energy supply, then it will never run out of energy and will therefore operate perpetually.

1.2.2 Dynamic Duty Cycling

To achieve energy neutral operation in the face of dynamic energy availability, adaptive duty cycling algorithms have been proposed [61, 68, 120, 137]. Basically, these algorithms aim to dynamically adjust a node's duty cycle given its current energy level, energy buffer capacity as well as current and future (predicted) harvesting rates. Duty cycling is not new and has been proposed as an energy conservation method in battery-powered WSNs because radio transceivers consume significant amounts of energy even when idle [50,52,72,83,133]. Duty cycling itself poses difficulties in the operation of networking protocols, and these are exacerbated with the addition of stochasticity.

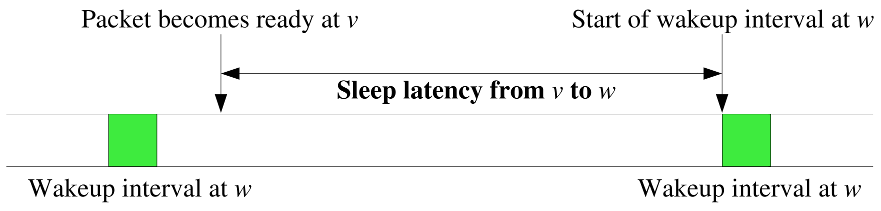

Packet becomes ready at v
Start of wakeup interval at w
Sleep latency from v to w
Wakeup interval at w
Wakeup interval at w

Figure 1.3: Sleep latency from a transmitting node v to a receiving node w. The latency is incurred because v must wait for w to be awake before it can transmit its packet that became ready earlier.

1.2.3 Dynamic Sleep Latency

Another challenge that directly impacts data delivery is *sleep latency*, a delay incurred due to the fact that a transmitting node must wait for the receiving node to be awake before it can commence packet transmission [50,83,130]. Figure 1.3 illustrates the sleep latency from a transmitting node v to a receiving node w. In either battery-powered WSNs and EPWSNs, sleep latency is a significant factor that contributes to the high end-to-end delay in these deployments [50,51]. As it is mostly determined by the duty cycle, in the context of EPWSNs which employ dynamic duty cycling, sleep latency is also time-varying and is therefore more challenging to address.

1.3 Problem Statement

The general objective of this thesis is to enable energy-neutral data delivery in EPWSNs. More precisely, the thesis aims to enable the transmission of data from the sensor nodes to a gateway, possibly through a multihop topology, while ensuring that energy-neutrality constraints are satisfied. To achieve this objective, we need to address the following problems:

Dynamic Wakeup Scheduling Duty cycling necessitates the use of a wakeup schedule, indicating the times at which a node wakes up to listen for transmissions from its neighbors. A key implication of dynamic duty cycling is that wakeup schedules must also be dynamic. Most importantly, it must address the challenge posed by dynamic sleep latency. While numerous wakeup scheduling schemes have been proposed for fixed duty cycling networks (*e.g.*, [133], [83], [72], [52]), few have been proposed for dynamic duty cycling networks [51]. A major limitation of the scheme proposed in [51] is the requirement of a fixed routing graph, thereby limiting its application to conventional single path routing schemes. In addition, it also requires high memory and communication overhead.

Low Latency and Reliable Path Selection When a path needs to traverse multiple hops, determining the path that provides the least delay can be difficult because the nodes may have different instantaneous duty cycles and therefore pose different sleep latencies. Existing metrics (*e.g.*, hop count, ETX [31]) do not consider sleep latency even though it is known to be a significant factor in the high end-to-end latency in EPWSNs. An equally important criteria in path selection is reliability which is mainly determined by the quality of the wireless links along a path. We therefore need to formulate a metric that simultaneously considers sleep latency and link quality to enable the selection of low latency and high reliability end-to-end paths.

Dynamic Duty Cycle Allocation The duty cycle of a node denotes the fraction of time that it can be active for data packet transmission and reception. Because duty cycles are dynamic, we expect that a static allocation of receive and transmit duty cycle will not provide the best performance. For instance, a node which has many backlog data packets may choose to allocate higher duty cycle for transmission than for reception. We therefore need to formulate a dynamic allocation scheme that can optimally apportion the duty cycle between packet reception and packet transmission.

Sustainable Bulk Transfer In many applications (*e.g.* [24,124]), sensor nodes are tasked to record time-series data at high sampling rates, resulting in large or bulk sensor data. Bulk transfer essentially refers to the delivery of bulk sensor data from sensor nodes to a gateway. While several bulk transfer schemes have been proposed [39, 40, 75, 100], they focus mainly on maximizing the throughput, neglecting the duty cycle constraints of sensor nodes. The use of existing schemes may therefore cause uncontrolled and rapid draining of the energy reserves, leading to the temporary unavailability of nodes along the transfer path. Ultimately, this will result in transfer disruptions which render the transfer of arbitrarily-sized sensor data difficult, if not infeasible.

1.4 Accomplishments and Contributions

This thesis aims to develop data delivery schemes for EPWSNs that can achieve performance requirements on reliability, end-to-end delay and throughput while ensuring that energy-neutrality constraints are satisfied. To this end, this thesis makes the following major findings and contributions:

- We have shown analytically that the expected sleep latency of a wakeup scheduling scheme is related to the variance of the intervals between receive wakeup slots. In particular, when the variance of the interval is low (high), the expected latency is low (high). Hence, the ideal scheduling scheme is the one where the receive wakeup slots are positioned at equal intervals since its variance is 0. We have designed a sequence-based scheduling scheme that uses bit-reversal permutation sequence (BRPS) and analytically obtained its worst-case sleep latency which is slightly worse than the ideal scheme but better than schemes where the intervals between receive wakeup slots are distributed uniformly or exponentially. BRPS entails low storage and communications overhead as the bit-reversal permutation sequence provides a compact representation of dynamic wakeup schedules. Simulation results show that BRPS provides low latency and can closely match the performance of Energy Synchronized Communication (ESC) [51], a state-of-the-art scheduling scheme for EPWSNs. Furthermore, BRPS's robustness results in lower scheduling error ratio which translates to better packet delivery ratio. Aside from having a lower storage and communication overhead, BRPS also has a lower computational complexity compared with ESC.

- To enable the selection of low latency and high reliability paths among dynamically duty cycled nodes, we have formulated a metric called expected transmission delay (ETD) which simultaneously considers sleep latency (due to duty cycling) and packet loss. We have proven that the metric is left-monotonic and left-isotonic, guaranteeing that its use in distributed algorithms such as the distributed Bellman-Ford will yield consistent, loop-free and optimal paths. Simulations show that compared with ETX and hop count and used in tandem with BRPS, ETD provides the best performance in terms of packet delivery ratio and delay.

- In the context of *receive-centric wakeup scheduling schemes*, there is a need to apportion the duty cycle between packet reception and transmission, which we refer to as the *duty cycle allocation problem*. We have derived analytical models for the packet arrival probability and the expected service time in the presence of contention. Using these models, we have formulated the duty cycle allocation problem as a constrained non-linear optimization problem that seeks to minimize the two-hop service time. We have developed LSLOTALLOC, a distributed low-complexity algorithm that uses linear search to find an optimal solution. Trace-driven simulation results show the significant performance gain of LSLOTALLOC over a static allocation scheme in terms of end-to-end delay.

- To address the problem of transferring bulk data in EPWSNs, we have proposed PUMP-AND-NAP, a packet train forwarding technique that maximizes throughput while simultaneously enforcing compliance to (dynamic) duty cycle limitations. PUMP-AND-NAP employs an *adaptive controller* to periodically compute the *optimal capacity*, that is, the maximum number of packets a node can receive and transmit in a train, given its duty cycle constraint. The controller uses prior input-output observations (capacity allocations and their corresponding duty cycle usage) to continuously tune its performance and adapt to wireless link quality variations. Its use of local information makes the controller easily deployable in a distributed fashion.

We have implemented PUMP-AND-NAP in TinyOS and evaluated its performance through experiments. Results show that PUMP-AND-NAP provides high transfer throughput while it simultaneously tracks the target duty cycle. More importantly, PUMP-AND-NAP enables sustainable bulk transfer compared to state-of-the-art techniques that greedily maximize throughput at the expense of downtime due to energy depletion.

1.5 Structure of the Thesis

This thesis begins with a survey of related literature in Chapter 2. The review covers the state-of-the-art in wakeup scheduling schemes, routing metrics, and bulk transfer schemes for wireless sensor networks. Our understanding of the advantages and disadvantages of these existing schemes will be useful in improving their performance.

In Chapter 3, an energy-neutral wakeup scheduling and forwarding scheme that addresses the deficiencies of existing schemes is presented in detail. A key finding on the effect of the wakeup schedule on the expected sleep latency is proven. Mathematical properties of the proposed wakeup scheduling scheme that uses BRPS are established, while monotonicity and isotonicity proofs of the ETD forwarding metric are shown in detail. Simulations are conducted to compare the performance of BRPS with ESC, and ETD with hop count and ETX.

In Chapter 4, the duty cycle allocation problem is introduced. Using discrete-time queueing theory and renewal theory, the packet arrival probability and the expected service time in the presence of contention are derived. LSLOTALLOC, a simple algorithm that solves the duty cycle allocation problem using the derived analytical models, is presented and its desirable mathematical properties are proven. Trace-driven simulations are then conducted to validate the packet arrival probability model and compare the performance of LSLOTALLOC with the best-performing static allocation scheme.

In Chapter 5, a packet-train forwarding technique for bulk data transfer is introduced. The design of the scheme, known as PUMP-AND-NAP, uses the principle of certainty equivalent to come up with an adaptive feedback control mechanism. Experimental results characterizing the performance of PUMP-AND-NAP are presented and discussed. Experiments involving a real energy-harvesting node are also conducted, comparing the performance of PUMP-AND-NAP to that of existing bulk transfer techniques.

Finally, in Chapter 6, the thesis concludes with a summary of the various accomplishments and contributions of this study. We also state several possible future research in the area of energy-neutral data delivery in EPWSNs.

Chapter 2

Review of Related Literature

Energy neutral operation entails dynamic duty cycling and dynamic sleep latency which pose difficult challenges on the design of data delivery schemes. In this chapter, we present a survey of related work, focusing on the problems enumerated in Section 1.3. In particular, we conduct thorough assessment, qualitative analysis and comparison of state-of-the-art wakeup scheduling schemes in Section 2.1, routing or forwarding metrics in Section 2.2 and bulk transfer schemes in Section 2.3. Our main objective in this chapter is to determine the suitability of these schemes in the context of EPWSNs, and pinpoint their weaknesses and strengths.

2.1 Wakeup Scheduling

At the most fundamental level, wakeup scheduling schemes can be classified based on their requirement for synchronization, *i.e.*, *synchronous* or *asynchronous*. As the name implies, synchronous scheduling schemes require that the time across nodes are synchronized. On the other hand, asynchronous scheduling schemes do not require any form of synchronization. Note that this section is an abridged version of our survey paper [118].

2.1.1 Considerations

Wakeup scheduling employed in EPWSNs must consider their unique characteristics and the underlying challenges posed by environmentally harvested energy supply. In the discussion of the various schemes, we examine how they measure up against the following important considerations:

Adaptation to Environment Dynamics EPWSNs are highly dynamic in terms of topology, energy supply, and data traffic among others [12]. Schemes that respond to some or all of these dynamics are expected to perform better than those schemes that are oblivious. But while some non-adaptive schemes can be easily modified to respond to changes in the operating environment, others are not flexible and are therefore not amenable for use in dynamic environments.

Latency-Aware Sleep latency is a delay incurred in duty cycling networks due to the fact that a transmitting node must wait for the receiving node to wakeup before it can commence packet transmission [50, 83, 130]. Sleep latency is a major challenge in both battery-powered and environmentally-powered WSNs and significantly contributes to the end-to-end delay [50, 51]. Schemes that explicitly tackle latency perform better than latency-oblivious schemes, but the awareness again comes at an additional cost.

Duty Cycle Range Certain WSNs operate in very low duty cycles and as such, schemes that are designed with high duty cycle in mind may not work well in these regimes. For instance, schemes that rely on random schedules may have poor performance because the probability of the sender and receiver being awake at the same time is low. On the other hand, scheduling schemes tailored for low duty cycles may have poor performance in high duty cycle regimes.

Processing Complexity Because EPWSN nodes have limited processing capability [12], schemes that use complex algorithms may not be suitable. Their use may require the deployment of special nodes with sufficient processing capability to perform the complex computations and this entails some form of centralized processing.

Overhead and Scalability Finally, constraints on channel and storage capacities [12] imply that schemes must have low communication and storage overhead. Sources of overhead are mainly schedule exchange and storage. Notice that high overhead may imply that a scheme is not scalable in terms of the number of nodes.

2.1.2 Terminology

We put forth the following basic terms to avoid confusion in the discussions.

Definition 1 (Wakeup Interval)**.** *The time duration at which the radio is **switched on** to enable the node to either receive or transmit packets. The literature sometimes refer to this as **active** or **on interval**.*

Definition 2 (Sleep Interval)**.** *The time duration at which the radio is **switched off** to enable the node to conserve energy. The literature sometimes refer to this as **inactive**, **off** or **dormant interval**.*

Wakeup scheduling schemes can be broadly grouped into two types: those that divide time into equal-length intervals called *slots* and those that treat time as continuous. The above definitions are usually applied to unslotted schemes. For *slotted* schemes, wakeup and sleep intervals are defined in terms of integer number of slots.

Definition 3 (Wakeup Schedule)**.** *A sequence of wakeup and sleep intervals that is usually specified for one cycle and repeats every cycle until otherwise modified by the wakeup scheduling scheme. This is sometimes referred to as **sleep schedule** or **sleep/wakeup schedule** in the literature.*

2.1.3 Data Exchange

The ultimate aim of wakeup scheduling schemes is to enable nodes to exchange data during wakeup intervals. Most wakeup scheduling schemes use a simple design wherein every wakeup interval can accommodate at most one data frame or packet. Several schemes have been proposed that can handle multiple data frames or packets in every wakeup interval. In the former, wakeup schedules are considered to be *receive-centric*, *i.e.*, the specified wakeup intervals in the schedule are meant for packet reception only[1]. In the latter, wakeup intervals are considered *bi-directional*, *i.e.*, wakeup intervals can be used for both packet transmission and reception.

2.1.4 Asynchronous Schemes

Asynchronous schemes were the earliest protocols proposed for wakeup scheduling. Their main distinguishing feature is that they operate in an asynchronous manner, meaning that nodes wakeup to transmit without regard on whether other nodes are awake to receive. Because of this, asynchronous schemes do not require time synchronization. This is one of its major advantages because as Wu, *et al.* [127] found in their study, efforts to periodically re-synchronize time across nodes can entail significant energy consumption. Another major advantage of asynchronous approaches is that they do not require any computation as well as communication and storage overhead since no schedules are exchanged and stored.

One of the main challenges of asynchronous scheduling is how to exchange data between two nodes which are not aware of each other's wakeup schedules. There are two major possible approaches to do this: (*i*) *transmitter-initiated*; and (*ii*) *receiver-initiated*.

Transmitter-Initiated

In transmitter-initiated protocols, a transmitting node v transmits a special frame to indicate to its neighbor nodes that it has data to transmit. When a neighbor node hears the special frame in one of its wakeup intervals, it awaits for the transmission of the data frame. A wakeup interval is receive-centric and can accommodate at

[1]In receive-centric wakeup scheduling schemes, a node u with data to transmit to v must wakeup at an interval where v is awake as specified by the latter's wakeup schedule. Note that this interval at which u wakes up to transmit its packet is not considered part of its wakeup schedule.

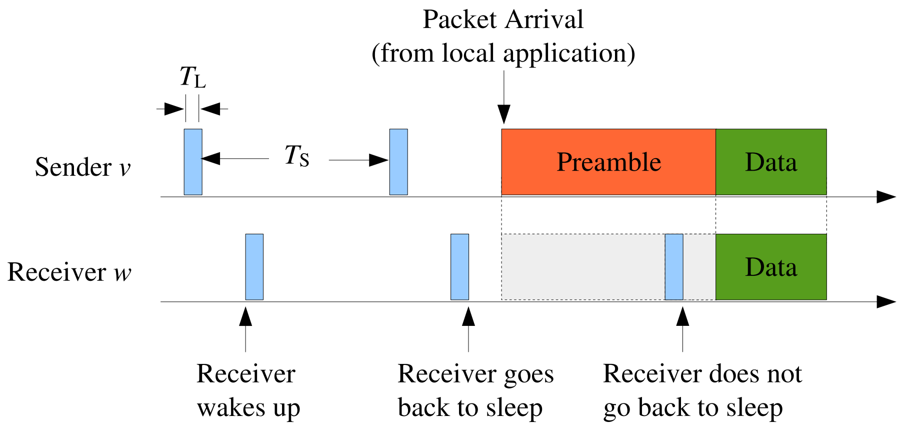


Figure 2.1: B-MAC operation. Nodes independently sleep and wakeup periodically but with the same wakeup and sleep durations. When node v needs to send data (a packet arrives from local application/higher layer), it must first send a preamble which should be at least as long as the sleep duration T_S (the *check interval*). When receiver node w detects the preamble, it remains awake to receive the preamble and data. If a node wakes up and does not detect a preamble within the wakeup duration T_L, it goes back to sleep.

most one data frame. (It is possible for a wakeup interval to accommodate more than one data frame depending on the data frame duration or the wakeup interval duration. However, the use of preamble acts as a reservation mechanism whereby only one node has the right to transmit one or more data frames within the wakeup interval.)

B-MAC The first protocol to use this approach is the B-MAC [98] protocol. In this protocol, nodes periodically wakeup for a duration of T_L and sleep for a duration of T_S. T_L is specified to be long enough for a node to detect the presence of a special signal known as *preamble*. When a node v has data to send, it immediately wakes up and transmits a preamble frame for a duration of T_S followed by the data frame. A node w that wakes up and detects the preamble will then remain awake for the remaining preamble duration until it receives the data frame. This process is shown in Figure 2.1.

B-MAC Enhancements B-MAC suffers from two major drawbacks. Firstly, sending nodes must transmit a long preamble which must be at least T_S, and secondly, overhearing nodes (*i.e.*, not the intended receiver) will also have to be awake during the entire preamble transmission and possibly until data transmission is completed. Several enhancements have been proposed to address these deficiencies. X-MAC [21] tackles the long preamble problem by replacing it with a *strobe short preamble*. In addition, the strobe preamble includes the intended receiver address, thereby allowing overhearing nodes to go back to sleep the moment they receive a strobe. BoX-MAC [87] further improves on X-MAC by replacing the short preamble transmissions with data transmissions. This however assumes that data packets are short enough to be effective replacements of strobe preamble.

Receiver-Initiated

Receiver-initiated protocols essentially pass the burden of energy consumption for the overhead from transmitters to receivers. That is, a receiving node w transmits a special frame every time it wakes up to indicate to potential transmitters that it is ready to receive data frames. When a node v has pending data to transmit, it immediately wakes up and awaits for the transmission of the special frame from its neighbors. The moment it receives the special frame from another node w for which it has data to transmit to, v commences data transmission to w. Similar to the transmitter-initiated protocols, a wakeup interval in receiver-initiated protocols is receive-centric and can accommodate at most one data frame.

RI-MAC Nodes periodically sleep for a duration of T_{S} and wakeup for a duration of T_{L}. Whenever a node wakes up, it transmits a *beacon* to indicate to potential transmitters that it is ready to receive data. If it does not receive a data frame after T_{L}, it goes back to sleep. From a transmitter perspective, if a node v has data to transmit, it waits for the beacon from the intended receiver before transmitting its data. Figure 2.2 shows the operation of RI-MAC [112], the protocol that first proposed this approach.

RI-MAC Enhancements Huang, *et al*. [62] proposed Receiver-Centric MAC (RC-MAC) that exploits the underlying routing tree structure to coordinate the transmission of a node's children. The coordination is done by piggybacking the ID of the next child that can transmit in the ACK. Meanwhile, Nguyen, *et al*. [90] ex-

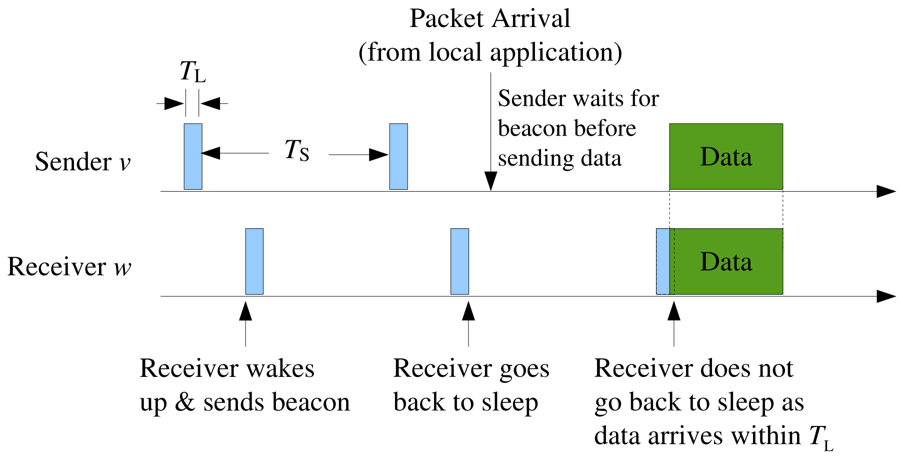

Packet Arrival
(from local application)
T_L
T_S
Sender v
Sender waits for beacon before sending data
Data
Receiver w
Data
Receiver wakes up & sends beacon
Receiver goes back to sleep
Receiver does not go back to sleep as data arrives within T_L

Figure 2.2: RI-MAC operation. Nodes independently sleep and wakeup periodically but with the same wakeup and sleep durations. Whenever a node wakes up, it transmits a beacon frame and stays awake for a duration of T_L. It goes back to sleep if it does not start to receive any data within T_L. If node v has data to send, it waits for the beacon before transmitting its data.

tended RI-MAC for EPWSNs by using the energy harvesting rate and the queue length to adjust the duty cycle of the nodes.

2.1.5 Periodic Synchronous Schemes

Numerous synchronous schemes have been proposed because the use of time synchronization somehow eases up the analysis and design of wakeup scheduling schemes. Time synchronization in the context of wireless sensor networks is a well-studied area and numerous protocols have been proposed for this purpose [41, 42, 85, 93]. Most synchronous schemes operate *periodically*, *i.e.*, a wakeup schedule repeats every *period*, *cycle* or *epoch* until a change is made by the scheduling algorithm. We present several of these schemes in this section. There are however *aperiodic* schemes that do not employ periodic schedules, and we will elaborate on them in Section 2.1.6.

Periodic wakeup scheduling schemes may operate either in a *slotted* or *unslotted* manner. In the former, the cycle is essentially broken up into equal-length slots as shown in Figure 2.3. Slotted cycles are usually employed by schemes that use more than one wakeup instance per cycle.

The main problem in periodic wakeup scheduling is to determine which time

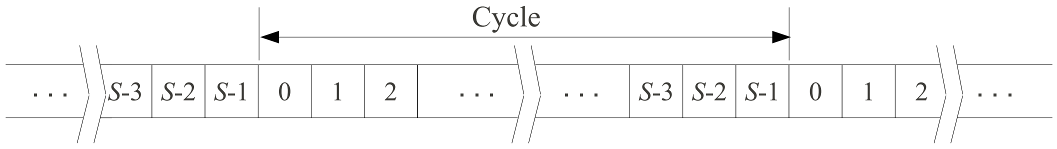


Figure 2.3: A cycle with S slots.

interval (or intervals) in a cycle to activate so that a node can perform packet transmission or reception. This problem may look trivial from a node-level perspective but from a network-level point-of-view, selecting intervals across nodes to optimize a certain performance metric can be difficult. To achieve a desired performance, nodes must *collaborate* with each other in the process of schedule computation. We can divide the various schemes into five groups depending on the level of collaboration: (*i*) *neighbor-coordinated*; (*ii*) *path-coordinated*; (*iii*) *network-coordinated*; (*iv*) *independent*; and (*v*) *centralized*. The latter two are actually non-collaborative approaches.

Neighbor-Coordinated

In this approach, a node establishes its own wakeup schedule by considering the wakeup schedules of its adjacent or neighbor nodes. To be precise, a node v calculates its wakeup schedule by consulting the schedule of all nodes $w \in \mathbf{N}_v$, where $\mathbf{N}_v$ is the set of one-hop neighbors of v. This is obviously the easiest among the collaborative schemes and requires the least effort.

The neighbor-coordinated schemes that we will be discussing in this survey are receive-centric and every wakeup interval or slot can accommodate at most one data packet.

S-MAC The simplest (and the first to be introduced) protocol that uses this approach is S-MAC [129]. As shown in Figure 2.4, the wakeup interval is divided into three parts: (*i*) a portion for SYNC, (*ii*) a portion for RTS, and (*iii*) a portion for CTS. S-MAC uses *scheduled rendezvous* communication scheme wherein nodes exchange SYNC packets (in the first portion of the wakeup interval) to coordinate sleep/wakeup periods. Before a node can send a SYNC packet, it must ensure

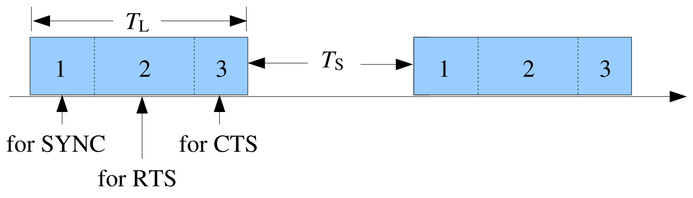


Figure 2.4: Components of an S-MAC wakeup interval.

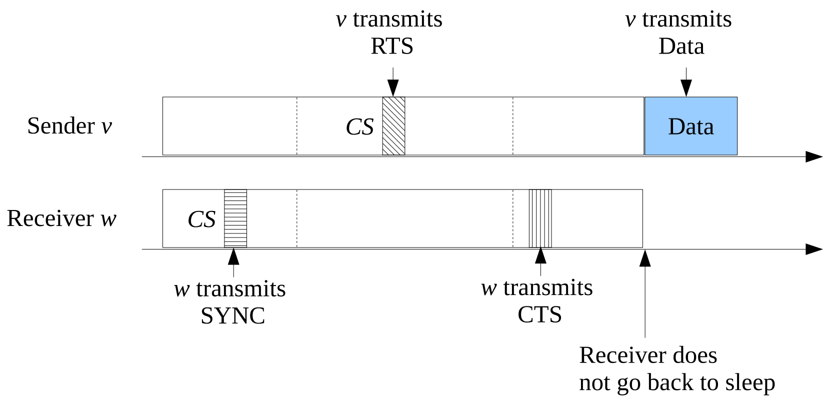


Figure 2.5: S-MAC protocol operation. In the first portion, w performs carrier sensing (denoted by CS) before sending a SYNC packet. In the second portion, v sends an RTS after performing carrier sensing. In the third portion, w sends back a CTS to v. After the regular wakeup interval, v transmits its data while w continues to be awake to receive the data.

that the channel is idle by performing carrier sensing. A node can either create its own schedule or follow a neighbor's schedule. When a node v has data to send to w, it performs carrier sensing and if the channel is idle, v sends an RTS to w in the second portion of the wakeup interval. If v detects the channel to be busy, it goes back to sleep. Node w sends back an RTS in the third portion of the wakeup interval. Data transmission occurs after the third portion of the wakeup interval. Figure 2.5 illustrates the operation of the protocol.

S-MAC Enhancements Since the introduction of S-MAC, numerous improvements have been proposed to make it more energy-efficient or adaptive to changes in the network conditions. In Timeout MAC (T-MAC) [119], the wakeup interval is shortened with the adoption of *adaptive active time*. Zheng, et al. [134] proposed Pattern-MAC (PMAC) to address the issues of fixed duty cycle through exchange of *sleep-wakeup patterns*. Dynamic S-MAC (DSMAC) [82] also attempts to make the duty cycle of S-MAC to be more dynamic, albeit in a limited matter. Specifically, DSMAC allows the duty cycle to be either 0.1, 0.2, or 0.4.

ESC Energy-synchronized communication (ESC) [51] is one of the earliest wakeup scheduling schemes proposed for EPWSNs. ESC operates in a slotted manner. The key idea is simple: increase the number of wakeup slots when the energy supply increases and conversely, decrease the number of wakeup slots when the energy supply decreases. ESC refers to the former as (bursty) active instance increment and the latter as (bursty) active instance decrement. To facilitate the increment and decrement processes, ESC uses the notion of *cross-traffic delay* (CTD). For a node v with predecessor nodes $\mathbf{P}_v$ and successor nodes $\mathbf{S}_v$, the cross-traffic delay at v is the expected delay of every packet from any node in $\mathbf{P}_v$ to any node in $\mathbf{S}_v$ passing through v. CTD considers both sleep latency and retransmission delay. The authors demonstrated that for given schedules of predecessors and successors of node v, the CTD at v is not affected when a packet reaches v as long as the packet arrives within a certain interval. (An interval is just the contiguous set of slots between any two consecutive wakeup slots of the combined wakeup slots of $\mathbf{P}_v$ and $\mathbf{S}_v$.) This observation termed as the *stair effect* was used by the authors to design a localized $O(1)$ algorithm for schedule adjustment that minimizes the CTD at v. Note however that the $O(1)$ complexity hinges on the assumption that nodes use extremely low duty cycles.

Discussion S-MAC and its variants rely on periodic scheduled rendezvous for synchronization, wherein the period or interval between rendezvous is determined by the duty cycle. In dynamically duty-cycled networks, the periodicity of these rendezvous will not hold and thus, communication among nodes will be difficult. As such, employing S-MAC or any of its variants in EPWSNs will be extremely challenging.

Meanwhile, ESC avoids the problem due to synchronization difficulties through the exchange of wakeup schedules among nodes. This enables any node u that receives a wakeup schedule advertisement from another node v to know the exact time instances at which v will be awake. ESC was designed for EPWSNs and as such, it adapts to changes in energy supply and more importantly, it minimizes sleep latency through the generation of wakeup schedules that minimizes the CTD. A major limitation of ESC is that it was designed for ultra low duty-cycled nodes. In high duty cycle scenarios, ESC will require excessive overhead to exchange and store wakeup schedules. In addition, the algorithm that computes the minimal CTD wakeup schedule will no longer be $O(1)$.

Path-Coordinated

As mentioned in Section 2.1.1, sleep latency is a major problem in duty-cycled networks that significantly contributes to the end-to-end delay. Path-coordinated scheduling was therefore proposed to allow nodes along a path to coordinate their wakeup schedules such that packets traversing along the path will encounter as little delay as possible. An important requirement of this scheme is that either (*i*) a routing tree rooted at the sink node must already be in place; or (*ii*) nodes know the location of the sink node and their respective location.

Except for the Adaptive Staggered Sleep Protocol (ASLEEP) [14], all the other path-coordinated schemes were designed such that a wakeup interval or slot could accommodate the reception of at most one data packet. In ASLEEP, the wakeup interval duration is specified such that a node can communicate with all its child nodes as well as its parent node.

Wakeup Patterns Keshavarzian, *et al*. [72] proposed several path-wide wakeup schedule patterns that aim to minimize the end-to-end sleep latency from every node to a common base station node (backward or uplink direction) and *vice versa* (forward or downlink direction). In the discussion of the wakeup patterns, it is assumed that the network is organized into levels, with the base station at level 0. The level of a node essentially indicates its minimum hop count to the base station node. Let H denote the maximum number of hops (or maximum number of levels) in the network.

Fully-Synchronized Pattern (FSP) In this pattern, all nodes in the network sleep and wakeup at the same time. Though this is strictly not a path-coordinated pattern, it is used as the basis of the subsequent path-coordinated wakeup schedules.

Shifted Even and Odd Pattern This pattern is derived from FSP by shifting the wakeup pattern of the nodes in even levels by $T/2$.

Ladder Pattern This pattern is similar to the idea of *green wave* traffic light scheduling, *i.e.*, synchronizing traffic lights to turn green just in time for the arrival of vehicles from the previous intersections. This pattern can also be derived from FSP by shifting the wakeup schedule of nodes in level k by τ from that of level $k-1$.

Two-Ladders Pattern One problem of the ladder pattern is that only the latency of the downlink traffic is reduced. To improve the latency of both traffic directions, two-ladders pattern is proposed, combining a forward ladder with a backward ladder. Note that nodes in the middle levels (*i.e.*, nodes in levels $1, 2, 3, \dots, H-1$) wakeup twice in every period T.

Crossed-Ladders Pattern This is an enhancement of the preceding wakeup schedule pattern where the two ladders are crossed so that the same wakeup is used for both downlink and uplink directions. The cross point can be in any of the middle levels (*i.e.*, levels $1, 2, 3, \dots, H-1$).

In addition to the five wakeup patterns, Keshavarzian, *et al.* [72] also proposed the *multi-parent method* which can be independently applied to any of the five wakeup patterns. In terms of latency, the crossed-ladders and two-ladders pattern provide the best performance for both uplink and downlink traffic. The ladders pattern yields the same latency for downlink traffic but worse latency for uplink traffic. Compared to the ladders pattern, the shifted even and odd pattern provides slightly better latency for uplink traffic but worse latency for downlink traffic. Finally, FSP performs the worst for both traffic directions.

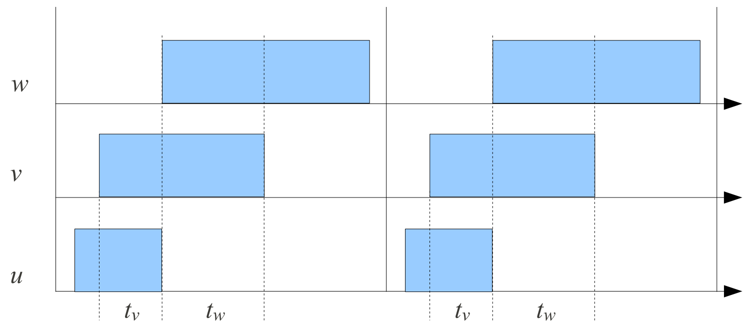


Figure 2.6: ASLEEP staggered scheduling. Wakeup interval of any node can actually be split into two parts which are called *talk intervals*. For node v in the figure, t_v is meant for talking to its children (one of which is u) while t_w is meant for talking to its parent w.

ASLEEP Adaptive Staggered Sleep Protocol (ASLEEP) [14] uses a staggered approach wherein nodes at the lower levels in the routing tree wakeup earlier than their ancestors. To clarify this, consider the wakeup schedules of nodes u, v, and w as shown in Figure 2.6. In this illustration, u is a child of v and v is a child of w. Note that the wakeup interval of any node can actually be split into two parts which are called *talk intervals*. The first part is meant for talking to its children while the second part is meant for talking to its parent. To establish a wakeup schedule, ASLEEP uses two control messages known as *direct beacon* and *reverse beacon*. The messages are used to propagate schedule information to downstream and upstream nodes, respectively.

Staggered Wakeup Scheduling with Multiple Parents Unlike ASLEEP which requires routing tree, Zhou and Medidi [136] proposed the use of location information to derive a staggered wakeup schedule. Prior to the computation of the wakeup schedule, the network is divided into concentric rings with the sink node located at the center. Every node must be able to identify its *ring level* with respect to the sink. Suppose that the K wakeup intervals in a cycle are $\{s_1, s_2, s_3, \ldots, s_K\}$, then a node belonging to ring n would choose to wakeup in intervals $\{s_1+n\Delta, s_2+n\Delta, s_3+n\Delta, \ldots, s_K+n\Delta\}$ where Δ is an estimated packet transmission delay. One advantage of using this approach is that a node belonging to ring level n can forward its packets to any of the nodes in ring level $n-1$, hence the approach is also called *multi-parent*.

Discussion A common advantage of path-coordinated schemes is that they provide low end-to-end sleep latency for all nodes in the network. However, the need to perform path-wide coordination makes the adaptation of these schemes to EPWSNs difficult. This is because a schedule change in one node causes all the other nodes in the path to possibly re-compute their wakeup schedules. In highly dynamic environments, this may lead to excessive communication and computational overhead and in the worst case, the scheduling algorithms may fail to converge to an optimal schedule.

Network-Coordinated

In network-coordinated scheduling, all nodes in the network collaborate to arrive at either a global wakeup schedule [81] or a per-node schedule that satisfies certain optimality goals [83]. Note that network-coordinated scheduling may either be *distributed* [81, 83] or *centralized* [55, 83]. In the latter, a single node is responsible for computing the wakeup schedules of all nodes in the network or at least a subset of nodes while in the former, every node is involved in the computation of their respective wakeup schedules.

Except for Sense-Sleep Trees (SS-Trees) [55], all the other network-coordinated schemes presented below were designed such that a wakeup interval or slot can be used to receive at most one data packet. In the former, nodes can transmit or receive one or more data packets in every wakeup interval.

GSA In the global schedule algorithm (GSA) proposed by Li, *et al.* [81], every schedule is tagged with a *schedule age* which indicates how long a schedule has existed in the network. Now consider a node v which uses a schedule with age A_v. When v receives a schedule from node w with age A_w and that $A_w > A_v$, then v adopts the schedule from w. GSA was proposed to enhance the performance of S-MAC protocol, *i.e.*, reduce the number of different schedules. This is because if a node v has neighbors with different schedules (*i.e.*, v is a border node), v must wakeup in all of its neighbors schedules which will result in higher energy consumption. Thus, by following the oldest schedule, after sufficient time, all nodes in the network will converge to a single schedule which is the oldest schedule.

Distributed DESS Lu, *et al.* [83] proposed two distributed algorithms to compute a wakeup schedule that minimizes the end-to-end delay. More specifically, the goal of the two algorithms is to find a single slot $s \in \{0, 1, 2, ..., S-1\}$ that minimizes the end-to-end delay for every source-destination pair in the network. The algorithms are called *Local-Neighbor* and *Local-DV*.

Centralized DESS Aside from the distributed DESS, Lu, *et al.* [83] also proposed a centralized approach for computing wakeup schedule that minimizes sleep latency. In particular, the goal of the algorithm is to find a slot $s_v \in [0, 1, 2, \ldots, S-1], \forall v \in \mathcal{N}$ that minimizes the *delay diameter*. The delay diameter D_f induced by a particular slot assignment f is defined as $D_f = \max_{v,w} P_f(v, w)$, where $P_f(v, w)$ is the delay along the shortest delay path between nodes v and w under the given slot assignment f.

Discussion For the distributed and centralized DESS schemes, the computation of minimal sleep latency paths is a big advantage. However, this comes at a high cost in terms of communication, storage and computational overhead. In general, all the above-mentioned network-coordinated wakeup scheduling schemes suffer from several drawbacks including long convergence time, high communication and computational overhead and low scalability. Between centralized and distributed schemes, the latter schemes are more feasible as they do not require the propagation of control information to a single node which can be prohibitively expensive. Except for GSA, all the network-coordinated schemes presented above can be easily adapted for dynamic wakeup scheduling. However, in highly dynamic environments, these schemes may fail to converge to an optimal wakeup schedule. In the case of GSA, its objective is to come up with a common global schedule which is opposite to the objective of dynamic wakeup scheduling.

Uncoordinated

In uncoordinated or non-collaborative schemes, a node does not use schedule information from other nodes to compute its own wakeup schedule. Rather, a node employs control theory, or other techniques that only require local information (*i.e.*, information within the node such as queue length or duty cycle.) This however does not mean that they are inferior to collaborative schemes. We discuss one such scheme below which is receive-centric, *i.e.*, every wakeup slot can accommodate the reception of at most one data packet.

Adaptive Duty Cycle Control with Queue Management Byun and Yu [22] proposed the use of a control-based technique to dynamically adjust a node's sleep interval at every cycle (and hence its wakeup schedule). Let $c_v(k)$ denote the sleep interval of a node v during the kth cycle. Then we have the following difference equation that can be used as a basis for designing a feedback controller:

$$c_v(k+1) = c_v(k) + \beta[q_v^{\mathrm{th}} - q_v(k+1)] - \gamma[q_v(k+1) - q_v(k)], \tag{2.1}$$

where $q_v(k)$ is the queue length at node v during the kth cycle, q_v^{th} is a specified queue length threshold for node v, and β and γ are control parameters that must be chosen. Note that as the queue length becomes smaller than the queue threshold, the sleep interval time increases linearly. Whereas, as the forward difference of queue length exceeds zero (because the increased forward difference of queue length induces a longer latency) the sleep interval time decreases. We highlight that the scheme only requires the local queue length information.

Discussion Uncoordinated wakeup scheduling schemes have two major advantages: (*i*) they do not require information from other nodes to compute their wakeup schedules resulting in low communication overhead; and (*ii*) schedule changes in other nodes will have no effect on a node resulting in low computational overhead. As such, these schemes are very agile and are therefore suitable for EPWSNs. One major disadvantage is that the schemes may generate wakeup schedules with high sleep latencies.

2.1.6 Aperiodic Synchronous Schemes

In periodic wakeup scheduling, a node's wakeup schedule usually repeats every cycle unless otherwise modified by the scheduling algorithm. In contrast, such repetition does not occur in aperiodic wakeup scheduling because the decision to wakeup or sleep in every slot is random. The aperiodic wakeup scheduling schemes presented in this section use bi-directional wakeup slots. Recall that a bi-directional wakeup slot can accommodate the transmission or reception of one or more data packets.

Dai et al. Dai *et al.* [30] proposed several random wakeup scheduling schemes where every node exchanges minimal schedule information with its neighbors to determine whether they are asleep or awake in a particular slot. In particular, every node exchanges its pseudo-random number generator (pRNG) seed and cycle position. A node can be in any one of the following states in a slot: ON-RX, ON-TX and OFF, with corresponding probabilities p_{rx}, p_{tx} and $p_{\text{off}} = 1 - p_{\text{rx}} - p_{\text{tx}}$, respectively. Note that with knowledge of the pRNG seed and cycle and the probabilities, any node v will be able to know the state of any other node w.

