2014

Building a more sustainable sensor network via protocol innovation

Yang Peng
Iowa State University

Building a more sustainable sensor network via protocol innovation

by

Yang Peng

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Iowa State University

Ames, Iowa

2014

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Traditionally, network protocols are designed based on the assumptions that network is powered by small batteries with scarce energy supply. However, emerging energy replenishment technologies such as ambient energy harvesting, wireless energy transferring, etc., provide alternatives to address the energy constraint problem but also introduce new challenges (e.g., energy heterogeneity). Been the core to achieve network sustainability, novel network protocols shall be designed to better exploit energy availabilities and tackle new challenges or issues exposed by emerging energy replenishment technologies. In this dissertation, we study how to build a more sustainable sensor network via network protocol innovation. Specifically, the study is conducted in four directions. First of all, we study how to improve energy utilization efficiency on individual sensor nodes as a foundation to improve the network sustainability. Secondly, we study how to prolong the network lifetime as a whole through dynamically and collaboratively tuning MAC layer operational parameters between neighboring nodes. Thirdly, we study the cross-layer design technique and propose a holistic routing and MAC protocol to further prolong the network lifetime. Fourthly, with given sensing coverage constraints, we jointly optimize the routing and sensing behaviors to further improve the network sustainability.

CHAPTER 1. INTRODUCTION

1.1 Sustainable Sensor Networks

When a sensor network is deployed for long-term monitoring, it is desired and critical to improve the network sustainability such that the network can operate as long as possible to collect valuable sensory data. *Network lifetime*, which is defined as the first time when certain application specified requirements cannot be satisfied, is a key measurement of network sustainability.

As sensor nodes are usually powered by small batteries and can be depleted after several days of operations, such an energy depletion problem has become one of the most important reasons that could render the network nonfunctional and limit the network sustainability. A lot of research has been conducted to address this problem. For example, as shown in Figure 1.1, it has been proposed that a sensor node's energy may be replenished after deployment through various methods, such as harvesting solar energy and wirelessly charging energy from mobile chargers to sensor nodes.

Though these energy replenishment technologies can provide extra energy supplies to the network, the amount of supplied energy may be restrained by weather condition, geographical accessibility, etc. In addition, these technologies alone cannot effectively solve but may even worsen the problem of heterogeneous energy distribution, which can lead to inefficient energy utilization in the network and may not really extend the network lifetime. Therefore, besides exploring and enhancing energy replenishment technologies, protocol innovation is also imperative to improve the network sustainability.

To cope with the heterogeneity in energy supply and consumption among sensor nodes, a major hurdle in sustaining network lifetime, new protocols should be developed for sensor nodes to collaborate in dynamically adjusting operational parameters in order to prolong network lifetime. Particularly, sensor nodes whose energy can be replenished efficiently (e.g., because they are deployed in open space exposed under sunshine or in locales closer to wireless chargers' moving tracks) may take more

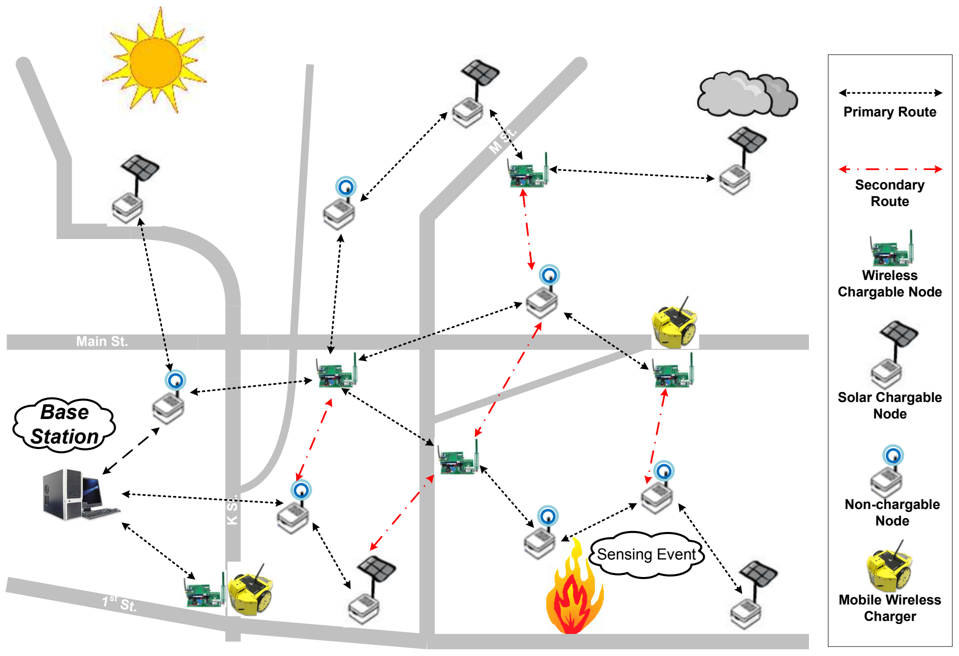


Figure 1.1 Overview.

workloads, to help saving the energy consumption of other nodes who are disadvantageous in energy replenishment. To facilitate such workload adjustment, it is desired to have a MAC protocol that can dynamically adjust nodal behaviors based on nodes' differential energy status and nodal lifetimes. It is also desired to have sophisticated cross-layer protocols that can jointly coordinate sensing, routing, and MAC behaviors, based on nodes' differential energy status, to utilize nodal energy more effectively and efficiently and thus maximize the network lifetime.

1.2 Challenges and Opportunities

As discussed above, protocol innovation is of critical importance to build a more sustainable sensor network. When developing new protocols, the following challenges should be taken into account:

- *Network resource heterogeneity*. In many sensor networks, heterogeneity is inevitable. As shown in Figure 1.1, the heterogeneity can be caused by diverse energy supplies (wireless charging, solar energy harvesting, or traditional batteries), device capabilities, geographical locations, etc. New protocols shall be designed with awareness of the existence of heterogeneity in network.

- *Energy efficiency and energy fairness*. As prolonging the network lifetime is essential to achieve the network sustainability, energy utilization efficiency and fairness must be considered in protocol design. On one hand, an individual node should not run out of energy much earlier than other nodes, which may render network disconnected and jeopardize data fidelity; on the other hand, when aiming at energy fairness, energy efficiency should not be neglected given that the total usable energy in the network may be limited. How to deal with the tradeoffs between energy efficiency and fairness is critical for prolonging the network lifetime.

- *Application Quality of Service requirements*. When building a sustainable sensor network, different Quality of Service (QoS) requirements, such as end-to-end data delivery delay and sensing coverage may be demanded by users. These requirements impose different constraints in protocol design. In addition, for different applications, the network traffics may vary over time; hence, a generic protocol design needs to be adaptive to these changes efficiently.

- *Coordination and collaboration among protocols in different network layers*. Though protocol innovation for a specific network layer can improve system performance, there are limitations that cannot be conquered by a single-layer design. Cross-layer design could be a promising solution to overcome the limitations and further prolong the network lifetime. However, a cross-layer design may increase design complexity and system overhead. Moreover, a cross-layer protocol without calibration may even result in a performance lower than that could be achieved by each single-layer protocol alone.

1.3 Research Themes

In this dissertation, we study how to design protocols for different network layers to build a sustainable sensor network. We aim at designing novel and practical schemes that can be implemented in commonly used sensor nodes and can offer better performance in terms of network lifetime, end-to-end packet delay, network power consumption, etc. The following research topics are included:

- *Delay-bounded MAC protocol design to improve nodal energy utilization efficiency*. We first study the problem of how to improve the nodal energy utilization efficiency, such that lifetime of individual nodes can be improved and network lifetime can be prolonged accordingly. We propose the CyMAC protocol, which plans the rendezvous schedules between neighboring nodes carefully, and adjusts the sensor nodes' radio duty cycles dynamically to the varying traffic condition, and therefore reduces idle listening time of a sensor node and prolongs nodal lifetime significantly. CyMAC can also guarantee the desired relative delay bound for data delivery services.

- *Collaborative MAC protocol design to prolong network lifetime*. By viewing the network as a whole, we study the problem of how to prolong the network lifetime rather than an individual node and propose a collaborative MAC protocol, called LB-MAC. Different from MAC protocols that focus on reducing energy consumption and extending lifetime of individual sensor nodes, the collaborative MAC protocol aims at prolonging the network lifetime through balancing the nodal lifetime between neighboring sensors. This way, the minimum nodal lifetime in the network can be prolonged; as a result, the network lifetime can be prolonged gradually.

- *Joint routing and MAC protocol design to prolong network lifetime*. Besides tuning the MAC layer parameters only, we also propose a novel holistic design, called I^2C, which is composed of two network lifetime improvement modules: the Intra-Route Coordination and the Inter-Route Coordination modules. As a cross-layer protocol, I^2C can leverage the advantages of both the lifetime prolonging schemes in MAC and routing layers with a sophisticated design that emphasizes the awareness and collaboration between the two schemes under different end-to-end delay constraints.

- *Joint routing and sensing protocol design to prolong network lifetime*. We conduct further study on how to prolong the network lifetime given the requirement of sensing coverage, existence of node redundancy and partitioned monitoring areas in the network. In this work, we propose J-RoS - a distributed and low-cost scheme, which can schedule routing and sensing activities between neighboring nodes collaboratively. Instead of performing lifetime balancing in network, J-RoS schedules routing and sensing activities to consume energy of sensing non-critical nodes on purpose, even at the cost of losing these nodes. This way, the energy of sensing critical nodes can be saved, and the network sustainability can be improved by running in desired sensing coverage for a longer period of time.

The rest of the dissertation is organized as follows. We first present the CyMAC design in Chapter 2, the lifetime-balancing MAC protocol, LB-MAC, is then presented in Chapter 3. Details of the two cross-layer protocols, I^2C and J-RoS, are presented in Chapters 4 and 5, respectively. Finally, Chapter 6 concludes this dissertation with a summary of the main contributions and discusses the future research topics.

CHAPTER 2. DELAY-BOUNDED MAC WITH MINIMAL IDLE LISTENING FOR SENSOR NETWORKS

2.1 Introduction

Wireless sensor networks should be energy efficient in order to operate for a long time. When a sensor node has its radio turned on, it operates at a similar power consumption level regardless whether it is transmitting, receiving or idle listening [1]. Hence, numerous MAC protocols have been proposed to reduce the idle listening time of a sensor node, which has been found to contribute substantially to a sensor node's total energy consumption [2, 3].

2.1.1 Related Work

Most of the existing MAC protocols are either synchronous or asynchronous. Representative synchronous protocols such as S-MAC [1], T-MAC [4], RMAC [5] and DW-MAC [6] require neighbor nodes to be time-synchronized. They align the active and sleep intervals of neighbor nodes, which wake up only during the common active time intervals to exchange packets. Since the active intervals usually are short, substantial energy can be saved. However, strictly synchronizing the clocks of neighbor nodes imposes high overhead, and the aligned and short active intervals can cause congestion when multiple flows cross the same node.

Asynchronous protocols such as B-MAC [7], WiseMAC [8], X-MAC [9] and RI-MAC [10] decouple the duty cycle schedules of different nodes and thus eliminate the overhead for synchronization. B-MAC, WiseMAC and X-MAC are sender-initiated preamble-based protocols which employ the low power listening technique. Particularly, B-MAC requires a sender to transmit a preamble longer than the sleep interval of its receiver to signal the receiver. WiseMAC shortens the preamble length by requiring a sender to learn the duty cycle schedule of its receiver and start a preamble shortly before the receiver wakes up. X-MAC improves B-MAC by replacing the long preamble with a sequence of short, strobed preambles. Nevertheless, these protocols are optimized mainly for light traffic conditions. In the scenarios of bursty or high traffic load, which can be caused by convergecast [11], correlated events [12] and data aggregation [13], the preambles may congest the channel and block data transmissions. Hybrid protocol such as SCP [14] combines a synchronous protocol with asynchronous low power listening but suffers the same clock synchronization overhead as synchronous protocols.

To work under a wider range of traffic conditions, RI-MAC [10] adopts a receiver-initiated beacon-based strategy. Each node periodically wakes up and sends out a short beacon to explicitly notify its neighbors that it is ready to receive data. When a node has data to transmit, it wakes up and waits for a beacon from its receiver. Once such a beacon is received, it starts sending the data. Compared to the sender-initiated preamble-based protocols, RI-MAC uses shorter and less frequent beacons which consume less bandwidth, and its receiver-initiated nature allows more efficient collision resolution. However, RI-MAC has the following limitations. A sender needs to remain awake after a data packet arrives, till the receiver wakes up to receive the packet, potentially wastes a lot of time on idle listening. Also, a receiver sends out beacons at a fixed time interval on average and does not adapt to changes of traffic pattern.

2.1.2 Motivations and Contributions

To further reduce idle listening and improve the energy efficiency of sensor networks, we propose a new MAC protocol called *CyMAC*. Similar to RI-MAC, CyMAC is a receiver-initiated beacon-based protocol. The difference is that CyMAC reduces the idle listening time significantly through establishing rendezvous times between sender and receiver. In addition, rendezvous schedules are adaptive to the changes of traffic condition so that sender and receiver can operate with minimal duty cycles while a certain desired delay bound for data delivery services can still be guaranteed. More importantly, CyMAC achieves the above goals without requiring clock synchrony between sensors. It functions properly as long as the desired delay bound is less stringent than the degree of clock asynchrony.

CyMAC targets to provide *relative delay bound* [15] guarantee for sensor data delivery services, which is defined as the ratio of the data delivery delay to the average data arrival interval. For example, if data packets arrive every 100 seconds and the delivery delay of a data packet is 10 seconds, the relative delay is 10%. This is in contrast to the absolute delay bound that usually is provided with a fixed beacon interval (e.g., in RI-MAC) so that the delivery delay of a data packet can be guaranteed less than the beacon interval. For sensor network applications, a relative delay bound could be more meaningful and important than an absolute delay bound. For example, the same delivery delay of one second may have different effects on two different sensor network applications: one with a data arrival interval of one second and the other with a data arrival interval of 100 seconds. The former situation could be far worse than the latter, since by the time when a data packet is delivered, it has become obsolete because a newer data packet has arrived. Relative delay bound may help sensor nodes conserve energy too. For example, if a 10% relative delay bound is acceptable, when the data arrival interval increases from 10 to 100 seconds, the number of beacons sent by the receiver and hence the energy consumed by the receiver can be reduced by an order of magnitude.

The contributions of this work are summarized as follows:

- We propose a new receiver-initiated MAC protocol, called CyMAC, for sensor networks. CyMAC attempts to minimize idle listening and hence duty cycles of sensor nodes via establishing rendezvous times between neighbors. It is adaptive to the changes in traffic condition, and can guarantee desired relative delay bound for sensor data delivery services under various traffic conditions. Different from existing synchronous MAC protocols, CyMAC does not require clock synchrony between sensor nodes.

- We have implemented CyMAC in TinyOS and evaluated it with small-scale experiments. We have also implemented it in the ns-2 simulator for evaluation in large-scale networks.

- Extensive experiments and simulations have demonstrated that CyMAC can always guarantee the desired delay bound, and has a lower duty cycle than RI-MAC in most cases except when the required delay bound is very tight. In this case, CyMAC can still provide the delay bound guarantee at the cost of having a slightly higher duty cycle than RI-MAC.

2.2 CyMAC Design

In the following, we give an overview of the proposed CyMAC protocol for sensor networks.

1) CyMAC is a receiver-initiated MAC protocol but with minimal idle listening time at the sender side. Similar to RI-MAC, the data exchange between CyMAC sender and receiver is initiated by the receiver with a beacon. However, different from RI-MAC which requires the sender to remain awake (upon a data packet arrival) and listen idly till the beacon arrives, CyMAC only requires the sender to wake up at pre-scheduled rendezvous times to communicate with the receiver, thus reducing the idle listening time significantly.

2) CyMAC provides delay-bounded data delivery services. A unique feature of CyMAC is its ability to adjust the duty cycles and rendezvous schedules of sensor nodes to provide the desired relative delay bound to data delivery services.

3) CyMAC adjusts the sensor nodes' duty cycles dynamically to the varying traffic condition. Another unique feature of CyMAC is dynamic duty cycling. When the traffic is light, CyMAC nodes sleep more and send fewer beacons to conserve more energy, while when the traffic is heavy, CyMAC nodes wake up more often to interact with each other so as to provide the desired delay bound.

4) CyMAC does not require clock synchrony between sensor nodes: Different from existing synchronous MAC protocols, CyMAC does not require clock synchrony between sensor nodes nor synchronization protocols executed on sensor nodes. CyMAC functions properly as long as the desired delay bound is less stringent than the degree of clock asynchrony between neighbor nodes. Section 2.2.3 discusses in detail how CyMAC handles clock asynchrony issues.

Next, we describe the design of CyMAC in detail. Table 2.1 lists the variables maintained at each CyMAC node.

2.2.1 Receiver's Behavior

The operation flowchart of a CyMAC receiver is shown in Figure 2.1. In CyMAC, the receiver wakes up at the scheduled beacon time $T_{\text{BEACON},i}$ to interact with sender i by sending a beacon and then waiting for a short dwell time[1]. As shown in the flowchart, if a new packet is received successfully,

[1]This short dwell time is platform dependent. In our implementation of CyMAC on MicaZ motes, it is set to 17.5ms.

Table 2.1 Variables maintained at each CyMAC node

Variable		Meaning
For each sender i	$T_{\text{LAST},i}$	arrival time of the last received data packet from sender i
	$T_{\text{BEACON},i}$	latest time to serve sender i (by sending a beacon) in order to satisfy the delay bound
$T_{\text{BEACON}} = \min_i (T_{\text{BEACON},i})$		scheduled next beacon time
For each receiver j	$T_{\text{LISTEN},j}$	scheduled next listen time for receiver j
	DONE_j	the set of packets that (i) have failed all transmission attempts (ii) arrive after the last successfully-delivered packet; the last successfully-delivered packet is also included in set DONE_j
	WAIT_j	the set of packets waiting to be transmitted
For each packet x in WAIT_j or DONE_j	$T_{\text{arrv}}(x)$	arrival time of packet x
	$D(x)$	delay between $T_{\text{arrv}}(x)$ and when x is transmitted
	$\theta(x)$	updated estimate of mean of packet arrival interval
	$\delta(x)$	updated estimate of variance of packet arrival interval

the receiver records the packet information, updates its estimates of the data traffic, and schedules the next beacon time using the I_{allow} information piggybacked in the packet by the sender (which tells the receiver when the next beacon should be sent); otherwise, it schedules the next beacon time for sender i based on (i) $T_{\text{BEACON},i}$; (ii) $T_{\text{LAST},i}$ – the arrival time of the last received data packet from sender i; and (iii) μ – the desired relative delay bound over a single hop. Note that, since a receiver may serve multiple senders, it performs the above routine for all senders and informs every one of its very next scheduled beacon time: $T_{\text{BEACON}} = \min_i(T_{\text{BEACON},i})$. This way, a sender may be able to forward a packet that arrives earlier than expected to the receiver opportunistically at an earlier beacon time that was scheduled for other senders, thus reducing the delivery delay further.

2.2.1.1 Online Traffic Estimation

Upon arrival of a data packet x, the receiver updates its estimate of the mean of data arrival interval as:

$$\theta(x) = \alpha(x)\theta(x') + (1 - \alpha(x))\theta_{\text{new}}(x), \tag{2.1}$$

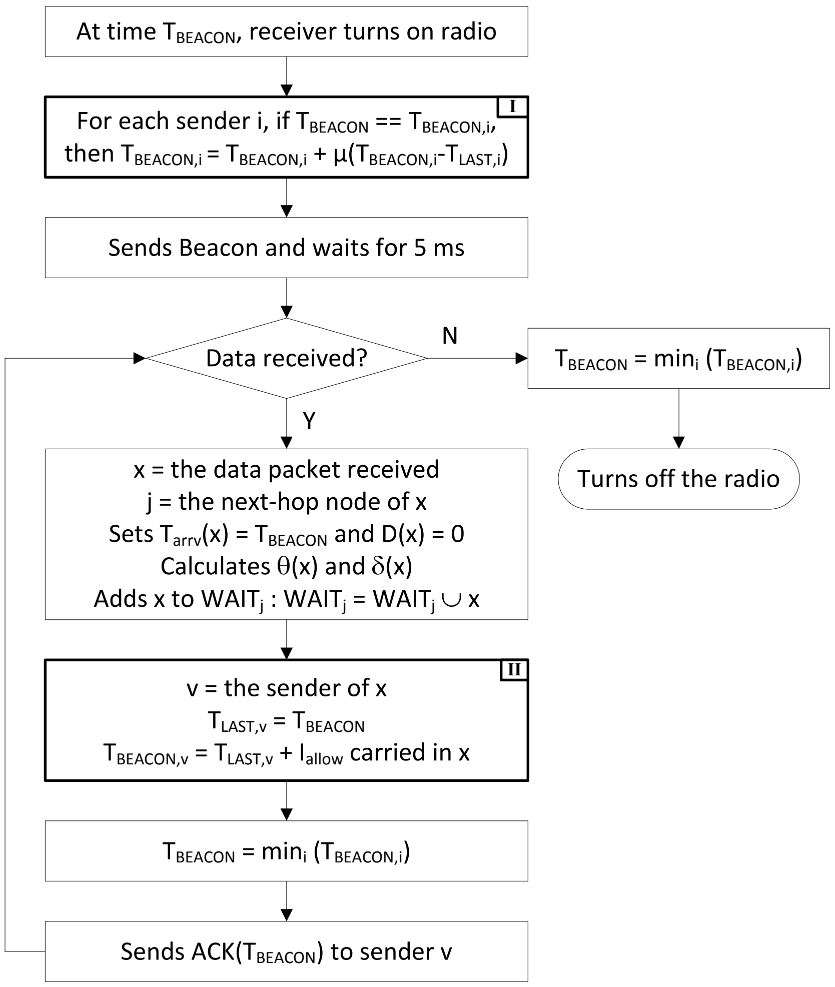

At time T_BEACON, receiver turns on radio
I
For each sender i, if T_BEACON == T_BEACON,i, then T_BEACON,i = T_BEACON,i + μ(T_BEACON,i-T_LAST,i)
Sends Beacon and waits for 5 ms
Data received?
N
T_BEACON = min_i (T_BEACON,i)
Turns off the radio
Y
x = the data packet received
j = the next-hop node of x
Sets T_arrv(x) = T_BEACON and D(x) = 0
Calculates θ(x) and δ(x)
Adds x to WAIT_j : WAIT_j = WAIT_j ∪ x
II
v = the sender of x
T_LAST,v = T_BEACON
T_BEACON,v = T_LAST,v + I_allow carried in x
T_BEACON = min_i (T_BEACON,i)
Sends ACK(T_BEACON) to sender v

Figure 2.1 Operation flowchart of a CyMAC receiver.

where x' is the last successfully-received data packet prior to x and has the same next-hop node as packet x. $\theta_{\text{new}}(x) = T_{\text{arrv}}(x) - T_{\text{arrv}}(x')$ is the new sample mean and $\alpha(x)$ is the smoothing factor: $\alpha(x) = 2^{\frac{-\theta_{\text{new}}(x)}{10 \cdot \theta(x')}} \cdot 0.9$. The reason for choosing such a smoothing factor for estimating the mean of data arrival interval is that, a larger $\theta_{\text{new}}(x)$ value implies that the previously estimated mean ($\theta(x')$) has become more obsolete, and hence a larger weight should be given to the new sample. For example, if $\theta_{\text{new}}(x) = 10 \cdot \theta(x')$, meaning that packet x arrives much later after the previous packet x' (10 times the mean arrival interval), then a larger weight ($0.55 = 1 - \alpha(x)$) is given to the new sample.