Ghidini and Das Ghidini and Das [44] proposed a random scheme that does not require any form of information exchange. Nodes therefore rely on the probability of being simultaneously awake to effect data transfer. To motivate the design of their random wakeup scheduling scheme, Ghidini and Das introduced the notion of *connection delay* for nodes v and w which is the time interval between the current slot and the first slot at which both v and w are simultaneously awake. A related concept is *connection duration* which is the time interval between the first and the last slot when v and w are simultaneously and continuously awake. The authors proposed a Markov Chain-based duty cycling scheme with control vector $[\delta, \tau, \gamma]^{\mathrm{T}}$ where δ is the target duty cycle, τ is the slot duration, and γ is the memory coefficient of the Markov Chain. The last parameter affects the transition probabilities

α (transition probability from sleep to wakeup) and β (transition probability from wakeup to sleep) as follows:

$$\alpha = \gamma\delta \tag{2.2}$$

and

$$\beta = \gamma - \alpha. \tag{2.3}$$

Note that $\gamma \in [0, 1/(1-\delta)]$, and setting $\gamma = 1$ means that the decision at every slot is totally independent from the previous decisions.

Discussion Aperiodic schemes are essentially random wakeup scheduling schemes where the decision to sleep or wakeup is performed at the beginning of every slot. As such, these schemes may potentially have higher computational overhead. But because no schedule is exchanged among the nodes, communication overhead is either zero or minimal. In terms of adaptability to dynamic environments, the schemes proposed by Dai *et al.* [30] are not amenable for adaptation because of their use of pRNG. Note that a node's wakeup slots are determined by its pRNG which is totally independent from the dynamics of the node's environment. As for the scheme proposed by Ghidini and Das [44], it can be easily adapted through the control parameter δ. In terms of sleep latency, the use of random wakeup slots in aperiodic schemes results in stochastic sleep latency as well. The scheme by Ghidini and Das [44] is slightly better as it provides a mechanism to improve the sleep latency, *i.e.*, through the minimization of connection delay.

2.1.7 Comparison Summary

Numerous wakeup scheduling schemes have been proposed for wireless sensor networks to address the unique challenges of duty-cycled node operation. We have summarized recent results on wakeup scheduling and classified the various approaches into two main categories, namely asynchronous and synchronous. Table 2.1 enumerates the various wakeup scheduling schemes that were presented in this section, consisting of MAC-layer and non-MAC-layer approaches. The table also qualitatively assesses the schemes against the considerations discussed in

Table 2.1: Comparison of Wakeup Scheduling Schemes

Scheme	Dynamics Adaptation	Latency-Aware	Duty Cycle Range	Processing Complexity	Overhead
B-MAC [98]	✓		M–H	N	L
X-MAC [21]	✓		M–H	N	L
BoX-MAC [87]	✓		M–H	N	L
RI-MAC [112]	✓		M	N	L
RC-MAC [62]	✓		M	N	L
S-MAC [129]			L–H	L	M
T-MAC [119]			L–H	L	M
PMAC [134]			L–H	M	M
DSMAC [82]			L–M (C)	L	M
ESC [51]	✓	✓	L	M	M
FSP [72]			L–H	M	M
Wakeup Patterns [72]		✓	L–H	M	H
ASLEEP [14]		✓	L–H	M	H
Zhou & Medidi [136]		✓	L–H	M	H
GSA [81]			L–H	M	H
Distributed DESS [83]	✓	✓	L–H	M	H
Centralized DESS [83]	✓	✓	L–H	H	H
Byun & Yu [22]	✓		L–H	H	L
Dai *et al.* [30]			U	M	M
Ghidini & Das [44]	✓		L–H	M	L

N–*None*; L–*Low*; M–*Medium*; H–*High*; C–*Coarse*; U–*Uncontrollable*

Section 2.1.1 to establish suitability for EPWSNs.

Asynchronous schemes were the earliest and simplest protocols proposed for wakeup scheduling. A major advantage of asynchronous operation is the absence of any form of time synchronization. Another major advantage of asynchronous approaches is that they do not require any computational and storage overhead since no schedules are stored. The only overhead is the transmission of special frames (*e.g.*, preamble or beacon) prior to data transmission. It is straightforward to make asynchronous schemes adaptive to the dynamics of its operating environment. However, both transmitter-initiated and receiver-initiated protocols cannot be used in the entire duty cycle range. One possibility to overcome this issue is to fix the wakeup rate and allow the exchange of multiple packets in a single wakeup. There is however a need to control the number of packets that can be exchanged to satisfy energy-neutral constraints.

In contrast to asynchronous schemes, synchronous schemes require nodes to be time synchronized. The majority of the proposed schemes presented fall under this category and we divide them further into two major sub-categories depending on the periodicity of the wakeup schedule. Most synchronous schemes operate periodically, *i.e.*, a wakeup schedule repeats every period, cycle or epoch until a change is made by the scheduling algorithm. There are however schemes wherein the decision to sleep or wakeup in every slot is random resulting in aperiodic wakeup schedules. In periodic wakeup scheduling, the main problem is to determine a subset of time intervals within a cycle to wakeup so that a node can perform packet transmission or reception. The selection of appropriate time intervals is usually driven by an objective to optimize a certain performance metric such as throughput or latency. Except for ESC which was specifically designed for low duty cycle networks, all periodic schemes can support low–high duty cycles. In terms of suitability for dynamic environments, path-coordinated schemes are not amenable for adaptation to such environments. This is because considerable coordination effort is needed to support dynamic wakeup schedules. Meanwhile, network-coordinated and path-coordinated schemes require high computational complexity. Notably, neighbor-coordinated schemes (except PMAC) entail low computational complexity. In terms of overhead, uncoordinated schemes entail the lowest overhead followed by neighbor-coordinated schemes. Path-coordinated as well as network-coordinated schemes require higher overhead because every node needs to coordinate their respective schedules with a larger number of nodes.

As mentioned in Section 2.1.1, sleep latency is a major challenge in duty-cycled networks. Path-coordinated schemes, including ESC and DESS appear to be heading in the right direction as they address this particular problem. However, the design of these schemes suffer from one major flaw: they assume that packet transmissions are always successful. In practical sensor networks where wireless link qualities have high variation, packet retransmissions are more the norm than the exception. As such, the low latency advantage of path-coordinated schemes will vanish in real-world deployments. Indeed, it might be difficult to have deterministic guarantees in stochastic environments.

Schedule representation is another important area that needs to be studied further. Note that in most synchronized schemes, every node needs to store the wakeup schedules of all its neighbors. Because sensor nodes have limited memory, schedules must be represented in a compact manner. The most straight-forward approach to represent a schedule is to use an S-bit array, where S is the number of slots per cycle. A '0' bit means that the corresponding bit position is a sleep slot while a '1' bit means that the corresponding bit position is a wakeup slot. Note however that this approach is not scalable. If a scheme uses high value for S and the network is dense, then considerable amount of memory is needed for schedule storage.

Another important consideration that needs particular attention is load balancing, a technique that can be used to balance the energy consumption among the nodes and can therefore increase the network lifetime [67]. While the scheme by Byun and Yu [22] is load aware, none of the proposed schemes has considered load balancing. To accomplish this, scheduling schemes may need to be coupled with the routing protocol or at least have knowledge of the underlying routing graph.

2.2 Routing Metrics

A routing or forwarding metric is critical in optimizing the performance of data delivery schemes. Because it determines the path to be used for data traffic, it must be carefully designed such that the chosen path provides optimal performance or satisfies the application requirements. In this section, we review the various routing metrics that have been proposed for wireless sensor networks.

The routing metrics that are discussed in this section can be classified into three classes, namely *traditional*, *loss-aware*, and *energy-aware*. The first class refers to metrics developed for wired networks such as hop count. Loss-aware metrics capture the impact of packet loss while energy-aware metrics consider the effect of energy.

2.2.1 Routing Metric Fundamentals

A routing metric is used to select the *least cost* path from a source to a destination. In proactive routing, least cost route selection normally employs the Bellman-Ford [17] or Dijkstra's [33] algorithms. In reactive routing, route selection is performed as and when a route reply is received, *i.e.*, a route is chosen if its cost is lower than the existing route. Formulating a metric is hard because of the fact that it must be able to capture the complex and dynamic characteristics of a *link* in a single scalar cost [97].

Link Cost The notion of a "link" is not well-defined in wireless networks. For the purpose of discussion, we define a wireless link as follows: we say that a link (u, v) exists between nodes u and v if v can receive a fraction of transmissions from u over a duration of time. We can associate a *link cost* to (u, v), which essentially represents the cost of sending one packet from u to v as measured or computed at u.

Path Cost In wired networks, the path cost is normally the sum of the individual link costs along the path [57, 84] or the sum of weighted costs [58]. This may not be sufficient in EPWSNs because wireless links are not totally independent from each other. Hence, the effect of link coupling also needs to be considered when obtaining the path cost. We can associate a *path cost* which indicates the cost of sending one packet over the path P_{st} from s to t computed at s.

2.2.2 Hop Count and Binary Link Abstraction

We first discuss hop count as it highlights the shortcomings of metrics that ignores wireless link quality. Hop count is the simplest and most widely used metric in many protocols including well-known mobile ad hoc network protocols such as DSDV [96], AODV [95], and DSR [66] and WSN protocols such as Gradient-Based Routing (GBR) [11] and IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) OF0 [114, 125].

For hop count, the cost of a link (u, v) from u to v as seen by u, denoted by c_{uv}^{HC}, is simply

$$c_{uv}^{\mathrm{HC}} = \begin{cases} 1 & \text{if link } (u,v) \text{ is “up”;} \\ \infty & \text{if link } (u,v) \text{ is “down”.} \end{cases}$$

Hop count assumes that links are symmetric, hence, the link status of (u, v) is taken to be the link status of the reverse link (v, u). The "up" and "down" link status is referred to as the *binary link abstraction*. Nodes that can be reached by one hop from u are called the *neighbors* of u. The cost of a path P_{st} from s to t is the sum of the individual metrics, or simply the path length:

$$C_{st}^{\mathrm{HC}} = \sum_{(u,v)\in P_{st}} c_{uv}^{\mathrm{HC}}$$

The binary link abstraction can cause difficulties in wireless networks. In wired networks, link status is easily determined by means of periodic hello packets. Because wired links are highly reliable, the loss of one hello packet indicates that the link is indeed "down" with very high probability. Unfortunately, this is not the case in wireless networks. Due to the lossy nature of wireless links [9], the loss of one hello packet does not necessarily mean that the link is "down". It is indeed trivial to devise a "link quality thresholding" rule that declares a link as "down" if more than a fraction of hello messages are not received. However, this technique does not work well in wireless networks [9]. Several experiments have confirmed that the binary link abstraction does not hold well in wireless networks [9,78], and that hop count performs poorly in real-world wireless multi-hop networks [27, 28, 35, 71]. The main reason for this is that hop count often selects paths with high loss rates that consequently degrade the network performance.

2.2.3 Packet Loss-Aware Metrics

Packet loss in wireless links is affected by many factors including data rate, transmit power, noise, multi-path, and RF interference [9,53,92]. For networks with low channel capacity, loss rate is also affected by packet size [63]. Given that loss rate is difficult to directly measure because of the many factors that influence it, various methods of estimation have been proposed. Some of the proposed estimators are:

Physical layer measurements (RSS, SNR, SINR, LQI) If readily provided by the physical layer, these are attractive estimators because they do not require additional measurement overhead. A model or mapping function is normally developed to transform the readings to packet delivery rate or throughput [77,101,132]. One drawback of this method is the hardware-dependence of the reliability and accuracy of measurements. Another disadvantage is that packet loss rate may have a weak correlation with physical layer measurements, as observed by De Cuoto *et al.* [32].

Probed packet delivery rate (active probing) This method entails the exchange of probe packets to measure packet delivery rate [31]. This is the most widely used estimation approach by packet loss-aware and flow interference-aware metrics. Packet transmission can either be broadcast (more common and less expensive) or unicast. Unlike physical layer measurements which requires the formulation of a model, this method is straightforward as it only requires the calculation of packet delivery ratio. It is also not hardware-dependent. However, probing is not accurate for two main reasons: (*i*) for broadcast probing, the broadcast data rate may be different from the unicast data rate; and (*ii*) the probe packet size is likely to be different from actual packet size.

Actual packet delivery rate ("passive" probing) This method takes advantage of the broadcast nature of wireless channels by basing the packet delivery rate on actual data packets [69,73]. Unlike the previous estimator, this one does not generate additional overhead. However, it has several disadvantages: (*i*) no measurement is available from non-transmitting nodes (must resort to probing or other means); (*ii*) if data is overheard (not intended for the listening node), the actual data rate when sender transmits to the listening node may be different.

LQI-Based Metrics The introduction of IEEE 802.15.4 [7], which require implementations to provide a link quality indication (LQI), motivated the development of several LQI-based metrics. According to the standard [7], the purpose of LQI is to characterize the strength and/or quality of a received packet. LQI values are expected to be within 0–255, with the higher value indicating better link quality. The MultihopLQI metric [115] is one of the earliest metrics to take advantage of this feature, and is implemented on top of the CC2420 radio [2] (an IEEE 802.15.4-compliant radio that is used in many sensor motes). The receiver sensitivity of CC2420 only allows the reception of packets with LQI at around 50 [2]. Suppose that v received a packet from u with LQI l_{uv}. Then the cost of a link (u, v) using LQI is

$$c_{uv}^{\mathrm{mLQI}} = \{[r^2(l_{uv}) >> 3] \times r(l_{uv})\} >> 3,$$

where $r(l_{uv}) = 80 - (l_{uv} - 50)$ and $>>$ denotes the right shift operator. The MultihopLQI of a path P_{st} is simply the sum of all the individual link costs from s to t, measured at s, given by

$$C_{st}^{\mathrm{mLQI}} = \sum_{(u,v)\in P_{st}} c_{uv}^{\mathrm{mLQI}}.$$

We note that the formulation of the link cost c_{uv}^{mLQI} is chip-specific. This severely limits the applicability of the metric, *i.e.*, to network deployments employing the CC2420 radio. Other LQI-based metrics include the ZigBee metric [8], MAX-LQI and RQI [25], and LQI-based ETX [48].

Expected Transmission Count (ETX) As the name implies, ETX predicts the number of data transmissions required to send a packet over a particular link, including retransmissions. The metric, proposed by De Couto *et al.* [31] can be considered as the ancestor of subsequent packet loss-aware metrics.

Given nodes u and v. Let p_{uv} denote the probability that a packet from u is received by v. Likewise, let p_{vu} denote the probability that a packet from v is received by u. Assuming that each transmission is independent, the probability that a packet sent by u is received by v *and* the acknowledgement by v is received by u is $p_{uv} * p_{vu}$. Then the expected number of transmissions over the link (u, v) as measured at u is:

$$c_{uv}^{\text{ETX}} = \frac{1}{p_{uv} * p_{vu}}.$$

To obtain the values of p_{uv} and p_{vu}, probe packets are periodically broadcast by the nodes. For example at node u, to get p_{vu}, u simply counts the number of probe packets received from v over a time window and divide it by the total number of probe packets expected over the time window. p_{uv} can be obtained by u through the probe packets received from v as v includes this value in its probe packets. The ETX of a path P_{st} from s to t, measured at s is defined as the sum of the metric of all the links in the route, or:

$$C_{st}^{\text{ETX}} = \sum_{(u,v)\in P_{st}} c_{uv}^{\text{ETX}}.$$

One drawback of ETX is the generation of additional communication overhead due to probe packets. The delivery rate probing scheme may also suffer from instability and unreliability when the network load is heavy and highly variable. While the formulation itself is stable as it does not consider load, the probing mechanism can be significantly affected by network load [105]. Another disadvantage of ETX is the inaccuracy of measured delivery rates. The term p_{uv} is supposed to account for data packet delivery rates while p_{vu} is meant to capture the probability of delivering MAC acknowledgements. However, the measured delivery rates correspond to probe (fixed-size broadcast) packets.

Notwithstanding these drawbacks, ETX have been demonstrated to outperform hop count in terms of throughput in several real-world testbeds [31,35]. The improvement in throughput is significant for paths with more than two hops. This may hint that ETX will become more useful as networks grow larger and paths become longer. In the context of sensor networks, ETX is used by several protocols including CTP [47], and RPL ETXOF [46]. In [47], results show that ETX significantly outperforms MultihopLQI in terms of packet delivery ratio.

ETX Enhancements The dramatic performance advantages offered by ETX triggered the development of several ETX enhancements. Expected Transmission Time (ETT) [36] improves on ETX by considering the impact of multi-rate transmission. Expected Transmission Count over Forward Links (ETF) [106] exploits highly asymmetric links, a common phenomenon in low-power links. Modified ETX (mETX) [76] improves response to short-channel variations while its related metric Effective Number of Transmissions (ENT) [76] aids in the selection of paths with bounded packet loss rate. Multicast ETX (METX) [104] and Success Probability Product (SPP) [104] are targeted for multicast routing.

2.2.4 Energy-Aware Metrics

Energy is a very important resource in wireless sensor networks. As such, path selection should also consider energy consumption and availability; otherwise, paths with high energy consumption and low energy availability might be consistently selected. In battery-powered sensors networks, this behavior will lead to certain nodes running out of energy which can lead to network partitioning.

Energy Metric One of the earliest works to consider energy in packet forwarding is Shah and Rabaey [108]. The authors proposed a link cost that considers the energy required for transmission and reception across the link, including the remaining energy of the sender. Hence for a link (u, v), the cost from the perspective of u, is given by

$$c_{uv}^{\mathrm{EM}} = e_{uv}^{\alpha} R_u^{\beta},$$

where e_{uv} denotes the energy used to transmit and receive on the link, and R_u is the residual energy of u, normalized to the initial energy of the node. Note that for battery-powered nodes, $R_u \leq 1$ since the battery level is non-increasing. The parameters α and β are weighting factors that can be tweaked to trade-off between energy consumption and energy availability, but which unfortunately requires further study (*i.e.*, the authors did not provide hints on how to adjust the parameters given certain objectives). The path cost from s to t, as computed at s, is given by

$$C_{st}^{\mathrm{EM}} = c_{sv}^{\mathrm{EM}} + C_{vt},$$

where C_{vt} denotes the average cost of reaching the destination through v, and is given by

$$C_{vt} = \sum_{w \in S_{vt}} p_w c_{vw}^{\mathrm{EM}}.$$

Before discussing the significance of p_w, we first discuss how packet forwarding is performed. To distribute packets according to residual energy, Shah and Rabaey [108] proposed the use of *probabilistic forwarding*. When a node v has a packet to forward, it randomly selects the successor node w from its successor set S_{vt}. The probability that w is selected, denoted by p_w, is inversely proportional to the cost of forwarding a packet through w, given by

$$p_w = \frac{1/c_{vw}^{\mathrm{EM}}}{\sum_{x \in S_{vt}} 1/c_{vx}^{\mathrm{EM}}}.$$

Energy Metric Enhancements The energy metric, in tandem with probabilistic forwarding, enables the network to attain a more balanced energy consumption and therefore longer network lifetime. Since its introduction, several extensions and variations have been proposed to address some of the deficiencies of Shah and Rabaey's work. A natural extension, proposed by Wang *et al.* [121], combines node residual energy with ETX with the aim of providing trade-off between energy consumption and path selection. A similar metric, proposed by Al-Jemeli *et al.* [13], uses RSSI instead of ETX. Several energy-aware metrics [10, 19, 59, 111] incorporate load-balancing and QoS-related metrics such as delay and throughput.

Harvesting-Aware Metrics The energy-aware metrics in the preceding discussions are designed with battery-powered WSNs in mind, hence their goal is to either select paths that will consume the least power or paths with high energy, or a trade-off between the two. In any case, the ultimate objective is to prolong the network lifetime. However, as highlighted in Section 1.2, network lifetime maximization is not a suitable objective in EPWSNs because of energy storage recharging opportunities. Several harvesting-aware metrics have therefore been proposed to exploit this important characteristic of EPWSNs.

Two of the earliest routing metrics to consider the impact of energy harvesting are the routing metric for the GREES-L and GREES-M routing protocols [131]. These protocols are based on geographic routing, and hence the metric uses physical distance as one of the parameters. A major drawback of these metrics is that a node u requires numerous information about every other node w in the successor set, including the harvesting rate, consumption rate, times of last packet transmission and hello transmission, and distance to the sink. In addition, the metrics have two parameters that need to be tuned.

Jakobsen *et al.* [64] introduced the notion of *energy distance*, which is essentially the sum of the hop count of a path and a *distance penalty* term. The latter encapsulates the amount of energy available on a node, and can be represented as a function that is monotonically decreasing with respect to the energy availability. For a node u with energy availability $e_u \in [0, 1]$, where 0 means empty and 1 means full, the distance penalty function must behave such that $f(e_u) \longrightarrow 0$ when $e_u \longrightarrow 1$ and $f(e_u) \longrightarrow \infty$ when $e_u \longrightarrow 0$. An example function proposed by Jakobsen *et* *al.* [64] is given below:

$$f(e_u) = \begin{cases} 0 & c < e_u \leq 1 \\ \beta \frac{e-c}{b-c} & b < e_u \leq c \\ (\alpha - \beta)\frac{e-b}{a-b} + \beta & a \leq e_u \leq b \\ \alpha & 0 \leq e_u < a, \end{cases}$$

where a, b and c are different thresholds of energy availability while α denotes the maximum penalty and β represents the penalty amplitude. Intuitively, paths will lower energy will tend to have longer energy distance, and hence, packet forwarding will favor paths with higher energy availability.

2.2.5 Comparison Summary

Routing metric plays a critical role in the performance of routing protocols. Formulating a metric is considered as the hardest problem in routing protocol design because of the fact that it must be able to capture the complex and dynamic characteristics of a link in a single scalar cost. Packet loss and energy availability are two of the most important factors that affect data delivery performance in wireless sensor networks.

Packet loss is affected by factors such as transmit data rate, transmit power, noise, and multi-path fading. Given the complex interplay of these factors, several loss rate estimation schemes have been proposed in the literature. MultihopLQI and ETX are two of the implemented and widely-used metrics in sensor networks. The former employs hardware-dependent link quality measurements while the latter uses active probing to estimate the packet loss rate. Experiments have shown that loss-aware metrics are significantly better than hop count in terms of packet delivery ratio. One major disadvantage of loss-aware metrics, as far as wireless sensor networks are concerned, is that they ignore the impact of energy.

In battery-powered sensor networks, metrics that consider energy consumption and residual energy have been shown to significantly prolong the network lifetime. When combined with loss-aware metrics, these metrics can yield energy-efficient and reliable data delivery. However, their underlying assumption that energy is finite makes them unsuitable for EPWSNs. Of the schemes that we have presented, the metrics for GREES-L and GREES-M, and the energy distance metric seem to be better candidates for EPWSNs as they have been designed to be aware of energy harvesting. However, a closer analysis shows that these metrics are deficient in terms of the following:

- The GREES-L and GREES-M metrics require the exchange and maintenance of numerous parameters for every forwarder node, entailing high communication and storage overhead. The metrics also require the physical distance to the sink as it is designed for geographic routing.
- The energy distance metric is based on hop count which is already well-known to perform poorly in real wireless sensor networks. While the authors proposed a distributed algorithm to calculate the energy distance, it was not shown whether the metric will yield consistent and loop-free paths.
- As mentioned in Section 1.2, sleep latency is a major challenge in EPWSNs. None of these schemes have tackled this critical issue.

2.3 Bulk Data Transfer

To understand how bulk transfer protocols will perform in the context of EPWSNs, we survey the state-of-the-art in bulk transfer. We categorize the various bulk transfer schemes into two, namely, *single packet-based*, and *packet train-based*. The ultimate aim of this section is to expose the shortcomings of existing bulk transfer schemes when nodes perform dynamic duty cycling.

2.3.1 Bulk Transfer Fundamentals

Bulk transfer refers to the transmission of large amount of sensor data from a source node to a destination node, typically a gateway or base station. Bulk transfer can actually be performed using *generic* transport protocols (Wang *et al.* [122] provides a good survey on this subject) but specific application requirements and tight resource constraints in terms of memory, channel capacity and energy have led to the development of *specialized* protocols for bulk transfers. In designing bulk transfer schemes for EPWSNs, the following important factors must be considered:

Dynamic Duty Cycling Compliance As mentioned, EPWSNs employ dynamic duty cycling to ensure energy neutrality. Bulk transfer schemes must therefore operate such that the duty cycle constraints of every node (along the bulk transfer path) are not exceeded. One main difference between battery-powered sensor networks and EPWSNs is that in the latter, unused energy (in times when the harvesting rate is high) will be wasted because of finite energy storage capacities. Thus, bulk transfer schemes must find the maximum achievable throughput that minimizes wastage. This is tantamount to saying that the bulk transfer must fully utilize the duty cycle.

Reliability and Flow Control Current sensor network protocol stacks lack a transport layer, hence, the bulk transfer scheme must implement its own reliability mechanism to ensure that all fragments are delivered. In addition, it must also implement some form of flow control to ensure that the network can accommodate its sending rate.

2.3.2 Single Packet-Based

Early bulk transfer schemes were heavily influenced by TCP, which can be considered a single packet-based scheme. In this scheme, the source splits the bulk data into fragments, and each fragment is individually sent to the destination. The main issue in single packet-based schemes is the determination of the interval in between packet transmissions.

Koala Koala [88] is one of the earliest schemes for bulk transfer. It uses RTT (round-trip time) to control the sending rate from the source to the sink. Specifically, Koala sends packets at a rate of RTT/2, relying on its underlying flexible control protocol to provide the RTT measurements and reliability. The key idea behind this sending interval is to ensure that a newly transmitted packet will not in any way interfere with the previously transmitted one as the latter should have already reached the sink. Koala also supports duty cycling and uses low-power probing, a technique akin to beacon transmission in receiver-initiated MAC protocols.

Unfortunately, RTT-based rate control performs poorly over long paths. Note that in wireless sensor networks, the use of short-range radios enable some form of spatial reuse. This basically means that node pairs that are out of range can actually communicate with each other without causing packet collisions. Hence, there are situations where the sender does not have to wait for the previously transmitted packet to reach the sink. It can transmit the next packet as soon as its transmission will not interfere with the last one.

Flush Flush [75] is one of the first bulk transfer schemes to take advantage of spatial reuse. It introduced the idea of "pipelining" packets to improve throughput, which is shown in Figure 2.7. To pipeline as many packets as possible, Flush needs to know the *interference range*. This is difficult in practice because of the complex radio propagation characteristics in real-world deployment environments. In their work, the authors proposed a simple SNR thresholding approach to identify "jammers", *i.e.*, nodes that can conflict with the transmission of another node but their signal cannot be heard. Now, once every node knows the interference range of every other node along a path, the sending rate can be maximized by using the following two simple rules: (*i*) transmit when the successor node is free from interference, and (*ii*) transmit at a rate below the successor node's sending rate. The first rule ensures collision-free transmissions, while the second rule is some form of flow control to avoid swamping the next hop with traffic that it cannot handle. For reliability, Flush uses a combination of end-to-end NACK (indicating lost fragments) and link layer acknowledgements to reduce end-to-end NACK.

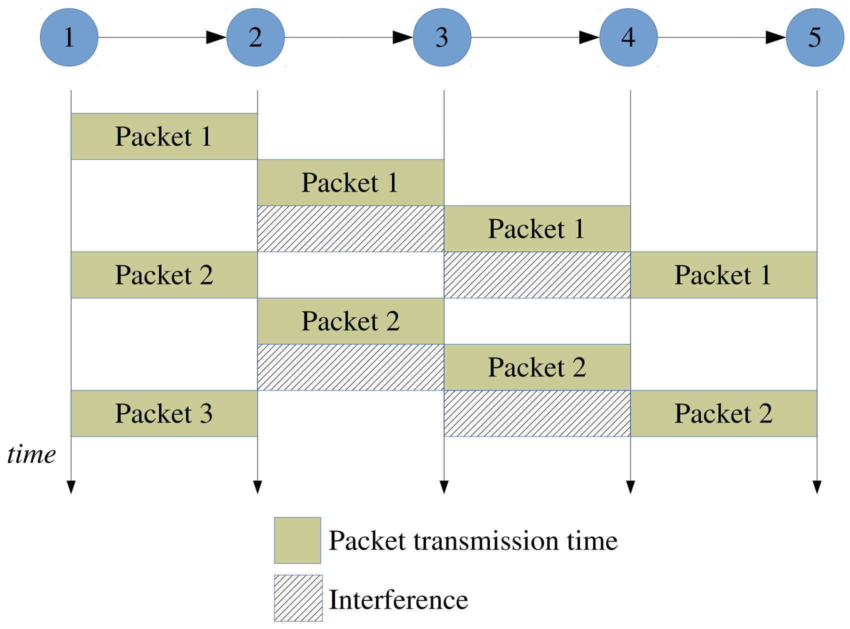


Figure 2.7: Illustrating the packet pipelining from the perspective of node 1, as proposed in Flush [75]. The interference range is 1, implying that a transmission by node 3 will reach back node 2. As such, node 1 must wait for node 3 to complete transmission before it can send its next packet. This is because if it transmits while node 3 is transmitting, its transmission to node 2 will collide with node 3's transmission.

Packets In Pipe (PIP) PIP [100] is another scheme that employs packet pipelining to improve throughput performance. PIP [100] took the idea of packet pipelining further through the use of a MAC protocol that is TDMA-based, centralized, connection-oriented and uses multiple channels. PIP essentially aims to tightly coordinate the packet pipelining from the source to the sink and further reduce intra-flow and inter-flow interference. Although PIP significantly outperforms Flush especially in transfers that involve longer paths, it may entail significant overhead for synchronization and other coordination.

2.3.3 Packet Train-Based

Flush and PIP are designed to maximize throughput without regard to the energy consumption of the sensor nodes. They need the radio to be turned on for the entire transfer duration to achieve the desired packet pipelining effect. This is obviously not suitable in EPWSNs where nodes are duty cycled for energy neutral operation. To attain energy sustainability, bulk transfer schemes clearly need to operate on top of wakeup scheduling schemes. However, such an arrangement poses difficulties on the single packet-based schemes. Note that in these schemes, the transmitting node needs to transmit at regular intervals for optimum performance. With wakeup scheduling, the transmitting node loses this control since it can only transmits when the intended next hop is awake.

Certain types of wakeup scheduling schemes entail subtle problems for bulk transfer schemes. As mentioned in Section 2.1.3, some schemes allow at most one data packet per wakeup slot. For asynchronous schemes, single packet per wakeup is highly inefficient as every packet needs to be preceded by some form of overhead (*i.e.*, preamble transmissions or listening for beacons). For synchronous schemes, single packet per wakeup is also wasteful since wakeup slots are designed to be larger than a single packet.

Fortunately, several wakeup scheduling schemes (*e.g.* X-MAC [21] and ContikiMAC [38]) support multiple packets per wakeup. Such a transmission approach, which we refer to as *packet train*, can clearly remedy the deficiencies of single packet-based schemes. Because wakeup scheduling somewhat limits the opportunities at which nodes can exchange packets, it makes sense to transmit as many packets as possible at every opportunity to improve efficiency. Duquennoy *et al.* [39] observed this deficiency with the single packet-based scheme as they introduced the first packet train-based scheme. They called their technique *packet bursting*, which they defined as the rapid transmission of successive packets after a single wakeup, in conjunction with the ContikiMAC [38] duty cycling. This is illustrated in Figure 2.8. Note that unlike the other presented schemes in this section, packet bursting is *not* complete, in the sense that it needs to be integrated with a transport protocol such as TCP for reliability and flow control. Experimental results show that packet bursting in conjunction with duty cycling can provide low power and high throughput performance.

Figure 2.8 illustrates the packet bursting technique performed across a single hop. For bulk transfers that traverse multiple hops, packet train transmissions are performed hop-by-hop, from the source until the destination. To illustrate, if a

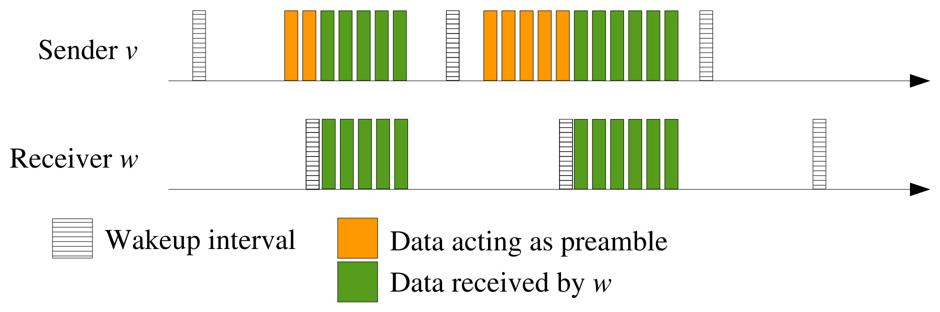


Figure 2.8: Illustrating the packet bursting technique. Sender node v transmits data frames as preambles. Once receiver w wakes up and receives a data frame, it sends back an ACK. Sender node v then transmits the next frame, and so on, until the last data frame. It indicates whether or not there is a succeeding frame using the frame pending bit in the IEEE 802.15.4 header.

source node 1 has bulk data to send to a destination node 5 which will traverse through the path 1-2-3-4-5, then packet train transmission will be performed at every hop, *i.e.*, at 1-2, 2-3, 3-4 and 4-5. The key challenge of this approach is the coordination in the hop-by-hop packet train transmissions to avoid both intra-flow and inter-flow interference.

2.3.4 Comparison Summary

In this section, we have differentiated bulk transfer schemes based on how they transmit packets. Single packet-based schemes transmit one packet per interval while packet train-based schemes transmit back-to-back packets for every transmission opportunity. Existing single packet-based techniques such as Flush and PIP yield high throughput but they are not suitable for EPWSNs because they require 100% duty cycle.

To satisfy energy neutrality constraints, bulk transfer schemes must work hand in hand with wakeup scheduling. From the perspective of a sending node, wakeup scheduling limits the opportunities at which it can transfer packets to its next hop node. Thus, sending one packet per wakeup is inefficient and that transmitting back-to-back packets for every wakeup seems to be clearly advantageous. Such an approach, which we call packet train-based, have been demonstrated by Duquennoy *et al.* [39] to result in low power high throughput bulk transfer. But while the technique yields low energy consumption, the outcome is *incidental* rather than *in-*

tentional, *i.e.*, the use of packet trains does not actively control the energy usage to be within specified bounds. In other words, the use of this scheme will not ensure energy-neutrality.

In summary, existing bulk transfer schemes are oblivious of the duty cycle constraints of sensor nodes. Such blindness can cause uncontrolled and rapid draining of the energy reserves, leading to the temporary unavailability of nodes along the transfer path. Ultimately, this will result in transfer disruptions which render the transfer of arbitrarily-sized sensor data difficult, if not infeasible.

Chapter 3

Energy-Neutral Scheduling and Forwarding

A major challenge in both static and dynamic duty cycling networks is *sleep latency* which is the delay incurred when a transmitting node must wait for the receiving node to wakeup before it can commence packet transmission [50, 83, 130]. In battery-powered WSN where duty cycles are static, static wakeup schedules that minimize sleep latency and end-to-end delay can be computed prior to the operation of the network [52, 83]. These pre-computed and fixed wakeup schedules cannot provide optimal performance in EPWSNs where duty cycles are dynamic and vary from node to node.

We therefore propose a dynamic wakeup scheduling scheme that enables every node to compute a wakeup schedule according to their respective prevailing duty cycle constraints. To reduce sleep latency, the scheme distributes the receive wakeup slots to be as evenly as possible across every epoch. This exploits our analytical result which states that the lower the variance of the intervals between receive wakeup slots, the lower the expected sleep latency. To reduce the overhead for storing and exchanging schedules, the scheme employs sequence-based scheduling to represent dynamic wakeup schedules in a compact manner. The resulting scheme uses *bit-reversal permutation sequence* (BRPS), and we analytically obtain its worst-case sleep latency to be slightly worse than the ideal scheme but better than schemes where the receive wakeup slots are spaced uniformly or exponentially.

As BRPS can only reduce the expected sleep latency on a single hop basis, we also propose a routing metric to enable the selection of multihop paths with low end-to-end latency. The metric, which we call *expected transmission delay* (ETD), simultaneously considers sleep latency and wireless link quality. We show that the metric is left-monotonic and left-isotonic, proving that its use in distributed algorithms such as the distributed Bellman-Ford algorithm will yield consistent, loop-free and optimal paths.

The rest of this chapter is organized as follows: Section 3.1 discusses the models and assumptions that are used in the development of the proposed scheme. Section 3.2 elaborates on the dynamic wakeup scheduling. It is also in this section that we derive an important finding regarding the expected sleep latency entailed by a dynamic wakeup schedule. Meanwhile, Section 3.3 provides a detailed presentation of the routing metric and forwarding scheme. The simulation models and parameters are discussed in Section 3.4 while the simulation results are presented in Section 3.5. We finally summarize our findings and contributions in Section 3.6.

3.1 Models and Assumptions

3.1.1 Network Model

We consider an EPWSN composed of N static nodes. No assumptions are made on the deployment or distribution of the nodes over the area of interest so long as the resulting network is connected. Every node is assigned a unique identifier and has a finite queue which is used for storing packets that need to be forwarded.

Application and Traffic Model We consider an environmental monitoring application wherein the sensor network is tasked to monitor the environmental conditions (*e.g.*, temperature, humidity, air quality) of an area of interest. Every node performs periodic sensing every T_s which are sent to a common data collection point. Similar to prior works [50, 52, 83], we assume that the data generation rate is low (*i.e.*, T_s is large) and does not cause significant congestion and queueing delay. Note that this assumption is reasonable as certain environmental monitoring applications require at most 1 reading every 5 minutes [116].

Cross-Layer Implementation Approach The proposed scheme has two main components: (*i*) a dynamic wakeup scheduling scheme; and (*ii*) a packet forwarding algorithm. In terms of implementation, the components are more appropriately implemented in separate layers. The first component can be implemented in the data link layer on top of a MAC protocol while the second component and can be implemented in the network layer.

Slot Synchronization The proposed scheme operates in a slotted fashion, thereby requiring slot synchronization or alignment. The slot duration is defined prior to the operation of the network and is given by $\tau = \frac{T}{S}$, where T is the duration of one cycle and S is the number of slots in a cycle. τ is defined such that it can accommodate the transfer of one maximum-length data packet and a corresponding ACK packet.

Medium Access Control and Link Estimation The proposed scheme requires the underlying MAC to support broadcast and two-way handshake packet transmission (unicast data followed by ACK). If the sending node fails to receive an ACK, the MAC layer informs the network layer of the failure. In addition, we assume that the underlying transceiver provides a link quality estimate. Note that this assumption is reasonable since in the IEEE 802.15.4 standard [7], link quality indication (LQI) is a feature required on radio transceivers.

Network Initialization As the proposed scheme focuses on the design of energy neutral wakeup scheduling and forwarding, it assumes that the network has been initialized, *i.e.*, (*i*) every node v is assumed to have established its set of one-hop neighbors $\mathbf{N}_v$; and (*ii*) nodes have aligned their slots. We will not propose schemes

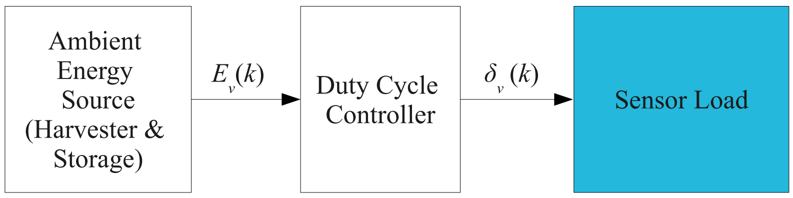


Figure 3.1: Node model showing the duty cycle controller, which takes energy level $E_v(k)$ from the ambient energy source as input and provides the duty cycle $\delta_v(k)$ as output.

for accomplishing these functions but the reader can refer to existing work on neighborhood discovery [86] and slot synchronization [29,43,85]. The latter methods also address the problem of drift due to low accuracy clocks or oscillators that are used in sensor nodes.

3.1.2 Energy-Harvesting Node Model and Duty Cycle

Every node, except for the sink node, is powered by ambient energy and uses adaptive duty cycling [51, 61, 68, 120, 137] to optimize its utilization of available energy for communication. The sink node does not perform duty cycling (*i.e.*, it is always awake) and has unlimited energy supply.

Energy-Harvesting Node Model Figure 3.1 shows the energy-harvesting node model used in this chapter. The duty cycle controller model, which is inspired by [120], requires the energy level of the buffer $E_v(k)$ as input and provides the duty cycle $\delta_v(k) \in [0, 1]$ as control output[2], where k denotes the cycle or epoch. In simple terms, the duty cycle controller [120] chooses $\delta_v(k)$ as the duty cycle that minimizes $|E_v^*(k) - E_v(k)|$, where $E_v^*(k)$ is a preset target energy level. (A more detailed discussion of this model is presented in Section 3.4.2.) In general, it can be said that the resulting duty cycle $\delta_v(k) \propto E_v(k)$.

Duty Cycle As mentioned, the duty cycle $\delta_v(k)$ indicates the fraction of time that v can be active at k, *i.e.*, all of its components (microcontroller, radio, sensors and miscellaneous peripherals) are powered up. To emphasize, this means that each

[2]Of course, other controller models (which may require more information such as power requirements of transmission and reception, current harvesting rate, etc.) can also be used as long as they can provide the energy neutral duty cycle which indicates the fraction of time that a node can be active.

of the components can have a duty cycle of $\delta_v(k)$. As our objective in this study is to perform wakeup scheduling of the radio, from hereon, we will only consider $\delta_v(k)$ as allocated to the radio component.