The receiver also updates its estimate of the variance of data arrival interval, but with a fixed smoothing factor:

$$\delta(x) = \beta\delta(x') + (1-\beta)\delta_{\text{new}}(x), \tag{2.2}$$

where $\delta_{\text{new}}(x) = |\theta_{\text{new}}(x) - \theta(x')|$ is the new sample variance and $\beta = 0.9$. This is because a late arriving packet (i.e., a larger $\theta_{\text{new}}(x)$ value) may skew the calculation of $\delta_{\text{new}}(x)$; hence we opt to not give a larger weight to the new sample in the estimation to avoid undesired complication.

2.2.1.2 Relative Delay Bound Guarantee

One of the key design goals of CyMAC is to provide delay-bounded data delivery services, meaning that if all packets (beacon, data and ACK) are transmitted successfully, the delivery delay of a data packet x over a single hop is

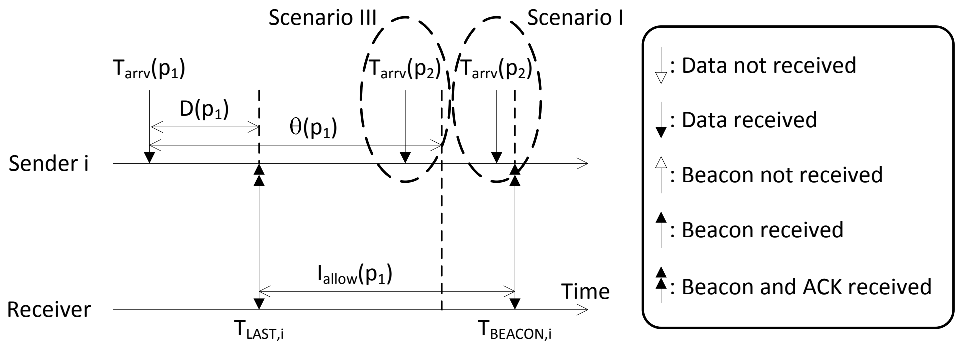
$$D(x) \leqslant \mu \max\{\theta(x), T_{\text{arrv}}(x) - T_{\text{arrv}}(x')\}, \tag{2.3}$$

where x' is the last successfully-received data packet prior to x and has the same next-hop node as packet x. μ is the desired relative delay bound over a single hop. In practice, a sensor network application often specifies its desired delay bound in terms of end-to-end delay (μ_{e2e}). Let ξ denote the hop-count diameter of the sensor network, we conservatively translate the application-specified end-to-end delay bound μ_{e2e} to the hop-by-hop relative delay bound μ as follows:

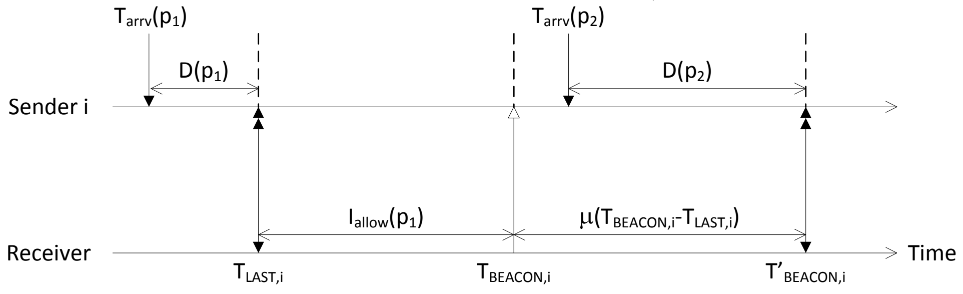
$$\mu = (1 + \mu_{\text{e2e}})^{1/\xi} - 1. \tag{2.4}$$

To illustrate how CyMAC guarantees Equation (2.3), we present a few example scenarios in Figure 2.2. Here, we assume that a CyMAC receiver only has one sender (sender i) to receive data packets from. As shown in the figure, after packet p_1 is delivered successfully from sender i to the receiver at time $T_{\text{LAST},i}$, the receiver schedules its next beacon time to

$$T_{\text{BEACON},i} = T_{\text{LAST},i} + I_{\text{allow}}(p_1), \tag{2.5}$$



(a) Scenario I: packet p_2 arrives between $T_{\text{arrv}}(p_1) + \theta(p_1)$ and $T_{\text{BEACON},i}$. Scenario III: packet p_2 arrives before $T_{\text{arrv}}(p_1) + \theta(p_1)$.



(b) Scenario II: packet p_2 arrives after $T_{\text{BEACON},i}$.

Figure 2.2 Example scenarios to illustrate how the desired delay bound is satisfied with CyMAC.

where $I_{\text{allow}}(p_1)$ is the information piggybacked in packet p_1 and set by sender i. For a relative delay bound of μ, let us set $I_{\text{allow}}(p_1)$ to

$$I_{\text{allow}}(p_1) = (1 + \mu)\theta(p_1) - D(p_1). \tag{2.6}$$

Then, depending on the arrival time of the next data packet p_2, there are three possible scenarios:

- Scenario I: $T_{\text{arrv}}(p_1) + \theta(p_1) \leqslant T_{\text{arrv}}(p_2) \leqslant T_{\text{BEACON},i}$. In this case, packet p_2 arrives before the scheduled beacon time $T_{\text{BEACON},i}$ but after $T_{\text{arrv}}(p_1) + \theta(p_1)$, as shown in Figure 2.2(a). The delivery delay for packet p_2 is then:

$$\begin{aligned} D(p_2) &= T_{\text{BEACON},i} - T_{\text{arrv}}(p_2) \\ &= T_{\text{LAST},i} + (1+\mu)\theta(p_1) - D(p_1) - T_{\text{arrv}}(p_2) \\ &= T_{\text{arrv}}(p_1) + (1+\mu)\theta(p_1) - T_{\text{arrv}}(p_2) \\ &\leqslant T_{\text{arrv}}(p_1) + (1+\mu)\theta(p_1) - (T_{\text{arrv}}(p_1) + \theta(p_1)) \\ &= \mu\theta(p_1) \leqslant \mu \max\{\theta(p_2), T_{\text{arrv}}(p_2) - T_{\text{arrv}}(p_1)\}. \end{aligned} \tag{2.7}$$

Therefore, the desired delay bound is guaranteed.

- Scenario II: $T_{\text{arrv}}(p_2) > T_{\text{BEACON},i}$. In this case, packet p_2 arrives after the scheduled beacon time, as shown in Figure 2.2(b). As a result, the receiver schedules the next beacon time to

$$T'_{\text{BEACON},i} = T_{\text{BEACON},i} + \mu(T_{\text{BEACON},i} - T_{\text{LAST},i}). \tag{2.8}$$

If packet p_2 arrives before $T'_{\text{BEACON},i}$, its delivery delay is bounded under the limit:

$$\begin{aligned} D(p_2) &= T'_{\text{BEACON},i} - T_{\text{arrv}}(p_2) < T'_{\text{BEACON},i} - T_{\text{BEACON},i} \\ &= T_{\text{BEACON},i} + \mu(T_{\text{BEACON},i} - T_{\text{LAST},i}) - T_{\text{BEACON},i} \\ &= \mu(T_{\text{BEACON},i} - T_{\text{LAST},i}) < \mu(T_{\text{arrv}}(p_2) - T_{\text{arrv}}(p_1)) \\ &= \mu \max\{\theta(p_2), T_{\text{arrv}}(p_2) - T_{\text{arrv}}(p_1)\}. \end{aligned} \tag{2.9}$$

If packet p_2 arrives after $T'_{\text{BEACON},i}$, a similar analysis can be applied to show that the desired delay bound is always satisfied.

- Scenario III: $T_{\text{arrv}}(p_2) < T_{\text{arrv}}(p_1) + \theta(p_1)$. In this case, since packet p_2 arrives before $T_{\text{arrv}}(p_1) +$

$\theta(p_1)$, as shown in Figure 2.2(a), its delivery delay would be

$$\begin{aligned} D(p_2) &= T_{\text{BEACON},i} - T_{\text{arrv}}(p_2) \\ &= T_{\text{arrv}}(p_1) + (1+\mu)\theta(p_1) - T_{\text{arrv}}(p_2) \\ &> T_{\text{arrv}}(p_1) + (1+\mu)\theta(p_1) - (T_{\text{arrv}}(p_1) + \theta(p_1)) \\ &= \mu\theta(p_1) > \mu \max\{\theta(p_2), T_{\text{arrv}}(p_2) - T_{\text{arrv}}(p_1)\}. \end{aligned} \tag{2.10}$$

This means that, for any packet that arrives within the estimated mean packet arrival interval, the delivery delay cannot be bounded under the desired limit. As a result, we may not be able to bound the average delivery delay (over all packets) under certain packet arrival distributions. One way to deal with this potential issue is to employ a more conservative approach by replacing θ with $(\theta - m\delta)$ in Equation (2.6):

$$I_{\text{allow}}(p_1) = (1+\mu)(\theta(p_1) - m\delta(p_1)) - D(p_1), \tag{2.11}$$

where $m \geqslant 1$ and larger m values may be used for more stringent delay requirements. This way, fewer packets would experience higher delay, and thus the average delivery delay may be bounded under the limit.

2.2.2 Sender's Behavior

The operation flowchart of a CyMAC sender is shown in Figure 2.3. In CyMAC, the sender acts in a leading role. It schedules the rendezvous times with each receiver by calculating I_{allow} and piggybacks such information in the packet transmissions to the receiver. For receiver j, the sender maintains two sets of packets (as listed in Table 2.1): (i) DONE$_j$ – the set of packets that have failed all transmission attempts and arrive after the last successfully-delivered packet, which itself is also included in the set; and (ii) WAIT$_j$ – the set of packets waiting to be transmitted. It also maintains $T_{\text{LISTEN},j}$ – the next listen time for beacons from receiver j. At $T_{\text{LISTEN},j}$, the sender forwards all the packets in WAIT$_j$ to receiver j with I_{allow} information piggybacked in each packet.