Transmit and Receive Slots Allocation At the kth cycle, the total amount of time for the radio of v to be active is $T\delta_v(k)$. As v must also perform relaying and not just transmitting of its own readings, it must allocate a fraction of its active time for reception as well. We propose the following simple allocation: First, v reserves a certain amount of time for transmitting its own readings. If there is excess time, v divides the remaining time such that the number of transmit and receive wakeup slots are equal. The rationale behind this is to ensure that v will have a chance to forward all packets generated (its own readings) and received within a cycle. Note that if v allocates less transmit slots and more receive slots, then packets will potentially accumulate if v receives more packets than its transmit slots. Let $n_v(k)$ be the number of receive slots of v at k. If T_s is the sensing interval, then at every cycle v has an average of $\frac{T}{T_s}$ readings which requires $\frac{T}{T_s}\tau$. Whereas, the time needed for receiving and transmitting transit packets is $2n_v(k)\tau$. Then $T\delta_v(k) = \frac{T}{T_s}\tau + 2n_v(k)\tau$. Solving for $n_v(k)$ and forcing the result to be an integer, we obtain

$$n_v(k) = \begin{cases} \left\lfloor \frac{T}{2}\left(\frac{\delta_v(k)}{\tau} - \frac{1}{T_s}\right)\right\rfloor & \text{if } \delta_v(k) > \frac{\tau}{T_s} \\ 0 & \text{if } \delta_v(k) \le \frac{\tau}{T_s}. \end{cases} \tag{3.1}$$

3.1.3 Wakeup Schedule

A duty-cycled node requires wakeup schedules for data transmission and reception. When node v needs to forward a packet to node w, v needs to know the receive wakeup schedule of w so that it can wakeup at the appropriate slot in the future to perform the actual transmission. We define the receive wakeup schedule as follows.

Definition 4 (Receive wakeup Schedule)**.** *Every node v has a receive wakeup schedule $\mathbf{\Gamma}_v(k)$ for the kth cycle which contains $n_v(k)$ time slots indicating the times at which v*

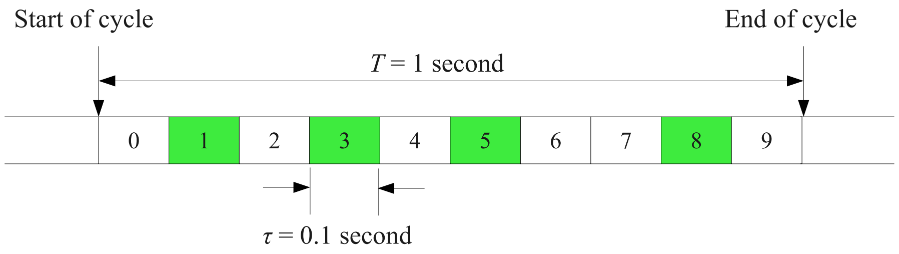


Figure 3.2: An example of a receive wakeup schedule of v where $\mathbf{\Gamma}_v(k) = \{1, 3, 5, 8\}$, $S = 10$ and $T = 1$ second. The shaded slots are the time slots where v listens for transmissions from its neighbors.

wakes up to listen for transmissions from its neighbors. A receive wakeup slot is represented by an integer which ranges from 0 to $S - 1$. A number $\eta \in \mathbf{\Gamma}_v(k)$ means that v should wake up in the interval $[\eta\tau, (\eta + 1)\tau]$ relative to the start of the cycle.

To clarify this definition, consider the receive wakeup schedule of node v, $\mathbf{\Gamma}_v(k) = \{1, 3, 5, 8\}$ where $S = 10$ and $T = 1$ second as shown in Figure 3.2. It must be noted that $\mathbf{\Gamma}_v(k)$ only contains the slots at which v wakes up to receive packets from its neighbors. To transmit its packets, v can do so in any unused slot $\beta \notin \mathbf{\Gamma}_v(k)$ but must ensure that the intended receiver node w is awake to listen for packet transmissions, *i.e.*, $\beta \in \mathbf{\Gamma}_w(k)$.

We complete the discussion on wakeup schedule by considering the following example. Suppose that v (with receive wakeup schedule given in Figure 3.2) intends to transmit a packet to w which has a receive wakeup schedule of $\mathbf{\Gamma}_w(k) = \{1, 3, 4, 7\}$. Node v can choose either slot 4 or 7 to transmit its packet but it cannot choose slots 1 and 3 because they are part of its own receive wakeup schedule. If v chooses slot 4, then v must transmit the data packet the moment slot 4 begins and should w correctly receive the data packet, it must send back an ACK packet within the same slot. This is possible because the slot duration τ has been defined to accommodate the transfer of one data packet and a corresponding ACK packet.

3.2 Dynamic Wakeup Scheduling

We now begin the development of a dynamic wakeup scheduling scheme that can reduce sleep latency to the least extent possible. Because it has to be ultimately executed on sensor nodes which have limited computational power, we want the scheduling scheme to have low computational complexity and low communication and storage overhead. We begin our discussion with a formal definition of sleep latency as it is a key concept in the chapter.

Definition 5 (Sleep Latency). *The sleep latency from node u to v, denoted by W_{uv}, is the delay from the time that a packet becomes ready for transmission at u until the actual packet transmission from u to v. The latency occurs because u must schedule its transmission in the future when v is awake to receive its transmission.*

As highlighted in Section 1.2, sleep latency is a major challenge in both battery-powered and environmentally-powered WSN as it is the main cause of high end-to-end delay. To reduce sleep latency, existing scheduling schemes [51, 52, 83] perform "tight" coordination wherein the receive slots of v are positioned at times that are close to the receive slots of its successor node w. The key idea is that when v receives a packet in its receive slot, it can quickly forward the packet to w thereby reducing sleep latency[3]. While such an approach may work in perfect conditions, it will face difficulties in situations where link qualities are not ideal. Because of lossy links, v may not be able to successfully transmit at the "nearest" receive slot of w. The retransmission of the lost packet will surely increase its delay which depends on how the receive slots of w are distributed over the cycle. Furthermore, because of the dependence of one node's schedule on another node's schedule, a change in one node may unnecessarily trigger a change in the schedule of other nodes.

We instead propose a "loose" coordination approach wherein the receive slots of v are distributed as evenly as possible within the cycle duration T without regard for the position of the receive slots of its neighbors. The basis of this simple approach is the following lemma which shows that minimizing the variance of intervals between receive slots leads to reduced sleep latency.

Lemma 1 (Expected Sleep Latency). *Suppose that n is the number of receive wakeup*

[3]This is possible because we assumed low data generation rate, *i.e.*, no congestion and queueing delay.

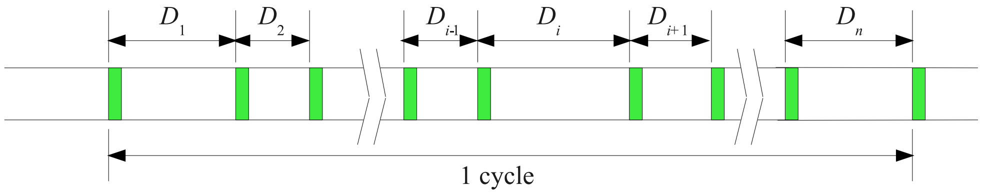


Figure 3.3: Model used in the derivation of expected sleep latency. Given a node with n receive slots, D_i is the interval between two successive receive wakeup slots, where $1 \leq i \leq n$.

slots of node v at a particular cycle. Let D_i denote the ith interval between two successive receive wakeup slots, where $1 \leq i \leq n$ (c.f. Figure 3.3). Suppose that the packet ready times at some neighbor node u is uniform in $[0, T]$. Whenever u has a packet to transmit to v, the expected sleep latency from u to v is

$$\mathsf{E}(W_{uv}) = \frac{1}{2}\mathsf{E}(D)\left[1 + \frac{\mathsf{Var}(D)}{\mathsf{E}^2(D)}\right], \tag{3.2}$$

where $\mathsf{E}(D)$ and $\mathsf{Var}(D)$ are the mean and variance, respectively, of all the intervals $\{D_i\}$.

Proof. Suppose that intervals $\{D_i\}$ are independent and identically distributed and let D be a random variable with distribution $F_D(x)$ and that $\mathsf{Pr}(D_i \leq x) = F_D(x)$, where $1 \leq i \leq n$. With $\{D_i\}$ being an i.i.d. sequence of positive random variables, we can use results from renewal theory, in particular renewal reward processes (see [103], page 441) to obtain the expected sleep latency.

Let $N(t) = \sup\{m \geq 0 : S_m = D_1 + D_2 + D_3 + ... + D_m \leq t\}$. The sleep latency from the current time t until the next receive wakeup slot at node v is simply the residual time $B(t)$ given by

$$B(t) = S_{N(t)+1} - t.$$

Since the packet ready times at node u are uniform, the expected sleep latency from u to v, denoted by $\mathsf{E}(W_{uv})$, is equal to

$$\mathsf{E}(W_{uv}) = \lim_{t \to \infty} \frac{\int_0^t B(\tau) d\tau}{t}$$

which is the average residual time. To obtain $\mathsf{E}(W_{uv})$ using renewal reward theory, we simply associate a reward that is equal to the residual time $B(t)$. Let $R(t)$ denote the total accumulated reward until time t. Then we have

$$R(t) = \int_0^t B(\tau)d\tau$$

Using Proposition 7.3 in [103] (page 433), we have

$$\mathsf{E}(W_{uv}) = \lim_{t\to\infty} \frac{R(t)}{t} = \frac{\mathsf{E}(R)}{\mathsf{E}(D)}, \tag{3.3}$$

where $\mathsf{E}(R)$ is the expected reward per renewal cycle or interval and $\mathsf{E}(D)$ is the expected duration of an interval. The former quantity can be easily obtained as follows:

$$\mathsf{E}(R) = \mathsf{E}\left\{\int_0^D (D-t)dt\right\} = \frac{\mathsf{E}(D^2)}{2}. \tag{3.4}$$

Substituting (3.4) in (3.3), we have

$$\mathsf{E}(W_{uv}) = \frac{\mathsf{E}(D^2)}{2\mathsf{E}(D)}. \tag{3.5}$$

Noting that $\mathsf{E}(D^2) = \mathsf{E}^2(D) + \mathsf{Var}(D)$ where $\mathsf{Var}(D)$ is the variance of $\{D_i\}$, we finally obtain (3.2).

□

Note that the result in Lemma 1 is not unique to EPWSNs. In fact, a similar result on bus waiting times have been shown in transportation studies [91]. The above result can also be explained by the waiting time paradox (also known as "inspection" paradox) [102] in renewal theory. In terms of packets, the paradox states that it is more likely for a packet to become ready for transmission at a larger interval than a shorter interval. The net effect is that the average waiting time will be higher than the typical value.

For a cycle k with duration T and $n_v(k)$, (3.5) can be rewritten as

$$\mathsf{E}(W_{uv}) = \frac{T}{2n_v(k)}\left(1 + C^2(D)\right), \tag{3.6}$$

where $C(D) = \frac{\sqrt{\mathsf{Var}(D)}}{\mathsf{E}(D)}$ is the coefficient of variation. (3.6) seems to imply that $\mathsf{E}(W_{uv})$ can be reduced by decreasing the duty cycle duration T. Recall however from (3.1) that $n_v(k) \propto T\delta_v(k)$ which really implies that $\mathsf{E}(W_{uv}) \propto \frac{1}{\delta_v(k)}$. Hence, given a duty cycle to operate on, the only other way to reduce $\mathsf{E}(W_{uv})$ is by reducing the variance or the coefficient of variation of the interval between receive slots.

Using Lemma 1, the ideal schedule is composed of equally-spaced receive wakeup slots. For a node v with $n_v(k)$ receive slots, the schedule

$$\mathbf{\Gamma}_v(k) = \left\{ \left\lceil \frac{S}{n_v(k)} \right\rceil n, n = 0, 1, 2, ..., n_v(k) - 1 \right\} \tag{3.7}$$

is the most ideal as it provides zero variance. However, one major disadvantage of (3.7) is that it is not *robust* to changes in $n_v(k)$. We shall discuss the notion of schedule "robustness" in Section 3.2.1.

3.2.1 Schedule Robustness

Consider two consecutive cycles k and $k+1$. Suppose that $n_v(k) \neq n_v(k+1)$, then the receive slots in $\mathbf{\Gamma}_v(k+1)$ may be entirely different from the receive slots in $\mathbf{\Gamma}_v(k)$. The implication is that if some other node u fails to receive a schedule update from v, u will transmit at a slot where v is not likely to be awake. We formally define this concept as follows.

Definition 6 (Schedule Robustness). *Given two cycles k and k', where $k \neq k'$, $n_v(k) \neq n_v(k')$ and $n_v(k), n_v(k') \neq 0$, let $\rho(k, k')$ be defined as*

$$\rho(k, k') = \frac{|\mathbf{\Gamma}_v(k) \cap \mathbf{\Gamma}_v(k')|}{\min[n_v(k), n_v(k')]}. \tag{3.8}$$

A schedule $\mathbf{\Gamma}_v$ is robust if for any k and k', $\rho(k, k') = 1$.

Note that in general, $0 \leq \rho(k, k') \leq 1$. A schedule with $\rho(k, k') = 0$ means that the receive slots in $\mathbf{\Gamma}_v(k)$ are entirely different from $\mathbf{\Gamma}_v(k')$ while a schedule with $\rho(k, k') = 1$ means that the receive slots in $\mathbf{\Gamma}_v(k)$ and $\mathbf{\Gamma}_v(k')$ are the same except for additional slots in one of the schedules.

It is easy to see that the schedule defined in (3.7) is not robust according to Definition 6. Consider two cycles k and k'. Let $a = \lceil \frac{S}{n_v(k)} \rceil$ and $b = \lceil \frac{S}{n_v(k')} \rceil$. Without loss of generality, suppose $a > b$ and let m be the least common multiple of the two numbers. Then $\mathbf{\Gamma}_v(k) \cap \mathbf{\Gamma}_v(k') = \{mn, n = 0, 1, 2, ..., r-1\}$. We then obtain r as follows. Since $a > b$, then $|\mathbf{\Gamma}_v(k)| < |\mathbf{\Gamma}_v(k')|$. The maximum slot number in $\mathbf{\Gamma}_v(k)$ is therefore the maximum possible common slot in the two schedules. That is,

$$\begin{aligned} m(r-1) &\leq a[n_v(k) - 1] \\ r &\leq \frac{a[n_v(k) - 1]}{m} + 1 \end{aligned} \tag{3.9}$$

Solving for $\rho(k, k')$ yields

$$\rho(k, k') \leq \frac{a}{m} + \frac{1}{n_v(k)} \left(1 - \frac{a}{m}\right),$$

which can only be at most 1 when either $a = m$ (*i.e.*, the schedules are the same) or $n_v(k) = 1$ (*i.e.*, one of the schedules has only one receive slot). The following lemma clarifies why a robust schedule is desirable.

Lemma 2. *Let k and k' be two cycles where $k < k'$, $n_v(k) \neq n_v(k')$ and $n_v(k), n_v(k') \neq 0$. Consider a node u with knowledge of $\mathbf{\Gamma}_v(k)$ alone and needs to transmit at k'. For a robust schedule, the probability that u transmits at a slot where v is awake, denoted by* $\mathsf{Pr}(c)$, *is given by*

$$\mathsf{Pr}(c) = \begin{cases} 1 & \textit{if } n_v(k) < n_v(k') \\ \frac{n_v(k')}{n_v(k)} & \textit{if } n_v(k) > n_v(k'). \end{cases} \tag{3.10}$$

Proof. If $n_v(k) < n_v(k')$, then the proof is obvious. The new schedule $\mathbf{\Gamma}_v(k')$ has more slots and that $\mathbf{\Gamma}_v(k) \subset \mathbf{\Gamma}_v(k')$. Since u knows $\mathbf{\Gamma}_v(k)$, it will transmit at $\eta \in \mathbf{\Gamma}_v(k) \Rightarrow \eta \in \mathbf{\Gamma}_v(k')$. Hence $\mathsf{Pr}(c) = 1$.

If $n_v(k) > n_v(k')$ the new schedule $\mathbf{\Gamma}_v(k')$ has fewer slots. Since u knows $\mathbf{\Gamma}_v(k)$

which has $n_v(k)$ slots and that $n_v(k')$ of these slots are contained in $\mathbf{\Gamma}_v(k')$, then $\mathsf{Pr}(c) = \frac{n_v(k')}{n_v(k)}$. □

A subtle implication of Lemma 2 is that if a node u does not know the new schedule of v but has knowledge of its old schedule, u can improve its chances of successfully sending to v by being conservative with its estimate of the number of slots in the new schedule.

3.2.2 Sequence-Based Wakeup Schedule

From Definition 4, we can view a receive wakeup schedule $\mathbf{\Gamma}_v(k)$ as a set of integers with cardinality of $n_v(k)$. Exchanging and storing raw schedules (*i.e.*, the entire contents of $\mathbf{\Gamma}_v(k)$) may therefore entail high overhead especially if $n_v(k)$ is high. To address this, we propose a *sequence-based wakeup schedule* pattern for exchanging and storing schedules in a compact manner.

Definition 7 (Sequence-Based Wakeup Schedule). *Let t_n be an integer sequence that satisfies the following conditions:*

(a) $0 \leq t_n \leq S-1, n = 0, 1, 2, ..., S-1$

(b) $t_m \neq t_n, \forall m \neq n$

A wakeup schedule of v at k is sequence-based if $\mathbf{\Gamma}_v(k) = \{t_n, n = 0, 1, 2, ..., n_v(k) - 1\}$.

Conditions (a) and (b) will ensure that the generated sequence will contain every possible slot number $\eta = 0, 1, 2, ..., S-1$ exactly once. With a sequence-based schedule, $\mathbf{\Gamma}_v(k)$ can be effectively specified by the tuple $\{t_n, n_v(k)\}$. If the sequence t_n is the same for all nodes, then only $n_v(k)$ is needed to completely specify $\mathbf{\Gamma}_v(k)$. Another important property of a sequence-based schedule is that it is robust according to Definition 6. We clarify this fact in the following lemma.

Lemma 3. *A sequence-based wakeup schedule is robust.*

Proof. Let k and k' be two cycles, where $k \neq k'$ and $n_v(k), n_v(k') \neq 0$. Without loss of generality, assume that $n_v(k) < n_v(k')$. Then $\mathbf{\Gamma}_v(k) = \{t_n, n = 0, 1, 2, ..., n_v(k) - 1\}$ and $\mathbf{\Gamma}_v(k') = \{t_n, n = 0, 1, 2, ..., n_v(k') - 1\}$, that is, the two schedules have the same terms up to $n_v(k)$ (since $n_v(k) < n_v(k')$). Hence, $|\mathbf{\Gamma}_v(k) \cap \mathbf{\Gamma}_v(k')| = n_v(k)$. Using (3.8), $\rho(k, k') = n_v(k)/n_v(k) = 1$ for any k and k'. □

3.2.3 Bit-Reversal Permutation Sequence

Definition 7 provides two conditions for an integer sequence t_n to be usable as a wakeup schedule generator. If we can obtain such t_n, then the schedule is guaranteed to be robust. We have also shown in Lemma 1 that a schedule with low variance can reduce the expected sleep latency. Thus, aside from satisfying the two conditions in Definition 7, we must also formulate t_n such that the generated schedule yields the minimum variance.

To obtain a suitable sequence, we proceed as follows. Let S be the number of slots in one cycle. If $n_v = 1$ (we drop the parameter k in this discussion as it is clear that we are at a specific cycle), then we can simply decide to position the slot, which we label as $\eta(0)$ at 0. (For the purpose of labeling the receive slots, we use the notation $\eta(i)$, where i is the index.) If $n_v = 2$, then we just add an active slot in the middle of the cycle at $S/2$ which we label $\eta(1)$. If $n_v = 3$, then we add a slot in the middle of $\eta(0)$ and $\eta(1)$. This new slot labeled $\eta(2)$ is at $S/4$. Figure 3.4 shows an example up to $n_v = 8$. Observe that for this method to work, S, $S/2$, $S/4$, $S/8$, ..., $S/(S/2)$, S/S must be integers. In other words, S must be a power of 2.

Table 3.1 provides a summary of the slot positions using the method above. We also show in columns 3 to 5 that when the numerator coefficient is represented in 3-bit binary, its bit-reversal yields the index of the slot label (column 1). The sequence in column 3 can be actually generated using *bit-reversal permutation* [70].

Definition 8. *Consider an integer $n \in [0, 2^r - 1]$ with binary representation*

$$(b_{r-1}, b_{r-2}, ..., b_1, b_0),$$

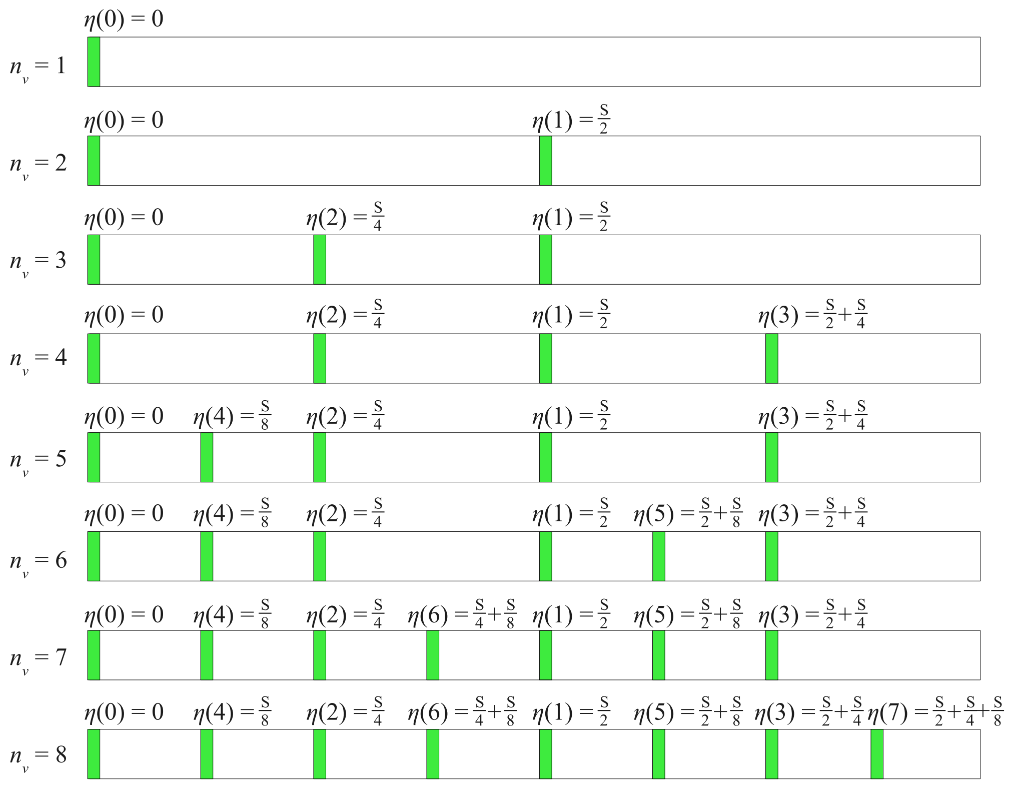


Figure 3.4: Illustrating the idea behind the proposed sequence for generating wakeup schedules.

Table 3.1: Summary of the Example Sequence for Generating Schedule

Label	Slot position	Numerator coefficient	Numerator coefficient (3-bit binary)	Bit reversal and decimal value
$\eta(0)$	0	0	000	$000 \Rightarrow 0$
$\eta(1)$	$\frac{S}{2} = \frac{4S}{8}$	4	100	$001 \Rightarrow 1$
$\eta(2)$	$\frac{S}{4} = \frac{2S}{8}$	2	010	$010 \Rightarrow 2$
$\eta(3)$	$\frac{S}{2} + \frac{S}{4} = \frac{6S}{8}$	6	110	$011 \Rightarrow 3$
$\eta(4)$	$\frac{S}{8}$	1	001	$100 \Rightarrow 4$
$\eta(5)$	$\frac{S}{2} + \frac{S}{8} = \frac{5S}{8}$	5	101	$101 \Rightarrow 5$
$\eta(6)$	$\frac{S}{4} + \frac{S}{8} = \frac{3S}{8}$	3	011	$110 \Rightarrow 6$
$\eta(7)$	$\frac{S}{2} + \frac{S}{4} + \frac{S}{8} = \frac{7S}{8}$	7	111	$111 \Rightarrow 7$

where $b_i \in \{0, 1\}$. Then its bit-reversal in r bits is

$$(b_0, b_1, ..., b_{r-2}, b_{r-1}).$$

If $B(n, r)$ denotes the decimal value of the bit-reversal of n in r bits, then

$$B(n, r) = \sum_{i=0}^{r-1} b_i 2^{r-1-i}.$$

We are now ready to generalize the method for obtaining the slot position of $\eta(n), n = 0, 1, 2, ..., n_v - 1$. Let $S = 2^s$. Suppose that $2^{a-1} \leq n_v < 2^a \leq 2^s$. Then we have

$$\eta(n) = B(n, a)\frac{S}{2^a} = B(n, a)\frac{2^s}{2^a}.$$

It is straightforward to show that $\eta(n)$ satisfies the conditions in Definition 7. The minimum value of $B(n, a)$ is the bit-reversal of $000 \cdots 000$ which is 0. Hence the minimum value of $\eta(n)$ is also 0. Whereas, the maximum value of $B(n, a)$ is the bit-reversal of $111 \cdots 111$ which is $2^a - 1$. Hence the maximum value of $\eta(n)$ is $(2^a - 1)\frac{2^s}{2^a} = 2^s - \frac{2^s}{2^a} \leq 2^s - 1$. Also, every n has a unique a-bit representation. Bit-reversal also yields a unique a-bit representation which yields a unique value. Multiplying this value by $\frac{2^s}{2^a}$ does not affect its uniqueness. Hence, $\eta(n)$ is an admissible sequence.

Expected Waiting Time We now obtain the expected sleep latency that results when bit-reversal permutation is used to generate a schedule. For conciseness, we drop the parameter k in $n_v(k)$ as it is clear that the results apply to a specific cycle or epoch k.

Theorem 1. *The expected sleep latency incurred using a bit-reversal permutation sequence schedule is*

$$\mathsf{E}(W_{uv}) = \frac{T}{2n_v}\left(1 + \frac{1}{2^{2a+1}}(n_v - 2^a)(2^{a+1} - n_v)\right), \tag{3.11}$$

where $n_v > 0$ is the number of receive slots and $a = \lfloor \log_2 n_v \rfloor$.

Proof. $a = \lfloor \log_2 n_v \rfloor \Rightarrow 2^a \leq n_v < 2^{a+1}$. The first 2^a slots divide T equally such that the interval between slots is $T/2^a$. The excess slots $x = n_v - 2^a$ further divides x of the 2^a intervals into 2. Thus, finally there are $2x$ intervals with duration $T/2^{a+1}$ and $n_v - 2x = 2^{a+1} - n_v$ intervals with duration $T/2^a$. Since $\mathsf{E}(D) = T/n_v$, the variance $\mathsf{Var}(D)$ of the intervals is

$$\begin{aligned}\mathsf{Var}(D) &= \left(\frac{2n_v - 2^{a+1}}{n_v}\right)\left(\frac{T}{2^{a+1}} - \frac{T}{n_v}\right)^2 + \\ &\quad \left(\frac{2^{a+1} - n_v}{n_v}\right)\left(\frac{T}{2^a} - \frac{T}{n_v}\right)^2 \\ &= \left(\frac{1}{2^{2a+1}}\right)\left(\frac{T}{n_v}\right)^2 (n_v - 2^a)(2^{a+1} - n_v). \qquad (3.12)\end{aligned}$$

Solving for $C^2(D) = \mathsf{Var}(D)/\mathsf{E}^2(D) = \mathsf{Var}(D)/(T/n_v)^2$ and substituting in (3.6), we obtain (3.11).

□

Note that when n_v is a power of 2, $n_v = 2^a$ which causes the coefficient of variation to vanish. The following corollary further shows that $C^2(D)$ has an upper bound and that $\mathsf{E}(W_{uv})$ therefore has an upper bound.

Corollary 1. *The expected sleep latency incurred using a bit-reversal permutation sequence schedule is bounded and* $0.5\frac{T}{n_v} \leq \mathsf{E}(W_{uv}) \leq 0.5625\frac{T}{n_v}$.

Proof. Since $2^a \leq n_v < 2^{a+1}$, let $d = 2^{a+1} - 2^a = 2^a$. Then for $0 \leq m < 2^a$, we can rewrite $(n_v - 2^a)(2^{a+1} - n_v)$ as $f(m) = (d - m)m = (2^a - m)m = 2^a m - m^2$. Solving for $f'(m)$ and $f''(m)$,

$$f'(m) = 2^a - 2m$$

and

$$f''(m) = -2.$$

Since $f''(m) < 0$, $f(m)$ is concave and setting $f'(m) = 0$ will yield the maximum value of $f(m)$. Doing this gives us $m = 2^{a-1}$. Hence, $n_v = 2^a + 2^{a-1}$ will give the

Table 3.2: Upper Bound of the Expected Sleep Latency under Different Distributions

Slot Distribution	Upper bound
Ideal (equally-spaced slots)	$0.5\frac{T}{n_v}$
Bit-reversal permutation sequence	$0.5625\frac{T}{n_v}$
Uniform	$0.6667\frac{T}{n_v}$
Exponential	$\frac{T}{n_v}$

maximum $C^2(D)$. Therefore,

$$C^2(D) = \frac{(2^a + 2^{a-1} - 2^a)(2^{a+1} - 2^a + 2^{a-1})}{2^{2a+1}} = \frac{1}{8} = 0.125.$$

Substituting this in (3.11), we finally have the upper bound $\mathsf{E}(W_{uv}) = 0.5625\frac{T}{n_v}$.

□

In Table 3.2, we compare the upper bound of the expected sleep latency under different distributions. For the uniform distribution, we let the slot intervals range from 0 to $2T/n_v$ such that the expected value is still T/n_v. The coefficient of variation of exponential distribution is always 1. Note that bit-reversal permutation sequence schedule yields lower expected sleep latency compared with uniform or exponential.

Per Node Sequence The disadvantage of using the same sequence $\eta(n)$ in all of the nodes is that they will have common wakeup schedules. This is not a desirable situation because more nodes may transmit at the same slot even though their intended receivers are different. Although traffic flow is assumed to be low, this may still result in higher occurrence of packet collisions. To reduce common receive wakeup slots across nodes, we introduce an offset to $\eta(n)$ for every node v as follows,

$$t_n = [v + \eta(n)] \mod S,$$

where v is the ID of node v. The modulo operation is necessary to ensure that t_n does not exceed $S - 1$.

Computational Complexity One of the key advantages of the proposed scheme is its low computational complexity. As a matter of fact, it does not require the computation of any schedule as it only needs to compute the number of receive wakeup slots. Hence, its complexity is $O(1)$.

Scheduling Overhead In terms of communication overhead, the advantage of a sequence-based wakeup schedule is that for any node v in the network, it only needs to send $n_v(k)$. In terms of storage overhead, the compact representation of the wakeup schedule in terms of $n_v(k)$ requires considerably lower overhead compared to schemes that require the storage of the entire schedule. If v has to store the schedule of every neighbor, then the overhead of a sequence-based wakeup schedule is $O(|\mathbf{N}_v|)$. Whereas, the overhead of schemes that require the storage of entire schedules is $O(|\mathbf{N}_v|S)$, where S is the number of slots in the cycle.

3.3 Low Latency and Reliable Forwarding

So far, our development assumed that the links are ideal and that packet transmissions are always successful on the first attempt. However in practical deployments, links in wireless sensor networks are far from being ideal [50]. The impact of lossy links is to essentially increase the expected sleep latency as will be shown in the following motivating example.

3.3.1 Motivating Example

In the example shown in Figure 3.5, the number of receive wakeup slots of nodes 2 and 3 are indicated. p_{vw} denotes the packet delivery probability from v to w. Thus, we can see that the link $(1,3)$ has better quality than link $(1,2)$. If only sleep latency were considered, then the selected forwarder would be node 2 since $\mathsf{E}(W_{12}) = \frac{T}{2n_2(k)} = \frac{T}{32} < \mathsf{E}(W_{13}) = \frac{T}{2n_3(k)} = \frac{T}{16}$. But because the delivery probability across $(1,2)$ is only 0.4, this implies that a packet needs to be transmitted $1/0.4 = 2.5$ on

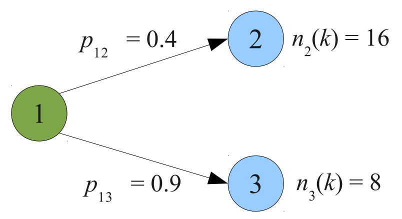


Figure 3.5: An example illustrating the adverse effect of neglecting link quality in forwarder selection. p_{vw} denotes the packet delivery probability from v to w.

the average[4] before being successfully received by node 2. Hence, the total latency would be roughly $(2.5)\mathsf{E}(W_{12}) = \frac{2.5T}{32}$. Whereas, for link $(1,3)$ which has a link quality of 0.9, the total latency would be roughly $(1.1)\mathsf{E}(W_{13}) = \frac{2.2T}{32}$.

3.3.2 Expected Transmission Delay

The example above clearly shows that considering sleep latency alone in selecting the next hop may result in poor performance. On the other hand, considering link quality alone will also not necessarily lead to the best performance. This is because the next hop node with the highest link quality might have a very high sleep latency. We therefore define a new metric that combines sleep latency and link quality.

Definition 9 (Transmission Delay). *The transmission delay between adjacent nodes u and v, denoted by $\mathcal{D}_{uv}$, is the delay from the time a packet becomes ready for transmission at u until it is successfully received by v.*

We now obtain the expected transmission delay (ETD) $\mathsf{E}(\mathcal{D}_{uv})$ from u to v. For a transmission to be deemed successful, the receiving node v must receive the data packet *and* the sending node u must receive the corresponding ACK packet. If p_{uv} and p_{vu} are the two-way delivery probabilities of link (u, v), then a packet is considered successfully transmitted with probability $p_{uv}p_{vu}$. Following the same argument as in [31], the expected number of transmissions is $\frac{1}{p_{uv}p_{vu}}$. Given that for every transmission attempt the delay is $\mathsf{E}(W_{uv})$, we have

[4]This argument is explained as follows. Each transmission attempt across link (u, v) can be considered as a Bernoulli trial. Because the probability of success is p_{uv}, then from elementary probability, the expected number of transmission attempts is simply $1/p_{uv}$.

$$\mathsf{E}(\mathcal{D}_{uv}) = \frac{\mathsf{E}(W_{uv})}{p_{uv}p_{vu}}. \tag{3.13}$$

Expected Transmission Delay of a Path Finally, the ETD of a path $\mathbf{P}$ from s to t, denoted by $\mathsf{E}(\mathcal{D}_{st})$, is defined as the expected delay measured from the time a packet becomes ready for transmission at s until it is successfully delivered to t. This is simply given by

$$\mathsf{E}(\mathcal{D}_{st}) = \sum_{\forall (u,v)\in\mathbf{P}} \mathsf{E}(\mathcal{D}_{uv}). \tag{3.14}$$

Distributed Calculation The search for the minimum ETD path from every sensor node v to the data collection point t can be performed in a distributed manner using the distributed Bellman-Ford algorithm. However, for such an algorithm to yield consistent, loop-free and minimum cost paths, we have to show that the ETD metric is both left-monotonic and left-isotonic [128].

Lemma 4. *The ETD metric is left-monotonic and left-isotonic.*

Proof. To show left-monotonicity, consider a path A from v to w with cost $\mathsf{E}(\mathcal{D}_A)$. Suppose that we prepend a path C from u to v with cost $\mathsf{E}(\mathcal{D}_C)$. Then the cost of the path from u to w through (C, A) is $\mathsf{E}(\mathcal{D}_{CA}) = \mathsf{E}(\mathcal{D}_C) + \mathsf{E}(\mathcal{D}_A)$. Clearly, $\mathsf{E}(\mathcal{D}_A) \leq \mathsf{E}(\mathcal{D}_{CA})$, proving its left-monotonicity.

To show left-isotonicity, consider two paths A and B from v to w with costs $\mathsf{E}(\mathcal{D}_A)$ and $\mathsf{E}(\mathcal{D}_B)$, respectively, and with $\mathsf{E}(\mathcal{D}_A) \leq \mathsf{E}(\mathcal{D}_B)$. Now, suppose that we prepend a path C from u to v with cost $\mathsf{E}(\mathcal{D}_C)$. Then the cost of the path from u to w through (C, A) is $\mathsf{E}(\mathcal{D}_{CA}) = \mathsf{E}(\mathcal{D}_C) + \mathsf{E}(\mathcal{D}_A)$ while the cost of the path from u to w through (C, B) is $\mathsf{E}(\mathcal{D}_{CB}) = \mathsf{E}(\mathcal{D}_C) + \mathsf{E}(\mathcal{D}_B)$. Clearly, $\mathsf{E}(\mathcal{D}_{CA}) \leq \mathsf{E}(\mathcal{D}_{CB})$, proving its left-isotonicity. ☐

Before ending the discussion on ETD, we highlight the following important differences between ETD and the energy-aware metrics presented in Section 2.2.4:

- ETD considers both sleep latency and packet loss. While several schemes have incorporated the latter (notably MultihopLQI and ETX), none of the state-of-the-art routing metrics have included sleep latency in their respective formulations.

- Sleep latency implicitly considers energy availability and dynamic duty cycling since sleep latency is inversely proportional to both, *i.e.*, the higher the energy and duty cycle, the lower the sleep latency.

- Unlike GREES-L and GREES-M that employ highly dynamic physical quantities such as energy harvesting rate, energy consumption, and fraction of energy used, ETD's use of sleep latency as a proxy to energy availability not only simplifies computation but also enhances its stability. Moreover, ETD considers packet loss and does not require physical node locations.

- Unlike the energy distance metric, we have shown that ETD is left-monotonic and left-isotonic, implying that it will yield consistent and loop-free paths. More importantly, ETD does not require the formulation of a penalty function and also considers packet loss.

3.3.3 Protocol Overview

We now present the details of a forwarding scheme that incorporates the BRPS and ETD to perform minimum-cost path computation and packet forwarding. As a matter of notation, we maintain the variable naming conventions but affix the node ID as superscript to indicate that a variable is maintained by a node.

State Variables and Data Structures Every node v maintains two global state variables and a neighbor table. The global variables are S^v_{vt} and $\mathsf{E}(\mathcal{D}^v_{vt})$ which are the ID of the next hop node and the minimum ETD, respectively, to the sink node t. The neighbor table contains an information tuple $(n^v_u, \mathsf{E}(\mathcal{D}^v_{vu}), \mathsf{E}(\mathcal{D}^v_{ut}), p^v_{vu}, p^v_{uv})$ about every node $u \in \mathbf{N}_v$, where n^v_u is the number of receive slots of u, $\mathsf{E}(\mathcal{D}^v_{vu})$ is the ETD from v to u, $\mathsf{E}(\mathcal{D}^v_{ut})$ is the minimum ETD from u to t, p^v_{vu} is the link delivery probability from v to u and p^v_{uv} is the link delivery probability from u to v.

Initial Values After a node has completed performing neighbor discovery using some suitable protocol (*e.g.*, [86]), it initializes every neighbor information tuple as follows: $n_u^v \leftarrow 0$, $\mathsf{E}(\mathcal{D}_{vu}^v) \leftarrow \infty$, $\mathsf{E}(\mathcal{D}_{ut}^v) \leftarrow \infty$, $p_{vu}^v \leftarrow 0$, and $p_{uv}^v) \leftarrow 0$. More importantly, it initializes its state variable $\mathsf{E}(\mathcal{D}_{vt}^v)$ as follows:

$$\mathsf{E}(\mathcal{D}_{vt}^v) = \begin{cases} 0 & \text{if } v \text{ is the sink node } t \\ \infty & \text{otherwise.} \end{cases} \tag{3.15}$$

Control Update and Node Update Slot At every cycle k, every node v must broadcast an update packet $\textsc{Update}(n_v(k), \mathsf{E}(\mathcal{D}_{vt}(k)), \{p_{uv}(k), \forall u \in \mathbf{N}_v\})$ at a designated *node update slot*. For simplicity, we assign the slot $(v \mod S)$ as the node update slot of v. Hence, every neighbor of v must wakeup at slot $(v \mod S)$ to listen for updates from v and must not use that slot for its own packet transmission.

Algorithm 1 Algorithm for processing an update packet.

```
1: if received $\textsc{Update}(n_u(k), \mathsf{E}(\mathcal{D}_{ut}(k)), \{p_{wu}(k), \forall w \in \mathbf{N}_u\})$ from $u$ then
2:     $n_u^v \leftarrow n_u(k)$
3:     $\mathsf{E}(\mathcal{D}_{ut}^v) \leftarrow \mathsf{E}(\mathcal{D}_{ut}(k))$
4:     $p_{vu}^v \leftarrow p_{vu}(k)$
5:     $p_{uv}^v \leftarrow$ PHY LQI estimate of $p_{uv}$
6: else
7:     $n_u^v \leftarrow \lfloor \alpha n_u^v \rfloor$
8: end if
```

Control Update Processing When v receives an update packet from u, it updates the corresponding information tuple for u in its neighbor table as indicated in lines (2)-(5) of Algorithm 1. Because the radio transceiver provides link quality information, v can also obtain an estimate of p_{uv}^v. Studies [23, 54] have shown that LQI is highly-correlated with packet delivery probability and can therefore be used to obtain the latter. If v does not receive an update packet from u on its designated slot, v sets $n_u^v \leftarrow \lfloor \alpha n_u^v \rfloor$, where $0 < \alpha < 1$. We refer to the parameter α

as the *receive slot discount factor*. By conservatively estimating $n_u(k)$, v is essentially improving its probability of successful transmission to u. This is possible because of the robustness property of the scheduling scheme as shown in Lemma 2.

3.3.4 Path Computation

Algorithm 2 provides a listing of the algorithm that is executed at every node v to obtain the minimum cost path from v to t. The algorithm is executed at the end of every node update slot of every neighbor node u.

Infinite ETD to Neighbor u Lines (2)–(12) lists the steps that are executed when the ETD to a neighbor node u becomes infinite. The ETD to u becomes infinite whenever the number of receive slots of u becomes zero (which happens at initial state or after several non-reception of update packets) or link delivery probability estimates are not available. If u is the successor node to t, v searches for an alternative neighbor w that provides the minimum ETD path to t.