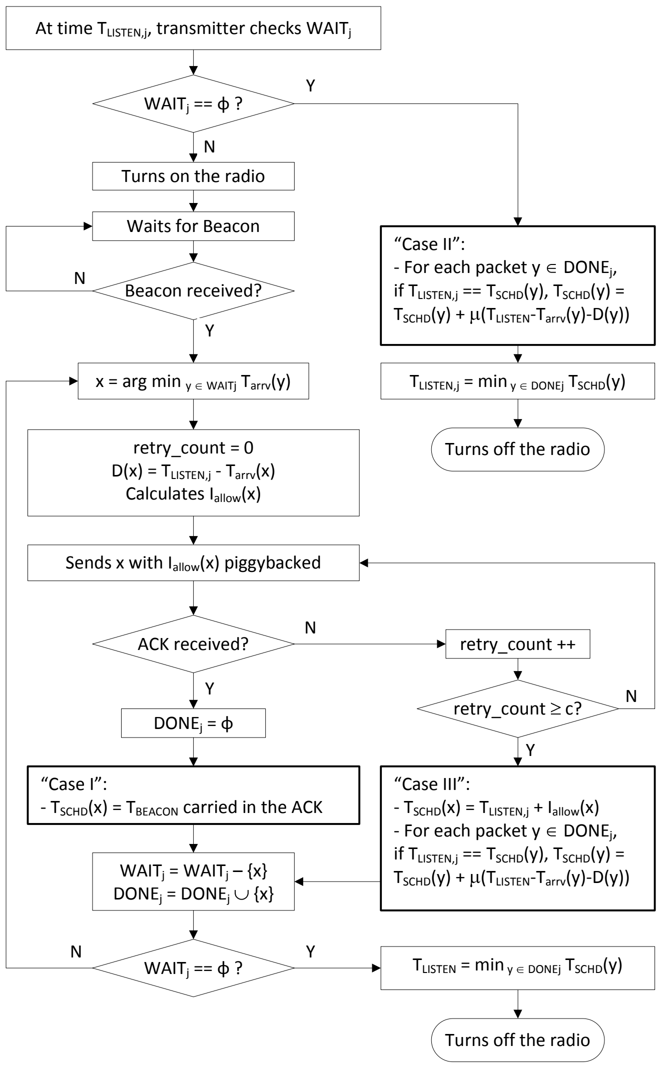
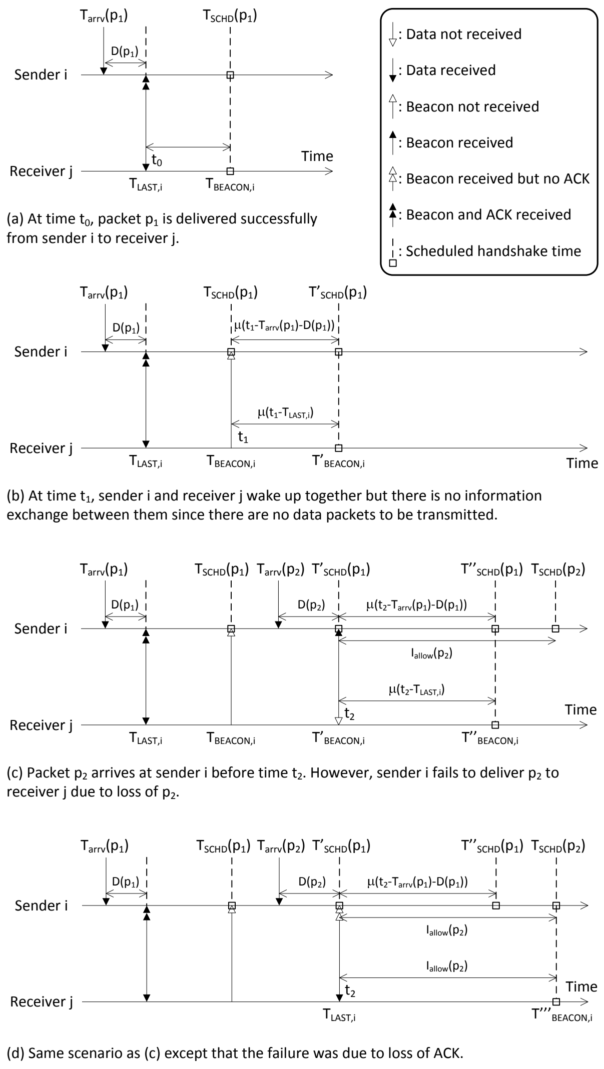

At time T_LISTEN,j, transmitter checks WAIT_j
WAIT_j == ϕ ?
Y
N
Turns on the radio
Waits for Beacon
Beacon received?
N
Y
"Case II":
- For each packet y ∈ DONE_j, if T_LISTEN,j == T_SCHD(y), T_SCHD(y) = T_SCHD(y) + μ(T_LISTEN-T_arrv(y)-D(y))
T_LISTEN,j = min y ∈ DONEj T_SCHD(y)
Turns off the radio
x = arg min y ∈ WAITj T_arrv(y)
retry_count = 0
D(x) = T_LISTEN,j - T_arrv(x)
Calculates I_allow(x)
Sends x with I_allow(x) piggybacked
ACK received?
N
retry_count ++
retry_count ≥ c?
N
Y
DONE_j = ϕ
Y
"Case I":
- T_SCHD(x) = T_BEACON carried in the ACK
"Case III":
- T_SCHD(x) = T_LISTEN,j + I_allow(x)
- For each packet y ∈ DONE_j, if T_LISTEN,j == T_SCHD(y), T_SCHD(y) = T_SCHD(y) + μ(T_LISTEN-T_arrv(y)-D(y))
WAIT_j = WAIT_j – {x}
DONE_j = DONE_j ∪ {x}
N
WAIT_j == ϕ ?
Y
T_LISTEN = min y ∈ DONEj T_SCHD(y)
Turns off the radio

Figure 2.3 Operation flowchart of a CyMAC sender with respect to receiver j.

2.2.2.1 Rendezvous between Sender and Receiver

As shown in Figure 2.3, there are three different cases when the sender schedules its next listen time differently. CyMAC is able to guarantee rendezvous between sender and receiver in all three cases, which will be explained with the help of example scenarios given in Figure 2.4.

- *Case I: after a successful data packet delivery.* In this case, the sender sets the next listen time to T_{BEACON} that is carried in the ACK. This case is illustrated in Figure 2.4(a) where we assume that there is only one sender (sender i) and one receiver (receiver j). We can see that, after packet



Tarrv(p1)
TSCHD(p1)
D(p1)
Sender i
Receiver j
t0
Time
TLAST,i
TBEACON,i
(a) At time t0, packet p1 is delivered successfully from sender i to receiver j.
: Data not received
: Data received
: Beacon not received
: Beacon received
: Beacon received but no ACK
: Beacon and ACK received
: Scheduled handshake time
T'SCHD(p1)
μ(t1-Tarrv(p1)-D(p1))
μ(t1-TLAST,i)
t1
T'BEACON,i
(b) At time t1, sender i and receiver j wake up together but there is no information exchange between them since there are no data packets to be transmitted.
Tarrv(p2)
T''SCHD(p1)
TSCHD(p2)
D(p2)
μ(t2-Tarrv(p1)-D(p1))
Iallow(p2)
μ(t2-TLAST,i)
t2
T''BEACON,i
(c) Packet p2 arrives at sender i before time t2. However, sender i fails to deliver p2 to receiver j due to loss of p2.
T'''BEACON,i
(d) Same scenario as (c) except that the failure was due to loss of ACK.

Figure 2.4 Example scenarios to illustrate how CyMAC guarantees rendezvous between sender and receiver.

p_1 is delivered successfully at time t_0, both sender and receiver schedule to wake up together at $T_{\text{SCHD}}(p_1) = T_{\text{BEACON},i} \triangleq t_1$.

- *Case II: when there are no data packets to be transmitted.* Despite that there is no information exchange between sender and receiver in this case, CyMAC can still guarantee that sender and receiver wake up together at future time instances. Figure 2.4(b) shows an example scenario when there are no data packets to be transmitted at time t_1. Sender i schedules the next listen time to (according to Case II in Figure 2.3 Flowchart)

$$T'_{\text{SCHD}}(p_1) = t_1 + \mu(t_1 - T_{\text{arrv}}(p_1) - D(p_1)), \tag{2.12}$$

and receiver j schedules the next beacon time to (according to Box I in Figure 2.1 Flowchart)

$$T'_{\text{BEACON},i} = t_1 + \mu(t_1 - T_{\text{LAST},i}). \tag{2.13}$$

These two time instances are indeed the same, meaning that sender and receiver will wake up together at $T'_{\text{SCHD}}(p_1) = T'_{\text{BEACON},i} \triangleq t_2$.

- *Case III: after a failed data packet delivery.* In the design, the sender assumes the data packet delivery is failed after retrying for c times (c is a configurable system parameter as the retry count threshold) without receiving an ACK from the receiver. This is the most complicated case as the sender is unsure whether the failure was due to loss of data packet or loss of ACK, when the receiver behaves differently. These two scenarios are illustrated in Figs. 2.4(c) and (d), where at time t_2 the receiver schedules the next beacon time to (loss of data packet; Box I in Figure 2.1 Flowchart)

$$T''_{\text{BEACON},i} = t_2 + \mu(t_2 - T_{\text{LAST},i}), \tag{2.14}$$

and (loss of ACK; Box II in Figure 2.1 Flowchart)

$$T'''_{\text{BEACON},i} = t_2 + I_{\text{allow}}(p_2), \tag{2.15}$$

respectively. In order to guarantee rendezvous between sender and receiver, CyMAC requires the sender to wake up at both time instances. To do so, the sender updates T_{SCHD} for all packets in set DONE and listen at all the updated T_{SCHD} time instances. In the example scenarios shown in Figs. 2.4(c) and (d), since sender i now has DONE$_j$ = $\{p_1, p_2\}$, it will listen at both

$$T''_{\text{SCHD}}(p_1) = t_2 + \mu(t_2 - T_{\text{arrv}}(p_1) - D(p_1)) \tag{2.16}$$

and

$$T'_{\text{SCHD}}(p_2) = t_2 + I_{\text{allow}}(p_2), \tag{2.17}$$

which match $T''_{\text{BEACON},i}$ and $T'''_{\text{BEACON},i}$, respectively.

2.2.2.2 Minimal Idle Listening Time

A major difference between CyMAC and RI-MAC is how a sender behaves upon a data packet arrival. In RI-MAC, a sender turns on the radio immediately after a data packet arrives, idly listening till it receives a beacon from the receiver. In comparison, a CyMAC sender only turns on the radio at scheduled listen times for possible interactions with receivers. So if a data packet arrives before the next scheduled listen time, the packet will be inserted into set WAIT but the radio won't be turned on till the scheduled listen time. This way, the idle listening time is reduced drastically.

2.2.2.3 Dynamic Duty Cycling

Another unique feature of CyMAC is that sensor nodes adjust their duty cycles dynamically to the varying traffic condition. When the traffic is light, sensor nodes sleep more and send less beacons to conserve more energy, while when the traffic is heavy, sensor nodes wake up more often to interact with each other so as to provide the desired delay bound.

Figure 2.5 shows the behavior of CyMAC nodes when the network turns idle (i.e., no more new data packets) after a packet is delivered successfully at T_{LAST}. As shown in the figure, the k-th ($k \geqslant 1$) rendezvous time after T_{LAST} will be scheduled at $T_{\text{LAST}} + (1 + \mu)^{i-1}\phi$, according to Case II in the sender flowchart and Box I in the receiver flowchart. For example, if T_{LAST} = 0 second, ϕ = 1 second and μ = 50%, the future rendezvous times will be at approximately $\{$1, 1.5, 2.3, 3.4, 5.1, 7.6, 11.4, 17.1, $\cdots\}$ seconds. This procedure goes on till new data packets arrive which will direct CyMAC nodes to reset their duty cycles based on their updated estimates of the data traffic. This shows that CyMAC nodes are able to adjust quickly to the varying traffic condition and operate in ultra-low duty cycles when the traffic is light.

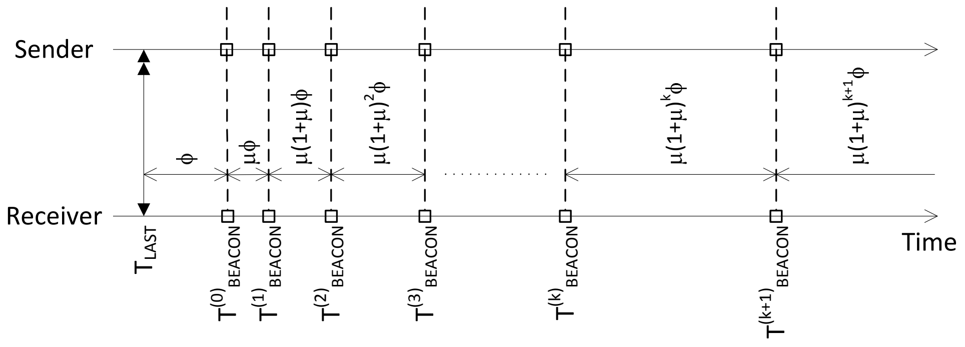


Figure 2.5 Dynamic duty cycling with CyMAC.

2.2.3 Effects of Time Asynchrony

In a practical sensor network, sender and receiver nodes are inevitably asynchronous. Typically, clocks of sensor nodes differ for two reasons: *clock skew* that is simply the initial difference between clocks, and *clock drift* that refers to different clocks counting time at slightly different rates, which results in varying clock skews over time. In general, clock asynchrony between sender and receiver nodes can be described with the following equation:

$$t_r = a \times t_s + b, \tag{2.18}$$

where t_s is the time instance at the sender, t_r is the corresponding time instance at the receiver, and a and b represent the clock drift and the clock skew, respectively. In this section, we analyze the effects of clock asynchrony on CyMAC performance, and discuss how we enhance CyMAC to deal with these issues.

2.2.3.1 $a < 1$

In this case, the sender clock counts time at a faster rate than the receiver clock, as shown in Figure 2.6(a). After the sender delivers a packet p_1 successfully to the receiver, both sender and receiver know that $T_{\text{sent}}(p_1)$ on the sender clock corresponds to T_{LAST} on the receiver clock, and schedule the next rendezvous time to $I_{\text{allow}}(p_1)$ time later. Since the sender clock counts faster, when the sender wakes up at $T_{\text{SCHD}}(p_1)$ to listen for beacon from the receiver, the receiver won't wake up till $I_{\text{allow}}(p_1)(\frac{1}{a} - 1)$ time later. As a result, an extra delay is introduced to the delivery of packet p_2:

$$D(p_2) = I_{\text{allow}}(p_1)\frac{1}{a} + D(p_1) - (T_{\text{arrv}}(p_2) - T_{\text{arrv}}(p_1)). \tag{2.19}$$

When the system stabilizes, $D(p_1) = D(p_2) \triangleq D$ and $T_{\text{arrv}}(p_2) - T_{\text{arrv}}(p_1) = \theta(p_1) \triangleq \theta$. Plugging in Equation (2.6), we have

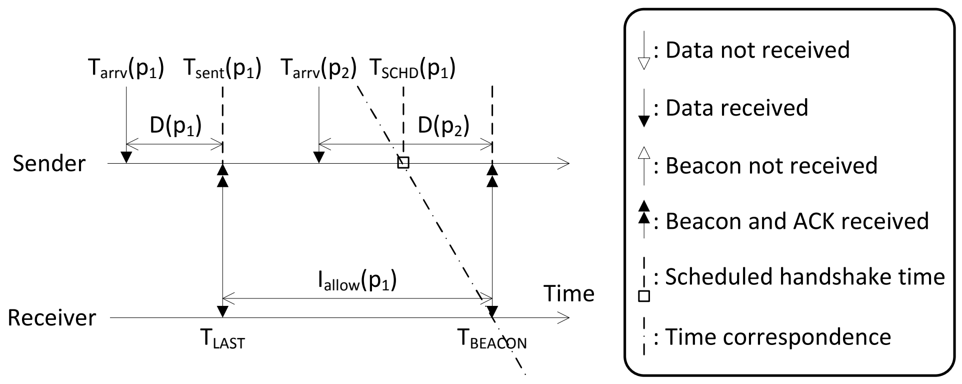
$$\begin{aligned} & D = ((1+\mu)\theta - D)\frac{1}{a} + D - \theta \\ \Longrightarrow \quad & D = (\mu + 1 - a)\theta. \end{aligned} \tag{2.20}$$

This means that an extra delay of $(1-a)\theta$ has been added to the packet delivery delay.

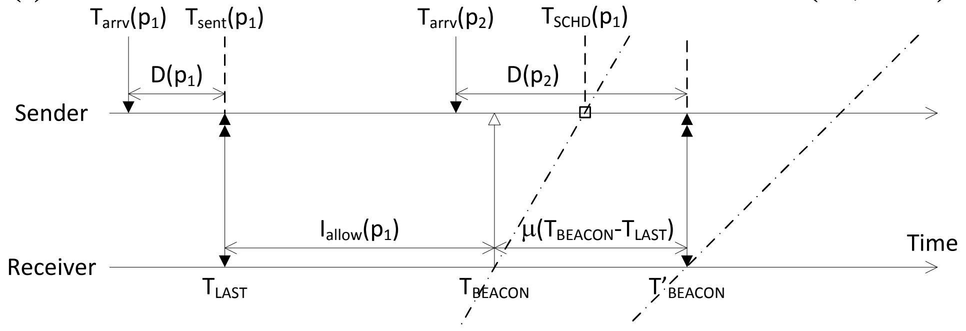
2.2.3.2 $a > 1$

In this case, the sender clock counts time at a slower rate than the receiver clock, as shown in Figure 2.6(b). After the sender delivers a packet p_1 successfully to the receiver, both sender and receiver schedule the next rendezvous time to $I_{\text{allow}}(p_1)$ time later. Since the sender clock counts slower, when the sender wakes up at $T_{\text{SCHD}}(p_1)$ to listen for a beacon from the receiver, the receiver has already finished its beacon transmission. As a result, the sender has to remain awake to wait for the next beacon. We have:

$$D(p_2) = (1+\mu)I_{\text{allow}}(p_1)\frac{1}{a} + D(p_1) - (T_{\text{arrv}}(p_2) - T_{\text{arrv}}(p_1)). \tag{2.21}$$



(a) When the sender clock counts time faster than the receiver clock (i.e., $a < 1$).



(b) When the sender clock counts time slower than the receiver clock (i.e., $a > 1$).

Figure 2.6 Effects of time asynchrony on CyMAC performance.

When the system stabilizes, $D(p_1) = D(p_2) \triangleq D$ and $T_{\text{arrv}}(p_2) - T_{\text{arrv}}(p_1) = \theta(p_1) \triangleq \theta$. Plugging in Equation (2.6), we have

$$\begin{aligned} & D = (1+\mu)((1+\mu)\theta - D)\frac{1}{a} + D - \theta \\ \Longrightarrow \quad & D = \left(\mu + 1 - \frac{a}{1+\mu}\right)\theta. \end{aligned} \tag{2.22}$$

This means that an extra delay of $(1 - \frac{a}{1+\mu})\theta$ has been added to the packet delivery delay.

To ameliorate the effects of time asynchrony, we have employed the following schemes in CyMAC:

- To guarantee a relative delay bound of μ, CyMAC does it more conservatively by replacing μ with $\mu^* = \mu - |1-\hat{a}|$ as the target delay bound in sensor nodes' operations, where $\hat{a}$ is the upper limit of clock drift between sensor nodes. When $|1-\hat{a}| < \mu$, CyMAC works fine. However, if $\mu \leqslant |1-\hat{a}|$, CyMAC won't be able to provide the desired delay bound. Fortunately, this situation rarely occurs in practice as it makes little sense to ask a sensor network to provide a delay bound that is even tighter than the degree of clock asynchrony between sensor nodes.

- In CyMAC, the sender wakes up a bit earlier prior to the scheduled listen time to wait for beacons. Specifically, if the time between the previous listen time and the next listen time is ψ seconds, the sender will wake up at $\left(\frac{\mu\psi}{2+2\mu}\right)$ prior to the next listen time.

With these two enhancements, time asynchrony can be dealt with effectively and the original relative delay bound of μ can be satisfied. The proof is as follows.

proof: 1 *As $\mu^* = \mu - |1-\hat{a}|$, we have $\mu^* = \mu - 1 + a$ when $a < 1$, and $\mu^* = \mu + 1 - a$ when $a > 1$.*

- *Case I: $a < 1$. By simply replacing μ with μ^* in Equation (2.20), we have $D = \mu\theta$, meaning that the desired delay bound is achieved. This indicates that when the sender clock counts time faster than the receiver clock (as shown in Figure 2.6(a)), using a conservative μ would guarantee the target delay bound.*

- *Case II: $a \geqslant \frac{2(1+\mu)}{2+\mu}$. Since $\mu^* = \mu + 1 - a < \mu$, we have $a \geqslant \frac{2(1+\mu)}{2+\mu} > \frac{2(1+\mu^*)}{2+\mu^*}$. Then, by replacing μ with μ^* in Equation (2.22), we have*

$$\begin{aligned} D &= \left(\mu^* + 1 - \frac{a}{1+\mu^*}\right)\theta \\ &= \left(\mu + 2 - a - \frac{a}{1+\mu^*}\right)\theta \\ &< \left(\mu + 2 - \frac{2(1+\mu^*)}{2+\mu^*} - \frac{\frac{2(1+\mu^*)}{2+\mu^*}}{1+\mu^*}\right)\theta \\ &= \left(\mu + 2 - \frac{2(1+\mu^*)}{2+\mu^*} - \frac{2}{2+\mu^*}\right)\theta \\ &= \mu\theta. \end{aligned} \tag{2.23}$$

  Thus the desired delay bound is achieved.

- *Case III: $1 < a < \frac{2(1+\mu)}{2+\mu}$. Since the sender clock counts time slower than the receiver clock, the* *scheduled listen time for the sender is at* $\left(1-\frac{1}{a}\right)\psi$ *after the beacon arrival time, where*

$$\left(1-\frac{1}{a}\right)\psi < \left(1-\frac{2+\mu}{2+2\mu}\right)\psi = \frac{\mu\psi}{2+2\mu}. \tag{2.24}$$

Therefore, if the sender wakes up at $\left(\frac{\mu\psi}{2+2\mu}\right)$ *prior to the scheduled listen time, we can guarantee that the sender receives the first beacon frame from the receiver. As a result, the packet delay must be smaller than the originally planned delay bound.*

Till now, we have proved that, with the proposed two enhancements, CyMAC can deal with time asynchrony effectively to guarantee the original relative delay bound of μ*.*

2.3 Performance Evaluation

Testbed-based experiment and ns-2 based simulation are conducted to evaluate the performance of CyMAC and compare it with RI-MAC, in terms of relative end-to-end delay and duty cycle.

2.3.1 Testbed Evaluation

We set up a testbed system composed of 9 MicaZ motes, forming a line or a star topology as illustrated in Figure 2.7. For each topology, CyMAC or RI-MAC is run respectively in the experiment. The average beacon interval in RI-MAC is set to one second. The only parameter for CyMAC is μ, the desired relative delay bound for a single hop. Depending on the desired end-to-end relative delay bound μ_{e2e}, μ is set to $(1+\mu_{\text{e2e}})^{1/\xi}-1$ where ξ is the hop-count diameter of the network, following the definition in Section 2.2.1.2.