Finite ETD to Neighbor u Lines (14)–(21) lists the steps that are executed when the ETD to a neighbor node u is finite. Node v computes the ETD to u using (3.13). If the resulting ETD through u is less than the current minimum ETD, then u is chosen as the new next hop node and the corresponding ETD is set as the minimum ETD from v to t.

Convergence Lines (5)–(11) and (18)–(21) ensure that after every execution of Algorithm 2, v either has infinite cost to t or a finite cost which satisfies the following:

$$\mathsf{E}(\mathcal{D}^v_{vt}) = \min_{u \in \mathrm{N}_v} \left[\mathsf{E}(\mathcal{D}^v_{vu}) + \mathsf{E}(\mathcal{D}^v_{ut})\right] \tag{3.16}$$

This equation is essentially the update rule of the distributed Bellman-Ford algorithm [18]. Together with the initial values in (3.15) and neighbor update processing in Algorithm 1, (3.16) ensures that the computation will converge to the correct minimum cost within finite time [18].

Algorithm 2 Distributed algorithm for path computation.

```
1: if $p^v_{uv} = 0$ or $p^v_{vu} = 0$ or $n^v_u = 0$ then
2:     $\mathsf{E}(\mathcal{D}^v_{vu}) \leftarrow \infty$
3:     if $S^v_{vt} = u$ then
4:         $\mathsf{E}(\mathcal{D}^v_{vt}) \leftarrow \infty$
5:         for $w \in \mathbf{N}_v \backslash u$ do
6:             $M \leftarrow \mathsf{E}(\mathcal{D}^v_{vw}) + \mathsf{E}(\mathcal{D}^v_{wt})$
7:             if $M < \mathsf{E}(\mathcal{D}^v_{vt})$ then
8:                 $\mathsf{E}(\mathcal{D}^v_{vt}) \leftarrow M$
9:                 $S^v_{vt} \leftarrow w$
10:            end if
11:        end for
12:    end if
13: else
14:    $a \leftarrow \lfloor \log_2 n^v_u \rfloor$
15:    $\mathsf{E}(W^v_{vu}) \leftarrow \frac{T}{2n^v_u} \left(1 + \frac{1}{2^{2a+1}}(n^v_u - 2^a)(2^{a+1} - n^v_u)\right)$
16:    $\mathsf{E}(\mathcal{D}^v_{vu}) \leftarrow \frac{\mathsf{E}(W^v_{vu})}{p^v_{uv} p^v_{vu}}$
17:    $M \leftarrow \mathsf{E}(\mathcal{D}^v_{vu}) + \mathsf{E}(\mathcal{D}^v_{ut})$
18:    if $M < \mathsf{E}(\mathcal{D}^v_{vt})$ then
19:        $\mathsf{E}(\mathcal{D}^v_{vt}) \leftarrow M$
20:        $S^v_{vt} \leftarrow u$
21:    end if
22: end if
```

3.3.5 Packet Forwarding

Algorithm 3 lists the packet forwarding algorithm of the protocol. Upon receipt of a data packet P from a neighbor node, from a local application, or from a previous unsuccessful transmission, v checks if it has a path to the sink. If it has no path, v drops P and the algorithm terminates. Otherwise, v checks the number of times that P has been transmitted previously. If P has been transmitted more than L, P is dropped and the algorithm terminates.

Slot Search If P is eligible for transmission, v searches for a time slot that is closest to the current slot in the schedule of the next hop node w. Node v ensures that a chosen transmit slot does not conflict with an update slot from a neighbor node or that the slot is not in v's own schedule. If a slot is found, P is scheduled for transmission at the specified slot $s + t_{\min}$. Otherwise, a *scheduling failure* is considered to have occurred and P is dropped.

Packet Transmission At the appropriate slot, v wakes up to initiate the transmission of the scheduled packet P. The packet is sent unicast to w and v waits for an ACK from w. When an ACK is received, v proceeds to schedule the next packet in the queue, if there is any. When no ACK is received, the number of retries counter r is incremented, and Algorithm 3 is invoked to try to schedule P again.

3.3.6 Reducing Control Overhead

As mentioned in the fourth paragraph of Section 3.3.3, every node in the network needs to broadcast an update packet exactly once every epoch. For convenience, every node v is assigned one *update slot* per epoch, in particular the slot $(v \mod S)$, to transmit such control packet.

In addition to the dedicated slot for transmitting update packets, every node must also wakeup at the update slot of all its known neighbors to know their respective wakeup schedules and other protocol parameters. As such, a node v may incur high overhead especially if it has a large number of neighbor nodes.

Algorithm 3 Packet forwarding algorithm.

```
1:  **if** $\mathsf{E}(\mathcal{D}_{vt}^{v}) = \infty$ **then**
2:    Drop $P$
3:    **return**
4:  **end if**
5:  $r \leftarrow$ Number of retransmissions of $P$
6:  **if** $r \geq L$ **then**
7:    Drop $P$
8:    **return**
9:  **end if**
10: $s \leftarrow$ Current slot
11: $w \leftarrow S_{vt}^{v}$
12: $\mathbf{\Gamma}_v \leftarrow \{t_n, n = 0, 1, 2, ..., n_v(k) - 1\}$
13: $\mathbf{\Gamma}_w \leftarrow \{t_w(n), n = 0, 1, 2, ..., n_w^v - 1\}$
14: $t_{\min} \leftarrow \infty$
15: **for** $\eta \in \mathbf{\Gamma}_w$ **do**
16:   **if** $\eta = u \mod S,\ \forall u \in \mathbf{N}_v$ or $\eta \in \mathbf{\Gamma}_v$ **then**
17:     continue
18:   **end if**
19:   $t_{\text{diff}} \leftarrow \eta - s$
20:   **if** $t_{\text{diff}} \leq 0$ **then**
21:     $t_{\text{diff}} \leftarrow t_{\text{diff}} + S$
22:   **end if**
23:   **if** $t_{\text{diff}} < t_{\min}$ **then**
24:     $t_{\min} \leftarrow t_{\text{diff}}$
25:   **end if**
26: **end for**
27: **if** $t_{\min} < \infty$ **then**
28:   Schedule $P$ for transmission at $s + t_{\min}$
29: **else**
30:   Drop $P$
31: **end if**
```

In what follows, we describe several mechanisms that can be used to reduce the overhead generated by the protocol:

Use Large Epoch Duration: Since update transmission and update reception (from every neighbor) is done on a per epoch basis, increasing the epoch duration could lower the overhead. Note however that a large epoch duration may slow down the convergence of the protocol, as shown in our simulation studies. As such, this approach may only be suitable in situations where the energy harvesting rates do not change rapidly.

Receive Updates from Selected Nodes: If a node is within a large neighborhood, it can limit its reception of update packets to a selected set of nodes. For instance, a node can choose to only receive updates from neighbors with routing metrics less than a certain *threshold*. This is because in the distributed computation of the optimal path, the most suitable successor node (or set of nodes) is likely to come from these nodes. The downside of this approach is that the set may turn out to be empty, resulting in the node being unnecessarily isolated. Another drawback is that due to changes in the environment, the optimal successor node may not belong to the chosen set, resulting in sub-optimal paths. Essentially, the challenge is for every node to choose the appropriate routing metric threshold that can reduce the set of nodes that it will listen to for updates.

Round Robin Listening: Another approach that can be employed to reduce the overhead entailed by neighbor update reception is to perform round robin listening. Instead of waking up to all neighbor updates every epoch, a node can choose to wakeup for at most n neighbors every epoch. If the node has more than n neighbors, it will just schedule the update reception by waking up for the first n neighbors in epoch k, the second n neighbors in epoch $k + 1$, and so on. The process will then repeat after all neighbors have been "listened to". Once again, this approach may only work well in situations where the energy harvesting rates do not change rapidly.

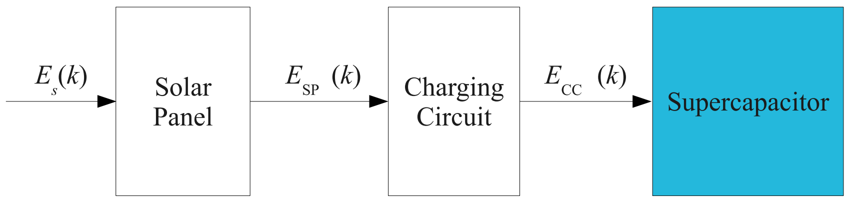


Figure 3.6: Solar energy harvesting source model.

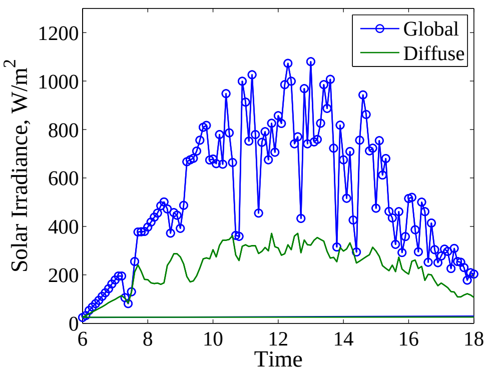
3.4 Simulation Models

To evaluate the performance of our proposed wakeup scheduling and forwarding scheme, we implemented a simulation model of the proposed wakeup scheduling and forwarding scheme in Qualnet [4], including other required components such as energy harvesting source and duty cycle controller. We set the slot duration τ to be 10 ms which is more than sufficient for the transmission of a 127-byte data packet (the maximum payload of an IEEE 802.15.4 PHY frame is 127 bytes [7]) and ACK.

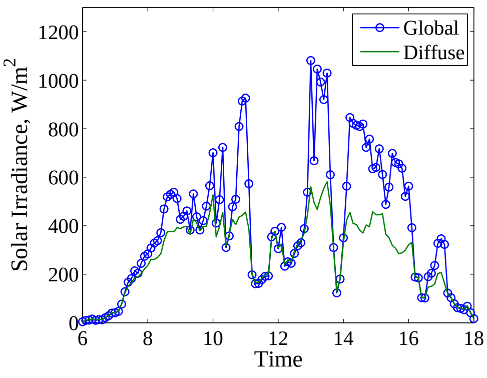
3.4.1 Energy Harvesting Source Model

Figure 3.6 shows the energy harvesting source model which is composed of a solar panel, a charging circuit, and a supercapacitor. This model captures the state-of-the-art in circuit design for micro-solar energy harvesting systems [26, 74]. The model requires solar irradiance data (in Watts per square meter) as input. For this purpose, we used real solar data traces from the National Renewable Energy Laboratory (NREL) [3]. We selected two days of solar radiation trace, representing "sunny" and "cloudy" scenarios, which are shown in Figure 3.7. For each node, the actual irradiance value used is a random number between the diffuse and global irradiance values.

The solar panel is characterized by its surface area A_{SP} and conversion efficiency η_{SP}. The energy output $E_{\mathrm{SP}}(k)$ of the solar panel at the kth epoch is given by $E_{\mathrm{SP}}(k) = A_{\mathrm{SP}}\eta_{\mathrm{SP}}E_s(k)$, where $E_s(k)$ is the solar irradiance. Whereas, the charging circuit is characterized by its charging efficiency η_{CC}. The amount of energy that reaches the supercapacitor is simply the output of the charging circuit which



(a) “Sunny” scenario solar irradiance



(b) “Cloudy” scenario solar irradiance

Figure 3.7: Solar irradiance data (at 6-minute resolution) from NREL Florida Solar Energy Center on July 2 and July 8, 2000.

is given by

$$E_{\mathrm{CC}}(k) = \eta_{\mathrm{CC}} E_{\mathrm{SP}}(k) = \eta_{\mathrm{CC}} A_{\mathrm{SP}} \eta_{\mathrm{SP}} E_s(k) = \eta_{EH} E_s(k).$$

We shall refer to $\eta_{EH} = \eta_{\mathrm{CC}} A_{\mathrm{SP}} \eta_{\mathrm{SP}}$ as the *effective harvesting efficiency*. In the simulations, we used $A_{\mathrm{SP}} = 0.01\ m^2$, $\eta_{\mathrm{SP}} = 0.1$, and $\eta_{\mathrm{CC}} = 0.5$, which yields $\eta_{EH} = 0.0005$. Note that these values are conservative compared to what is currently available or achievable in the literature. For instance, current photovoltaic cells can achieve as high as 25% energy conversion efficiency while state-of-the-art supercapacitor charging circuits can attain as high as 89% efficiency [74]. We also conduct simulations where η_{EH} is varied from 0.0001 to 0.0005. Finally, we used 25 Farad 4 Volt supercapacitor as energy buffer.

Before proceeding further, we would like to make the following remarks about the effect of energy harvesting types and the suitability of the design. Note that in this study, we selected solar energy to drive the simulations because real-world sensor motes such as Waspmote [5] require modest energy (in the order of 10s to 100s of milliwatts) to deliver usable performance. Nevertheless, the schemes that we have proposed (BRPS and ETD) are general and can be used in tandem with any environmental energy supply. Note that in the development of these schemes, we have not stipulated any strong assumptions or special requirements about the underlying environmental energy supply. The resulting performance, however, will depend on the gap between the amount of power that can be supplied by the source and the node consumption. The lower the gap, the better the performance because the sensor node can operate at higher duty cycles.

3.4.2 Duty Cycle Controller Model

The duty cycle controller is modeled after the LQ-Tracker algorithm proposed by Vigorito *et al.* [120] to attain energy-neutral operation (ENO). Our main motivation for using LQ-Tracker is that it represents the state-of-the-art in adaptive duty cycling algorithms and is easily implementable in simulations. We provide a brief description of the controller below while interested readers can refer to [120] for a detailed discussion of the controller.

The controller's objective is to achieve ENO-Max, which entails two simultaneous objectives: (*i*) to ensure that energy consumed is always less than or equal to the energy harvested; and (*ii*) to maximize task performance by maximizing energy consumption. To derive the adaptive control law, the authors modeled the dynamics of the battery level as a first order, discrete time, linear dynamical system with colored noise which conforms to

$$y(k+1) = ay(k) + bu(k) + cw(k) + w(k+1),$$

where y is the battery level, u is the control, w is a zero-mean input noise, and a, b, c are real-valued coefficients. The objective of the control system is to minimize the error $|y(k) - y^*|$ for all k, where y^* is the target battery level. Note that this objective is equivalent to minimizing the average squared tracking error

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \left[y(k) - y^*\right]^2, \tag{3.17}$$

which is the ENO-Max objective. The optimal control law that minimizes (3.17) is

$$u(k) = \frac{y^* - (a+c)y(k) + cy^*}{b}. \tag{3.18}$$

The authors proposed an on-line algorithm based on standard gradient descent techniques to estimate the coefficients a, b, and c.

3.4.3 Network Parameters

The network consists of 200 static nodes that are uniformly-distributed in a 500 m × 500 m area. A single sink node is positioned at (0, 0), *i.e.*, the bottom-left part of the area. Figure 3.8 shows the histogram of hop count to the sink of a typical scenario. We can see that the hop count ranges from 1 to 12 and that a large fraction of nodes are 5 to 10 hops away from the sink. The positioning of the sink at (0, 0) results in a challenging scenario where data traffic converges to a "narrow spot"

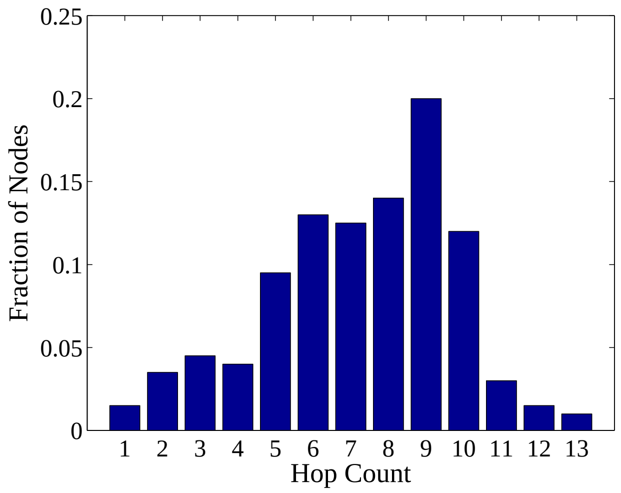


Figure 3.8: Hop count distribution of a typical scenario.

in the network. Note that the hop count statistics were obtained using ETX as the path metric. As such, the paths tend to be longer than expected, as ETX uses wireless links that are usually shorter.

The sensor node is modeled after a Libelium Waspmote equipped with XBee-802.15.4, an IEEE 802.15.4-compliant radio transceiver. The transceiver is configured to send at a data rate of 250 kbps and a transmit power of 0 dBm. With these configurations, the node consumes approximately 180 mW, 195 mW, and 240 μW in transmit, receive (active and idle), and deep sleep modes, respectively at 4 volts [5,6].

To model lossy wireless links, we modeled the packet reception model using the bit-error (BER) based reception model that is available in Qualnet [4]. Briefly, the model works as follows: (*i*) Upon receipt of a packet, the receiving node calculates the signal-to-interference plus noise ratio (SINR) of the received packet. (*ii*) The BER table is then consulted to obtain the bit error rate, given the SINR. In our simulations, we used the binary phase-shift keying (BPSK) error table. (*iii*) Let p_b denote the BER of the received packet. The packet reception success rate is then computed as

$$P_{suc} = (1 - p_b)^M,$$

where M is the packet length in bits. (*iv*) A random number R between 0 and 1 (inclusive) is then generated. If $R \leq P_{suc}$, then the packet is considered to be error-free and is therefore successfully received. Otherwise, the packet is corrupted and deemed not successfully received.

A constant bit rate (CBR) traffic generator is used to generate data traffic. Each data packet is 64 bytes and every node (except the sink) generates data at intervals of 60, 120, 180, 240, 300 seconds. These data generation rates are already considered to be high for certain environmental monitoring applications which require around 1 sample every 5 minutes [116]. Packet generation times are randomly staggered among different nodes. Each data point is obtained by averaging the results from 20 seed values, with every simulation run configured for 43,200 seconds (12 hours) in simulation time.

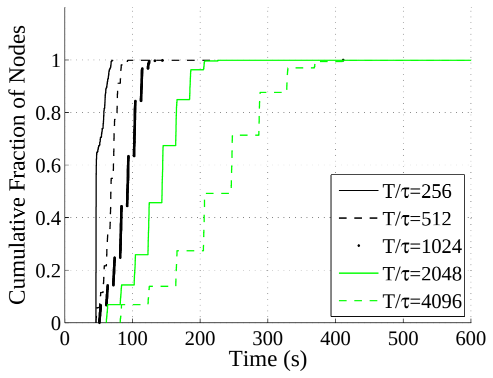
3.5 Simulation Results

We first evaluate the effect of the three design parameters, namely, the duty cycle duration T, maximum retry limit L, and receive slot discount factor α on the performance of the proposed scheduling and forwarding scheme.

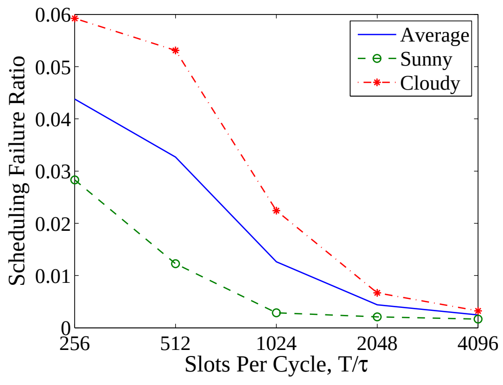
Duty Cycle Duration T To determine the effect of T, we varied the number of slots per cycle T/τ, such that $T/\tau \in \{256, 512, 1024, 2048, 4096\}$. Note that these values are exact powers of two as required by the bit-reversal permutation scheduling scheme. The smallest value of 256 is chosen since there are 200 nodes in the network. This ensures that every node will have its own slot for transmitting its update packet.

We first study the convergence time of the distributed path computation algorithm. To measure the convergence time, we forced the nodes to use fixed duty cycles throughout the simulation. The convergence time is affected by T since each node sends updates at an interval equal to T. As shown in Figure 3.9(a), the algorithm takes longer time to converge at higher T since the propagation of updates is slower.

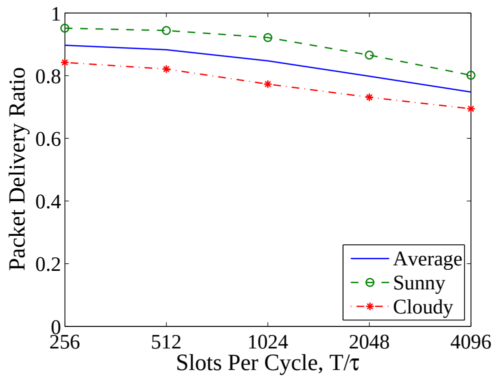
However, one advantage of using a larger T is that it reduces the occurrence of



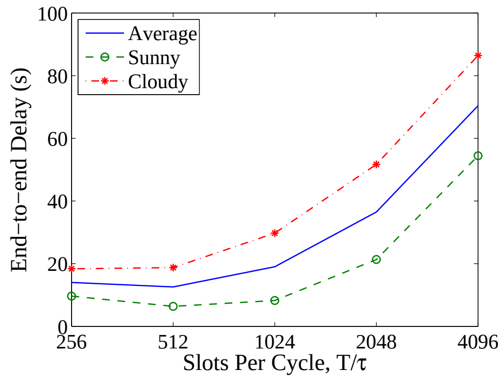
(a) Convergence time



(b) Scheduling failure



(c) Packet delivery ratio



(d) End-to-end delay

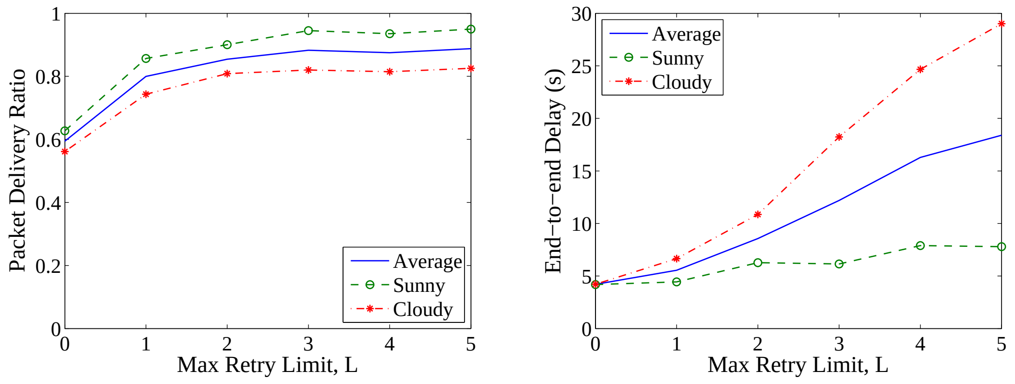
Figure 3.9: Effect of duty cycle duration.

scheduling failures (see Figure 3.9(b)). As discussed in Section 3.3.5, a scheduling failure occurs when the forwarding algorithm fails to find a receive wakeup slot at the intended receiver that does not conflict with neighbor update slots. Note that when T is small (*i.e.*, number of slots per cycle is small), there are fewer receive wakeup slots for the same duty cycle; hence the probability of successfully obtaining a non-conflicting receive wakeup slot is lower.

Ultimately, the packet delivery ratio and end-to-end delay results (see Figures 3.9(c) and 3.9(d)) show that using a smaller duty cycle duration provides better performance. The poor performance obtained when large T is used is due to the effect of slow convergence time. Especially in conditions where duty cycles are highly dynamic, slow convergence may result in the forwarding of packets through sub-optimal paths.

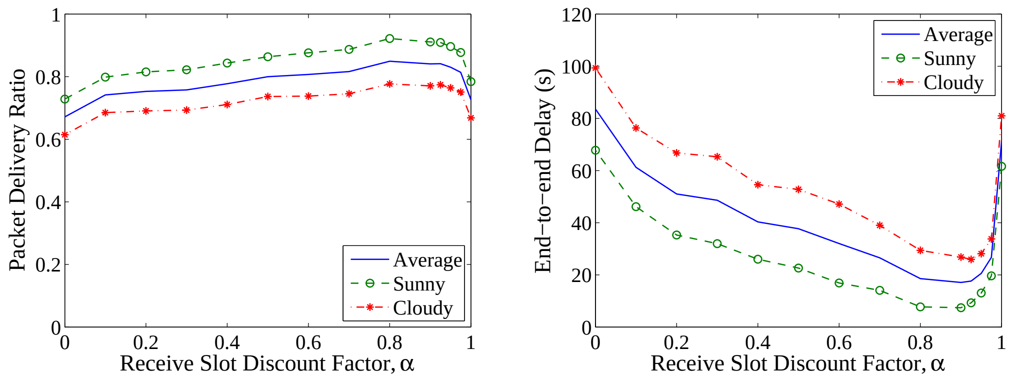
Maximum Retry Limit L The maximum retry limit L determines the level of reliability provided by the forwarding scheme. The packet delivery ratio results (see Figure 3.10(a)) demonstrate the positive effect of allowing higher number of retransmissions. The delivery ratio improves significantly when L is increased from 0 to 1; thereafter, the improvement is marginal. The downside of allowing higher retransmissions is increased end-to-end delay (see Figure 3.10(b)). The increase is more dramatic in the cloudy scenario; this is due to the fact that every packet retransmission incurs a higher latency because of the lower node duty cycles in the cloudy scenario, in which the receive slots are generally spaced further apart.

Receive Slot Discount Factor α The receive slot discount factor α in the proposed forwarding scheme is used to estimate the number of receive wakeup slots of a neighbor node u when node v fails to receive an update from u. A value of $\alpha = 0$ implies that whenever node v fails to receive an update from node u, the number of receive slots of u is set to 0 immediately, while $\alpha = 1$ means that the number of receive slots of u is retained without any change. We can see that these values are extreme and as shown by the results (see Figure 3.11), using either 0 or 1 does not provide the best performance. The optimal performance can be obtained



(a) Packet delivery ratio (b) End-to-end delay

Figure 3.10: Effect of maximum retry limit.



(a) Packet delivery ratio (b) End-to-end delay

Figure 3.11: Effect of slot discount factor.

when α is around 0.8 – 0.9 for both the sunny and cloudy scenarios. Using a value less than the optimal value underestimates the number of receive slots of u which unnecessarily increases the sleep latency. Likewise, using a value greater than the optimal value overestimates the number of receive slots of u which increases the scheduling errors thereby increasing the latency due to retransmission.

3.5.1 Scheduling Performance Comparison

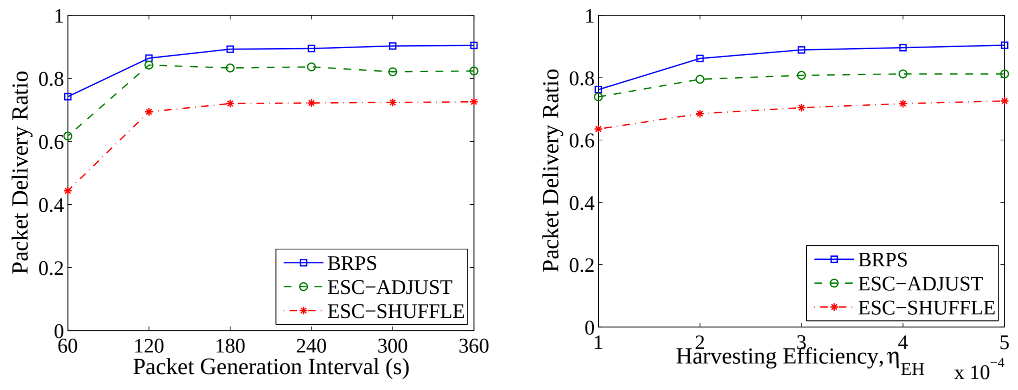
We now compare the performance of the proposed wakeup scheduling scheme (bit-reversal permutation sequence based scheduling or BRPS) with the energy-synchronized communication (ESC) scheme [51]. As mentioned in Section 2.1.5, ESC is one of the first schemes proposed for EPWSNs and represents the state-of-

Table 3.3: Comparison Between BRPS and ESC (Adjust and Shuffle)

Criteria	BRPS	ESC-ADJUST	ESC-SHUFFLE
Storage overhead per neighbor	$O(1)$	$O(n)$	$O(n)$
Communication overhead	$O(1)$	$O(n)$	$O(n)$
Schedule computation complexity	$O(1)$	$O(m)$	$O(m)$
Robustness	Yes	Yes	No

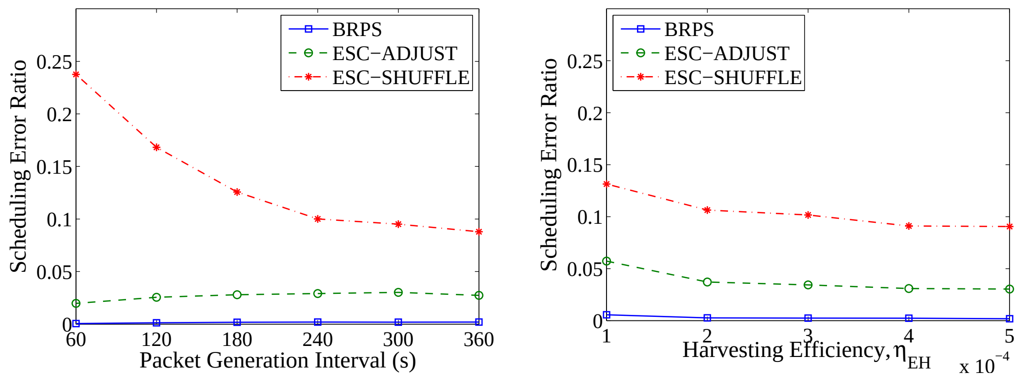
the-art in dynamic wakeup scheduling. We implemented a simulation model of ESC as described in [51] with the following notable features: (*i*) To reduce storage and communication overhead, we represented the wakeup schedules as a bitmap instead of an integer array. Hence, a cycle with 512 slots only requires 512/8 = 64 bytes regardless of the number of active slots. Using integers, a cycle with n active slots requires $2n$ bytes. In the bitmap representation, a bit 1 at position i implies that slot i is active whereas a bit 0 implies that slot i is inactive. (*ii*) To further reduce communication overhead, schedule updates are piggybacked in neighbor updates. (*iii*) In ESC, every node requires the packet ready times at its predecessor nodes. As this is difficult to obtain *a priori*, we used the receive wakeup slots of these predecessor nodes for this purpose. This is reasonable because packets are most likely to become ready for transmission after a node wakes up in its receive wakeup slot. (*iv*) We implemented both the adjustment-based approach and the shuffle-based approach. Note that the former satisfies the properties of robustness while the latter does not. Table 3.3 provides a brief comparison of BRPS and the two variants of ESC. The variable n denotes the number of active slots while m denotes the sum of the active slots of a node's successors and the number of packets sent by its predecessors. The scheduling computation complexity of BRPS is $O(1)$ because it only requires knowledge of the number of active slots. Whereas, ESC requires $O(m)$ as it computes the optimal wakeup slot positions by considering all the individual active slots of a node's successors and the packet ready times at its predecessors.

In the comparison, we paired the above scheduling schemes with ETD. Note



(a) (b)

Figure 3.12: Packet delivery ratio of different scheduling schemes.



(a) (b)

Figure 3.13: Scheduling error ratio of different scheduling schemes.

that using other routing metrics (ETX or hop count) yielded inferior performance. (Section 3.5.2 provides a comparison of the different routing metrics.) We fixed the maximum retry limit L to 3 and the receive slot discount factor α to 0.8. The results plotted in Figures 3.12 – 3.13 are the averages of the sunny and cloudy scenarios.

In terms of packet delivery ratio (see Figure 3.12), we can see that BRPS outperforms ESC-SHUFFLE and ESC-ADJUST by 20% and 10%, respectively on the average. We did not expect to see significant difference in the data delivery performance because the schemes used the same routing metric. The performance advantage of BRPS is due to its robustness and the frequency of schedule updates. Note that the schedule information of BRPS (*i.e.*, the number of receive wakeup slots) is always included in every neighbor update. Because of the large overhead

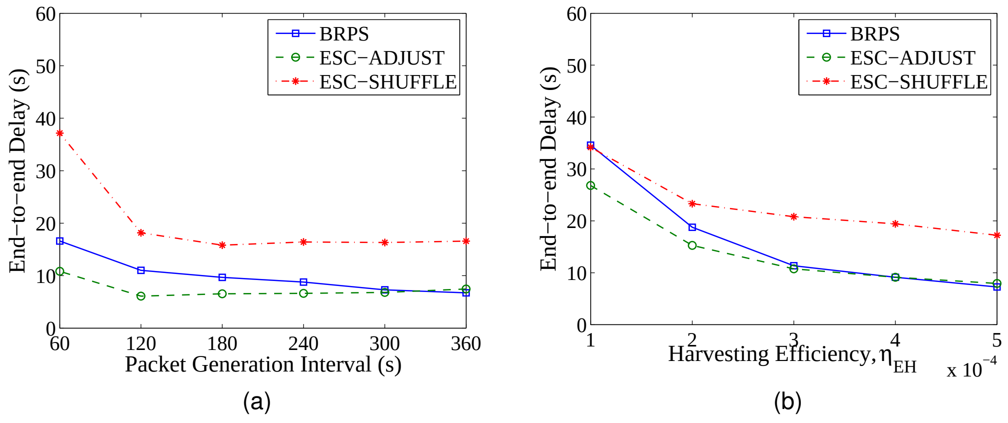


Figure 3.14: End-to-end delay of different scheduling schemes.

entailed by exchanging entire schedules in ESC, schedule updates are only transmitted when schedule changes have occurred. The net effect is that both ESC variants have higher scheduling error, as shown in Figure 3.13. Between ESC-ADJUST and ESC-SHUFFLE, we can see the significant benefit of robustness as the former has significantly better performance.

In terms of delay (see Figure 3.14), we can see that the performance of BRPS is well within the performance of the two ESC variants. ESC-ADJUST shows the best performance while ESC-SHUFFLE shows the worst performance. The performance difference between ESC-ADJUST and ESC-SHUFFLE can be attributed to the robustness and non-robustness of the former and the latter, respectively. That is, the non-robustness of the latter causes higher scheduling error ratio. A high scheduling error ratio implies more packet retransmissions which naturally leads to higher delay. Comparing BRPS and ESC-ADJUST, we can see that at moderate harvesting efficiency values and packet generation rates, BRPS can closely match the performance of ESC-ADJUST. We have expected both ESC variants to perform well as they employ a high complexity algorithm to generate wakeup schedules. The comparable performance of BRPS, despite its simplicity and low complexity, is a strong demonstration of the validity of the theory behind its design. BRPS can therefore be employed as an alternative to ESC for generating low latency wakeup schedules, especially in sensor nodes that face severe resource constraints (com-

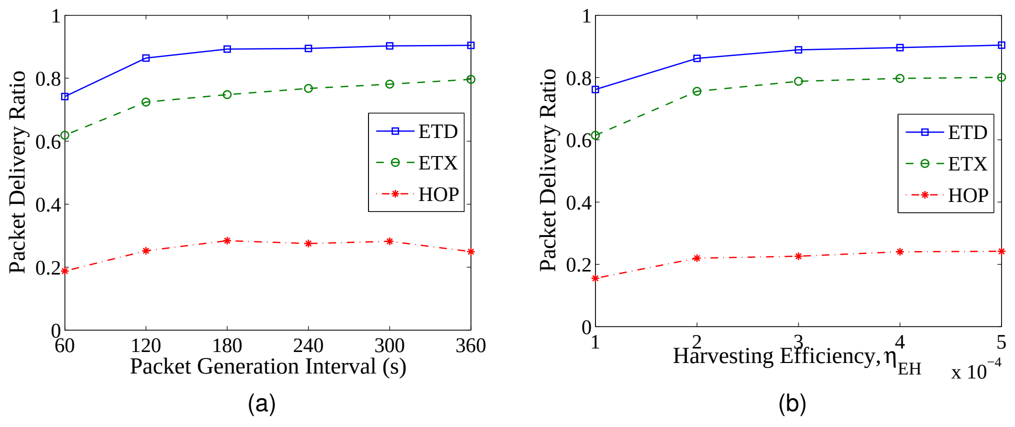


Figure 3.15: Packet delivery ratio of different path metrics (in tandem with BRPS).

putation, memory, and channel capacity) or in situations where ESC is rendered infeasible due to its high memory or communications overhead.

3.5.2 Routing Metric Performance Comparison

To complete our simulation studies, we compare the performance of the proposed path metric (ETD) with ETX [31] and hop count in tandem with BRPS. Note that the relative performance of the three metrics remain the same when they are respectively paired with ESC-SHUFFLE and ESC-ADJUST. Likewise, the relative performance of the scheduling schemes remain the same in each of the three metrics.

As in the scheduling performance comparison, we also fix the maximum retry limit L to 3 and the receive slot discount factor α to 0.8. The results of the sunny and cloudy scenarios are averaged to obtain the packet delivery ratio and end-to-end delay results, which are shown in Figures 3.15 and 3.16, respectively. We can see that in both performance metrics, ETD shows the best performance. Because of its awareness of both sleep latency and link quality, ETD considerably outperforms ETX and hop count by 10% and 60%, respectively, in most of the traffic conditions and harvesting efficiency values.

The end-to-end delay results (see Figure 3.16) also show the significant advantage of ETD over ETX and hop count. Note that ETD's delay is less than 1/10

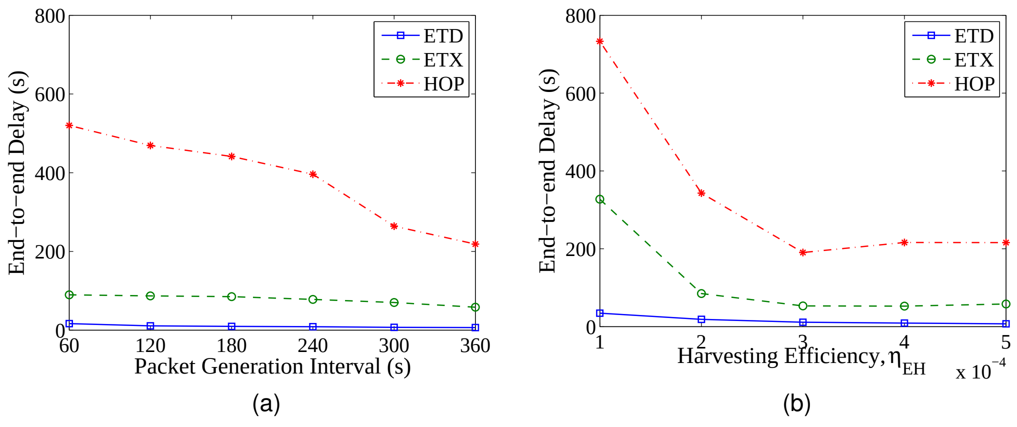


Figure 3.16: End-to-end delay of different path metrics (in tandem with BRPS).

and 1/30 that of ETX and hop count, respectively. These results demonstrate the significant impact of both sleep latency and link quality on end-to-end packet forwarding. Note that ETX, a metric that considers link quality, performed much better than hop count but still significantly behind ETD. These results strongly suggest that blindness to sleep latency can cause considerable delay degradation.

3.6 Summary

In EPWSNs, low latency wakeup scheduling and packet forwarding is challenging due to dynamic duty cycling which necessitates the use of dynamic wakeup schedules and poses time-varying sleep latencies.

We showed analytically that the expected sleep latency is affected by the variance of the intervals between receive wakeup slots: when the variance of the intervals is low (high), the expected latency is low (high). This is because when the intervals between receive wakeup slots are highly uneven, it is more likely for a packet to become ready for transmission at a larger interval than a shorter interval. We therefore introduced a scheduling scheme that aims to position receive wakeup slots as evenly as possible. To reduce storage and communication overhead, the schedule of a node is represented compactly using an integer sequence formula.

We analytically obtained the worst-case sleep latency of a scheduling scheme that uses the bit-reversal permutation sequence (BRPS) and found it to be slightly worse than the ideal scheme (scheduling scheme where the receive wakeup slots are equally-spaced) but better than schemes where the intervals between receive wakeup slots are taken from a uniform or exponential distribution. But while the ideal scheme is not robust to changes in the duty cycle, the BRPS is robust.

A low sleep latency schedule does not necessarily lead to low end-to-end latency paths because other factors such as link quality play a significant role in the performance of packet forwarding. We therefore formulated a metric called expected transmission delay (ETD) which simultaneously considers sleep latency (due to duty cycling), and wireless link quality. We showed that the metric is left-monotonic and left-isotonic, proving that its use in distributed algorithms such as the distributed Bellman-Ford will yield consistent, loop-free and optimal paths.

We have conducted simulations to evaluate the performance of the proposed schemes. We compared the performance of BRPS with ESC, a scheduling scheme that represents the state-of-the art. Results show that BRPS provides low latency and can closely match the performance of ESC. Furthermore, BRPS has a lower scheduling error ratio which translates to better packet delivery ratio. Aside from having a lower storage and communication overhead, BRPS also has a lower computational complexity compared with ESC. Compared with ETX and hop count, and used in tandem with BRPS, ETD provides the best performance in terms of packet delivery ratio and delay.

Chapter 4

Dynamic Duty Cycle Allocation

The main objective of a wakeup schedule is to enable duty cycling nodes to exchange data packets. As elaborated in Section 2.1, numerous wakeup scheduling schemes have been designed to allow the exchange of at most one packet per wakeup interval or slot. In such schemes, a wakeup slot can be used either for transmission, reception or both. Based on this, we can then classify wakeup scheduling schemes into two categories based on the usage of a wakeup slot: (*i*) *bi-directional*; and (*ii*) *receive-centric*.

Bi-directional Wakeup Schedule In this category, a wake-up slot can be used for either reception or transmission. The actual usage is opportunistic: if a node has data to transmit, it transmits during the wakeup slot; otherwise, the node listens for transmissions from its neighbors.

Receive-Centric Wakeup Schedule In this category, a wakeup slot is intended for reception only. A node with data to transmit simply waits for the next wakeup slot of its intended receiver and performs the transmission within this wakeup slot. Most wakeup scheduling schemes presented in Section 2.1 fall under this category. The BRPS wakeup scheduling scheme introduced in the preceding chapter also belongs to this category.

Note the most important difference between a bi-directional and a receive-centric wakeup schedule: in the former, transmissions and receptions are both accounted for in the schedule whereas in the latter, only receptions are included. This implies that a receive-centric scheme cannot use the entire duty cycle to generate wakeup schedules. Since packet transmissions are not accounted for in the schedule, nodes that need to forward packets need to explicitly reserve a portion of their duty cycle for transmission. Otherwise, such nodes run the risk of exceeding their respective duty cycle allotments which may consequently lead to short-term energy supply shortages. We refer to this apportioning of the duty cycle between packet reception and transmission as the *duty cycle allocation problem*.

Unfortunately, current receive-centric wakeup scheduling schemes do not address the problem of duty cycle allocation. In most of the proposed schemes (*e.g.*, [50,51,83]), it is implicit that the entire duty cycle is used for reception, leaving no allocation for transmission. This approach is clearly not suitable for multihop topologies where the nodes must also forward packets and not just perform packet reception.

While the BRPS wakeup scheduling scheme allocates duty cycle for transmission, it uses a static approach wherein the duty cycle is divided equally (*i.e.*, half for reception and half for transmission). In practical settings where data traffic and energy supply are dynamic, static allocation schemes will not be able to provide the optimal performance. For instance, a node which has a lot of backlog data packets may choose to allocate more duty cycle for transmission than for reception.

In this chapter, we investigate the duty cycle allocation problem in EPWSNs and we make the following contributions:

Packet Arrival Probability Model The issue of packet contention in duty cycled sensor networks have been largely sidestepped in the literature [50, 51, 83, 117]. In this chapter, we consider the impact of contention and using the CSMA/CA method in IEEE 802.15.4 [7], we derive the packet arrival probability when several nodes attempt to transmit in a wakeup slot of a receiver node.

Service Time/Sleep Latency Model Using discrete-time queueing analysis, we derive the service time of packets in the context of duty cycled nodes and in the presence of contention. The service time is essentially equivalent to the *sleep latency* which is a major challenge in EPWSNs. A key insight of our result is that the variance of the intervals between the wakeup slots affects the service time, *i.e.*, a higher (lower) variance yields higher (lower) expected service time.

Optimal Duty Cycle Allocation Using the packet arrival probability and expected service time models, we formulate a constrained non-linear optimization problem to apportion the duty cycle, with the objective of minimizing the two-hop expected service time. We propose LSLOTALLOC, a distributed low-complexity algorithm that linearly searches for the optimal solution of the problem. To the best of our knowledge, this work is the first to propose a low-complexity algorithm for computing the optimal duty cycle allocation.

Validation and Performance Evaluation Through simulations, we validate the analytical models and evaluate the performance of LSLOTALLOC. Results show the significant performance advantage of LSLOTALLOC over the static allocation scheme in terms of delay. Notably, LSLOTALLOC's delay is around half the delay of the static allocation scheme in most scenarios.

The rest of the chapter is organized as follows. In Section 4.1, we introduce the system models used in the chapter and present the derivation of the packet arrival probability. In Section 4.2, we derive the expected service time using discrete-time queueing analysis while in Section 4.3, we formulate the optimization problem and propose the LSLOTALLOC algorithm. In Section 4.4, we present the simulation models and the results of the model validation and performance evaluation. Finally, we conclude the chapter in Section 4.5.

4.1 System Models

4.1.1 General

The network is modeled as a tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ rooted at sink node t, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Edge $(v, w) \in \mathcal{E}$ if v and w can directly receive packets from each other. The tree topology implies that every node v has a set of predecessor nodes $\mathbf{P}_v$ and a single successor node w.

Epochs and Slots Similar to existing work [51, 113, 117], time is divided into *epochs*. An epoch is further subdivided into S *slots*. The nodes employ a time synchronization protocol (*e.g.,* [29]) for slot synchronization or alignment. A slot can be in one of three possible states: (*i*) active for data packet reception; (*ii*) active for data packet transmission; or (*iii*) inactive. As shown in Figure 4.1, the slot duration τ is designed to accommodate several time components, *i.e.,*

$$\tau = \tau_{\text{cr}} + \tau_{\text{data}} + \tau_{\text{ackto}}, \tag{4.1}$$

where τ_{cr} is the contention resolution time, τ_{data} is the time needed to send or receive a maximum-length data packet, and τ_{ackto} is the acknowledgment (ACK) timeout. The usage of the time components τ_{cr} and τ_{ackto} are discussed in Section 4.1.3. We make the following remarks about slots:

- When a slot is active for data packet reception, the node actually switches its radio to transmit mode after it receives a data packet to send an ACK packet.
- In a similar fashion, when a slot is active for data packet transmission, the node switches its radio to receive mode after it transmits a data packet to listen for an ACK packet.
- τ_{data} caters for the maximum-length data packet and as such, a portion of the slot may be wasted in cases where the transmitting node sends shorter packets. To save energy, the receiving node may immediately go back to sleep after receiving a packet *and* transmitting the corresponding ACK. Likewise,

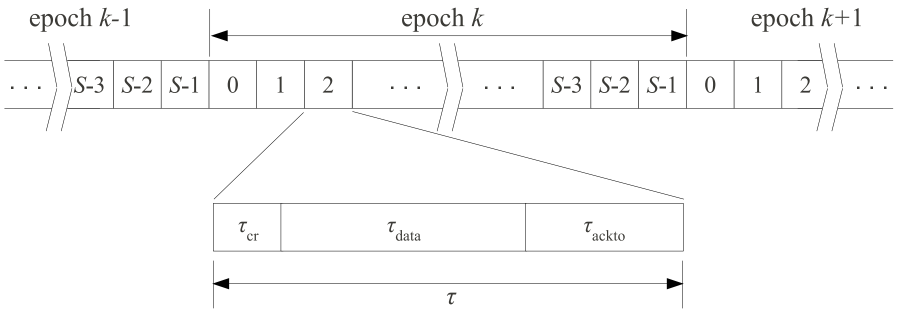


Figure 4.1: Epoch, slot and time components of a slot.

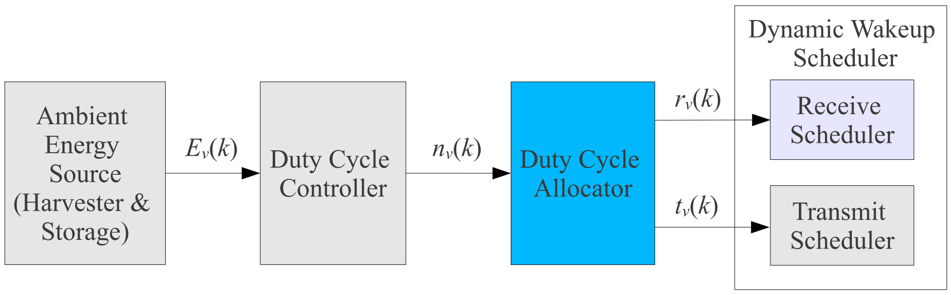


Figure 4.2: Energy harvesting node model. The focus in this chapter is the design of the duty cycle allocator.

the transmitting node may immediately go back to sleep after receiving an ACK.

Duty Cycle Every node $v \in \mathcal{N}$ uses the energy-harvesting node model shown in Figure 4.2. The duty cycle controller performs adaptive duty cycling to ensure that v is energy-neutral. It determines the operating duty cycle $\delta_v(k)$ which indicates the fraction of slots in epoch k that v can be active for reception and transmission, or

$$\delta_v(k) \triangleq \frac{n_v(k)}{S}, \tag{4.2}$$

where $n_v(k) \in [0, S]$ is the number of active slots of v in epoch k. The *duty cycle allocator*, which is the focus of this study, is responsible for apportioning $n_v(k)$ into two parts: $r_v(k)$ slots for reception and $t_v(k)$ slots for transmission. The quantities $r_v(k)$ and $t_v(k)$ are used by the dynamic wakeup scheduler which is presented in Section 4.1.2.

4.1.2 Dynamic Wakeup Schedule Model

We now formulate a generic dynamic wakeup schedule model that captures the essential elements of existing receive-centric wakeup scheduling schemes.

Receive Wakeup Schedule Whenever u needs to forward a packet to its successor node v, u needs to know the receive wakeup schedule of v so that it can wakeup at the appropriate slot in the future to perform the actual transmission. We have formally defined this notion in Definition 4.

Transmit Wakeup Schedule The receive wakeup schedule only specifies the slots at which v can receive packets from its predecessor nodes. If v is a relay node, it must also wakeup to relay packets. To do so, v can use any slot $\bar{s} \notin \mathbf{\Gamma}_v(k)$ but must ensure that the intended receiver node w is awake to listen for packet transmissions, *i.e.*, $\bar{s} \in \mathbf{\Gamma}_w(k)$. We formalize this notion as follows:

Definition 10. *The transmit wakeup schedule of a node v for epoch k consists of the slots at which v wakes up to transmit its packets to its successor node w. It consists of at most $t_v(k)$ slots.*

The main difference between the receive and transmit wakeup schedules is that the former can be completely specified prior to the start of an epoch while the latter cannot be specified in any way. This is because packet transmissions by v depend on the presence of packets in v and the receive wakeup schedule of the successor node w, $\mathbf{\Gamma}_w(k)$.

4.1.3 Medium Access Control Protocol

Prior studies [51, 113, 117] have neglected the impact of packet collisions, arguing that a very low sending rate will not lead to significant packet collisions. However, as pointed out by Du, *et al.* [37], the use of synchronization increases the probability of contention and as such, we must consider contention in the development of our models.

Contention Resolution The contention resolution process is modeled after the CSMA/CA method in IEEE 802.15.4 [7]. With reference to Figure 4.1, the first part of the slot (τ_{cr}) is meant to accommodate the contention resolution process. At the start of a slot of node v, a node u with data to transmit to v selects a random backoff between 0 and $2^B - 1$, where $B = 3$ is the default backoff exponent value. At the end of its backoff period, u performs clear channel assessment twice. If the channel is idle, u commences transmission; otherwise, it goes back to sleep and repeats the same process in the next wakeup slot of v.

Channel Access Success Probability We will now derive the probability that a packet transmitted by $u \in \mathbf{P}_v$ is successfully received by v in the presence of contention in epoch k. Suppose that $\mathcal{P}_v \subseteq \mathbf{P}_v$ have pending packets to be relayed to v. Denote D_v as the (expected) number of packets that every node needs to transmit to v. Then the probability that any of these nodes will access the channel in a receive slot of v, denoted by $p_v^{\text{ca}}(k)$, is

$$p_v^{\text{ca}}(k) = \min\left[\frac{D_v}{r_v(k)}, 1\right], \tag{4.3}$$

where $r_v(k)$ is the number of receive slots allocated by v in epoch k. Note that (4.3) implies that the packets are equally distributed among the nodes in $\mathcal{P}_v$. This may not be realistic but from an analytical point of view, such a situation will result in the worst-case contention and the result will therefore be conservative.

Denote $p_v^{\text{cas}}(k)$ as the probability of successful channel access in epoch k from any of the nodes in $\mathcal{P}_v$ to v. Since all channel access attempts are synchronized, we can obtain $p_v^{\text{cas}}(k)$ as follows: a channel access attempt by a predecessor node u is successful if exactly m nodes out of $|\mathcal{P}_v|$ access the channel but u selects a value for its backoff counter that is unique and the lowest. Let $\Pr(E|M = m)$ denote the conditional probability of this event. Supposing that V_x corresponds to the random backoff value selected by $x \in \mathcal{P}_v$, we have

$$\begin{aligned}
\mathsf{Pr}(E|M=m) &= \sum_{i=0}^{2^B-1} \mathsf{Pr}(V_u = i) \times \\
&\quad \prod_{x \in \mathcal{P}_v \setminus u} \mathsf{Pr}(i+1 \le V_x \le 2^B - 1) \\
&= \sum_{i=0}^{2^B-1} \frac{1}{2^B} \left[\frac{2^B - (i+1)}{2^B} \right]^{m-1} \\
&= \frac{1}{2^{mB}} \sum_{i=0}^{2^B-1} \left[2^B - (i+1) \right]^{m-1}. \qquad (4.4)
\end{aligned}$$

The probability that exactly m nodes out of $|\mathcal{P}_v|$ access the channel, denoted by $\mathsf{Pr}(M = m; k)$, is simply

$$\mathsf{Pr}(M = m; k) = \binom{|\mathcal{P}_v|}{m} [p_v^{\text{ca}}(k)]^m [1 - p_v^{\text{ca}}(k)]^{|\mathcal{P}_v| - m}. \qquad (4.5)$$

Using the law of total probability,

$$p_v^{\text{cas}}(k) = \sum_{m=1}^{|\mathcal{P}_v|} \mathsf{Pr}(E|M=m)\mathsf{Pr}(M=m;k). \qquad (4.6)$$

Packet Arrival Probability Denote $p_v(k)$ as the probability of one packet arrival in a receive slot of v when $r_v(k)$ slots are allocated for packet reception. Neglecting channel errors, v will receive a packet in a receive slot whenever a node $u \in \mathcal{P}_v$ successfully accesses the channel. This is because if u wins in the contention process, it will transmit exactly one packet. Thus we have

$$p_v(k) = p_v^{\text{cas}}(k). \qquad (4.7)$$

Data Transmission Once u successfully acquires the channel, it commences data transmission. For a unicast transmission from u to v, u waits for a corresponding ACK within τ_{ackto}. Node u subsequently informs the network layer whether the transmission was a success (if it received an ACK from v) or a failure (if it did not receive any ACK).

4.2 Analysis of Wakeup Scheduling

Consider a node v with active predecessor nodes $\mathcal{P}_v$ and a successor node w. It wakes up at predefined slots in $\mathbf{\Gamma}_v(k)$ to listen for transmissions. Whenever v has a packet in its queue, it wakes up at the next earliest slot in $\mathbf{\Gamma}_w(k)$ to forward its packet to w. The operation of v can therefore be modeled after a discrete-time queue, where packet arrivals can occur at any slot in $\mathbf{\Gamma}_v(k)$ and packet departures can occur at any slot in $\mathbf{\Gamma}_w(k)$. In the following analysis, we will derive the service time in both ideal conditions and in the presence of contention.

Remarks To reduce notational clutter, we drop the parameter k (to denote epoch) in the queueing analysis as it is understood that the analysis is within a single epoch. We also stipulate that events, *e.g.*, packet arrivals and service completions, can only occur at the end of a slot.

4.2.1 Intervals Between Wakeup Slots

Before delving into the details of the model derivation, we first introduce an idealization of the epoch duration. As presented in Section 4.1.1, an epoch has finite duration consisting of S equal-length slots. To apply results from discrete-time queueing theory and renewal theory, we idealize the epoch to contain an infinite number of slots. Now, consider the receive wakeup schedule of a node w as shown in Figure 4.3. We define the length of the ith interval in the schedule denoted by L_i, in terms of number of slots, as

$$L_i \triangleq s_{i+1} - s_i, \tag{4.8}$$

where s_i and s_{i+1} are consecutive receive wakeup slots of w. Let us assume that $\{L_i\}$ are i.i.d. and taken from a random variable G_w with probability mass function (PMF) $g_w(n) = \mathsf{Pr}(G_w = n)$. We will see later that the distribution of $\{L_i\}$, and hence G_w, has a direct effect on the service time of packets emanating from any node v that need to be relayed to w.

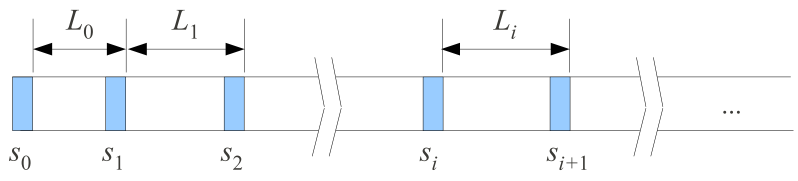


Figure 4.3: Receive wakeup schedule showing the intervals between wakeup slots. Note that the interval length L_i includes slot s_{i+1}. This means that from the perspective of a transmitter node v, a packet that becomes ready for transmission at slot s_{i+1}, which according to our convention occurs at the end of the slot, is not eligible for transmission at slot s_{i+1}. Rather, the packet has to wait until slot s_{i+2}.

4.2.2 Ideal Service Time

Because w is not always awake to receive packets, v needs to wait for the nearest receive wakeup slot of w to transmit its packet. In queuing theory, service time is defined as the time from the moment a customer starts to get served until its departure. In the context of our problem, service time is the time (in slots) from the moment a packet becomes ready for transmission at v until its successful reception by w which is basically the *sleep latency* from v to w [117, 130].

Before proceeding further, we need to prove the following result about the mean forward recurrence time of a discrete-time renewal process.

Lemma 5. *Let $\{S_n, n \geq 0\}$ be a discrete-time renewal process, where $S_n = L_0+L_1+L_2+\ldots+L_n$ and $\{L_i\}$ is an i.i.d. sequence of non-negative integers, taken from a distribution L. Then the mean forward recurrence time of this process is $\frac{\mathsf{E}[L(L-1)]}{2\mathsf{E}(L)}$, where $\mathsf{E}(L)$ denotes the mean of L.*

Proof. Suppose that in the current interval i, the interval duration or length is L_i. (Without loss of generality, we say that the current interval has L_i slots.) Let K denote the index of a randomly selected slot within interval i (*i.e.*, $K = 1$ and $K = L_i$ mean the first and last slots in interval i, respectively). Then by definition, the backward recurrence time is $K - 1$. This is because at slot K, there are exactly $K - 1$ slots prior to itself since the start of the interval.

To obtain the mean forward recurrence time (which is equal to the mean backward recurrence time), we simply have to obtain $\mathsf{E}(K - 1)$. There are several ways to obtain this quantity but here, we use renewal-reward theory. To do so, we associate a reward that is equal to the backward recurrence time. This means that if slot k is chosen, then the corresponding reward is $k-1$. With this, the total reward R within an interval with L_i slots is

$$R = 1 + 2 + 3 + \ldots + L_i - 1 = \frac{L_i(L_i - 1)}{2}.$$

Note that the maximum reward is $L_i - 1$ which is obtained when the randomly chosen slot $k = L_i$. From renewal-reward, $\mathsf{E}(K-1) = \mathsf{E}(R)/\mathsf{E}(L)$. Hence, we have

$$\mathsf{E}(K-1) = \frac{\mathsf{E}[L(L-1)]}{2\mathsf{E}(L)}.$$

$\square$

Assuming that the queue follows a first-in first-out (FIFO) discipline, then a packet is ready for transmission when it reaches the head of queue. Service time is dependent on the state of the queue, as will be elaborated in the following:

Case 1: Empty Queue Upon Arrival As illustrated in Figure 4.4, when a packet arrives at v and its queue is empty, the arriving packet can be immediately transmitted at the next receive wakeup slot of w. Recall that G_w is the interval between receive wakeup slots in $\mathbf{\Gamma}_w(k)$. Let $\mathsf{E}(S_v^*|Q_v = 0)$ be the ideal conditional expected service time at v when the queue is empty. Since $L_i \geq 0$ and $\{L_i\}$ are i.i.d., then we can use renewal theory [16] to obtain $\mathsf{E}(S_v^*|Q_v = 0)$. From Figure 4.4, we can see that $\mathsf{E}(S_v^*|Q_v = 0)$ is essentially the mean forward recurrence time of a discrete-time renewal process, which from Lemma 5, gives us:

$$\mathsf{E}(S_v^*|Q_v = 0) = \frac{\mathsf{E}[G_w(G_w - 1)]}{2\mathsf{E}(G_w)} = \frac{\mathsf{E}(G_w^2) - \mathsf{E}(G_w)}{2\mathsf{E}(G_w)}. \tag{4.9}$$

Case 2: Non-Empty Queue Upon Arrival When a packet arrives at v and its queue is not empty, the arriving packet enters the queue and can only be processed after the service completion of all earlier packets. This is illustrated in Figure 4.5. In this case, the service time is determined solely by the interval lengths in $\mathbf{\Gamma}_w(k)$. If $\mathsf{E}(S_v^*|Q_v \neq 0)$ denotes the ideal conditional expected service time given that the

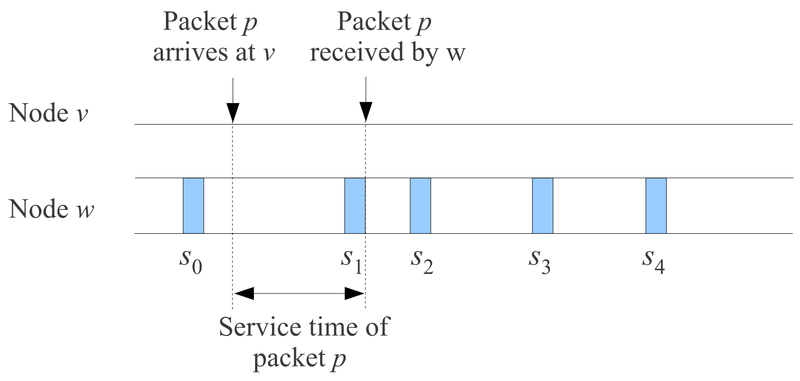


Figure 4.4: Service time to send a packet from node v to w when the arriving packet p at v encounters an empty queue.

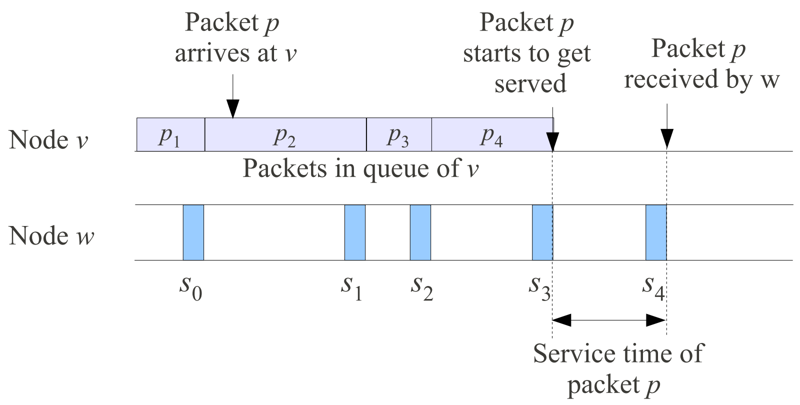


Figure 4.5: Service time to send a packet from node v to w when the arriving packet p at v encounters a busy queue. Since the queue is FIFO, the packets $p_1 - p_4$ in the queue are serviced before p gets served. Note that the service time is exactly equal to one interval length.

queue is not empty, then

$$\mathsf{E}(S_v^*|Q_v \neq 0) = \mathsf{E}(G_w). \tag{4.10}$$

Finally, we can obtain the unconditioned ideal expected service time at v, denoted by $\mathsf{E}(S_v^*)$, by using the law of total expectation. Let ρ_v be the utilization factor at v. From elementary queueing theory [20], $\mathsf{Pr}(Q_v = 0) = 1 - \rho_v$ and $\mathsf{Pr}(Q_v \neq 0) = \rho_v$. We therefore have $\mathsf{E}(S_v^*) = (1 - \rho_v)\mathsf{E}(S_v^*|Q_v = 0) + \rho_v \mathsf{E}(S_v^*|Q_v \neq 0)$. Substituting (4.9) and (4.10) into this yields

$$\mathsf{E}(S_v^*) = \mathsf{E}(G_w)\left[\frac{1-\rho_v}{2}c_{G_w}^2 + \frac{1+\rho_v}{2}\right] - \frac{1-\rho_v}{2}, \tag{4.11}$$

where $c_{G_w} = \sqrt{\mathsf{Var}(G_w)}/\mathsf{E}(G_w)$ is the coefficient of variation of G_w.

Some insights provided by (4.11) are worth mentioning at this point. As expected, $\mathsf{E}(S_v^*)$ increases as $\mathsf{E}(G_w)$, the mean interval between the wakeup slots of w, increases. Interestingly, $\mathsf{E}(S_v^*)$ is also affected by the variations of the intervals between wakeup slots G_w. In particular, a higher (lower) variance in G_w yields higher (lower) expected service time. Thus, a simple way to reduce the $\mathsf{E}(S_v^*)$ is to generate wakeup schedule patterns wherein the receive slots are positioned at equal intervals as this will cause c_{G_w} to vanish.

4.2.3 Service Time with Contention

The preceding discussion assumes that a packet transmitted by v will always be received by w in the first attempt. However, if several predecessor nodes of w are contending for the channel, collisions may occur and v may need to transmit a packet several times to be successfully relayed to w. Nevertheless, the expected service time in the presence of contention, which we denote by $\mathsf{E}(S_v)$, can be obtained by extending the preceding results.

To begin, denote p_w^{txs} as the probability of successful packet delivery from v to w in an epoch. This probability can be obtained as follows: Suppose that w allocated r_w slots for packet reception and every active predecessor node $x \in \mathcal{P}_w$ needs to send D_w packets to w. If p_w is the corresponding packet arrival probability in every receive slot of w, then w is expected to receive $p_w r_w$ packets out of $|\mathcal{P}_w| D_w$. This yields

$$p_w^{\text{txs}} = \frac{p_w r_w}{|\mathcal{P}_w| D_w}. \tag{4.12}$$

We remark that (4.12) is an approximation because it is actually dependent on the retransmission limit. Recall that p_w and D_w are interdependent and as the retransmission limit is increased, both p_w and D_w increase, albeit at different rates.

Case 1: Empty Queue Upon Arrival Let l denote the number of transmission attempts at v for a particular packet. In the first attempt, the expected service time is simply given by the ideal conditional expected service time at v when the queue is empty, $\mathsf{E}(S_v^*|Q_v = 0)$. In the nth transmission attempt, the expected service time will include $(n-1)\mathsf{E}(G_w)$, hence,

$$\mathsf{E}(S_v|Q_v=0,l=n)=\frac{\mathsf{E}[G_w(G_w-1)]}{2\mathsf{E}(G_w)}+(n-1)\mathsf{E}(G_w). \tag{4.13}$$

Case 2: Non-Empty Queue Upon Arrival This case is straightforward because at each attempt, the expected service time increases by $\mathsf{E}(G_w)$. Hence at the nth transmission attempt,

$$\mathsf{E}(S_v|Q_v\neq 0,l=n)=n\mathsf{E}(G_w). \tag{4.14}$$

Using (4.13), (4.14) and the law of total expectation, the expected service time from v to w conditioned on n transmission attempts, can be simplified to

$$\mathsf{E}(S_v|l=n)=\mathsf{E}(S_v^*)+(n-1)\mathsf{E}(G_w). \tag{4.15}$$

Let $\mathsf{Pr}(l=n)$ denote the probability that a packet is successfully transmitted on the nth attempt. From elementary probability,

$$\mathsf{Pr}(l=n)=\frac{p_w^{\text{txs}}(1-p_w^{\text{txs}})^{n-1}}{1-(1-p_w^{\text{txs}})^L}, \tag{4.16}$$

where L is the maximum transmission limit. Using (4.15) and (4.16), the unconditioned expected service time can be easily obtained by applying the law of total expectation, that is,

$$\mathsf{E}(S_v)=\sum_{n=1}^{L}[\mathsf{E}(S_v^*)+(n-1)\mathsf{E}(G_w)]\frac{p_w^{\text{txs}}(1-p_w^{\text{txs}})^{n-1}}{1-(1-p_w^{\text{txs}})^L}. \tag{4.17}$$

4.2.4 Equal-Interval Wakeup Schedule

The insights provided by (4.11) motivates the use of wakeup schedule patterns wherein the receive slots are positioned at equal intervals. We focus on this and express (4.17) in terms of the wakeup scheduling parameter r_v, *i.e.*, the number of receive slots in the schedule of v. For an equal-interval wakeup schedule, the coefficient of variation c_{G_w} vanishes, hence the ideal expected service time simplifies to

$$\mathsf{E}(S_v^*) = \mathsf{E}(G_w)\left[\frac{1+\rho_v}{2}\right] - \frac{1-\rho_v}{2}. \tag{4.18}$$

Substituting (4.18) into (4.17), we can then get a closed form expression for (4.17) using [126] as follows:

$$\mathsf{E}(S_v) = \mathsf{E}(G_w)\left[\frac{Q_w-(1-\rho_v)}{2}\right] - \frac{1-\rho_v}{2}, \tag{4.19}$$

where

$$Q_w = \frac{2}{p_w^{\text{txs}}} - 2L\left[\frac{1}{1-(1-p_w^{\text{txs}})^L} - 1\right]. \tag{4.20}$$

Note that when $p_w^{\text{txs}} = 1$, then $Q_w = 2$ and $\mathsf{E}(S_v) = \mathsf{E}(S_v^*)$ which is what we expected.

By definition, the utilization factor is $\rho_v = \lambda_v/\mu_v$, where λ_v is the packet arrival rate at v, and μ_v is the service rate at v. Since the packet arrival probability in every receive slot of v is p_v, the expected number of packets to arrive at v in the epoch is

$$\mathsf{E}(A_v) = p_v r_v + d_v, \tag{4.21}$$

where d_v is the expected number of self-generated packet arrivals. Note that d_v is independent of r_v because v will always receive its self-generated packets regardless of r_v. An epoch has S slots, hence the packet arrival rate is $\lambda_v = \mathsf{E}(A_v)/S$. The service rate μ_v is simply the reciprocal of the service time $\mathsf{E}(S_v)$. Thus $\rho_v = \mathsf{E}(A_v)\mathsf{E}(S_v)/S$. Substituting this into (4.19) and solving for $\mathsf{E}(S_v)$ yields

$$\mathsf{E}(S_v) = \frac{S[\mathsf{E}(G_w)(Q_w-1)-1]}{2S-[\mathsf{E}(G_w)+1]\mathsf{E}(A_v)}. \tag{4.22}$$

Note that $\mathsf{E}(G_w)$ is simply the mean interval length between the receive wakeup slots of w and this can be computed as $\mathsf{E}(G_w) = S/r_w$. Finally, we introduce the mean service time function at v, denoted by $\mathcal{S}_v(r_v, r_w)$:

$$\mathcal{S}_v(r_v, r_w) = \frac{S[S(Q_w-1)-r_w]}{2Sr_w-(S+r_w)(p_v r_v+d_v)}. \tag{4.23}$$

4.3 Optimal Duty Cycle Allocation

We are now ready to tackle the problem of apportioning n_v (the number of active slots in an epoch) for packet reception and transmission. Our solution approach will be as follows. First, we will formulate a constrained non-linear optimization problem to minimize the two-hop expected service time of packets that are traversing through node v. Second, we will develop LSLOTALLOC, an algorithm with $O(n)$ complexity that can search for the minimizer r^* of the two-hop expected service time. Finally, we will discuss several practical issues that must be considered in the implementation of the proposed algorithm in real-world sensor networks. As in the preceding section, we drop the parameter k (which denotes the current epoch) to reduce notational clutter.

4.3.1 Two-Hop Service Time

When v allocates r slots for packet reception, it affects not only its own service time but also the service time of all its active predecessor nodes $u \in \mathcal{P}_v$. For every node u, its expected service time is $\mathcal{S}_u(r_u, r)$, where A_u is the expected number of packet arrivals at u in the current epoch. If the fraction of packets transmitted by u to v in the epoch is ϕ_u, then its total contribution to the two-hop service time is $\phi_u \mathcal{S}_u(r_u, r)$. Thus, the two-hop expected service time of packets traversing through v, denoted by $T_v(r)$ is

$$T_v(r) = \mathcal{S}_v(r, r_w) + \sum_{u \in \mathcal{P}_v} \phi_u \mathcal{S}_u(r_u, r). \tag{4.24}$$

From the point of view of v, it actually does not need to know r_u for every predecessor node u. Rather, v only needs to have knowledge of the number of packets that every node u expects to receive, *i.e.*, $\mathsf{E}(A_u)$. Thus, we can rewrite (4.24) as

$$T_v(r) = \mathcal{S}_v(r, r_w) + \sum_{u \in \mathcal{P}_v} \frac{\phi_u S[S(Q_v - 1) - r]}{2Sr - (S + r)\mathsf{E}(A_u)}. \tag{4.25}$$

4.3.2 Optimization Problem

If r slots are allocated for packet reception, $n_v - r$ slots are left for packet transmission. With this allocation, the expected number of packets that v can receive is simply rp_v. Node v may also be generating its own data packets, denoted by d_v. The sum $rp_v + d_v$ is essentially $\mathsf{E}(A_v)$ which is defined in (4.21). To ensure that v can relay all the received and generated packets to its successor node w, it must allocate at least $(rp_v + d_v)/p_w^{\text{txs}}$ transmit slots. Hence, r must be chosen such that $r + (rp_v + d_v)/p_w^{\text{txs}} \leq n_v$. The optimization problem can be initially formulated as follows:

$$\begin{array}{lll} \text{minimize} & T_v(r) & \\ \text{subject to} & (1) & r \leq n_v \\ & (2) & r + \frac{rp_v + d_v}{p_w^{\text{txs}}} \leq n_v \\ & (3) & r \in \mathbb{Z}^+ \end{array} \tag{4.26}$$

Constraints (1) and (3) specify the entire domain of the problem, which is the set $\{1, 2, 3, ..., n_v\}$, while constraint (2) is the traffic flow feasibility condition. Problem (4.26) is a mixed integer non-linear optimization problem, with the feasibility set consisting of positive integers in a finite interval defined by the intersection of the constraints. With this, we can combine constraints (1) and (3) to simplify problem (4.26) as follows:

$$\begin{array}{ll} \text{minimize} & T_v(r) \\ \text{subject to} & r + \frac{rp_v + d_v}{p_w^{\text{txs}}} \leq n_v \\ & r \in \{1, 2, 3, \ldots, n_v\} \end{array} \tag{4.27}$$

Algorithm 4 LSlotAlloc

```
1: $T_{\min} \leftarrow \infty$
2: $r^* \leftarrow 0$
3: for $r \leftarrow 1$ to $n_v$ do
4:     if $r + \frac{rp_v + d_v}{p_w^{\mathrm{txs}}} > n_v$ then
5:         return $r^*$
6:     end if
7:     $T \leftarrow \mathcal{S}_v(r, r_w) + \sum_{u \in \mathcal{P}_v} \frac{\phi_u S[S(Q_v - 1) - r]}{2Sr - (S + r)\mathsf{E}(A_u)}$
8:     if $T < T_{\min}$ then
9:         $T_{\min} \leftarrow T$
10:        $r^* \leftarrow r$
11:    end if
12: end for
```

4.3.3 Algorithm

Time and space complexities are important in EPWSNs because of the limited processing and storage capacities of sensor nodes. Fortunately, the integer constraint in problem (4.27) can be exploited to design a straightforward algorithm. The listing in Algorithm 4 provides a pseudo-code of the LSlotAlloc algorithm that searches for the optimal solution r^* by linear traversal of the set $\{1, 2, 3, \ldots, n_v\}$. The algorithm is executed at the start of every epoch and independently by every node v in the network.

Input Variables A node v that executes Algorithm 4 requires inputs from all its active predecessor nodes $u \in \mathcal{P}_v$ and its successor node w. In particular, v requires r_w and p_w^{txs} from w, as well as $\mathsf{E}(A_u)$ from every predecessor node u. This can be accomplished by requiring every node x in the network to broadcast a message containing r_x, p_x^{txs} and $\mathsf{E}(A_x)$ once every epoch.

Estimation of Input Traffic Aside from input variables that are needed from its neighbor nodes, v also needs to estimate the input traffic D_v and the respective contribution of every active predecessor node u, ϕ_u. For v to obtain D_v, since every $u \in \mathcal{P}_v$ has pending packets,

$$D_v = \frac{1}{|\mathcal{P}_v|} \sum_{u \in \mathcal{P}_v} \mathsf{E}(A_u).$$

Whereas, ϕ_u can be calculated using

$$\phi_u = \frac{\mathsf{E}(A_u)}{\sum_{x \in \mathcal{P}_v} \mathsf{E}(A_x)}.$$

Properties In the following, we will obtain the space and time complexities of LSLOTALLOC.

Lemma 6. *The space complexity of* LSLOTALLOC *is $O(1)$.*

Proof. The algorithm requires several variables that are re-used in every iteration and not dependent on n_v. Hence, the space complexity is constant. □

Lemma 7. *The time complexity of* LSLOTALLOC *is $O(n)$ where n is the number of active slots.*

Proof. This is obvious since in the worst case, the algorithm traverses the entire domain $\{1, 2, 3, \ldots, n_v\}$. Since every iteration requires constant computation and the algorithm iterates for at most n_v times its time complexity is linear with respect to n_v, or $O(n)$. □

4.3.4 Practical Considerations

Lemma 6 and 7 show that LSLOTALLOC has low-complexity and are good indications about its suitability for implementation in resource-constrained sensor nodes. However, two important practical issues need to be addressed: (*i*) exchange of parameters among nodes; and (*ii*) allocation strategy when the optimization problem is infeasible.

Exchange of Scheduling Parameters As mentioned, every node v is required to broadcast a message containing r_v, p_v^{txs} and $\mathsf{E}(A_v)$. We need to ensure that the broadcasts will not collide with each other. As a matter of convenience, the actual slot that v uses to broadcast the message is the slot number that corresponds to its ID. This approach is similar to the strategy in [117] and reduces broadcast collisions. Note that this requires the nodes to forgo any transmission in a slot s if a node with ID s is one of their neighbors.

Infeasible Problem If the problem is not feasible because the input traffic is higher than what a node can accommodate, LSLOTALLOC will not provide the optimal allocation r^*. This happens when it is not possible to find r such that the constraint $r + (rp_v + d_v)/p_w^{\text{txs}} \leq n_v$ is satisfiable. A straightforward approach is to allocate $r = \lfloor \beta n_v \rfloor$, where $\beta \in (0, 1)$, for reception whenever the optimization problem is infeasible. In the evaluation, we will study the sensitivity of LSLOTALLOC with respect to β.

4.3.5 Control Packet Piggybacking

In Section 4.3.4, we highlighted the requirement of LSLOTALLOC to have access to the most current variables (*i.e.*, r_v, p_v^{txs} and $\mathsf{E}(A_v)$) for every neighbor node v along the routing graph. Without these variables or when they are stale, LSLOTALLOC may yield suboptimal results. As such, the performance of LSLOTALLOC is sensitive to the loss of these information. Moreover, the need to separately exchange these variables could entail high overhead.

To remedy these deficiencies, we propose the use of control packet piggybacking, wherein a node v piggybacks r_v, p_v^{txs} and $\mathsf{E}(A_v)$ in every data packet that it transmits to its successor node, and to every acknowledgement packet that it transmits to its predecessor nodes. Note that the inclusion of the three variables only adds a few bytes to the data and acknowledgement packets. To illustrate, if each of the variables was represented with a 16-bit number, then a total of 6 bytes would need to be piggybacked. This approach clearly addresses the issue of high control overhead. And as data and acknowledgement packets are sent several times in an epoch, the loss is also mitigated.

Note however that this technique will only work (*i*) when there are ongoing data transmissions and (*ii*) when the MAC protocol can be modified to support the piggybacking of additional information. The first point is not an issue since LSLOTALLOC is designed to perform slot allocation when there is on-going data traffic to the sink. If there is no data traffic, then the scheme can be configured to output a default allocation. As for the second point, we can overcome the problem (of not being able to piggyback in the MAC acknowledgements) by disabling MAC-layer acknowledgements and employing network-layer acknowledgements which are more amenable to modifications.

4.4 Evaluation

In this section, we perform simulations using the Qualnet network simulator [4] to (*i*) validate the analytical models for the packet arrival probability (4.7) and the expected service time (4.23); and to (*ii*) determine the performance of LSLOTALLOC.

4.4.1 Simulation Models

We implemented a simulation model of a dynamically duty cycled node in Qualnet [4] as shown in Figure 4.6(a). Aside from LSLOTALLOC, the implemented simulation components include the duty cycle generator, wakeup scheduling algorithm, and collection tree routing protocol. We want to highlight the fact that the Qualnet simulation model also implements the periodic transmission of overhead packets as elaborated in Section 4.3.4. Thus, the simulation results take into account the effect of such overhead in terms of energy consumption and packet collisions.

Duty Cycle Trace To make the simulations realistic, a duty cycle trace from an Arduino-based sensor node (*cf.* Figure 4.6(b)) is collected and used to drive the network simulations. The node is equipped with a 5 Watt solar panel and a 2 Ah lithium polymer rechargeable battery. The duty cycle of the node is controlled

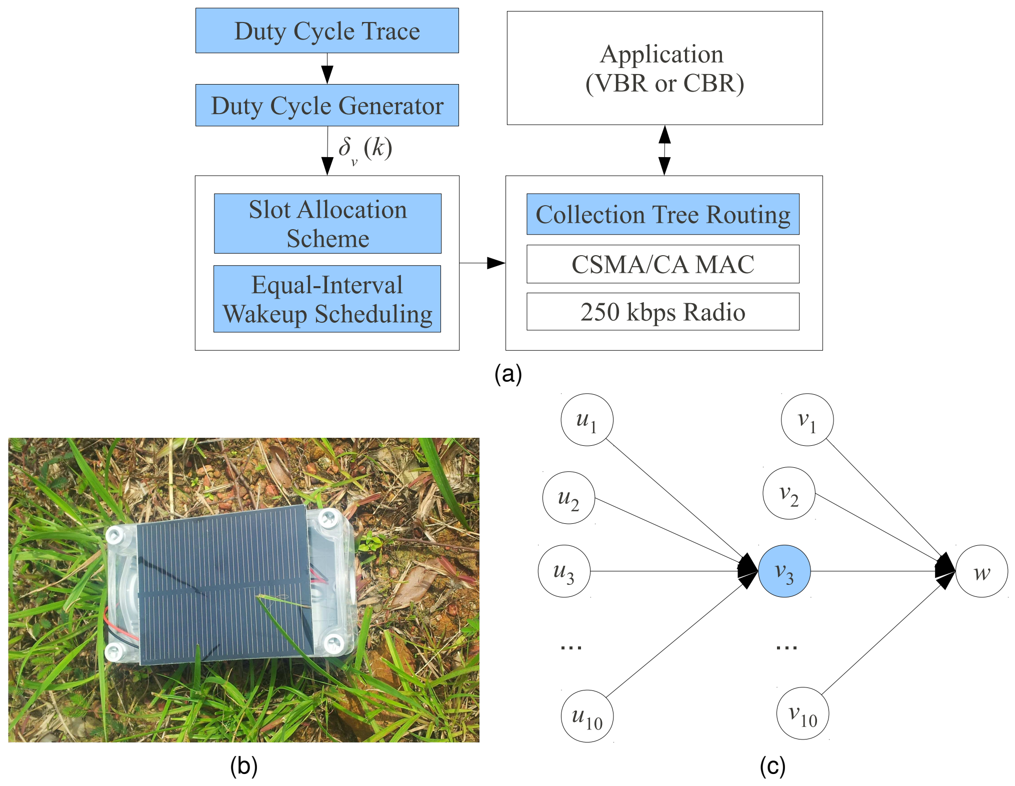


Figure 4.6: (a) Qualnet simulation model; (b) Arduino node used to collect duty cycle traces; and (c) Network topology used in the model validation.

using the LQ-Tracker duty cycle control algorithm [120] which is executed once per minute.