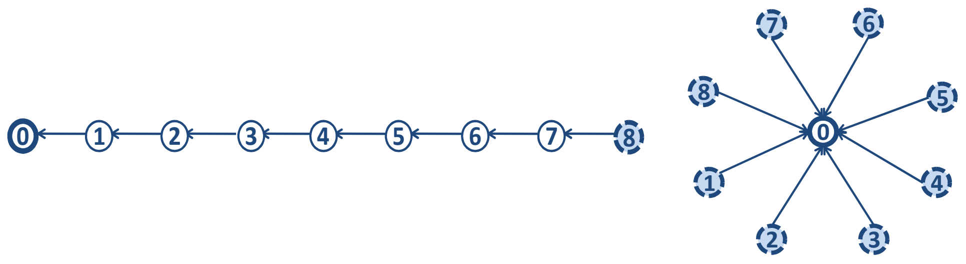
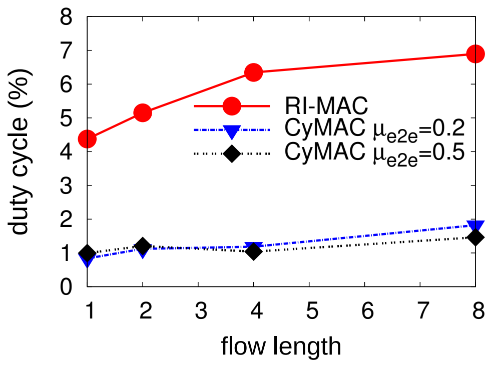


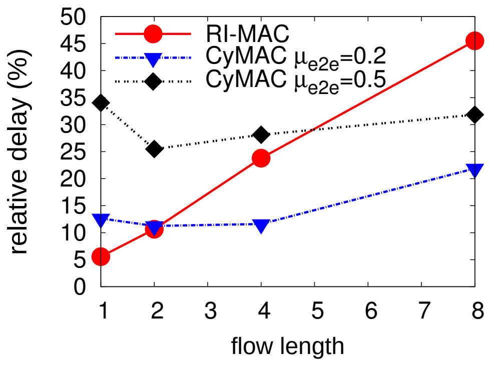
Figure 2.7 The line and star topologies of the testbed system.

2.3.1.1 Line Topology

In each experiment, there is a single data packet flow starting from node 1, 2, 4 or 8 to sink node 0 with flow length of 1, 2, 4 or 8 hops, respectively. The performances of CyMAC and RI-MAC are compared with varying flow length, data packet generation interval τ and μ_{e2e}.



(a) duty cycle



(b) relative delay

Figure 2.8 Comparison of CyMAC and RI-MAC with the line topology as the flow length and the end-to-end relative delay bound μ_{e2e} vary. For each flow, data packets are generated at the source node at an average interval of $\tau = 10s$ with 10% variance. μ_{e2e} is 0.2 or 0.5.

With varying flow length and μ_{e2e}, the duty cycles of CyMAC and RI-MAC are compared in Figure 2.8(a). In RI-MAC, as each node sends a beacon every one second regardless of the traffic condition, and each sender needs to idly listen for 0.5 seconds (on average) to send a packet, a lot of energy is consumed. In contrast, CyMAC establishes rendezvous times between neighbors adaptively to the packet arrival interval; hence, it saves much idle listening and has significantly lower duty cycle than RI-MAC. Figure 2.8(b) shows the relative delay with CyMAC and RI-MAC. CyMAC provides the desired delay bound as expected, while the end-to-end delay in RI-MAC increases linearly with the flow length. When the flow length is large, RI-MAC cannot provide the desired delay bound even with a higher duty cycle than CyMAC.

CyMAC and RI-MAC are compared in Figure 2.9 with varying μ_{e2e} and τ. As μ_{e2e} increases, the relative delay achieved by CyMAC increases accordingly and the average duty cycle of sensor nodes decreases. This is because CyMAC attempts to schedule the rendezvous times between neighbor nodes

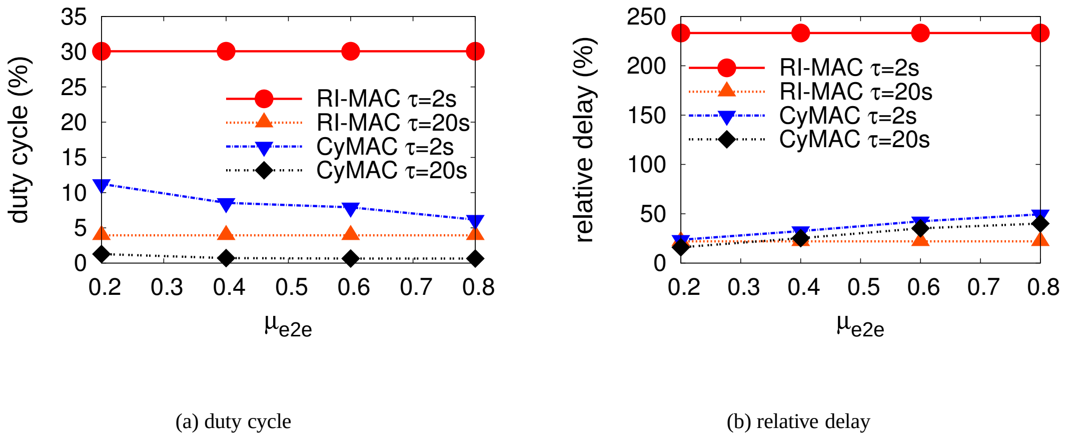


Figure 2.9 Comparison of CyMAC and RI-MAC with the line topology as the desired end-to-end relative delay bound μ_{e2e} and the average packet generation interval τ vary. The flow length is fixed at 8 hops.

in the way that the duty cycle of the nodes is as low as possible provided that the desired delay bound is guaranteed. However, RI-MAC does not change its beacon interval as μ_{e2e} changes, and therefore keeps the same duty cycle and relative delay. Similar to the reasons explained for Figure 2.8, RI-MAC has higher duty cycle and relative delay than CyMAC.

Figure 2.10 demonstrates a trace of instantaneous changes in duty cycle and relative delay as τ varies over time. Each delay or duty cycle point in the figure represents the measurement during a 20s period ending at the corresponding time instance. As we can see, CyMAC always guarantees the desired end-to-end delay bound except for a short duration when τ drops suddenly from 20s to 10s around time 2500s. In this case, some packets (with $\tau = 10$s) are queued and their end-to-end delay may exceed the desired bound. The instantaneous duty cycles in this duration also increase because packets need to be exchanged in a higher frequency in order to reach new rendezvous times. Nevertheless, CyMAC can adapt to the traffic changes and re-stabilize the system quickly. Comparing with CyMAC, RI-MAC does not adapt to the traffic changes and has higher duty cycle and relative delay during most of the time.

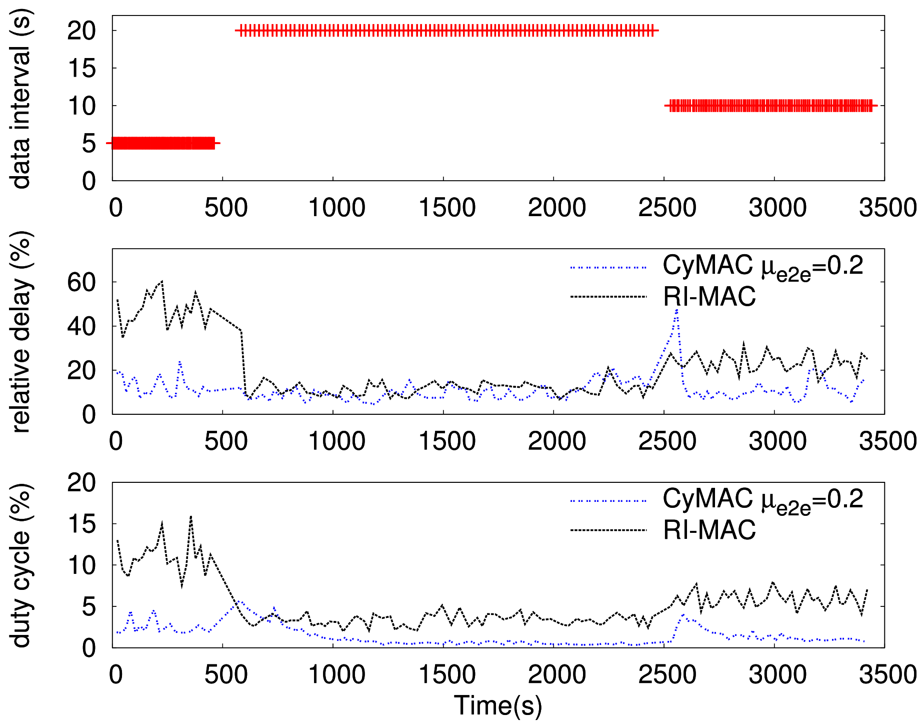


Figure 2.10 A trace demonstrates the instantaneous changes in duty cycle and relative delay as the packet generation interval τ varies over time. The flow length is fixed at 4 hops. μ_{e2e} is fixed to 0.2.

2.3.1.2 Star Topology

We deploy the testbed network in a star topology, as illustrated in Figure 2.7, where node 0 is the sink and other nodes can be data sources. We vary the number of source nodes and the packet generation interval τ in the experiment.

Results are shown in Figure 2.11. As a receiver in CyMAC sends out beacons at the scheduled beacon times to each of its senders, the time spent on sending beacons increases with the number of senders and with τ. A receiver in RI-MAC, on the other hand, sends out beacons at a constant rate regardless of the number of senders or τ. Also considering that a receiver in CyMAC and RI-MAC spends similar time for packet reception, the overall duty cycle of a receiver in CyMAC has higher duty cycle than its counterpart in RI-MAC when the number of senders is large and/or τ is small, as illustrated in Figure 2.11(a). In this case, however, a sender in CyMAC has a much lower duty cycle than its counterpart in RI-MAC, as illustrated in Figure 2.11(b), because CyMAC can significantly reduce the idle listening time for senders through setting up rendezvous times between sender and

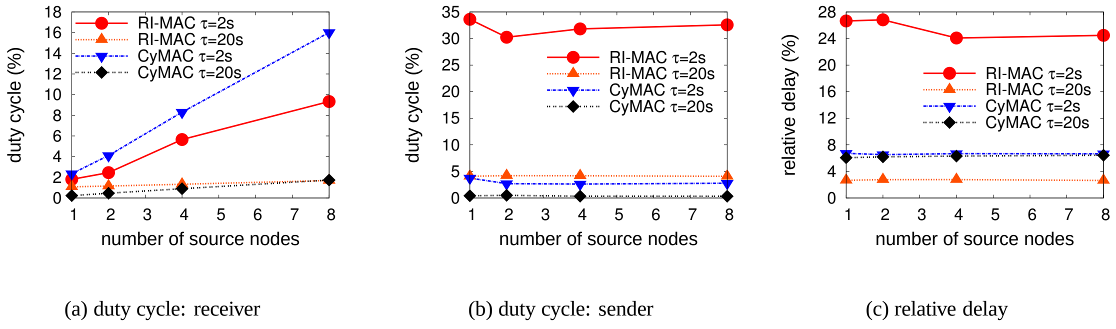


Figure 2.11 Comparison of CyMAC and RI-MAC with the star topology as the number of source nodes and the packet generation interval τ at each source node vary. μ_{e2e} is 0.1.

receiver.

Figure 2.11(c) demonstrates that CyMAC always achieves the desired relative delay, regardless of the number of source nodes or τ. RI-MAC limits the average absolute delay to half of the beacon interval, and thus the relative delay increases as τ decreases. Therefore, the relative delay in RI-MAC is not affected much by the number of source nodes but by τ.