Duty Cycle Generator The duty cycle trace is used to generate the duty cycle $\delta_v(k)$ for every node v in the network. The trace is divided into segments such that every segment is stationary. Time series analysis is employed and every segment is modeled as an autoregressive AR(1) process. The noise/error variance is computed and is used to introduce random variations into every $\delta_v(k)$.

Equal-Interval Wakeup Scheduling The wakeup scheduling algorithm is responsible for selecting r out of S slots for packet reception. As highlighted in Section 4.2.4, an equal-interval wakeup scheduling scheme can reduce the expected waiting time. In the evaluation, we use this simple scheme as the focus of the study is on the impact of r on the expected waiting time at v. The slot duration τ is set to 10 ms while $S = 500$.

Collection Tree Routing A simulation model of the Collection Tree Protocol [47] is implemented using ETX as the routing metric. The forwarding engine is modified to control the time instances at which a node v can transmit its packet. Essentially, the forwarding engine at v is only allowed to transmit at the beginning of slots wherein its successor node w is awake.

4.4.2 Model Validation

To determine the validity of the packet arrival probability model (4.7) and the expected service time model (4.23), we conducted simulations where r_v is varied for different (but fixed) values of input traffic D_v. The validation used the network topology shown in Figure 4.6c. The results are taken from node v_3 which has 10 predecessor nodes (labeled u_1 – u_{10}) and successor node w.

Figure 4.7(a) shows the packet arrival probability p_v as a function of r_v, comparing the model with the simulation results for four different D_v values. The plots clearly show strong agreement between the model and simulation results. Both results show that as r_v increases, p_v decreases. This is expected since for a fixed input traffic D_v, the arrival probability in every receive slot drops when there are more allocated receive slots. There is a slight difference in the results at lower values of r_v in the case of $D_v = 15$ and $D_v = 20$. The simulation results show slightly higher packet arrival probability as compared to the model. This difference is due to the conservative nature of the model and this property is magnified at lower r_v, *i.e.*, the effect of contention is worse when there are fewer slots for contention.

Figures 4.7(b) and 4.7(c) show the expected service time at v as a function of r_v for different transmission limit L values and different D_v values, respectively. In Figure 4.7(b), we fixed $D_v = 20$ while in Figure 4.7(c), we used $L = 3$. Once again, the results show a close agreement between the model and simulations. Both results show that the service time increases as r_v is increased and that the increase is more noticeable at higher L or D_v values. The noticeably higher service time results obtained using the model (in lower values of r_v and higher values of

D_v) can once again be attributed to the conservative nature of the model.

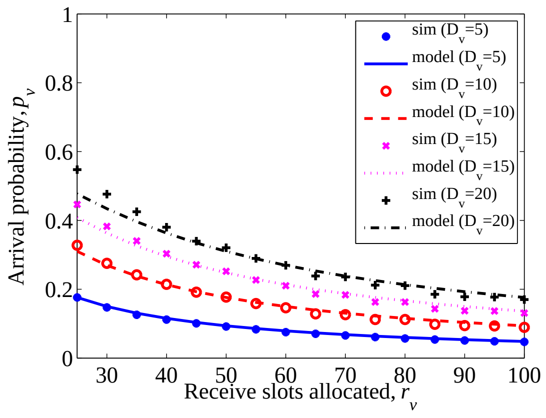
Before ending the discussion, we highlight the significance of the increasing service time with respect to increasing r_v. This is due to the fact that for a fixed input traffic, v receives more packets when it allocates more receive slots. This results in higher utilization factor ρ_v. The increasing service time is also a good reason for considering the two-hop service time in the optimization problem. If the problem was posed such that only v's service time was minimized, then the optimal solution would always be to allocate zero receive slots as this would provide the lowest service time at v. Note however that this allocation would incur infinite service times for the predecessor nodes of v.

4.4.3 Performance Evaluation

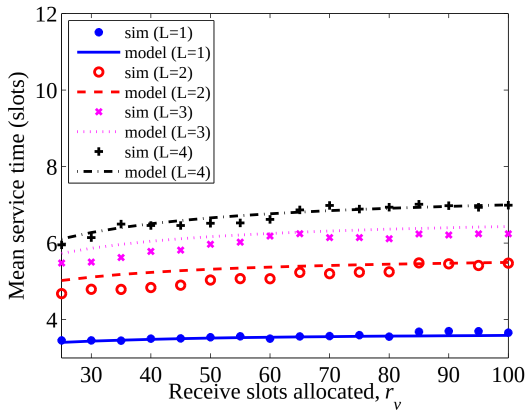
We now proceed to evaluate the performance of LSlotAlloc using a larger scale wireless sensor network topology. To be precise, the network consists of 300 static nodes that are uniformly-distributed in a 500 m $\times$ 500 m area. A single sink node is positioned at (0, 0), *i.e.*, the bottom-left part of the area. Figure 4.8 shows the histogram of hop count to the sink of a typical scenario. We can see that the hop count ranges from 1 to 12, with a tiny fraction even higher than 12 hops. Note that a large fraction of nodes are 5 to 10 hops away from the sink. The positioning of the sink at (0, 0) results in a challenging scenario where data traffic converges to a "narrow spot" in the network. Note that the hop count statistics were obtained using ETX as the path metric. As such, the paths tend to be longer than expected, as ETX uses wireless links that are usually shorter.

The radio transceiver is configured to send at a data rate of 250 kbps and a transmit range of 100 m. Data is generated by randomly-selected nodes which generate variable bit rate traffic with mean packet inter-arrival time of 1 s for a duration of 30 s. This bursty pattern mimics the traffic in event detection and target tracking applications [123].

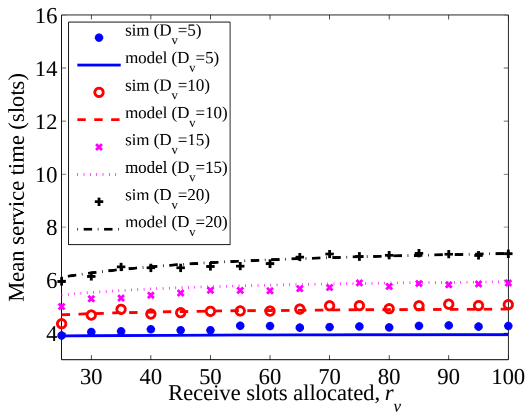
In the simulations, we modeled the physical layer after the bit-error (BER) based reception model that is available in Qualnet [4]. As elaborated in Sec-



(a)



(b)



(c)

Figure 4.7: Validation of packet arrival probability and mean service time models.

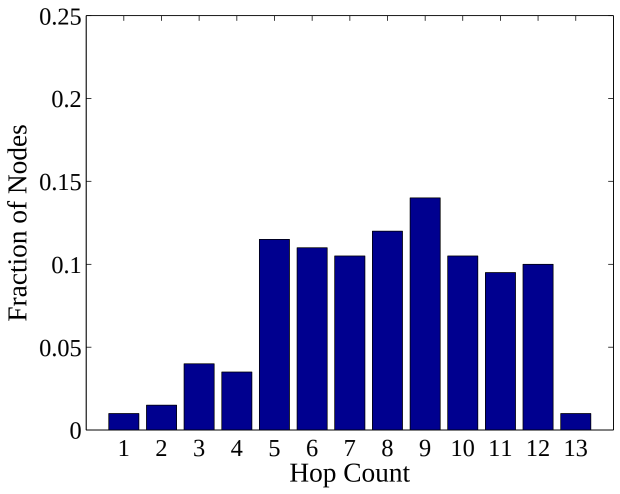


Figure 4.8: Hop count distribution of a typical simulation scenario.

tion 3.4.3 this model uses probabilistic reception of packets based on the BER, thereby simulating the effect of lossy wireless links.

The performance metrics of interest are the (average) end-to-end delay and packet delivery ratio. Since the slot allocation algorithm is formulated to minimize the two-hop service time, end-to-end delay is the primary metric of interest that can indicate the effectiveness of LSLOTALLOC. Each data point is obtained by averaging the results from 20 seed values, with every simulation run configured for 7,200 seconds (2 hours) in simulation time.

Static Allocation

It is possible to implement a simple static allocation scheme such that the number of receive slots $r_v(k) = \lfloor \gamma n_v(k) \rfloor$, where $\gamma \in (0, 1)$. To determine if there is a γ that optimizes the network performance, we ran simulations wherein γ is varied from 0.1 to 0.9.

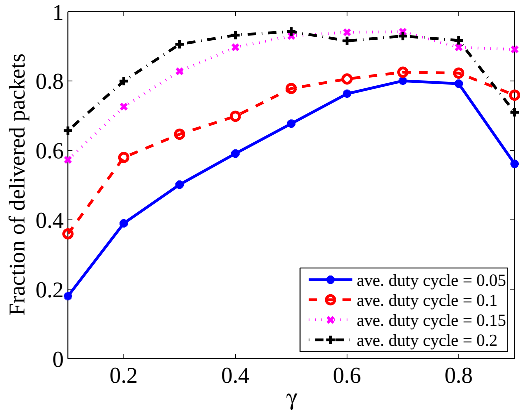
Figures 4.9(a)–4.9(b) show the delivery ratio and delay, respectively, using four duty cycle values. The results show that both metrics are affected by γ and that there are indeed optimal values. In particular, it is apparent that the larger the duty cycle, the larger the range for the optimal values. To illustrate, consider the results for the duty cycles of 0.05 and 0.2. In the former, the delivery ratio peaks at around $0.6 \leq \gamma \leq 0.8$ while the delay is minimal at $\gamma = 0.6$. In the latter, the delivery ratio is optimal when $0.3 \leq \gamma \leq 0.8$ while the delay is minimal when $0.2 \leq \gamma \leq 0.5$.

To gain more insight into the performance of the scheme, we plot the fraction of times that a node has pending packets in its queue but it ran out of transmit slots. We refer to this condition as *transmit slot shortage*. Figure 4.9(c) shows the transmit slot shortage for the static scheme and we can clearly see an increase in the shortage at higher γ. This is understandable since when γ is high, a node has more receive slots (and can receive more packets) and fewer transmit slots. Note however that a non-zero shortage is not totally undesirable. Likewise, a zero shortage is not totally desirable since it may mean that too few receive slots are allocated resulting in negligible number of received packets that need to be forwarded. To illustrate, consider the results for $\gamma = 0.1$ and $\gamma = 0.6$. The former shows zero shortage while the latter shows roughly 10% shortage but yet, the latter provides better performance in both metrics.

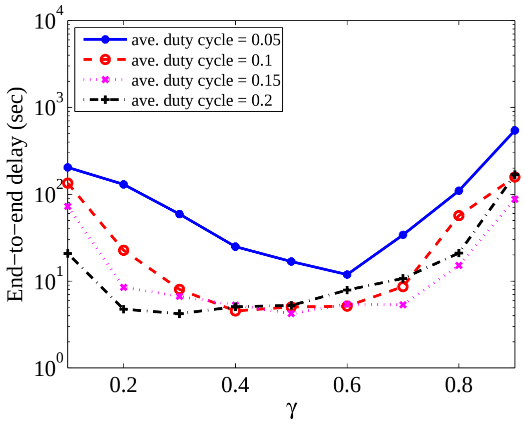
LSLOTALLOC Performance

In Section 4.3.4, we remarked that LSLOTALLOC may not be always feasible because $n_v(k)$ may not be enough to accommodate the expected input traffic to v. In this case, $r = \lfloor \beta n_v(k) \rfloor$ slots are allocated for reception, where $\beta \in (0, 1)$. In terms of delivery ratio (*cf.* Figure 4.10(a)), LSLOTALLOC can obtain the optimal value regardless of β, except when the average duty cycle is 0.05. This is because at this duty cycle, a significant number of executions resulted in the problem being infeasible.

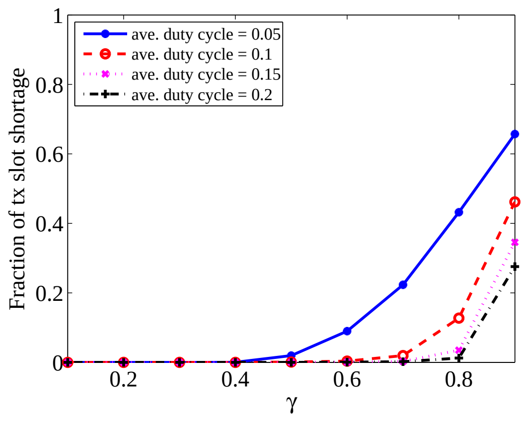
The end-to-end delay seems to be more sensitive to β, with the extreme values causing higher delay (*cf.* Figure 4.10(b)). The result for transmit slot shortage also shows the same trend (*cf.* Figure 4.10(c)). These results indicate the importance of having an appropriate allocation to handle infeasible cases. We can comfortably set $0.3 \leq \beta \leq 0.6$ to ensure both consistent end-to-end delay and transmit slot shortage performance.



(a) Packet delivery ratio

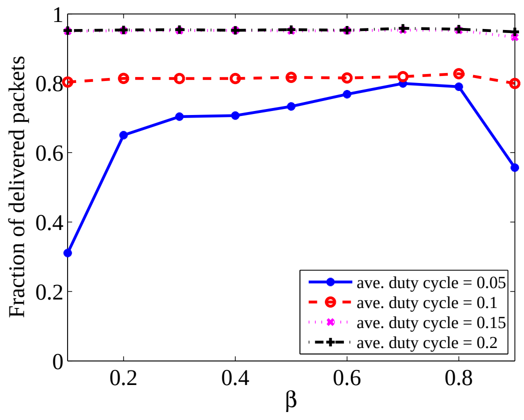


(b) Delay

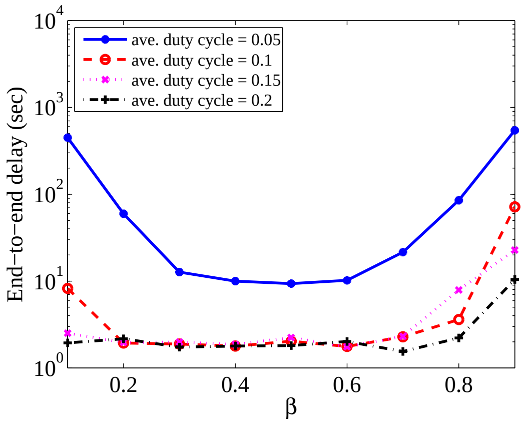


(c) Transmit slot shortage

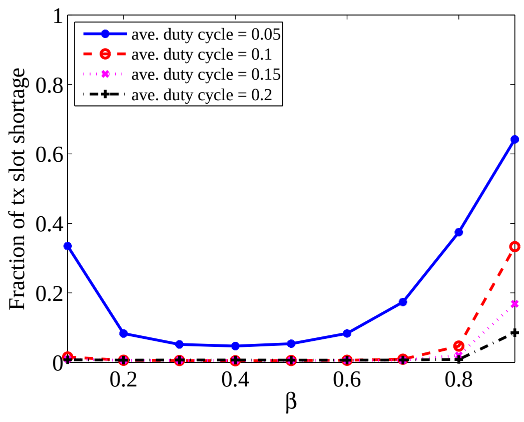
Figure 4.9: Performance of a simple static allocation scheme as a function of γ.



(a) Packet delivery ratio



(b) Delay



(c) Transmit slot shortage

Figure 4.10: Performance of LSlotAlloc as a function of β.

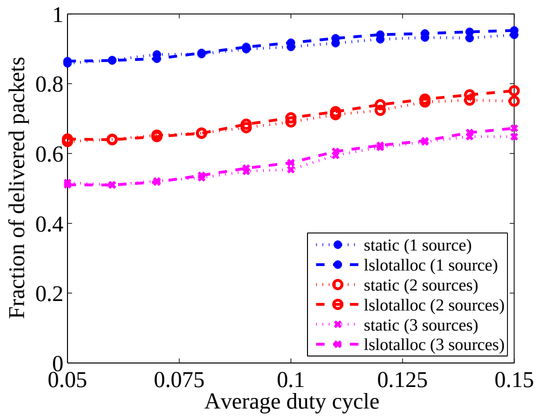
Performance Comparison

We compare the performance of LSLOTALLOC and the static allocation scheme. We use $\beta = 0.6$ and $\gamma = 0.6$ for the two schemes, respectively, since these parameter values provide a good trade-off between the delivery ratio and the end-to-end delay for both schemes. We varied the average duty cycle of the nodes from 5% to 15% and conducted tests with 1, 2 and 3 simultaneous source nodes.

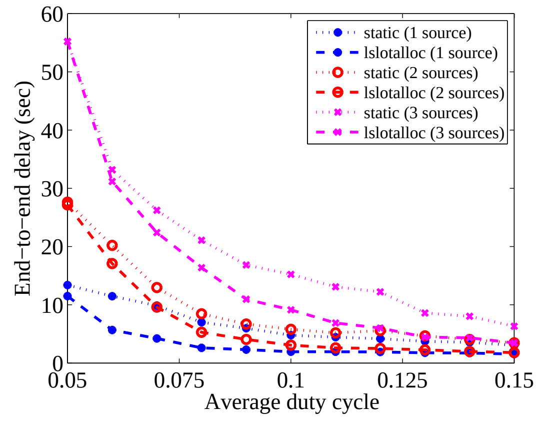
Looking at the packet delivery ratio (*cf.* Figure 4.11(a)), we cannot see any significant difference between the performance of the two schemes in all settings. The results for the end-to-end delay however show a different picture. From Figure 4.11(b), we can see that LSLOTALLOC significantly outperforms the static scheme in almost all settings. To get a clear sense on the performance gain achieved by LSLOTALLOC, we plot the ratio of the delay of LSLOTALLOC over the delay of the static scheme in Figure 4.11(c). The ratio is less than 1 in all settings, indicating that the delay of LSLOTALLOC is better than that of the static scheme. Notably, the delay of LSLOTALLOC is around half the delay of the former in more than half of the settings used.

There is an observable trend where the ratio decreases as the duty cycle increases, reaching minima at a particular value, and increasing from thereon. This behavior can be explained as follows: At lower duty cycles, the gain of LSLOTALLOC is lower because the algorithm encounters significant infeasible traffic flow conditions. Since most of the allocation will default to $r = \lfloor 0.6 n_v(k) \rfloor$ which is the same as the static scheme, the performance gain is limited. As the duty cycle rises, the frequency of infeasible traffic flow decreases, resulting in better performance. The tapering of the performance advantage is due to the fact that at higher duty cycles, the simple allocation is able to provide sufficient number of receive and transmit slots with respect to the input traffic. However, LSLOTALLOC still retains significant performance advantage as its delay is around half that of the static scheme.

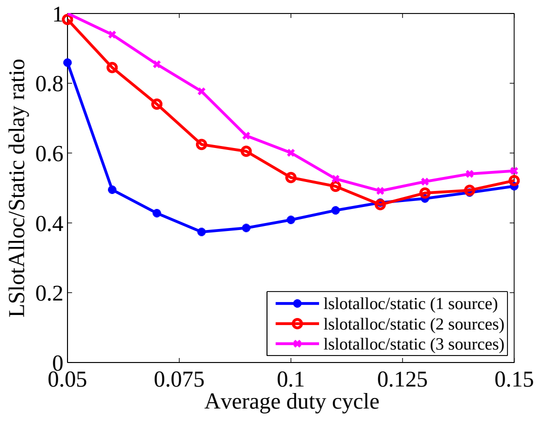
Figure 4.11(c) also indicates that the performance gain of LSLOTALLOC seems to decrease as the number of source nodes increases. This observation is expected



(a) Packet delivery ratio



(b) End-to-end delay



(c) LSLOTALLOC/Static delay ratio

Figure 4.11: Performance comparison between LSLOTALLOC and static scheme.

because as the number of source nodes increases, the number of infeasible traffic flow conditions also increases. This leads to a reduction in the performance gain, as elaborated in the preceding paragraph. In the worst case, *i.e.*, when the number of sources is sufficiently high to cause all problems to be infeasible, the performance of LSLOTALLOC will be on par with that of the static scheme.

4.5 Summary

In environmentally-powered wireless sensor networks, the nodes employ adaptive duty cycling to optimize the utilization of dynamic energy supply and attain energy-neutral operation. A node's duty cycle indicates its budget for both packet reception and transmission. In this chapter, we tackled the problem of apportioning the duty cycle for packet reception and transmission in receive-centric wakeup scheduling schemes. Using discrete-time queueing theory, we derived an analytical model for the service time (which also corresponds to the sleep latency) in the presence of contention. We formulated the duty cycle allocation problem as a constrained non-linear optimization problem that seeks to minimize the two-hop service time. To search for the optimal allocation, we developed a low-complexity algorithm called LSLOTALLOC and using trace-driven simulations, we demonstrated that it significantly outperforms a static allocation scheme in terms of end-to-end delay. Notably, LSLOTALLOC's delay is around half the delay of the static allocation scheme in most scenarios while maintaining the same packet delivery ratio.

To the best of our knowledge, this work is the first to address the problem of duty cycle allocation in receive-centric wakeup scheduling. Naturally, there are many possible avenues for extension that can be undertaken in this area. Some of the most important aspects that are worthwhile to investigate are (*i*) the impact of link quality or channel errors; (*ii*) allocation strategies in the presence of non-stationary duty cycles; and (*iii*) allocation strategies in non-tree-based network topologies or multi-parent collection trees.

Chapter 5

PUMP-AND-NAP: Enabling Sustainable Bulk Transfer

Applications such as indoor and outdoor environmental monitoring generate low data rates, typically in the order of a few bytes to at most several tens of bytes at every sensing interval. This essentially implies that every sensor reading can be encapsulated in a single packet, or a couple of packets at the most. For such applications, the data delivery objectives are to maximize the packet delivery ratio and minimize the end-to-end delay.

There are however applications (*e.g.*, volcano monitoring [124] and railway bridge monitoring [24]), wherein the sensor nodes are tasked to record time-series data at high sampling rates. Such tasks generate large or bulk sensor data, typically in the order of tens to hundreds of kilobytes. These bulk data need to be transferred to a gateway (for eventual transmission to the backend, where further processing and analysis can be undertaken). In bulk transfer, the objective is rather different, in that the entire data needs to be completely delivered to the gateway at the highest possible throughput. Bulk transfer in EPWSNs is challenging because as mentioned, nodes need to perform adaptive duty cycling to ensure uninterrupted operation [68, 120, 137]. In other words, every node must strictly operate according to a specified duty cycle, or risk downtime due to short-term energy shortage.

In this chapter, we tackle the problem of bulk data transfer in EPWSNs where adherence to duty cycle constraints is a primary concern. While several bulk transfer schemes have been proposed [39,40,75,100], they focus mainly on maximizing the throughput, neglecting the duty cycle constraints of sensor nodes. The use of existing schemes may therefore cause uncontrolled and rapid draining of the energy reserves, leading to the temporary unavailability of nodes along the transfer path. Ultimately, this will result in transfer disruptions which render the transfer of arbitrarily-sized sensor data difficult, if not infeasible.

We introduce PUMP-AND-NAP, a forwarding technique that uses *controlled packet trains* to simultaneously maximize throughput and enforce compliance to (dynamic) duty cycle limitations. At the heart of PUMP-AND-NAP is an *adaptive controller* that determines a node's *optimal capacity*, defined as the maximum number of packets the node can receive and transmit in a train within its duty cycle constraints. The controller uses prior input-output observations (capacity allocations and their corresponding duty cycle usage) to continuously tune its performance and adapt to wireless link quality variations.

We implement PUMP-AND-NAP in TinyOS [60] and perform experiments in the Indriya testbed [34], a 139-node indoor testbed, to evaluate its performance. Experimental results show that PUMP-AND-NAP can adaptively track duty cycles and provide high bulk transfer throughput at the same time. More importantly, we demonstrate in energy harvesting experiments that PUMP-AND-NAP can truly enable sustainable bulk transfer compared to state-of-the-art techniques [39, 75] that greedily maximize throughput at the expense of downtime due to energy depletion.

The rest of the chapter is organized as follows. In Section 5.1, we elaborate on the challenges that need to be address by bulk transfer schemes in the context of EPWSN. In Section 5.2, we describe PUMP-AND-NAP in detail while in Section 5.3, we evaluate its performance and compare it with existing bulk transfer techniques. We conclude the chapter in Section 5.4.

5.1 Bulk Transfer In EPWSNs

As highlighted in Section 2.3.3, bulk transfer schemes in EPWSNs need to work with wakeup scheduling to support duty cycling. In this work, we motivate our design using asynchronous schemes because as emphasized in Section 2.1.4, they offer two distinct advantages over synchronous schemes: (*i*) they do not require periodic re-synchronization which can entail significant energy consumption; and (*ii*) they do not require the storage and exchange of wakeup schedules which can entail significant memory and communication overhead. Nevertheless, our resulting scheme can also be used on top of synchronous MAC protocols after slight modifications.

Recall that in asynchronous schemes, a packet transmission is preceded either by a *beacon listening phase* or *preamble(s) transmission phase*[5]. The former is employed in *receiver-initiated* schemes (*e.g.*, [112]) while the latter is used in *transmitter-initiated* schemes (*e.g.*, [21,38,98]). Regardless, the transmitting node always incurs this overhead before it can have the opportunity to transmit its packets. For simplicity, we introduce a common term to refer to either overhead:

Definition 11 (Pre-transmission Overhead)**.** *The duration from the moment a transmitting node v has a packet ready for transmission until the time the receiving node w wakes up. During this time, v's radio is active, either awaiting for a beacon (receiver-initiated) or transmitting preamble(s) (transmitter-initiated).*

Figure 5.1 illustrates the pre-transmission overhead, denoted by Θ_v, of a transmission from node v to node w. Note that Θ_v is heavily influenced by the sleep time T_S. We will elaborate on this in the following discussion.

Now, consider a multi-hop bulk transfer from node s to t. Supposing that we can modify the single packet-based schemes Flush and PIP to operate on top of an asynchronous MAC, the fastest sending rate that a transmitting node v can achieve is to transmit once every wakeup of its successor node w, or $1/(T_\mathrm{L} + T_\mathrm{S})$ (*cf.* Figure 5.1). This is because v needs to wait for the ACK before it can transmit the next

[5]In [38], preambles are replaced by actual data packets.

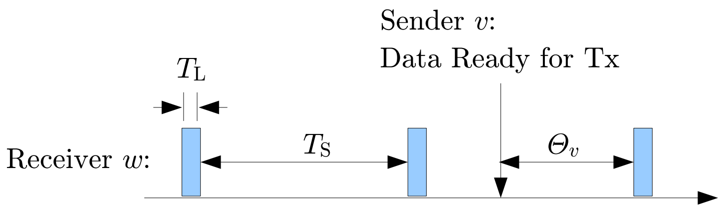


Figure 5.1: Illustrating the pre-transmission overhead of a transmission from v to w, denoted by Θ_v. The shaded boxes denote the wakeup intervals of w.

packet and obviously, it can only receive the ACK when w is awake. Because of rate control, v does not transmit immediately (after receiving an ACK) and hence, v goes back to sleep. Once v transmits later, it must wait again for w to be awake to receive the ACK.

In addition to the low throughput, every packet transmission will have a very high pre-transmission overhead. To see why this is the case, consider Figure 5.1. The moment v becomes ready to transmit, it needs to wait for the next wakeup interval of w, which is Θ_v seconds into the future. If the probability of a packet becoming ready for transmission at v is the same any time, then

$$\Theta_v \sim \mathcal{U}(0, T_S), \tag{5.1}$$

where $\mathcal{U}(0, T_S)$ denotes the uniform distribution in $[0, T_S]$. From (5.1), we can see that on the average, the pre-transmission overhead is $T_S/2$. Hence, transmitting a single packet yields an average efficiency of $\tau/(\tau + T_S/2) = 2\tau/(2\tau + T_S)$, where τ is the transmission time of a packet.

The use of packet trains can clearly remedy the deficiencies of single packet-based schemes. Because wakeup scheduling somewhat limits the opportunities at which nodes can exchange packets, it makes sense to transmit as many packets as possible at every opportunity to improve efficiency. For clarity, we define the notion of a packet train in the context of asynchronous wakeup scheduling as follows:

Definition 12 (Packet Train). *A series of packet transmissions, where only the first packet transmission is preceded by a pre-transmission overhead.*

Note that if L packets are transmitted in a train, the average efficiency increases to $2L\tau/(2L\tau+T_\mathrm{S})$ while the throughput rises to $L/(T_\mathrm{L}+T_\mathrm{S})$ which is L times that of the single packet transmission approach. One difficulty that immediately pops up is what value of L should be used. We have therefore identified the first problem:

Problem 1. *What packet train length should a transmitting node use, given the duty cycle constraints of itself and the receiving node?*

In single-hop scenarios, answering Problem 1 may be sufficient to achieve a duty cycle-compliant bulk transfer. For multi-hop transfers, we need to consider the operation of relay nodes. Consider a relay node v with predecessor node u and successor node w. If v allocates its entire duty cycle for packet train reception from u, then surely, it will use extra duty cycle (*i.e.*, beyond its allocation) to forward them to w. Thus, we have exposed the second problem:

Problem 2. *For relay nodes, how should they allocate their respective duty cycles between packet train reception and packet train transmission?*

The bulk transfer may take considerable amount of time and during this, node duty cycles as well as wireless link qualities may fluctuate. Thus, solving Problems 1 and 2 once is not sufficient. We state the final and third problem as follows:

Problem 3. *Every node along the transfer path needs to periodically review the duty cycle allocation (and hence packet train lengths) to adapt to changes in duty cycle target and wireless link quality and attain optimal performance over time.*

5.2 PUMP-AND-NAP Design

In the design of PUMP-AND-NAP, we focus on a single bulk transfer from s to t that uses a path P_{st}. This is the usual *modus operandi* in data collection, as simultaneous transfers cause *inter-flow interference* which can severely degrade the throughput performance [75] and in severely-constrained sensor nodes, this may entail excessive resource consumption leaving insufficient resources for sensing and data processing. The recommended strategy is to let the gateway or sink node initiate all data transfers to ensure that at most one transfer is on-going at any point in time.

Epoch Time is divided into epochs with fixed duration T. Nodes need not be synchronized, *i.e.*, the start of epochs in nodes u and v need not occur simultaneously. The main reason for dividing time into epochs is to facilitate "periodic review" of PUMP-AND-NAP operating parameters at the start of every epoch.

Target Duty Cycle Nodes employ adaptive duty cycling to balance the dynamic energy supply and demand [68,120,137]. We let $\delta_v(k)$ denote the target duty cycle of v in epoch k which indicates the fraction of time that v can be active for reception and transmission. Note that $\delta_v(k) \in [0, 1]$.

Wakeup Scheduling Scheme PUMP-AND-NAP is designed to work with any asynchronous scheduling scheme that supports back-to-back packet transmissions or packet trains. We employ X-MAC [21] because of its implementation availability in TinyOS and more importantly, it supports packet trains. In the TinyOS implementation, this is possible because a duty-cycled node waits for a specified amount of time (`DELAY_AFTER_RECEIVE`) after its last packet reception before going back to sleep.

5.2.1 Architecture

Figure 5.2 shows the architecture of PUMP-AND-NAP with the major functional blocks. The two main functions provided by PUMP-AND-NAP are hop-by-hop packet train transmission using the pump and nap strategy, and dynamic computation of packet train length using adaptive capacity control. The former will be elaborated in Section 5.2.2 while the latter will be discussed in detail in Section 5.2.3.

PUMP-AND-NAP is specifically designed for dynamic duty cycling sensor networks and as such, it is assumed that an adaptive duty cycle controller provides the optimal operating duty cycle. Nevertheless, PUMP-AND-NAP can also be used

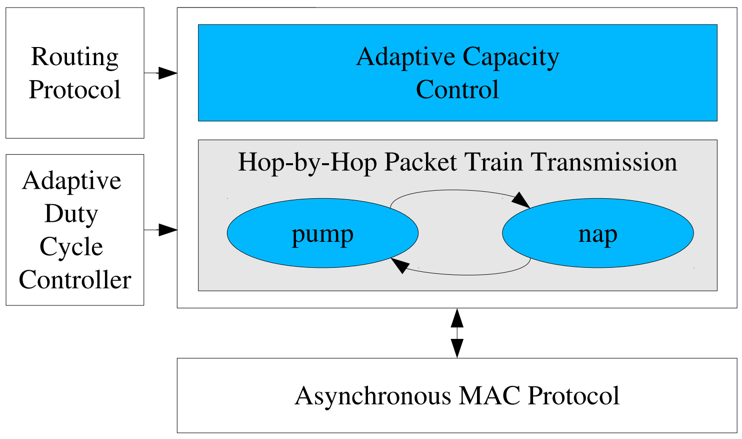

Routing Protocol
Adaptive Duty Cycle Controller
Adaptive Capacity Control
Hop-by-Hop Packet Train Transmission
pump
nap
Asynchronous MAC Protocol

Figure 5.2: PUMP-AND-NAP architecture.

even in static duty cycle scenarios, as will be elaborated at the end of this section. In either case, PUMP-AND-NAP's goal is to ensure that the radio duty cycle will comply with the stipulated duty cycle to ensure long-term sustainability. To perform packet forwarding, PUMP-AND-NAP requires knowledge of the successor node which can be obtained through a routing protocol. Finally, to control the wakeup scheduling of the wireless transceiver and perform efficient packet transmissions, PUMP-AND-NAP relies on an asynchronous MAC protocol that supports back-to-back packet transmissions.

5.2.2 Operation

PUMP-AND-NAP is a forwarding technique that can be used in conjunction with existing bulk transport protocols. As such, PUMP-AND-NAP focuses on two areas: (*i*) the computation of packet train lengths, and (*ii*) the manner by which packet trains are exchanged at every hop, from the source to the sink. In what follows, we describe the operation of PUMP-AND-NAP in a multihop bulk transfer from s to t along a path P_{st}. We assume that every node has a queue for storing packets. We also assume that the transfer has been initiated, and that every node $v \in P_{st}$ has started the operation of its *adaptive controller*, the details of which are presented in Section 5.2.3. For now, it is sufficient to know that the adaptive controller is responsible for computing $r_v(k)$ and $t_v(k)$ at every epoch k, the maximum number of packets that v can receive and transmit, respectively, given its current duty cycle

$\delta_v(k)$.

To commence the transfer, s starts a nap cycle timer which will time out after T seconds (1 epoch). Node s then sends a *train request* to its successor node, say v. When v receives the request, it sends back a *train reply* to s indicating $r_v(k)$, the maximum number of packets that v can receive in the current epoch. Node s then *pumps* at most $\rho_s(k)$ packets back-to-back to v, where $\rho_s(k) = \min[t_s(k), r_v(k)]$. After this *pumping session*, s takes a *nap*, *i.e.*, stops transmissions until the next cycle.

Basic Packet Train Forwarding

Let us look at how an arbitrary relay node v will perform packet transmissions. After receiving a train of packets from its predecessor node u, v performs its own pump-and-nap transmission strategy to its successor node w. That is, v sends a train request to w. After receiving a train reply which indicates $r_w(k)$, v pumps at most $\rho_v(k)$ packets back-to-back to w, where

$$\rho_v(k) = \min[t_v(k), r_w(k)],$$

and immediately takes a nap after this. Note a subtle difference between how s and v performs the pump-and-nap strategy: while s uses a nap timer to trigger pumping sessions, v does not employ any such timer. This is because v's trigger for its pumping session is the end of its packet train reception from its predecessor node u. In the ideal case, the duty cycle usage of v, denoted by $\hat{\delta}_v(k)$, is

$$\hat{\delta}_v(k) := \frac{\tau_v(k)}{T}, \tag{5.2}$$

where $\tau_v(k)$ is the total time that v has been active. This includes the packet train reception time from u, pre-transmission overhead, train request/reply overhead, and packet train transmission time to w.

Wakeup-Synchronized Packet Train Forwarding

In the preceding approach, v commences packet train transmission to w immediately after completing a packet train reception from u. Note that it is possible for v to optimize its duty cycle usage by timing its transmission to begin at the moment that w wakes up. This requires v to know the exact wakeup intervals of w. But for transferring large bulk data, this overhead is justified because it will reduce, if not eliminate, the pre-transmission overhead. This will result in v consuming a lower duty cycle for the same packet train length.

Regardless of whether the basic or wakeup-synchronized packet train transmission is employed, the hop-by-hop packet train transmission strategy is repeated until the sink node t. PUMP-AND-NAP relies on the link layer for reliability and error detection. When a node v fails to receive an ACK after exhausting the specified retransmission limit, the packet being transmitted is *not* dropped; rather, v stops the packet train transmission and immediately takes a nap. Note that when a packet train transmission is abnormally terminated due to such failures, the subsequent packet train transmission will commence from the last unsuccessful packet.

5.2.3 Adaptive Capacity Control

We shall now discuss the design of an adaptive controller that can simultaneously address the three problems posed in Section 5.1. In our design, we adapted the methodology described by Goodwin and Sin [49]. First, we seek a dynamic model that describes the evolution of the quantity that we want to control, *i.e.*, the node duty cycle usage. This dynamic model will contain an unknown system parameter. Second, we formulate the problem as consisting of two parts: estimation of the unknown parameter, and calculation of optimal control law using the parameter estimate.

The motivating problem at node v is to determine $r_v(k)$ and $t_v(k)$, the maximum number of packets that v can receive and transmit, respectively, given its current duty cycle $\delta_v(k)$. Taken together, the sum of $r_v(k)$ and $t_v(k)$ is the *node* *capacity* $C_v(k)$, that is,

$$C_v(k) := r_v(k) + t_v(k).$$

The goal of the adaptive capacity controller is to let the duty cycle usage $\{\hat{\delta}_v(k)\}$ track the target duty cycle $\{\delta_v(k)\}$, for all epoch k, while at the same time maximize $\{C_v(k)\}$.

Input-Output Model

As a matter of convention, we assume that control decisions are done at the start of every epoch k. There is a unit epoch delay before the effects of the control decision can be observed. Thus, if v decides to receive $r_v(k)$ packets and transmit $t_v(k)$ packets at epoch k, we can only ascertain the corresponding duty cycle usage at epoch $k+1$, denoted by $\hat{\delta}_v(k+1)$, which can be obtained by measuring the active time of the radio and using (5.2).

If α and β are the duty cycle 'consumed' for every *successful* packet reception and transmission, respectively, then $\hat{\delta}_v(k+1) = \alpha r_v(k) + \beta t_v(k)$. Note however that this formulation ignores two overheads: (*i*) the pre-transmission overhead as discussed in Section 5.1; and (*ii*) the duty cycle used for the intervals at which v wakes up to listen for transmissions (for transmitter-initiated schemes) or transmit beacons (for receiver-initiated schemes). Denoting U_v for the former and L_v for the latter, we have

$$\hat{\delta}_v(k+1) = \alpha r_v(k) + \beta t_v(k) + U_v(k+1) + L_v. \tag{5.3}$$

The parameter L_v can be treated as a constant since v incurs the same overhead at every epoch. Because there is at most one packet train transmission every epoch, the duty cycle usage of pre-transmission overhead is simply

$$U_v(k) = \frac{\Theta_v(k)}{T},$$

where $\Theta_v(k)$ is a random variable defined in (5.1). With this, $U_v(k)$ is effectively uniform in $[0, T_S/T]$. Note that the index of U_v is $k+1$ because of the fact that

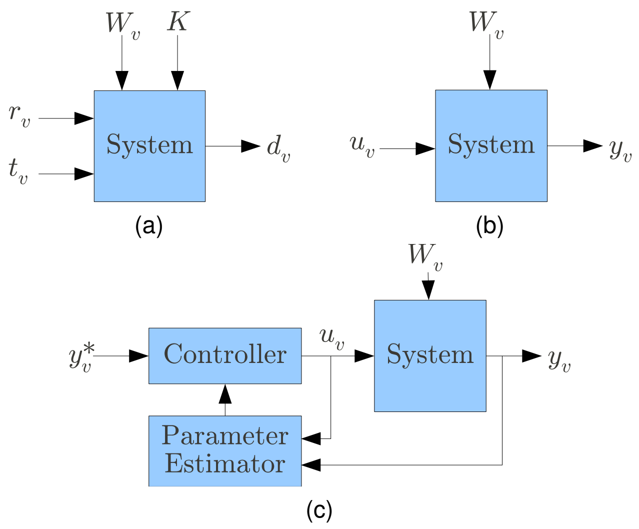


Figure 5.3: Modeling of the system for adaptive feedback control: (a) input-output model; (b) simplified model; and (c) system with adaptive controller.

its effect is only measured together with the measurement of $\hat{\delta}_v(k+1)$. $U_v(k)$ can actually be expressed as the sum of a constant and uniform random variable, that is,

$$U_v(k) = \frac{T_\mathrm{S}}{2T} + W_v(k),$$

where $W_v(k) \sim \mathcal{U}(-T_\mathrm{S}/2T, T_\mathrm{S}/2T)$. Lumping together all the constants as K, (5.3) can be rewritten as

$$\hat{\delta}_v(k+1) = \alpha r_v(k) + \beta t_v(k) + W_v(k+1) + K, \tag{5.4}$$

where $K := L_v + T_\mathrm{S}/(2T)$. A block diagram representation of (5.4) is shown in Figure 5.3(a).

Simplified Model

In what follows, we refine (5.4) to address important considerations such as queue stability and capacity maximization.

Queue Stability Ensuring that queues are stable is important to reduce packet loss due to buffer overflows. For the queue at v to be stable in the long-run, the number of incoming packets must be at most equal to the number of packets that

v can transmit [89], or

$$r_v(k) \le t_v(k). \tag{5.5}$$

Capacity Maximization As mentioned, a key objective of the adaptive controller is to maximize node capacity $C_v(k)$. Given the constraint provided by (5.5), it is easy to see that in order to maximize $C_v(k)$, v must be allowed to receive as much as possible. That is,

$$r_v(k) = t_v(k). \tag{5.6}$$

From (5.6), by letting $u_v(k)$ denote either $r_v(k)$ or $t_v(k)$ and introducing a parameter b, that is

$$u_v(k) := r_v(k) = t_v(k), \tag{5.7}$$

$$b := (\alpha + \beta), \tag{5.8}$$

(5.4) can be rewritten as

$$\hat{\delta}_v(k+1) = bu_v(k) + W_v(k+1) + K. \tag{5.9}$$

We remark that as b encapsulates the "duty cycle cost" to successfully receive and transmit a packet, it is affected by the variations of its incoming and outgoing wireless links. For convenience, we make the following change of variables:

$$y_v(k) := \hat{\delta}_v(k) - K. \tag{5.10}$$

Substituting this into (5.9) yields

$$y_v(k+1) = bu_v(k) + W_v(k+1) \tag{5.11}$$

which is our desired form and is pictorially depicted in Figure 5.3(b). Note that the original control objective is to let $\{\hat{\delta}_v(k)\}$ track $\{\delta_v(k)\}$, for all epoch k. But because $\hat{\delta}_v(k)$ is 'hidden' in (5.11) due to the change of variables, we also define the following for convenience:

$$y_v^*(k) := \delta_v(k) - K. \tag{5.12}$$

The above essentially means that the equivalent control objective is for $\{y_v(k)\}$ to track $\{y_v^*(k)\}$.

Estimation and Control

We shall now use (5.11) to obtain the optimal control $u_v(k)$ that maximizes the capacity of v while ensuring that the duty cycle usage $\{y_v(k)\}$ tracks the target duty cycle $\{y_v^*(k)\}$, for all epoch k. We structure the control system as in Figure 5.3(c). A key component of the system is the parameter estimator, which is responsible for estimating the value of b and essentially makes the controller adaptive. Because (5.11) is linear and the "noise" term has zero mean (*i.e.*, $\mathsf{E}[W_v(k)] = 0$), the least squares estimate of b, denoted by $\hat{b}$, is given by [79]

$$\hat{b} = \frac{\sum_{i=0}^{k-1} y_v(i+1)u_v(i)}{\sum_{i=0}^{k-1} u_v^2(i)}. \tag{5.13}$$

The optimal control law can be obtained by invoking the *principle of certainty equivalence* [79]. It means that we use $\hat{b}$ as though it were the true parameter b. Hence, $u_v(k)$ can be obtained by replacing $y_v(k+1)$ and b in (5.11) with $y_v^*(k+1)$ and $\hat{b}$, respectively, and cancelling $W_v(k)$. This yields $u_v(k) = y_v^*(k+1)/\hat{b}$. Noting that $u_v(k)$ must be an integer, we simply take the floor and obtain

$$u_v(k) = \left\lfloor \frac{y_v^*(k+1)}{\hat{b}} \right\rfloor. \tag{5.14}$$

Estimation and Control for Wakeup-Synchronized Scheme

The estimator $\hat{b}$ in (5.13) and control law $u_v(k)$ in (5.14) are applicable when the basic packet train forwarding scheme is employed. When the wakeup-synchronized approach is used, we need to slightly modify the parameter estimate and control law. Note that in the latter, the pre-transmission overhead vanishes, hence, we can rewrite (5.9) as

$$\hat{\delta}_v(k+1) = bu_v(k) + X_v(k+1) + L_v,$$

where $X_v(k)$ denotes the uncertainty between the time that v commences packet transmission and the exact time that w exactly wakes up. This uncertainty is present because even though v transmits at the wakeup intervals of w, errors in clocks of both v and w are still possible. Following the same arguments as in Section 5.2.3 and assuming that $X_v(k)$ are i.i.d. for all k with mean $\bar{X}_v$, the above can be rewritten as

$$\hat{\delta}_v(k+1) = bu_v(k) + \tilde{W}_v(k+1) + \tilde{K},$$

where $\tilde{W}_v(k)$ is a zero-mean random variable and $\tilde{K} = L_v + \bar{X}_v$. We can therefore define analogues of (5.10) and (5.12) as:

$$\tilde{y}_v(k) := \hat{\delta}_v(k) - \tilde{K} \tag{5.15}$$

$$\tilde{y}^*_v(k) := \delta_v(k) - \tilde{K} \tag{5.16}$$

Finally, the parameter estimator for the wakeup-synchronized packet train forwarding is given by

$$\tilde{b} = \frac{\sum_{i=0}^{k-1} \tilde{y}_v(i+1)u_v(i)}{\sum_{i=0}^{k-1} u_v^2(i)} \tag{5.17}$$

while the optimal control law is given by

$$u_v(k) = \left\lfloor \frac{\tilde{y}^*_v(k+1)}{\tilde{b}} \right\rfloor. \tag{5.18}$$

We have just completed the design of the adaptive controller at v, which will allocate $u_v(k)$ packets for both reception and transmission in the current epoch k. To ensure that v can store all the packets in a train, it must limit the number of packets that it indicates to its predecessor node to

$$r_v(k) = \min[u_v(k), Q - Q_v(k)],$$

where Q is the maximum queue size and $Q_v(k)$ is the queue length at v. Before ending the discussion, we remark the following desirable properties of the controller.

Applicability to any node type The controller was initially designed for a relay node v. However, the use of a single control $u_v(k)$ makes the model applicable for the source and sink nodes as well. In the latter two types, $u_v(k)$ provides the optimal transmit and receive allocations, respectively, without any change.

Adaptation to wireless link variations As mentioned, the parameter b encapsulates the effect of link quality variations. Since b is continuously estimated, the control $u_v(k)$ also automatically adjusts to the link quality fluctuations.

Support for synchronous and asynchronous MAC Our design assumed that the underlying MAC is asynchronous. By removing the third term (due to pre-transmission overhead) in (5.3) and slightly re-defining $y_v(k)$ and $y_v^*(k)$, we can use the controller in conjunction with synchronous MAC protocols.

Usability in static and dynamic duty-cycling In the development of the controller, we did not make any assumption about the target duty cycle $\delta_v(k)$, other than $\delta_v(k) \in [0, 1]$. As such, the controller can also be used in situations where $\delta_v(k)$ is constant, *i.e.*, static duty cycling.

Supporting Simultaneous Transfers

At the start of Section 5.2, we remarked that PUMP-AND-NAP is designed to work well in data collection scenarios wherein the bulk data transfer is managed by a single entity (*i.e.*, the gateway node) and that this single entity ensures that every node in the network is involved in at most one bulk transfer. Nevertheless, PUMP-AND-NAP can be extended to support simultaneous data transfers through the following modifications.

Suppose that a node v is currently supporting a single bulk transfer, and it currently allocates $r_v(k)$ for reception and $t_v(k)$ for transmission. When v receives another *train request*, it simply divides $r_v(k)$ equally into 2. Likewise, if the successor node is different, v divides $t_v(k)$ equally into 2. This process can be repeated for every new bulk transfer, dividing $r_v(k)$ and $t_v(k)$ equally among the distinct number of predecessor and successor nodes, respectively.

We note that the above modifications pose some challenges on the performance of the adaptive capacity controller especially in the case of the basic forwarding scheme. This is because in the design of the controller, only one pre-transmission overhead per epoch is considered. If a node v needs to perform packet train transmissions to several successor nodes, then every such successor node will entail a pre-transmission overhead. As such, the extension of PUMP-AND-NAP to support simultaneous bulk transfers will only work well for the wakeup-synchronized forwarding scheme. The basic scheme can only be employed in scenarios where there is a single data collection point.

5.3 Evaluation

To empirically evaluate PUMP-AND-NAP, we implemented it in TinyOS 2.1.2 [60] and deployed in TelosB motes. Experiments to characterize the performance of PUMP-AND-NAP and simulate energy harvesting scenarios were conducted in the 139-node Indriya indoor testbed [34] while energy-harvesting experiments were conducted in indoor and outdoor locations.

PUMP-AND-NAP Implementation TelosB uses the CC2420 radio which is duty cycled by a component called `PowerCycle`. To measure the duty cycle usage, we implemented two event "hooks" that are invoked from `PowerCycle`, namely `radioStarted()` and `radioStopped()` to indicate the exact instances at which the radio is turned on and off, respectively.

To implement the wakeup-synchronized scheme, we used a readily-available component in TinyOS called `CC2420TimeSyncMessageC`, which enables a node v to inform another node u of the exact time at which an event has occurred. Note that this timing information is piggy-backed in data packets, and thus, no additional message overhead is generated. In our implementation, v always piggybacks its last wakeup interval in any data packet transmission. This enables a receiving node u to deduce all future wakeup intervals of v, since T_L and T_S (*cf.* Figure 5.1) are fixed.

Experiment Settings PUMP-AND-NAP and the underlying X-MAC protocol have several important parameters that need to be specified prior to deployment. For the X-MAC protocol, the wakeup interval T_L is set to 15 ms while the sleep interval T_S is varied to 485, 235, and 110 ms. These values correspond to wakeup rates of 2, 4 and 8 wakeups/second, respectively. The 15 ms wakeup interval was chosen because it provided a good trade-off between overhead and preamble reception probability. For reliability, we used CC2420 software-based ACKs (default setting) and set the retry limit to 7. The latter value was chosen to improve the reliability of individual packet transmissions and to essentially reduce abnormal termination of packet train transmissions. Note that the IEEE 802.15.4 standard [7] allows for the retry limit to be chosen from 0 to 7, with 3 as the default value.

For PUMP-AND-NAP, the epoch duration T and packet buffer space Q are the two key parameters. We chose $T = 3$ seconds in our evaluation. Selecting a shorter T requires more frequent computations but faster reaction to environmental changes while a longer T requires less frequent computations but slower reaction to environmental changes. T also has a direct impact on the efficiency of packet trains. A shorter (longer) T implies shorter (longer) packet trains and therefore less (more) efficient. However, supporting longer packet trains requires nodes to maintain larger packet buffer space. In this work, we used a buffer size of 60 packets which was more than sufficient for the tested scenarios.

We focused our evaluation on the performance of the adaptive controller and packet train forwarding scheme so we used fixed network topologies. In addition, we used a packet size of 64 bytes for transmitting fragments of the bulk data, which is generated on-the-fly at the source nodes.

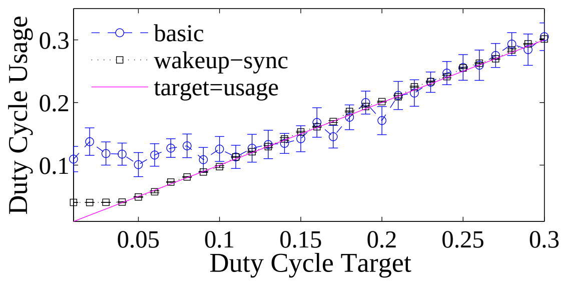
5.3.1 PUMP-AND-NAP Performance

Dynamic Performance

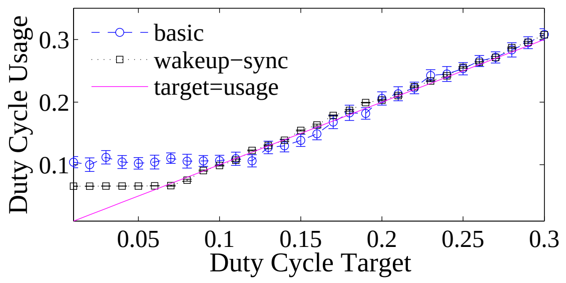
To evaluate the performance of the proposed controllers, we performed experiments in the Indriya testbed [34] involving 3 nodes, namely a source, a relay, and a sink. We selected 10 sets of combinations from the testbed, where the link delivery probabilities from source to relay, and from relay to sink were more than 0.8. The source and sink duty cycles were fixed at 50% whereas the relay duty cycle was changed every minute to a random value in $[1\%, 30\%]$ that had not yet been previously selected. This setup ensured that all possible duty cycle values in the range were tested, and that the relay duty cycle was the bottleneck.

Figure 5.4 and 5.5 show the average duty cycle usage and relay capacity, respectively, of *basic* and *wakeup-synchronized* approaches (as defined in Section 5.2.2) under different X-MAC wakeup rates. The error bars indicate the 95% confidence intervals. As far as tracking is concerned, we can see that both schemes can follow the duty cycle target, except at lower duty cycles. The latter is due to the X-MAC wakeup overhead and pre-transmission overhead (in the case of *basic*). To illustrate, at 4 wakeup/s, the wakeup overhead is $(4\times 15)/1000 = 6\%$, hence we can see in Figure 5.4(b) that the usage of *wakeup-synchronized* does not go below 6%. For *basic*, there is an additional overhead of roughly $T_{\mathrm{S}}/(2T) = 235/(2 \times 3000) \approx 4\%$, hence, its usage is 10% at the minimum. We want to highlight the importance of duty cycle tracking in EPWSNs: if the usage is lower than the target, then it means that the controller is not taking full advantage of the duty cycle. Whereas, if the usage is higher than the target, then it means that the node is using up more duty cycle than allocated. In the long run, this can cause the node to fail due to energy depletion.

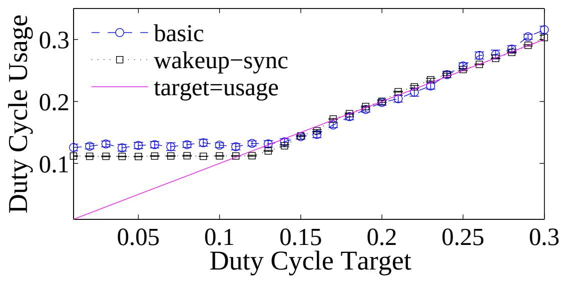
Another noticeable aspect is that *basic* shows higher variation especially at lower wakeup rates while *wakeup-synchronized* provides highly consistent performance regardless of the X-MAC wakeup rate. The higher variation of *basic* is expected because the uncertainty due to the pre-transmission overhead is significantly higher than the clock uncertainty in *wakeup-synchronized*. Moreover, the



(a) 2 wakeup/s

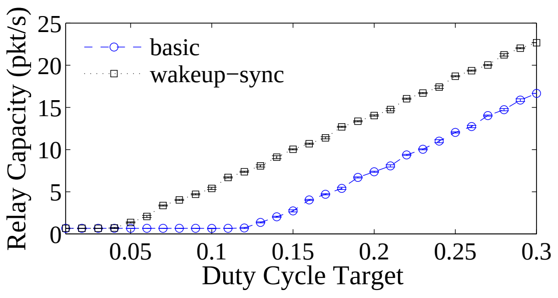


(b) 4 wakeup/s

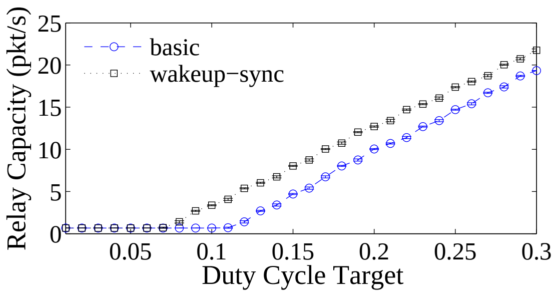


(c) 8 wakeup/s

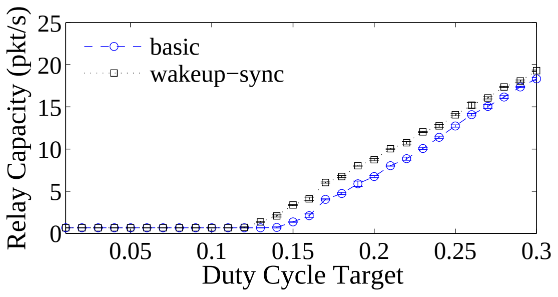
Figure 5.4: Comparing the duty cycle tracking performance of basic and wakeup-synchronized under different X-MAC wakeup rates.



(a) 2 wakeup/s

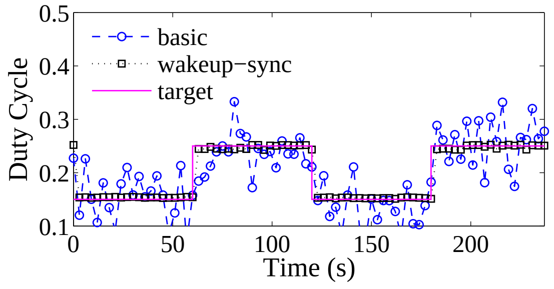


(b) 4 wakeup/s

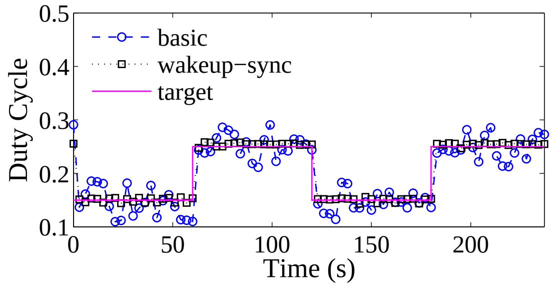


(c) 8 wakeup/s

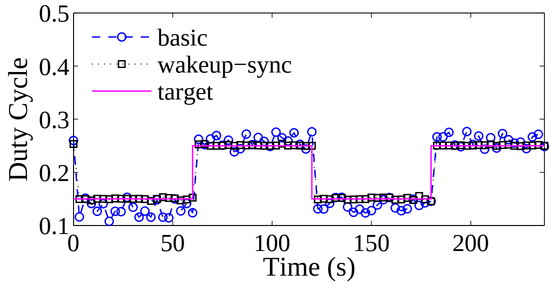
Figure 5.5: Comparing the relay capacity of basic and wakeup-synchronized under different X-MAC wakeup rates.



(a) 2 wakeup/s



(b) 4 wakeup/s



(c) 8 wakeup/s

Figure 5.6: Snapshots of controller response when duty cycle target abruptly changes, under different X-MAC wakeup rates.

former is sensitive to the X-MAC wakeup rate, *i.e.*, at lower wakeup rates, the variation is higher because T_S is larger (*cf.* Figure 5.1).

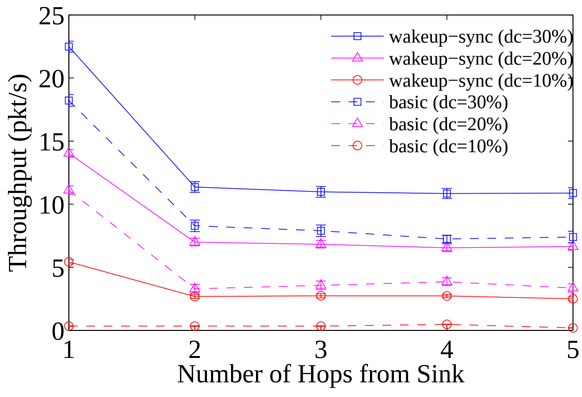
To emphasize the sensitivity of *basic*, we show snapshots of the controller response when the duty cycle target changes abruptly from 15% to 25% (and vice versa) every minute in Figure 5.6. Note the stable dynamic response of *wakeup-synchronized*, regardless of the X-MAC wakeup rate. Compare this with *basic* which is highly oscillatory because of the high pre-transmission overhead uncertainty. The lower the wakeup rate, the higher the uncertainty which ultimately results in wider oscillations.

With respect to the relay capacity, we can observe that both schemes provide consistent (low variation) capacity. At lower duty cycles, both schemes yield negligible capacity because the X-MAC wakeup overhead and pre-transmission overhead (in the case of *basic*) use up the entire duty cycle. The advantage of synchronization is noticeable, as *wakeup-synchronized* shows better performance compared to *basic* in all wakeup rates due to the elimination of pre-transmission overhead. Its advantage is higher at lower wakeup rates because of the lower X-MAC wakeup overhead in those settings. At 8 wakeup/s, the performance of both schemes are comparable because the X-MAC wakeup overhead becomes dominant.

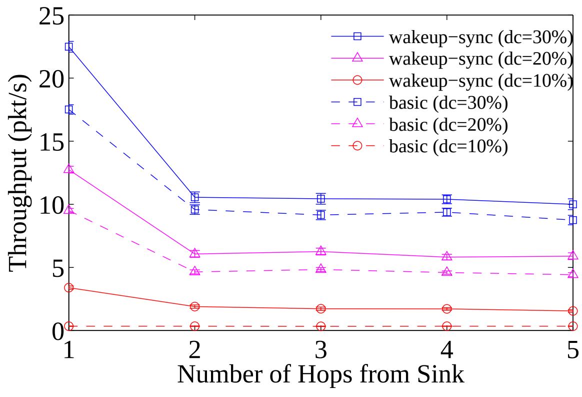
Multihop Performance

To see how PUMP-AND-NAP will perform in multihop deployments, we ran experiments where the number of hops from the source to the sink is varied from 1 to 5 hops. For every hop count, we tested three duty cycle targets, namely 10%, 20%, and 30%. Each experiment was run for 1 minute and repeated 10 times. We monitored the bulk transfer throughput, and these are shown in Figure 5.7. The plots also show the 95% confidence intervals.

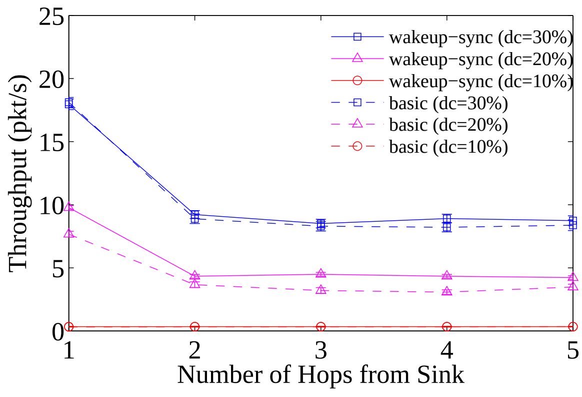
Except for the settings that yielded negligible throughput (caused by the usage of the entire duty cycle for X-MAC wakeup overhead and pre-transmission overhead), we can see a big drop from 1 to 2 hops for the rest, with the latter throughput being just around half of the former. This is expected because for single hop trans-



(a) 2 wakeup/s



(b) 4 wakeup/s



(c) 8 wakeup/s

Figure 5.7: Throughput performance of basic and wakeup-synchronized under different X-MAC wakeup rates.

fers, the source does not need to allocate any duty cycle for reception and that it can allocate its entire duty cycle for transmission. For 2–5 hops, the throughput is almost the same for every scheme, due to the fact that the relay nodes are able to maximize the allocated duty cycle. Once again, we can see the distinct advantage of *wakeup-synchronized*, as its throughput is higher than *basic* especially at lower X-MAC wakeup rates. The lower throughput of *basic* is due to the pre-transmission overhead, which is relatively higher at lower wakeup rates.

The flat throughput results for 2–5 hops is counter-intuitive. We have expected the throughput to decrease with path length because as the number of hops increases, intra-flow interference due to contention worsens. To understand why this is the case, we pictorially analyze the "airtime usage" of a 5-hop bulk transfer in Figure 5.8. We define the airtime usage as the total time (in an epoch) that the nodes used for transmission and reception of packet train from the source to the sink. Suppose that the target duty cycle of all the nodes is 30%. Then node 1 will allocate all of its duty cycle, *i.e.*, 30% for packet transmission. Relay nodes 2, 3 and 4 will split their duty cycles accordingly, say 15% for reception and 15% for packet transmission, for simplicity. Finally, sink node 5 will allocate its entire 30% for packet reception. While node 1 can utilize 30% to transmit to node 2, it will only be able to use 15% because node 2 limits the packet train transmission. Likewise, while node 5 can use 30% for reception, node 4 limits its usage to only 15%. Thus, the total airtime usage is 60% of the epoch duration which is the sum of the following: 15% from node 1 to 2; 15% from node 2 to 3; 15% from node 3 to 4; and 15% from node 4 to 5. We can use the same figure to analyze the airtime usage of 2, 3, and 4 hops to show that the airtime usage of these transfers are well below 100% and will therefore sidestep the problem of intra-flow interference. To clarify why intra-flow interference will not occur, note that because the airtime usage is below 100%, all packet train transmissions by nodes 2–4 have already completed before node 1 initiates a new packet train in the next epoch.

We now want to emphasize the following: for all transfers involving 2, 3, 4 and 5 hops, the relay nodes limit the packet train size or duration to 15% of the

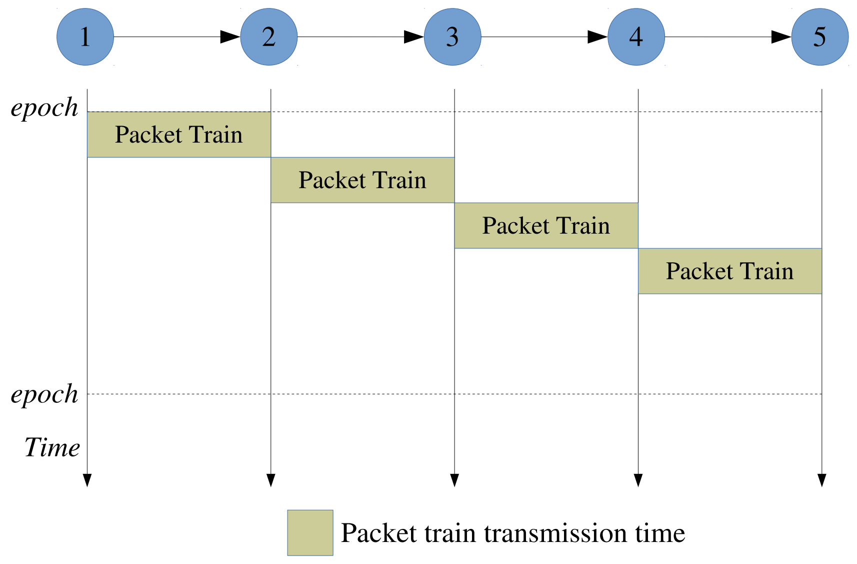


Figure 5.8: Airtime usage of a packet train transmission from the source to the sink. If the target duty cycle of the nodes is 30%, then each packet train requires 15% of the epoch. Hence the total airtime is $4 \times 15 = 60$% of the epoch duration.

epoch duration. Since the source is allowed to initiate 1 packet train transmission every epoch, then the throughput is determined by the packet train duration. This explains the flat throughput for 2–5 hops.

5.3.2 Energy Harvesting Experiments

Finally, we conduct experiments involving a real energy-harvesting node to determine whether PUMP-AND-NAP can indeed provide sustainable bulk transfer. The setup involves a 2-hop bulk transfer: the source and sink nodes are powered through the USB port while the relay node uses energy-harvesting. Figure 5.9 shows the schematic diagram of the energy-harvesting relay node. It uses 4 solar panels that can generate up to 22 mW of power, and a 1 Farad supercap as energy store. BQ25504 EVM [1] is a power management circuit that controls the supercap charging and energy supply to the mote. It is configured to charge the supercap to 3.1 V. We implemented a simple *voltage-duty cycle mapping* to generate the target duty cycle $\delta(k)$ at every epoch k, given by:

$$\delta(k) = \max[b(k) - 2.5, 0],$$

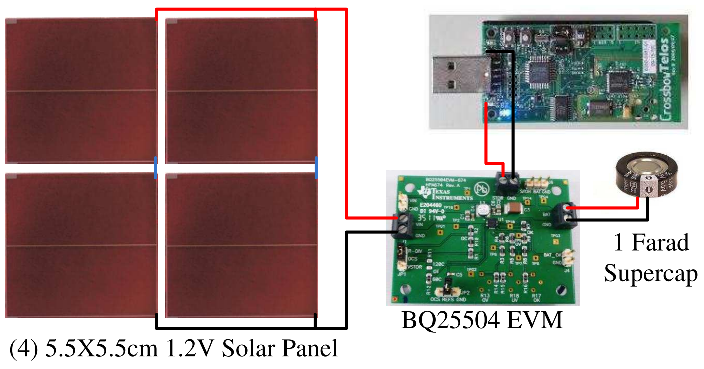


Figure 5.9: Energy harvesting experimental setup.

where $b(k)$ denotes the supercap voltage at epoch k. Note that this simple mapping allows a maximum duty cycle of 60% since $b(k) \leq 3.1$. Note also that when the supercap voltage is less than or equal to 2.5 V, $\delta(k) = 0$. We made this stipulation because based on observations, the mote does not function when the voltage drops below 2.5 V. This simple mapping is definitely not optimal and as such, we anticipate that PUMP-AND-NAP can benefit from more sophisticated adaptive duty cycle schemes such those proposed in [68, 120].

We compare PUMP-AND-NAP with some of state-of-the-art bulk transfer techniques presented in Section 2.3: (*i*) packet train-based transmissions which is employed in [39], and (*ii*) Flush [75]. For the former, we tested both unsynchronized and wakeup-synchronized. Two energy harvesting scenarios were used: (*a*) *indoor scenario* – solar panel was exposed to a lamp with 10 klux illuminance; and (*b*) *outdoor scenario* – solar panel was exposed under direct sunlight with 100 klux illuminance. (The Extech HD450 Lux Meter was used to measure the illuminance.) Every scheme was run 10 times, and every run was scheduled for at most 2 hours for practical reasons. For fairness, all experiments under (b) were performed when the sun was unobstructed by any cloud. We checked that the supercap was at 3.1 V prior to the start of every run.

Rationale for Using 10 klux and 100 klux

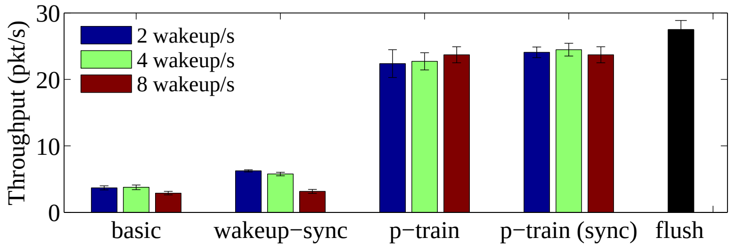
We used these values because based on our measurements, outdoor daytime illuminance ranges from 10–100 klux during a fair sunny day. In [45], the authors have shown through measurements that for solar altitudes above 10 degrees, the direct normal illuminance is more than 10 klux. It is easy to consistently obtain 100 klux outdoors, which happens when the sun is not obstructed by any cloud at around 1-4 pm. It is however difficult to obtain a consistent 10 klux. Hence, we performed indoor experiments with a lamp that was placed at a distance such that the illuminance reaching the solar panel is around 10 klux. By choosing these two values of illuminance, *i.e.*, 10 klux and 100 klux, we can use the results to infer that (*i*) the scheme should be able to provide sustainable bulk data transfer within 10–100 klux of illuminance or a fair sunny day, and (*ii*) the scheme automatically adjusts the throughput according to the variations in energy availability (klux).

Experimental Results

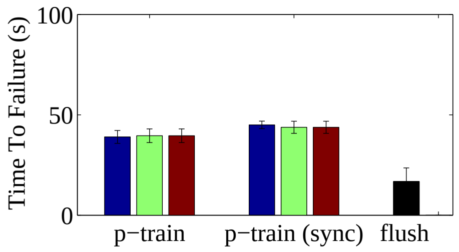
Figures 5.10 and 5.11 show the throughput and mean time before the relay node failed due to energy exhaustion, in indoor and outdoor scenarios, respectively, with the error bars indicating the 95% confidence intervals. Note that *p-train* and *p-train (sync)* denote unsynchronized and wakeup-synchronized packet-train forwarding schemes, respectively. PUMP-AND-NAP is not included in Figures 5.10(b) and 5.11(b) because it can achieve uninterrupted operation, *i.e.*, it lasted for the entire duration of the experiments (2 hours).

In either scenario, *flush* yields the highest throughput at around 28 pkt/s. However, the transfer is short-lived, lasting for only 16.8 s indoors and 36.4 s outdoors. Meanwhile, the use of packet trains can indeed improve the energy-efficiency of bulk transfer. Although its throughput is slightly lower than *flush* by at most 18%, *p-train* can last more than twice that of *flush* in both illuminance conditions. Comparing *p-train* and wakeup-synchronized *p-train*, we can observe a slight advantage of the latter. While both schemes yield comparable throughput, the latter can last slightly longer by at most 17 s, due mainly to the energy savings from pre-transmission overhead.

At this point, we highlight that our *p-train* implementation is sub-optimal compared to the implementation in [39] because we used a burst duration of 1500 ms

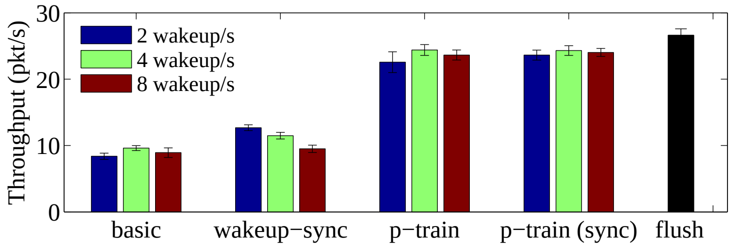


(a) Throughput

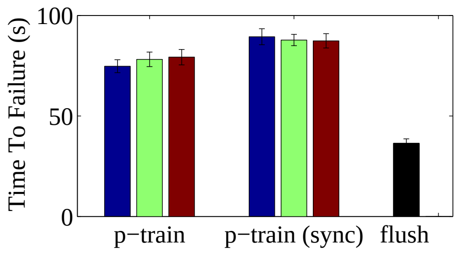


(b) Mean Time to Failure

Figure 5.10: Throughput of PUMP-AND-NAP (basic and wakeup-synchronized), packet train (unsynchronized and wakeup-synchronized) and flush, and mean time to relay node failure of the latter two, in indoor scenario (10 klux).



(a) Throughput



(b) Mean Time to Failure

Figure 5.11: Throughput of PUMP-AND-NAP (basic and wakeup-synchronized), packet train (unsynchronized and wakeup-synchronized) and flush, and mean time to relay node failure of the latter two, in outdoor scenario (100 klux).

and did not employ optimizations such as storage interleaving and multichannel support. We expect that with these techniques, *p-train* can attain an even higher throughput. Nevertheless, these techniques are not meant for *intentional duty cycle usage control* and are therefore not expected to improve the sustainability of the scheme. As for PUMP-AND-NAP, it is the only scheme that can provide uninterrupted transfer, regardless of illuminance and X-MAC wakeup rate. It accomplishes this by adjusting its throughput according to the energy availability. This is evident in the results as we observe that the throughputs of both *basic* and *wakeup-synchronized* in indoor experiments are around 1/2 that of outdoor experiments. And though PUMP-AND-NAP's best throughput is around 1/4 and 1/2 that of *p-train* in indoor and outdoor scenarios, respectively, the transfer can last for an indefinite amount of time. This will enable PUMP-AND-NAP to transfer bulk data of any size. This is clearly not possible with either *flush* or *p-train*.

5.3.3 Energy Harvesting Simulations

To further study the effect of energy harvesting rate and path length in a controlled setting, we perform simulations in the Indriya testbed [34], wherein the energy harvesting and consumption processes are emulated.

Energy Harvesting Process

We model the energy harvesting process after the energy harvesting node shown in Fig. 5.9. To characterize its harvesting process, the load (*i.e.*, the TelosB mote) is disconnected and the voltage across the supercap is sampled at every epoch (3 s), as the solar panel is exposed to 10 klux and 100 klux light. The supercap is first discharged to around 2 V prior to the characterization. Fig. 5.12 shows the voltage over time across the supercap. We only consider the charging rate at and above 2.5 V because once a node goes below this voltage, it is considered non-operational. Now, from elementary circuit theory,

$$I(t) = CdV(t)/dt,$$

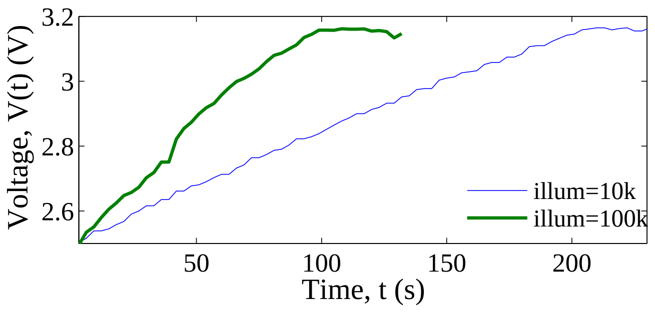


Figure 5.12: Voltage across the supercap, under 10 klux and 100 klux illuminance.

where $V(t)$ is the supercap voltage at t, C is the capacitance, and $I(t)$ is the current flowing in or out of the supercap at t. Since $C = 1$, the harvesting rate is simply

$$I(t) = dV(t)/dt$$

or the slope of $V(t)$. From Fig. 5.12, we can see that the voltage increases almost linearly with time from 2.5–3.1 V, suggesting that the harvesting rate can be approximated by a constant value within this region. Using the measurements obtained, the average charging current at 10 klux and 100 klux are 3.81 and 6.48 mA, respectively. In the simulations, we vary the harvesting rate from 3.8–6.5 mA to mimic the above conditions.

Energy Consumption Process

To emulate the energy consumption, we measured the current consumption of TelosB at 3 V in three modes of operation and obtained the following: (*i*) MCU is active and radio is in deep sleep: $I_1 = 2$ mA; (*ii*) MCU is active and radio is in receive mode: $I_2 = 23$ mA; and (*iii*) MCU is active and radio is in transmit mode at 0 dBm: $I_3 = 21$ mA. These are similar to the findings in [99].

We implemented a TinyOS module to emulate the above harvesting and consumption processes. Briefly, the supercap voltage $V(k)$ evolves according to this difference equation:

$$V(k) = \mathcal{Q}_{\rm C}(k-1) - \mathcal{Q}_{\rm D}(k-1) + V(k-1), \tag{5.19}$$

where $\mathcal{Q}_{\mathrm{C}}(k)$ denotes the charge accumulated at k due to the energy harvesting process while $\mathcal{Q}_{\mathrm{D}}(k)$ denotes the total discharge at k due to the energy consumption process. Eq. 5.19 is the discrete version of the well-known relation

$$V(t) = \frac{1}{C}\int_{t_0}^{t} I(t)dt + V(t_0) = \int_{t_0}^{t} I(t)dt + V(t_0)$$

since $C = 1$. To mimic the uncertainty in the harvesting process,

$$\mathcal{Q}_{\mathrm{C}}(k) = I_{\mathrm{C}}T + \omega(k),$$

where $I_{\mathrm{C}} \in [3.8, 6.5]$ mA is the simulated charging current, T is the epoch duration, and $\omega(k)$ is a random number generated using the `RandomC` component. Note that $\omega(k) \sim \mathcal{U}(-\Omega, \Omega)$, where Ω is chosen to capture the variability of the harvesting rate on an epoch by epoch basis. We make this simplification because TinyOS only provides modules that can generate uniformly distributed random numbers. Meanwhile,

$$\mathcal{Q}_{\mathrm{D}}(k) = I_1 T + I_2 \tau_{\mathrm{rx}}(k) + I_3 \tau_{\mathrm{tx}}(k),$$

where $\tau_{\mathrm{rx}}(k)$ and $\tau_{\mathrm{tx}}(k)$ are the times spent in receive and transmit modes at epoch k, respectively.

We ran simulations in the Indriya testbed [34] for PUMP-AND-NAP (wakeup-synchronized), packet train (wakeup-synchronized) and Flush, and fixed the X-MAC wakeup rate to 4 per second. Unlike the experimental setup in Section 5.3.2, all nodes in the simulations are powered by energy harvesting. Every scheme was run 10 times.

Influence of Energy Availability

To investigate the performance of the schemes with respect to energy availability, we conducted testbed simulations where the harvesting rate is varied from 3.8–6.5 mA, representing the energy that can be scavenged from 10–100 klux. Fig. 5.13a shows the throughput of the three schemes, while Fig. 5.13b shows the mean time to failure of packet train and Flush. PUMP-AND-NAP is not included in the latter plot because it can sustain the bulk transfer indefinitely. The results are obtained using a 5-hop bulk transfer. The error bars show the 95% confidence interval.

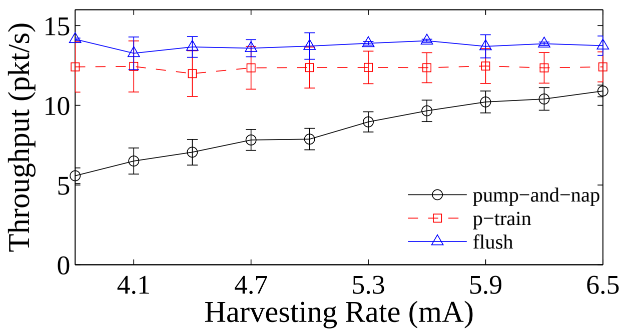
The throughput results suggest that PUMP-AND-NAP is the only scheme that adapts to energy availability. While packet train and Flush respectively yields the same throughput regardless of the harvesting rate, PUMP-AND-NAP shows a throughput that increases as the harvesting rate increases. Specifically, its throughput almost doubles from 5.5 pkt/s at 3.8 mA to 10.9 pkt/s at 6.5 mA. This important result demonstrates the effectiveness of employing the adaptive capacity controller to automatically adjust the relay capacity of nodes according to what energy availability can support.