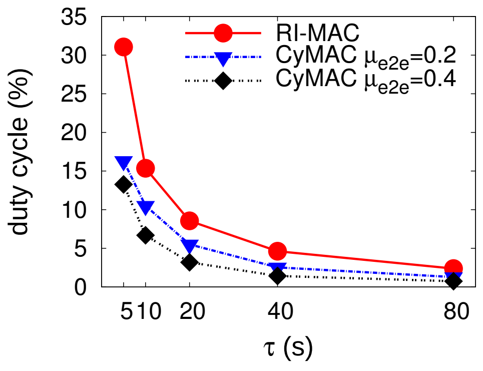
2.3.2 Simulation Evaluation

CyMAC is evaluated in large-scale networks with the ns-2 simulator. Two scenarios are considered: a grid sensor network where one node is the sink and every other node is a data source; a random mesh network with multiple data flows.

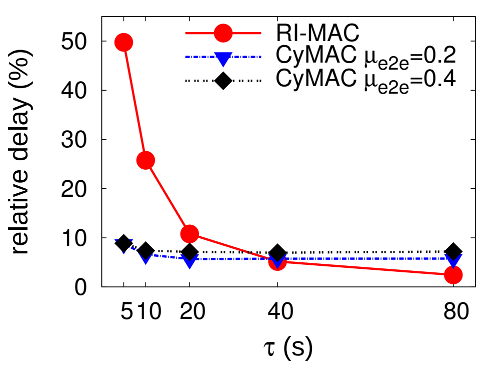
2.3.2.1 Grid Topology

A total of 49 sensor nodes are deployed to form a 7×7 grid where nearby nodes are 70 meters apart. The node at the center is the sink while every other node is a data source. CyMAC and RI-MAC are run respectively in the network to compare their performances. The packet generation interval τ at each source node varies from 5 seconds to 80 seconds, and the desired end-to-end relative delay bound μ_{e2e} is set to 0.2 or 0.4.

As showed in Figure 2.12, CyMAC always has lower duty cycle than RI-MAC. When the network



(a) duty cycle



(b) relative delay

Figure 2.12 Comparison of CyMAC and RI-MAC with the grid topology as the packet generation interval τ at each source node and the desired end–to-end relative delay bound μ_{e2e} vary.

traffic is heavy (e.g., $\tau = 5$s), a node in CyMAC may spend more time sending beacons to signal its senders than its counterpart in RI-MAC, but it spends much less time on idle listening for each packet that it sends; as the result of these two factors, CyMAC has lower duty cycle than RI-MAC, which is demonstrated by the simulation results. When the network traffic is light (e.g., $\tau = 80$s), CyMAC also has lower duty cycle than RI-MAC because a node in CyMAC has less beacons to send due to the larger τ, but a node in RI-MAC still needs to send beacons at the same rate regardless of the change in traffic condition.

Figure 2.12(b) depicts the changes of the end-to-end relative delay as τ varies. RI-MAC's absolute end-to-end delay is not affected much by τ because it is mainly determined by the beacon interval and the network hop-count diameter. Hence, as τ decreases, its relative delay, which is the ratio of the absolute delay to τ, increases accordingly. On the other hand, CyMAC can adapt the rendezvous times between nodes to the change of τ and maintain a stable relative delay below the desired bound.

2.3.2.2 Mesh Topology with Multiple Flows

A total of 49 nodes form a mesh topology with five data flows passing through 25 nodes, as shown in Figure 2.13. In this scenario, different flows have different sources, destinations, flow lengths and data generation intervals. They co-exist in the network and affect each other, which represents a more realistic situation than the line, star or grid topology.

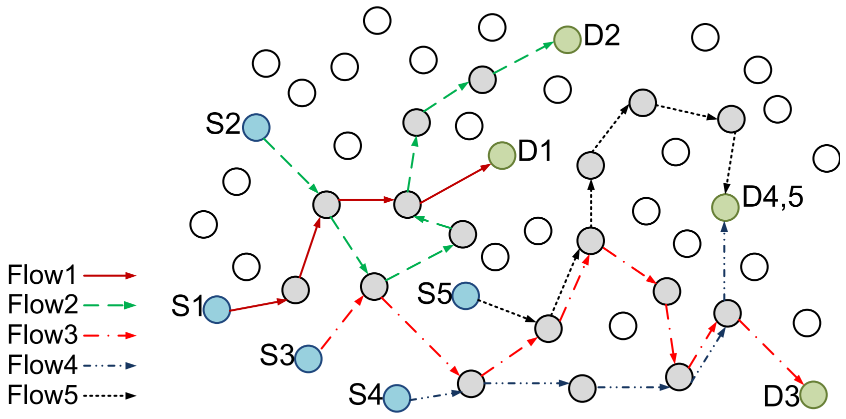
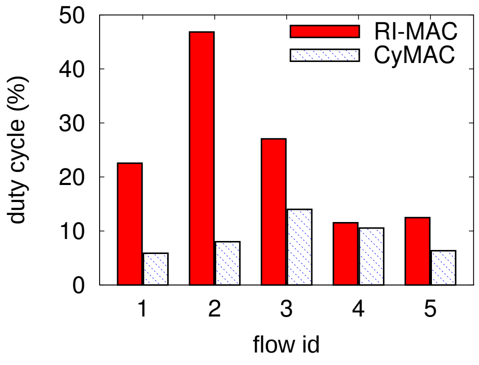


Figure 2.13 Mesh topology with multiple flows. A total of 49 nodes are in the network and five flows pass 25 nodes. The numbers of nodes on these flows are 4, 7, 8, 5 and 6, respectively. The data generation intervals of the flows are 20s, 10s, 30s, 50s and 40s, respectively.

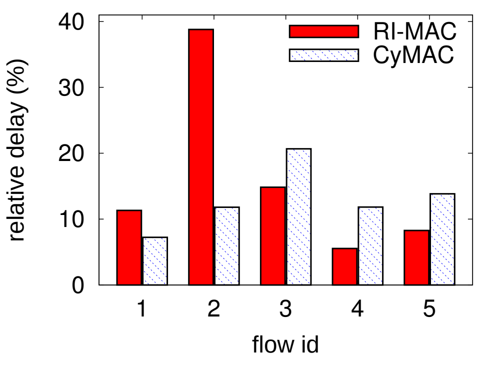
Figure 2.14 shows that CyMAC has lower duty cycle than RI-MAC for nodes on every flow and all flows can achieve the desired delay bound. As the flow length is different in each flow and the per-hop delay bound is conservatively selected based on the network hop-count diameter, shorter flows achieve lower delay than longer ones. For example, flow 1 has a relative delay of 0.072, flow 3 has a relative delay of 0.207, and their flow lengths are 4 and 8 respectively.

2.4 Conclusions

In this chapter, we propose a new receiver-initiated sensor MAC protocol called CyMAC, and implement it in both TinyOS and the ns-2 simulator. Theoretical analysis and in-depth experiments/simulations demonstrate that CyMAC guarantees the desired delay bound for data delivery services under various traffic conditions. It yields a lower duty cycle than RI-MAC in most cases except when the required delay bound is very tight. In this case, CyMAC can still provide the delay bound guarantee at the cost of having a slightly higher duty cycle than RI-MAC. In addition, CyMAC can tolerate time asynchrony between sensor nodes.



(a) duty cycle



(b) relative delay

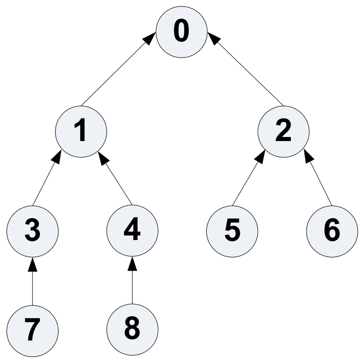
Figure 2.14 Comparison of CyMAC and RI-MAC with the mesh topology. The desired end-to-end relative delay bound is $\mu_{\text{e2e}} = 0.2$.

CHAPTER 3. LB-MAC: A LIFETIME-BALANCED MAC PROTOCOL FOR SENSOR NETWORKS

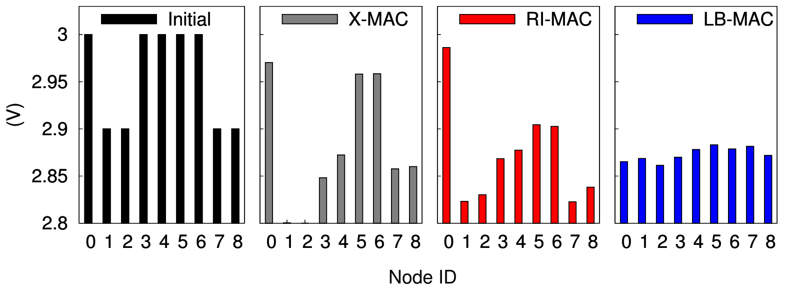
3.1 Introduction

Energy conservation is perhaps the most important issue in battery-operated sensor networks. It is always desirable to extend the operational lifetime of a sensor network as much as possible. For many sensor network applications [16–19], the *network lifetime* is often defined as the minimal nodal lifetime among all sensor nodes in the network. This is because, the depletion of battery energy of bottleneck sensor nodes, such as the nodes close to the root in a tree topology network, may cause network disconnection and render the sensor network nonfunctional. Although energy saving techniques such as energy-aware routing can be used to reduce the workload and extend the lifetime of bottleneck sensor nodes, they may still consume more energy than other nodes in the network and thus bound the network lifetime. Besides, sensor nodes with a similar level of workload may have different nodal lifetime due to environmental [20, 21] or system reasons. For example, nodes with poorer-quality batteries or solar-rechargeable nodes deployed at shady locales may have shorter lifetime than their peers. Therefore, to maximize the network lifetime, it is important to extend the shortest nodal lifetime among all sensor nodes.

Despite the need for a holistic approach to address the energy conservation challenge and to prolong the network lifetime, most of the current research on MAC protocol design has focused on reducing the energy consumption and extending the operational lifetime of individual sensor nodes. For example, as shown in Figure 3.1, when sensor nodes run X-MAC [9] or RI-MAC [10], which are two state-of-the-art MAC protocols for sensor networks, they experience severe imbalance in residual nodal energy after 1.4 hours of network operation. As a result, the network lifetime is limited due to such energy bottleneck effect.



(a) A tree topology where nodes 5, 6, 7, 8 are source nodes.



(b) Initial and residual nodal energy after the network has operated for 1.4 hours with X-MAC and RI-MAC, respectively. Note that, nodes 1 and 2 are bottleneck nodes in the network.

Figure 3.1 The energy bottleneck effect with two state-of-the-art MAC protocols. The data generation rate is 2 packets/second and the wakeup interval is one second for all protocols in the experiment.

To remedy this deficiency, we investigate the MAC protocol design from the perspective of network lifetime maximization and propose a new solution, called LB-MAC (Lifetime-Balancing MAC), to achieve this goal via balancing the nodal lifetime between neighboring sensor nodes. We have implemented LB-MAC in TinyOS and experiment results show that LB-MAC outperforms the state-of-the-art MAC protocols in terms of network lifetime while maintaining comparable levels of data delivery ratio, average nodal power consumption, and end-to-end data delivery delay.

LB-MAC emphasizes collaboration between sensor nodes to benefit the network as a whole, even at the expense of a single node. The key idea is that neighboring nodes adjust their MAC-layer behaviors together (only when there are data communications between them) via the following tunable parameters: *wakeup interval* and *channel checking period* at the receiver side, and *data retry interval* and *idle listening period* at the sender side. These parameters are tuned carefully in a certain manner so that (i) the rendezvous between sender and receiver can always be guaranteed; (ii) the incurred communication overhead (for rendezvous maintenance) can be shifted between them; and (iii) the packet delivery delay between neighboring nodes shall be preserved. This way, the node with a shorter expected lifetime than its communicating neighbor can extend its lifetime by shifting more communication overhead to the neighbor. As a result, the network lifetime can be prolonged.