In terms of the mean time to failure, we can observe that Flush performs poorly compared to packet train. Moreover, its performance seems to be almost the same regardless of the harvesting rate. A closer inspection of the results, however, reveal that Flush slightly improves its performance from 24.6 s at 3.8 mA to 37.2 s at 6.5 mA. Whereas, the effect of harvesting rate is significantly noticeable in the case of packet train. In fact, its bulk transfer at 6.5 mA lasts 334.8 s, almost three times that at 3.8 mA which only lasts by 112.2 s. The advantage of packet train over Flush can be attributed to its use of duty cycling.

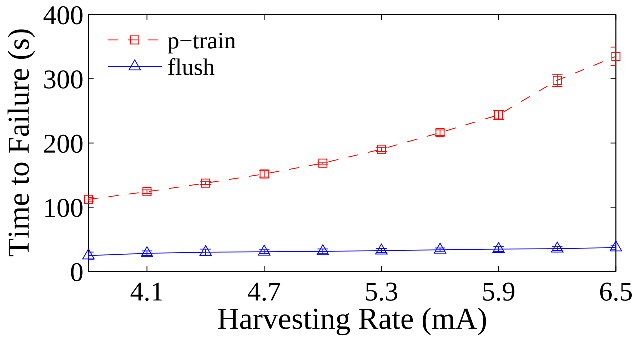
Influence of Hop Count

Fig. 5.14a shows the throughput of the PUMP-AND-NAP, packet train and Flush, as the number of hops between the source and the sink increases from 1 to 5. Meanwhile, Fig. 5.14b plots the mean time to failure of packet train and Flush. Once again, PUMP-AND-NAP is not included in the plot because it can sustain the bulk transfer indefinitely. The results simulate 6.5 mA harvesting rate, which is equivalent to the harvesting rate at 100 klux.

The throughput of packet train and Flush are comparable, and both show a decline as the path length increases. This is expected because as the path length increases, the increasing contention due to intra-flow interference causes these transfer schemes to throttle down their respective sending rates. For PUMP-AND-NAP,



(a) Throughput



(b) Mean Time to Failure

Figure 5.13: Throughput of PUMP-AND-NAP, packet train and Flush, and mean time to relay node failure of the latter two, of a 5-hop bulk transfer, as a function of energy harvesting rate.

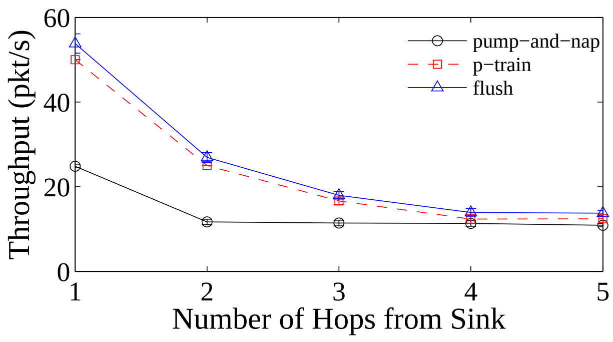
we observe a slightly different trend. We can see a big drop from 1 to 2 hops, with the latter throughput being just around half of the former. This is expected because for single hop transfers, the source does not need to allocate any duty cycle for reception and that it can allocate its entire duty cycle for transmission. For 2–5 hops, the throughput remains flat because of the effect of controller action to limit the usage of the radio. Essentially, the duty cycle enforced is sufficiently low that intra-flow interference is avoided.

With respect to the mean time to failure, Flush shows a flat response regardless of the path length. This is because in Flush, the radios are always on, resulting in roughly the same energy consumption regardless of the path length. As for packet train, we observe an interesting trend where the nodes last longer as the number of hops increases. This interesting result is due to the fact that as the number of hops increases, the frequency of packet train transmissions decreases, as evidenced by the decreasing throughput. This leads to the reduction of the amount of time that the radios need to be active. In other words, the duty cycle usage of packet train is highly dependent on the path length, with the duty cycle usage decreasing as the number of hop increases. This indicates the possibility for packet train to attain sustainable data transfer at a certain number of hops. We however remark that such sustainability is achieved *incidentally*, compared with the sustainability provided by PUMP-AND-NAP that is attained *intentionally* at all hop counts.

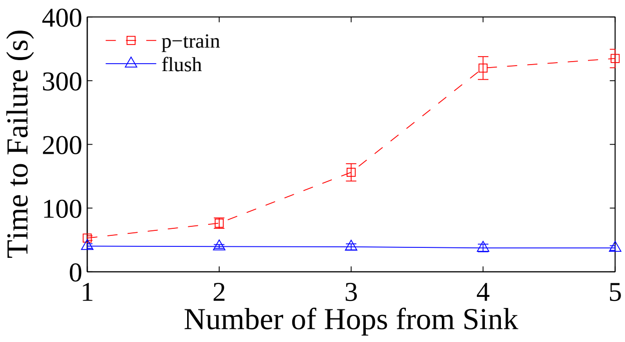
5.4 Summary

This work addresses the problem of bulk data transfer in EPWSNs where duty cycle compliance is critical. While several bulk transfer schemes have been proposed in the literature, they focus mainly on maximizing the transfer throughput, neglecting the duty cycle constraints of sensor nodes.

We proposed PUMP-AND-NAP, a packet train forwarding technique that uses adaptive feedback control to calculate the optimal packet train length for both reception and transmission. The adaptive feedback control aims to control the duty



(a) Throughput



(b) Mean Time to Failure

Figure 5.14: Throughput of PUMP-AND-NAP, packet train and Flush, and mean time to relay node failure of the latter two, at harvesting rate of 6.5 mA, as a function of path length.

cycle usage of a node, modeled as a linear system with zero-mean disturbance. The latter is mainly due to the uncertainty induced by the pre-transmission overhead in asynchronous wakeup scheduling.

The controller uses prior input-output observations (capacity allocations and their corresponding duty cycle usage) to continuously tune its performance and adapt to wireless link quality variations. Because of its reliance on local information, the controller is amenable to distributed implementation. We implemented PUMP-AND-NAP in TinyOS and evaluated its performance in real energy harvesting experiments and testbed simulations. Results show that PUMP-AND-NAP provides high transfer throughput while it simultaneously tracks the target duty cycle. More importantly, energy harvesting experiments show PUMP-AND-NAP is the only scheme that can provide sustainable bulk transfer compared to the other state-of-the-art techniques that we have tested, as the latter greedily maximize throughput at the expense of high and uncontrolled energy consumption.

Chapter 6

Conclusion and Future Work

An environmentally-powered wireless sensor network is an ad hoc deployment of sensor nodes powered by energy harvested from the environment. EPWSNs are becoming more viable due to breakthroughs in energy harvesting technologies and ultra low-power computing and communication devices. One of the major appeals of EPWSNs is their potential to address the problem of limited lifetime which is a major drawback of battery-powered wireless sensor networks. The utilization of renewable energy can enable EPWSNs to operate indefinitely without the need for battery replacement which is not only laborious or expensive but also infeasible in certain scenarios.

But while energy harvesting can theoretically facilitate perpetual network operation, it poses a major constraint on energy availability: the amount of energy available for consumption at any given instant can be unpredictable and changes over time. Thus, unlike battery-powered WSNs where the aim is to maximize network lifetime through energy conservation, the key objective in EPWSNs is to efficiently and adaptively utilize available energy to optimize the network performance. The new guiding principle in EPWSNs is *energy-neutrality* or *energy neutral operation*, which means balancing the energy demand and consumption while at the same time optimizing a desired network performance objective. Energy neutrality in conjunction with dynamic energy supply entails two major challenges, namely, dynamic duty cycling and dynamic sleep latency.

In this thesis, we have tackled the problem of data delivery amidst energy harvesting nodes, following the design principle of energy-neutrality to enable sustainable node operation while maximizing network performance. To this end, we have proposed schemes for dynamic wakeup scheduling, reliable and low latency path selection, dynamic duty cycle allocation and sustainable bulk transfer.

6.1 Dynamic Wakeup Scheduling

A major challenge in EPWSNs is sleep latency due to the fact that a transmitting node must wait for the intended receiver node to be awake before it can commence transmission. Our first contribution is therefore the design of a wakeup scheduling scheme that not only supports dynamic duty cycling but also tackles the problem of sleep latency. In addressing this issue, we have established an important result to characterize the sleep latency that is entailed by a dynamic wakeup schedule. We have demonstrated that the expected sleep latency is affected by the variance of the intervals between the receive wakeup slots: When the variance of the intervals is low (high), the expected latency is low (high). This is because when the intervals are highly uneven, it is more likely for a packet to become ready for transmission at a larger interval than a shorter interval. This result calls for the design of wakeup schedules wherein the receive wakeup slots are positioned at equal intervals. However, such a schedule is not robust to changes in the duty cycle. We have therefore proposed sequence-based scheduling which is essentially a method for representing a receive wakeup schedule using a mathematical integer sequence. More than being robust, this scheme also allows wakeup schedules to be represented in a compact manner, thereby reducing communication and storage overheads.

We have introduced a particular sequence-based scheduling scheme that uses the *bit-reversal permutation sequence* (BRPS). Through analysis, its worst-case expected sleep latency has been proven to be slightly worse than the ideal scheme (*i.e.*, a scheduling scheme where the receive wakeup slots are equally-spaced) but better than schemes where the receive wakeup slots are spaced at intervals with uniform or exponential distributions.

We have performed Qualnet simulations to compare the performance of BRPS with ESC, a scheduling scheme that represents the state-of-the art. Results show that BRPS provides low latency and can closely match the performance of ESC. Furthermore, BRPS has a lower scheduling error ratio due to its robustness property, translating to better packet delivery ratio. Aside from having a lower storage and communication overhead, BRPS also has a lower computational complexity compared with ESC.

6.2 Reliable and Low Latency Path Selection

A low sleep latency schedule does not necessarily lead to low end-to-end latency paths because wireless link quality plays a significant role in the performance of packet forwarding. Our second contribution is therefore the formulation of a routing metric that simultaneously considers sleep latency and packet loss. Called *expected transmission delay* (ETD), the metric is novel in the following manner: (*i*) none of the state-of-the-art routing metrics have included sleep latency in their respective formulations; (*ii*) while energy availability does not explicitly appear in the formulation, it is encapsulated in the sleep latency; and (*iii*) unlike metrics that employ highly dynamic physical quantities such as energy harvesting rate, energy consumption, and fraction of energy used, ETD's use of sleep latency as a proxy to energy availability not only simplifies computation but also enhances its stability.

Two important properties that make ETD useful in many routing protocols are left-monotonicity and left-isotonicity. With these properties, ETD can be employed in any distributed algorithm that searches for the least cost path such as the distributed Bellman-Ford, and that its use is guaranteed to yield consistent, loop-free and optimal paths. To take advantage of these properties, we have designed a distributed algorithm that searches for the path with the least ETD. Through Qualnet simulations, we have shown that when compared with hop count and the state-of-the-art routing metric ETX, ETD provides the best performance in terms of packet delivery ratio and end-to-end delay.

6.3 Dynamic Duty Cycle Allocation

Numerous receive-centric wakeup scheduling schemes have been proposed in the literature. In such schemes, a wakeup slot or interval is meant for receiving packets only. Thus, a node that needs to perform packet relaying cannot use the entire duty cycle to generate wakeup schedules and it needs to explicitly reserve a portion of its duty cycle for packet transmissions. We refer to this apportioning of the duty cycle between packet reception and transmission as the *duty cycle allocation problem*.

Our third contribution therefore addresses the problem of dynamic duty cycle allocation in the context of receive-centric synchronous wakeup scheduling schemes. Synchronous schemes are more prone to contention because of synchronized access to the medium. As such, we have investigated how packet contention affects the packet arrival probability at a node that employs synchronous wakeup scheduling. Using discrete-time queueing and renewal theory, we have derived the service time of packets in the context of duty cycled nodes and in the presence of contention. The service time is essentially equivalent to the sleep latency which is a major challenge in EPWSNs. Our result essentially generalizes our key finding about the expected sleep latency: in the presence of contention, the variance of the intervals between the wakeup slots affects the service time, *i.e.*, a higher (lower) variance yields higher (lower) expected service time.

We have expressed the expected service time in terms of wakeup scheduling parameters, in particular the number of slots to be apportioned in an epoch. A closer inspection of the quantity reveals that when a node allocates a certain number of slots for packet reception, not only does it affect the service time of the packets that it is relaying, but also the service time of the packets that its predecessor nodes are relaying. This therefore led to the formulation of a non-linear optimization problem that aims to minimize the two-hop expected service time. By exploiting the integer constraints of the problem, we have proposed LSLOTALLOC, a distributed low-complexity algorithm that linearly searches for the optimal solution of the problem.

Through extensive simulations in Qualnet, we have validated the analytical models (packet arrival probability and expected service time). Moreover, we have demonstrated the significant performance advantage of LSLOTALLOC over the best-performing static allocation scheme in terms of delay. Notably, LSLOTALLOC's delay is around half the delay of the static allocation scheme in most scenarios.

6.4 Sustainable Bulk Transfer

Our fourth and final contribution is on bulk data transfer in EPWSNs. We have handled bulk data transfer separately because its objective is slightly different from data delivery schemes for monitoring applications such as indoor and outdoor environmental or habitat monitoring. These applications generate low data rates, typically in the order of a few bytes to at most several tens of bytes at every sensing interval. For such applications, the data delivery objectives are to maximize the packet delivery ratio and minimize the end-to-end delay[6]. In bulk transfer, however, the objective is to deliver the entire bulk data which is typically in the order of tens to hundreds of kilobytes, at the highest possible throughput. Indeed, state-of-the-art bulk transfer schemes have been designed with this objective. But as we have found out, these schemes do not work well in EPWSNs because throughput maximization without regard for energy availability leads to uncontrolled and rapid draining of the energy reserves. This will lead to the temporary unavailability of nodes along the transfer path. Ultimately, this will result in transfer disruptions which render the transfer of arbitrarily-sized sensor data difficult, if not infeasible.

We have therefore designed PUMP-AND-NAP, a forwarding technique that uses *controlled packet trains* to simultaneously maximize throughput and enforce compliance to (dynamic) duty cycle limitations. Two forwarding techniques have been introduced, namely, a basic scheme that commences packet train transmis-

[6]We note that optimizing both objectives simultaneously is hard, so a trade-off is usually employed, depending on the application requirements.

sions without regard for the wakeup schedule of the receiver, and a wakeup-synchronized scheme that commences packet train transmissions when the receiver is known to be awake. While the latter looks attractive, it is more complex than the former because it needs knowledge of the wakeup schedule of the receiver which may entail some cost. However, in the case of long transfer durations, the latter may be more advantageous as it reduces energy wastage due to pre-transmission overhead.

PUMP-AND-NAP uses an *adaptive controller* to determine a node's *optimal capacity*, defined as the maximum number of packets the node can receive and transmit in a train within its duty cycle constraints. We have modeled the duty cycle usage as a linear system with zero-mean disturbance and accordingly designed a certainty equivalent adaptive feedback controller. In essence, the controller uses prior input-output observations (capacity allocations and their corresponding duty cycle usage) to continuously tune its performance and adapt to wireless link quality variations.

Finally, we have implemented PUMP-AND-NAP in TinyOS [60] and performed experiments in the Indriya testbed [34], a 139-node indoor testbed, to evaluate its performance. Our results show that PUMP-AND-NAP can adaptively track duty cycles and provide high bulk transfer throughput at the same time. More importantly, we have demonstrated in energy harvesting experiments that PUMP-AND-NAP can truly enable sustainable bulk transfer compared to state-of-the-art techniques that greedily maximize throughput at the expense of downtime due to energy depletion.

6.5 Open Research Issues

This thesis focused on the challenges posed by the energy-neutral design principle on data delivery schemes. Notwithstanding its contributions, we identify several important research problems that need to be tackled to further enhance the performance and robustness of data delivery schemes for EPWSNs.

Wakeup Scheduling in the Presence of Mobility Many of the existing wakeup scheduling schemes assume that the sensor nodes and sink are stationary. As the use of EPWSNs become more prevalent, we expect EPWSNs to be employed in scenarios where some or all of the nodes are mobile. There is therefore a need to design wakeup scheduling schemes that can support node mobility. As far as the existing schemes are concerned, asynchronous and non-collaborative synchronous schemes are good candidates for these scenarios because their lack of coordination requirement makes them robust to network topology changes. Note that in the presence of node mobility, schemes that require coordination may not converge to an optimal schedule or may generate excessive overhead.

Opportunistic Routing in Dynamic Duty Cycling Networks In this thesis, we used conventional hop-by-hop packet forwarding, wherein the sending node explicitly specifies the successor node of every packet that it forwards. This approach, also known as *best-path routing*, may entail packet retransmissions or path re-discoveries in the presence of highly variable links due to external interference, multi-path fading and weather [135]. Opportunistic routing have therefore been proposed to mitigate the above-mentioned issues. It takes advantage of the broadcast nature of wireless transmissions, whereby any node that overhears a packet and has a better metric (with respect to a sink) than the sender can forward the packet. Obviously, duty cycling will reduce, if not eliminate, the overhearing opportunities. Thus, for opportunistic routing to work in such scenarios, there might be a need to coordinate the wakeup schedules such that overhearing is still possible.

Non-myopic Dynamic Duty Cycle Allocation The dynamic duty cycle allocation problem was formulated as a myopic optimization problem in Chapter 4, *i.e.*, the apportioning only considers the current epoch. The myopic formulation is warranted because regardless of the slot allocation at epoch k, the number of available slots for allocation in the next epoch $k+1$ is considered to be independent, *i.e.*, not in any way affected by the prior allocation. However, in real energy-harvesting nodes, the number of slots for allocation at consecutive epochs are likely to be correlated. Thus, it is possible to design non-myopic schemes based on Markov decision process or dynamic stochastic optimization. Such approaches are more complicated so there is a need to strike a balance between complexity and performance.

Incorporating End-to-end Reliability and Rate Control in PUMP-AND-NAP We have proposed PUMP-AND-NAP as a hop-by-hop forwarding technique that can be used by nodes with (dynamic) duty cycle constraints. PUMP-AND-NAP requires other functions to be complete, including end-to-end reliability and rate control. For the former, we can use a NACK-based scheme similar to the one employed in Flush. As for the latter, there is a need for further study because rate control and duty cycle compliance seem to be interdependent. Note that if the sum of all duty cycle constraints along a path is less than 100%, rate control may not be necessary. This is because even if every node maximizes its respective duty cycle usage, the probability that two or more nodes will transmit at the same time is still low. However, when the duty cycle sum is more than 100%, simultaneous transmissions become more probable. In these cases, rate control needs to be enforced to avoid intra-flow interference. Indeed, determining when to apply or not to apply rate control seems to be a crucial issue in EPWSNs.

Network-Level Energy Neutrality The thesis was motivated by the principle of energy-neutrality, applied on a per-node basis. This approach might be too rigid, as it essentially does not allow any node in the network to (temporarily) deplete its energy store. In highly-dense deployments, this restriction may not be necessary, as the network can still satisfy its mission even if a fraction of the nodes are not alive. This calls for energy-neutral designs at the network level. There are two important questions to answer: (*i*) what is the performance difference between protocols that use node-level energy-neutrality and protocols that employ network-level energy-neutrality; and (*ii*) the added complexity and cost of protocols that use network-level energy-neutrality.

Appendix A

Publications

Alvin C. Valera, Wee-Seng Soh, and Hwee-Pink Tan. Energy-Neutral Scheduling and Forwarding in Environmentally-Powered Wireless Sensor Networks. *Ad Hoc Networks (Elsevier)*, vol. 11, no. 3, May 2013.

Alvin C. Valera, Wee-Seng Soh, and Hwee-Pink Tan. Survey on Wakeup Scheduling for Environmentally-Powered Wireless Sensor Networks. *Computer Communications (Elsevier)*, May 2014.

Alvin C. Valera, Wee-Seng Soh, and Hwee-Pink Tan. On Duty Cycle Allocation in Environmentally-Powered Wireless Sensor Networks. *Ad Hoc Networks (Elsevier)*, submitted, October 2014.

Alvin C. Valera, Wee-Seng Soh, and Hwee-Pink Tan. Pump-and-Nap: Enabling Sustainable Bulk Transfer in Environmentally-Powered Wireless Sensor Networks. *2015 Twelfth Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, submitted, December 2014.

Bibliography

[1] BQ25504 Battery Management Evaluation Board. [Online]. Available: http://www.ti.com/tool/bq25504evm-674 [Accessed: Sep 6, 2014].

[2] CC2420: Single-Chip 2.4 Ghz IEEE 802.15.4 Compliant and ZigBee Ready RF Transceiver. [Online]. Available: http://focus.ti.com/docs/prod/folders/print/cc2420.html [Accessed: May 31, 2012].

[3] National Renewable Energy Laboratory. [Online]. Available: http://http://www.nrel.gov [Accessed: May 31, 2012].

[4] The Qualnet Simulator. [Online]. Available: http://www.scalable-networks.com/products/developer.php [Accessed: May 31, 2012].

[5] Waspmote Datasheet. [Online]. Available: http://www.libelium.com/.../waspmote/waspmote-datasheet_eng.pdf [Accessed: May 31, 2012].

[6] XBee Datasheet. [Online]. Available: http://http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf [Accessed: May 31, 2012].

[7] IEEE 802.14.5-2006 Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), 2006.

[8] Zigbee Specification, versio r17, January 2008. ZigBee Alliance.

[9] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proc. ACM SIGCOMM*, August 2004.

[10] K. Akkaya and M. Younis. An energy-aware qos routing protocol for wireless sensor networks. In *Proc. IEEE ICDCS Workshops*, pages 710–715, May 2003.

[11] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, May 2005.

[12] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[13] M. Al-Jemeli, F.A. Hussin, and B.B. Samir. A link-quality and energy aware routing metric for mobile wireless sensor networks. In *Proc. International Conference on Intelligent and Advanced Systems*, volume 1, pages 211–216, June 2012.

[14] G. Anastasi, M. Conti, and M. Di Francesco. Extending the lifetime of wireless sensor networks through adaptive sleep. *IEEE Trans. Industrial Informatics*, 5(3):351–365, 2009.

[15] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 9(3):537–568, May 2009.

[16] V.S. Barbu and N. Limnios. *Semi-Markov Chains and Hidden Semi-Markov Models toward Applications*. Springer Science+Business Media, 2008.

[17] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[18] Dimitri P. Bertsekas and Gallager. *Data Networks (2nd Edition)*. Prentice Hall, 2 edition, January 1992.

[19] Najet Boughanmi and Ye-Qiong Song. A new routing metric for satisfying both energy and delay constraints in wireless sensor networks. *Journal of*

Signal Processing Systems for Signal, Image, and Video Technology, 51(2):137–143, 2008.

[20] Herwig Bruneel and Byung Kim. *Discrete-time models for communication systems including ATM*. Kluwer Academic Publishers, 1993.

[21] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: A short preamble mac protocol for duty-cycled wireless sensor networks. In *Proc. ACM SenSys*, 2006.

[22] Heejung Byun and Junglok Yu. Adaptive duty cycle control with queue management in wireless sensor networks. *IEEE Trans. Mobile Computing*, 12(6):1214–1224, 2013.

[23] Q. Cao, T. He, L. Fang, T. Abdelzaher, J. Stankovic, and S. Son. Efficiency centric communication model for wireless sensor networks. In *Proc. IEEE INFOCOM*, 2006.

[24] Kameswari Chebrolu, Bhaskaran Raman, Nilesh Mishra, Phani Kumar Valiveti, and Raj Kumar. Brimon: A sensor network system for railway bridge monitoring. In *Proc. ACM MobiSys*, pages 2–14, 2008.

[25] B. Chen, K-K Muniswamy-Reddy, and M. Welsh. Ad-hoc multicast routing on resource limited sensor nodes. In *Proc. ACM REALMAN*, 2006.

[26] Chien-Ying Chen and Pai Chou. Duracap: a supercapacitor-based, power-bootstrapping, maximum power point tracking energy-harvesting system. In *Proc. ACM ISLPED*, 2010.

[27] K.W. Chin, J. Judge, A. Williams, and R. Kermode. Implementation experience with manet routing protocols. *SIGCOMM Comput. Commun. Rev.*, 32(5):49–59, 2002.

[28] D. De Couto, D. Aguayo, B. Chambers, and R. Morris. Performance of multihop wireless networks: shortest path is not enough. *ACM SIGCOMM Comput. Commun. Rev.*, 33(1):83–88, 2003.

[29] Hui Dai and Richard Han. Tsync: a lightweight bidirectional time synchronization service for wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8:125–139, January 2004.

[30] L.L. Dai and P. Basu. Energy and delivery capacity of wireless sensor networks with random duty-cycles. In *Proc. IEEE ICC*, 2006.

[31] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. ACM MobiCom*, 2003.

[32] D. De Couto, D. Aguayo, B. Chambers, and R. Morris. Effects of loss rate on ad hoc wireless routing. Technical Report MIT-LCS-TR-836, MIT Laboratory for Computer Science, March 2002.

[33] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerical Mathematics*, October 1959.

[34] M. Doddavenkatappa, M.C. Chan, and A.L. Ananda. Indriya: A low-cost, 3d wireless sensor network testbed. In *Proc. TRIDENTCOM*, 2011.

[35] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. *ACM SIGCOMM Comput. Commun. Rev.*, 34(4):133–144, 2004.

[36] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proc. MobiCom 2004*, pages 114–128, New York, NY, USA, 2004. ACM.

[37] S. Du, A.K. Saha, and D.B. Johnson. Rmac: A routing-enhanced duty-cycle mac protocol for wireless sensor networks. In *Proc. IEEE INFOCOM*, pages 1478–1486, 2007.

[38] A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne. The announcement layer: Beacon coordination for the sensornet stack. In *Proc. EWSN*, 2011.

[39] Simon Duquennoy, Fredrik Österlind, and Adam Dunkels. Lossy links, low power, high throughput. In *Proc. ACM SenSys*, 2011.

[40] G. Ekbatanifard, P. Sommer, B. Kusy, V. Iyer, and K. Langendoen. Fastforward: High-throughput dual-radio streaming. In *Proc. IEEE MASS*, 2013.

[41] Jeremy Elson and Kay Römer. Wireless sensor networks: A new regime for time synchronization. *SIGCOMM Comput. Commun. Rev.*, 33(1):149–154, January 2003.

[42] Saurabh Ganeriwal, Deepak Ganesan, Hohyun Shim, Vlasios Tsiatsis, and Mani B. Srivastava. Estimating clock uncertainty for efficient duty-cycling in sensor networks. In *Proc. ACM SenSys*, pages 130–141, 2005.

[43] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *Proc. ACM SenSys*, 2003.

[44] Giacomo Ghidini and Sajal Das. An energy-efficient markov chain-based randomized duty cycling scheme for wireless sensor networks. In *Proc. IEEE ICDCS*, pages 67–76, 2011.

[45] Gary Gillette, William Pierpoint, and Stephen Treado. A general illuminance model for daylight availability. *Journal of Illuminating Engineering Society (IES)*, pages 330–337, July 1984.

[46] O. Gnawali and P. Levis. The etx objective function for rpl, February 2010. Internet-Draft.

[47] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proc. ACM SenSys*, pages 1–14, 2009.

[48] Carles Gomez, Antoni Boix, and Josep Paradells. Impact of lqi-based routing metrics on the performance of a one-to-one routing protocol for ieee 802.15.4 multihop networks. *EURASIP Journal on Wireless Communications and Networking*, 2010.

[49] Graham Goodwin and Kwai Sang Sin. *Adaptive Filtering Prediction and Control*. Dover Publications, 1984.

[50] Yu Gu and Tian He. Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links. In *Proc. ACM SenSys*, 2007.

[51] Yu Gu, Ting Zhu, and Tian He. Esc: Energy synchronized communication in sustainable sensor networks. In *Proc. IEEE ICNP*, 2009.

[52] S. Guha, Chi-Kin Chau, and P. Basu. Green wave: Latency and capacity-efficient sleep scheduling for wireless networks. In *Proc. IEEE INFOCOM*, 2010.

[53] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan. Understanding and mitigating the impact of rf interference on 802.11 networks. *SIGCOMM Comput. Commun. Rev.*, 37(4):385–396, 2007.

[54] V.C. Gungor, C. Sastry, Zhen Song, and R. Integlia. Resource-aware and link quality based routing metric for wireless sensor and actor networks. In *Proc. IEEE ICC*, 2007.

[55] Rick W. Ha, Pin-Han Ho, X. Sherman Shen, and Junshan Zhang. Sleep scheduling for wireless sensor networks via network flow model. *Computer Communications*, 29(13–14):2469–2481, 2006.

[56] Adnan Harb. Energy harvesting: State-of-the-art. *Renewable Energy*, 36:2641–2654, October 2011.

[57] C. Hedrick. Routing information protocol, June 1988. IETF RFC 1058.

[58] C. Hedrick. An introduction to igrp, August 1991. http://www.cisco.com/warp/public/103/5.html.

[59] Junyoung Heo, Jiman Hong, and Yookun Cho. Earq: Energy aware routing for real-time and reliable communication in wireless industrial sensor networks. *IEEE Trans. Industrial Informatics*, 5(1):3–11, February 2009.

[60] J. Hill, R. Szewczyk, A. Woo, P. Levis, K. Whitehouse, J. Polastre, D. Gay, S. Madden, M. Welsh, D. Culler, and E. Brewer. Tinyos: An operating system for sensor, 2003.

[61] Jason Hsu, Sadaf Zahedi, Aman Kansal, Mani Srivastava, and Vijay Raghunathan. Adaptive duty cycling for energy harvesting systems. In *Proc. ACM ISLPED*, 2006.

[62] Pei Huang, Chen Wang, Li Xiao, and Hongyang Chen. Rc-mac: A receiver-centric medium access control protocol for wireless sensor networks. In *Proc. IWQoS Workshop*, 2010.

[63] J J. Korhonen and Y. Wang. Effect of packet size on loss rate and delay in wireless links. In *Proc. IEEE WCNC*, volume 3, pages 1608–1613, March 2005.

[64] M.K. Jakobsen, J. Madsen, and M.R. Hansen. Dehar: A distributed energy harvesting aware routing algorithm for ad-hoc multi-hop wireless sensor networks. In *Proc. IEEE WoWMoM*, pages 1–9, 2010.

[65] Xiaofan Jiang, Joseph Polastre, and David Culler. Perpetual environmentally powered sensor networks. In *Proc. IEEE IPSN*, 2005.

[66] D. Johnson, D. Maltz, and J. Broch. Dsr the dynamic source routing protocol for multihop wireless ad hoc networks. In C. Perkins, editor, *Ad hoc Networking*, pages 139–172. Addison-Wesley, 2001.

[67] Rahim Kacimi, Riadh Dhaou, and Andre-Luc Beylot. Load balancing techniques for lifetime maximizing in wireless sensor networks. *Ad Hoc Networks*, 11(8):2172–2186, 2013.

[68] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. Power management in energy harvesting sensor networks. *ACM Trans. Emb. Comput. Sys.*, 6:1–38, September 2007.

[69] G. Karbaschi and A. Fladenmuller. A link-quality and congestion-aware cross layer metric for multi-hop wireless routing. In *Proc. IEEE MASS*, pages 7–13, 2005.

[70] Alan Karp. Bit reversal on uniprocessors. *SIAM Review*, 38(1):1–26, 1996.

[71] V. Kawadia and P.R. Kumar. A cautionary perspective on cross-layer design. *IEEE Wirel. Commun.*, 12(1):3–11, February 2005.

[72] Abtin Keshavarzian, Huang Lee, and Lakshmi Venkatraman. Wakeup scheduling in wireless sensor networks. In *Proc. ACM MobiHoc*, 2006.

[73] K.H. Kim and K.G. Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proc. ACM MobiCom*, pages 38–49, 2006.

[74] Sehwan Kim, Keun-Sik No, and Pai Chou. Design and performance analysis of supercapacitor charging circuits for wireless sensor nodes. *IEEE J. Emerging and Selected Topics in Circuits and Systems*, 1(2), September 2011.

[75] Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica. Flush: A reliable bulk transport protocol for multihop wireless networks. In *Proc. ACM SenSys*, 2007.

[76] C. Koksal and H. Balakrishnan. Quality-aware routing metrics for time-varying wireless mesh networks. *IEEE JSAC*, 24(11):1984–1994, November 2006.

[77] R. Kortebi, Y. Gourhant, and N. Agoulmine. On the use of sinr for interference-aware routing in wireless multi-hop networks. In *Proc. ACM MSWiM*, pages 395–399, 2007.

[78] D. Kotz, C. Newport, R. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proc. ACM MSWiM*, October 2004.

[79] P.R. Kumar and Pravin Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice-Hall, 1986.

[80] Emanuele Lattanzi, Edoardo Regini, Andrea Acquaviva, and Alessandro Bogliolo. Energetic sustainability of routing algorithms for energy-harvesting wireless sensor networks. *Computer Communications*, 30(14-15):2976–2986, 2007.

[81] Yuan Li, Wei Ye, and J. Heidemann. Energy and latency control in low duty cycle mac protocols. In *Proc. IEEE WCNC*, volume 2, pages 676–682, 2005.

[82] Peng Lin, Chunming Qiao, and Xin Wang. Medium access control with a dynamic duty cycle for sensor networks. In *Proc. IEEE WCNC*, volume 3, pages 1534–1539, 2004.

[83] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay efficient sleep scheduling in wireless sensor networks. In *Proc. IEEE INFOCOM*, 2005.

[84] G. Malkin. Rip version 2, November 1998. IETF RFC 2453.

[85] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *Proc. ACM SenSys*, 2004.

[86] Michael J. McGlynn and Steven A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proc. ACM MobiHoc*, 2001.

[87] D. Moss and P. Levis. Box-mac: Exploiting physical and link layer boundaries in low-power networking. Technical report, 2008. Technical Report SING-08-00.

[88] Razvan Musaloiu-E., Chieh-Jan Mike Liang, and Andreas Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proc. ACM IPSN*, 2008.

[89] Michael J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[90] Kien Nguyen, Vu-Hoang Nguyen, Duy-Dinh Le, Yusheng Ji, DucAnh Duong, and Shigeki Yamada. A receiver-initiated mac protocol for energy

harvesting sensor networks. In Young-Sik Jeong, Young-Ho Park, Ching-Hsien (Robert) Hsu, and James J. (Jong Hyuk) Park, editors, *Ubiquitous Information Technologies and Applications*, volume 280 of *Lecture Notes in Electrical Engineering*, pages 603–610. Springer Berlin Heidelberg, 2014.

[91] E.E. Osuna and G.F. Newell. Control strategies for an idealized public transportation system. *Transportation Science*, 6(1):52–72, February 1972.

[92] J. Padhye, S. Agarwal, V. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of link interference in static multi-hop wireless networks. In *Proc. IMC 2005*, pages 28–28, Berkeley, CA, USA, 2005. USENIX Association.

[93] Santashil PalChaudhuri, Amit Kumar Saha, and David B. Johnson. Adaptive clock synchronization in sensor networks. In *Proc. ACM IPSN*, pages 340–348, 2004.

[94] J.A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. *IEEE Pervasive Computing*, 4(1):18 – 27, Jan-March 2005.

[95] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. IETF RFC 3561.

[96] C. Perkins and P. Bhagwat. Dsdv routing over a multihop wireless network of mobile computers. In C. Perkins, editor, *Ad hoc Networking*, pages 53–74. Addison-Wesley, 2001.

[97] L. Peterson and B. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, San Francisco, California, USA, 2000.

[98] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proc. ACM SenSys*, New York, NY, USA, 2004. ACM.

[99] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *Proc. IEEE IPSN*, 2005.

[100] Bhaskaran Raman, Kameswari Chebrolu, Sagar Bijwe, and Vijay Gabale. PIP: A connection-oriented, multi-hop, multi-channel tdma-based mac for high throughput bulk transfer. In *Proc. ACM SenSys*, 2010.

[101] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based models of delivery and interference in static wireless networks. *ACM SIGCOMM Comput. Commun. Rev.*, 36(4):51–62, 2006.

[102] Sheldon Ross. *Stochastic Processes*. Wiley and Sons, 1982.

[103] Sheldon Ross. *Introduction to Probability Models*. Elsevier, 9 edition, 2007.

[104] S. Roy, D. Koutsonikolas, S. Das, and Y. Hu. High-throughput multicast routing metrics in wireless mesh networks. In *Proc. IEEE ICDCS*, page 48, 2006.

[105] T. Salonidis, M. Garetto, A. Saha, and E. Knightly. Identifying high throughput paths in 802.11 mesh networks: a model-based approach. In *Proc. IEEE ICNP 2007*, pages 21–30, October 2007.

[106] L.F. Sang, A. Arora, and H.W. Zhang. On exploiting asymmetric wireless links via one-way estimation. In *Proc. ACM MobiHoc*, pages 11–21, 2007.

[107] Winston K.G. Seah, Zhi Ang Eu, and Hwee-Pink Tan. Wireless sensor networks powered by ambient energy harvesting (wsn-heap) – survey and challenges. In *Proc. Wireless VITAE*, May 2009.

[108] R.C. Shah and J.M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Proc. IEEE WCNC*, pages 350–355, 2002.

[109] Farhan Simjee and Pai H. Chou. Efficient charging of supercapacitors for extended lifetime of wireless sensor nodes. *IEEE Trans. Power Electronics*, 23(3):1526–1536, May 2008.

[110] Sujesha Sudevalayam and Purushottam Kulkarni. Energy harvesting sensor nodes: Survey and implications. *IEEE Communications Surveys Tutorials*, (99):1–19, 2010.

[111] Wenzhong Sun, Yingxiong Song, and Min Chen. A load-balanced and energy-aware routing metric for wireless multimedia sensor networks. In *Proc. IET International Conference on Wireless, Mobile and Multimedia Networks*, pages 21–24, September 2010.

[112] Yanjun Sun, Omer Gurewitz, and David B. Johnson. RI-MAC: A receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In *Proc. ACM SenSys*, 2008.

[113] Shaojie Tang, Jie Wu, Guihai Chen, Cheng Wang, Xuefeng Liu, Tao Li, and Xiang-Yang Li. On minimum delay duty-cycling protocol in sustainable sensor network. In *Proc. IEEE ICNP*, 2012.

[114] P. Thubert. Objective function zero for the routing protocol for low-power and lossy networks (rpl), March 2012. Internet-Draft.

[115] Gilman Tolle. Multihoplqi, October 2004. http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI.

[116] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong. A macroscope in the redwoods. In *Proc. ACM SenSys*, 2005.

[117] Alvin C. Valera, Wee-Seng Soh, and Hwee-Pink Tan. Energy-neutral scheduling and forwarding in environmentally-powered wireless sensor networks. *Ad Hoc Networks*, 2013.

[118] Alvin C. Valera, Wee-Seng Soh, and Hwee-Pink Tan. A survey on wakeup scheduling in environmentally-powered wireless sensor networks. *Computer Communications*, 2014.

[119] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proc. ACM SenSys*, 2003.

[120] Christopher Vigorito, Deepak Ganesan, and Andrew Barto. Adaptive control of duty cycling in energy-harvesting wireless sensor networks. In *Proc. IEEE SECON*, 2007.

[121] Binhua Wang, Yongrui Chen, and Weidong Yi. An energy-aware routing for maximum lifetime in wireless sensor networks. In Ruchuan Wang and Fu Xiao, editors, *Advances in Wireless Sensor Networks*, volume 334 of *Communications in Computer and Information Science*, pages 413–423. Springer Berlin Heidelberg, 2013.

[122] Chonggang Wang, K. Sohraby, Bo Li, M. Daneshmand, and Yueming Hu. A survey of transport protocols for wireless sensor networks. *Network, IEEE*, 20(3), May 2006.

[123] Qinghua Wang and Tingting Zhang. Source traffic modeling in wireless sensor networks for target tracking. In *Proc. ACM PE-WASUN*, pages 96–100, 2008.

[124] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2), March 2006.

[125] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander. Rpl: Ipv6 routing protocol for low-power and lossy networks, March 2012. IETF RFC 6550.

[126] Wolfram Research Inc. *Mathematica Edition: Version 8.0*. Wolfram Research, Inc., Champaign, Illinois, 2010.

[127] Yan Wu, Sonia Fahmy, and N.B. Shroff. Optimal sleep/wake scheduling for time-synchronized sensor networks with qos guarantees. *IEEE/ACM Trans. Networking*, 17(5):1508–1521, 2009.

[128] Y. Yang and J. Wang. Design Guidelines for Routing Metrics in Multihop Wireless Networks. In *Proc. IEEE INFOCOM*, pages 1615–1623, April 2008.

[129] Wei Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proc. IEEE INFOCOM*, volume 3, 2002.

[130] Wei Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. In *Proc. IEEE INFOCOM*, 2002.

[131] Kai Zeng, Kui Ren, Wenjing Lou, and Patrick J. Moran. Energy aware efficient geographic routing in lossy wireless sensor networks with environmental energy supply. *Wireless Networks*, 15:39–51, January 2009.

[132] J. Zhang, L. Cheng, and I. Marsic. Models for non-intrusive estimation of wireless link bandwidth. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Pers. Wirel. Commun.*, pages 334–348. Springer Berlin, 2003.

[133] Rong Zheng, Jennifer C. Hou, and Lui Sha. Asynchronous wakeup for ad hoc networks. In *Proc. ACM MobiHoc*, 2003.

[134] Tao Zheng, S. Radhakrishnan, and V. Sarangan. PMAC: An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proc. IEEE Parallel and Distributed Processing Symposium*, 2005.

[135] Zifei Zhong and S. Nelakuditi. On the efficacy of opportunistic routing. In *Proc. IEEE SECON*, pages 441–450, June 2007.

[136] Yuanyuan Zhou and M. Medidi. Sleep-based topology control for wakeup scheduling in wireless sensor networks. In *Proc. IEEE SECON*, pages 304–313, 2007.

[137] Ting Zhu, Ziguo Zhong, Yu Gu, Tian He, and Zhi-Li Zhang. Leakage-aware energy synchronization for wireless sensor networks. In *Proc. ACM MobiSys*, 2009.