In brief, the behavior of LB-MAC can be generalized as follows.

- **Shifting communication overhead from a sender to a receiver.** If a receiver finds itself with a longer expected lifetime than sender, it may decrease the wakeup interval or increase the channel checking period, which allows the sender to choose a longer data retry interval (to reduce its communication energy consumption) while the rendezvous between the sender and the receiver can still be guaranteed.

- **Shifting communication overhead from a receiver to a sender.** On the other hand, to save energy at the receiver side, the sender may attempt data transmissions more frequently (with a shorter data retry interval) so that the receiver can increase the wakeup interval or shorten the channel checking period to reduce its communication energy consumption. The sender may even choose to keep listening idly upon a data arrival; this way, receiver can reduce the channel checking period to minimal, and the rendezvous between the sender and the receiver is triggered solely by the receiver's periodic beacons.

3.2 Related Work

3.2.1 Fixed Duty Cycle MAC Protocols

For many duty cycle MAC protocols, the MAC operational parameters are predetermined before deployment for simplicity of usage and implementation, and the parameter settings are usually the same on all nodes in the network.

Among these protocols, B-MAC [7] and X-MAC [9] are representative sender-initiated asynchronous MAC protocols. In B-MAC, the rendezvous between a sender and a receiver is established through long preambles initiated by the sender. X-MAC improves over B-MAC by replacing the long preamble with a sequence of short, strobed preambles. A node running X-MAC may stop sending short preambles upon receiving an EarlyACK from its target receiver, thus saving more energy than B-MAC.

As B-MAC and X-MAC are optimized mainly for light traffic conditions, the preambles may congest the channel and block data transmissions in the scenarios of bursty or high traffic load. To work under a wider range of traffic conditions, RI-MAC [10] and A-MAC [22] adopt a receiver-initiated beacon-based strategy. Each node wakes up periodically and sends out a short beacon to explicitly notify its neighbors that it is ready to receive data. When a node has data to transmit, it wakes up and waits for a beacon from the target receiver. Once such a beacon is received, it starts sending the data. Compared to the sender-initiated preamble-based protocols, a receiver-initiated protocol only requires a receiver to keep radio on for a short period after sending a beacon (i.e., tx-rx turnaround time) and therefore saves the receiving energy cost. Additionally, the receiver-initiated nature allows efficient collision resolution which can effectively save the transmission energy cost when the channel contention is severe. However, it is worth noting that under very light traffic, the receiver-initiated protocols may incur higher energy cost than the sender-initiated protocols due to the overhead of sending receiver's beacons and waiting for incoming traffics.

3.2.2 Dynamic Duty Cycle MAC Protocols

Different from the above fix duty cycle MAC protocols, MAC parameter tuning in duty cycle sensor networks has also been studied in [8, 23–31].

Particularly, SEESAW [23] was proposed to balance the energy consumption between a sender and a receiver through adapting the data retry interval at the sender side and the channel checking period at the receiver side. Though SEESAW yields a longer network lifetime than B-MAC and S-MAC, the effectiveness of SEESAW is limited by several factors. Firstly, as a sender-initiated only protocol, SEESAW mandates a minimum channel checking period at the receiver side, which may incur unnecessary energy consumptions. Secondly, the policies used in SEESAW for balancing nodal lifetime are empirical and not adaptive to varying network conditions. Thirdly, MAC parameters such as the wakeup interval and the idle listening period are fixed in SEESAW, which, if tuned properly, could prolong the network lifetime further.

Both DDCC [27] and CyMAC [28] target at improving individual nodal energy efficiency. In DDCC [27], a controller is implemented on individual sensor nodes to dynamically adjust the radio duty cycle based on the network traffic condition. CyMAC [28] was proposed to reduce radio duty cycle by scheduling rendezvous between neighboring nodes based on the relative end-to-end delay requirement and the network traffic condition. Though these schemes may reduce individual nodal energy consumption, they may not effectively improve the network lifetime due to the lack of collaboration between nodes. MaxMAC [32] is a MAC protocol that can adapt between X-MAC and pure CSMA mode of operations given different network traffic conditions to deal with the tradeoff between energy-efficiency and throughput/delay. More recently, AEDP [33] is proposed to dynamically adjust the radio CCA threshold to improve network reliability and duty cycle based on application-specified bounds. Although these protocols can improve nodal energy-efficiency, deal with the exposed throughput or latency drawbacks of duty cycle MAC protocols, they cannot significantly improve the network lifetime as a whole.

ZeroCal [26] is a MAC layer protocol which adaptively tunes the wakeup intervals between a sender and a receiver to balance their energy consumption; however, the proposed scheme may cause increased end-to-end packet delivery delays as the wakeup interval may be extended indefinitely to save nodal energy. Additionally, ZeroCal does not consider the adjustment of other MAC parameters such as channel checking period and data retry interval, which, if tuned properly, could further prolong the network lifetime. GDSIC [29] is another work targeting at improving the fairness of energy utilization in duty cycle sensor networks. It proposes a similar idea as in ZeroCal by dynamically tuning the nodal wakeup interval. Different from ZeroCal, GDSIC decides the individual nodal wakeup interval through solving distributed convex optimization problems. Though the network lifetime can be prolonged in GDSIC, the side effect of increased data delivery delay may also be observed.

pTunes [30] is a recent work that adjusts the MAC parameters dynamically for low-power sensor networks. It formalizes three optimization problems, in each of which the network lifetime, the end-to-end reliability, or the end-to-end latency is the optimization objective while the other two are the optimization constraints, and the MAC-layer parameters including radio-on duration, radio-off duration, and the number of retransmission attempts are the output. Furthermore, pTunes is a centralized solution that requires periodic network state collection and parameter dissemination. Hence, it may not be feasible in practice.

3.2.3 Uniqueness of Proposed LB-MAC Protocol

Different from existing works, our proposed LB-MAC protocol aims to improve the network lifetime while satisfying a certain delay preservation requirement. It achieves this goal with a unique approach that adjusts nodal radio duty cycles (i) collaboratively between neighbors, and (ii) systematically via a comprehensive set of tunable operational parameters at the MAC layer. It is a distributed, lightweight, and scalable solution as the control information is only exchanged locally between neighbors.

3.2.4 Techniques Beyond MAC Layer

Multiple energy-aware routing protocols [18, 34, 35] have been proposed to prolong sensor networks' lifetime. Recently, the authors in [16,36,37] proposed specially-designed energy-aware routing schemes for duty cycle sensor networks. In all these works, the main idea is to route packets through nodes with a higher residual energy or a longer nodal lifetime such that nodes with a lower energy or a shorter lifetime can participate less in data transmission activities. As a result, the minimum nodal lifetime in the network may be extended and the network lifetime may be prolonged. In addition, approaches to prolonging the network lifetime through cross-layer design are proposed in [38–42]. In these works, [38] attempts to maximize the network lifetime via joint routing and MAC design, [42] solves the problem via joint routing and congestion control, and [39] tackles the problem through joint optimal design of physical, MAC, and routing layers in time slotted networks.

Complementarily, LB-MAC can be integrated with the above schemes to further improve the network lifetime.

3.3 Analysis

In this section, we define a generic model for duty cycle MAC protocols in sensor networks. Based on this model, an analytical study is conducted to provide a theoretical foundation for the design of our proposed LB-MAC protocol.

3.3.1 Duty Cycle MAC Protocols: A Generic Model

Figure 3.2 illustrates the behaviors of sensor nodes in a generic duty cycle MAC protocol, which are explained below. Table 3.1 lists the parameters to characterize a MAC protocol.

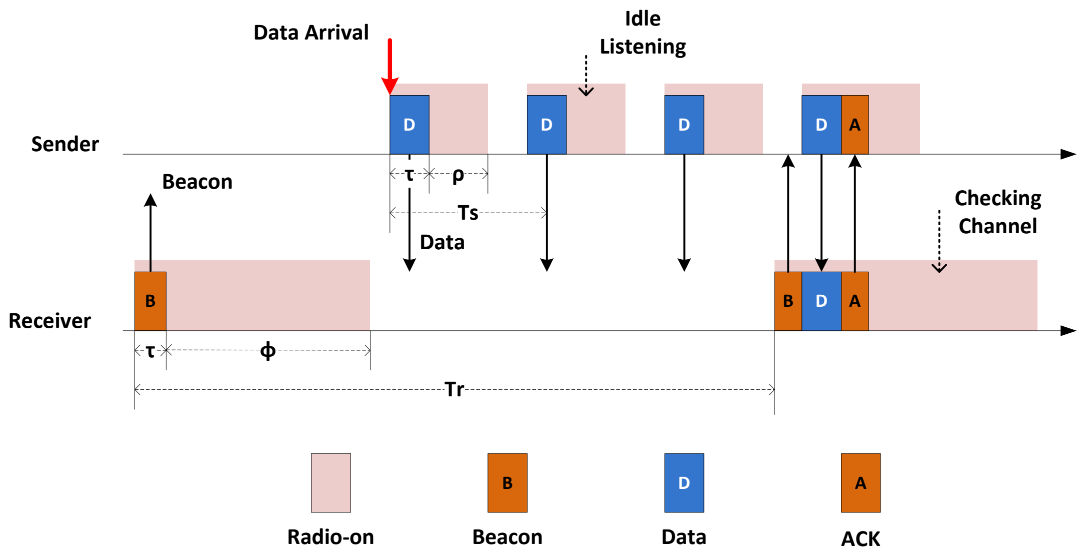


Figure 3.2 A generic model for duty cycle MAC protocols.

Table 3.1 Duty cycle MAC protocol parameters

T_s	sender's data retry interval
ρ	sender's idle listening period
η_s	Boolean value: 1 - sender sends a probe; 0 - no
T_r	receiver's wakeup interval
ϕ	receiver's channel checking period
η_r	Boolean value: 1 - receiver sends a beacon; 0 - no
τ	duration of a probe/beacon transmission

As a receiver, a sensor node wakes up every T_r interval to interact with potential senders. At the beginning of each wakeup, the sensor node may send out a beacon message to waiting senders (the transmission duration of the beacon message is τ), or silently wait for its senders to transmit packets. During the wakeup period, the sensor node checks the channel activity for ϕ time for incoming messages. If a data packet is received within ϕ time, it replies with an ACK; otherwise, it goes back to sleep.

On the other hand, when a sensor node has a data packet to send, it wakes up every T_s interval to interact with the target receiver. At the beginning of each wakeup, the sensor node may transmit the data packet[1] immediately or wait silently for the target receiver's beacon to start the data transmission. During the idle listening period ρ, if an ACK is received, the procedure ends as the data packet has been delivered successfully; if a beacon is received instead, it retransmits the data packet; if neither ACK nor beacon is received, it goes back to sleep and wakes up at the next T_s interval and to repeat the above procedure.

Note that, a sensor node may participate in the network activity as a sender, a receiver, or both at the same time.

The above model can be instantiated to a specific MAC protocol by assigning proper values to the parameters. For example, as shown in Table 3.2, the X-MAC [9] protocol can be obtained by setting $\eta_r = 0$ (i.e., receiver does not send any beacon), $\eta_s = 1$, $T_s = \epsilon$ (which is the sum of τ and *tx-rx turnaround time*), $\rho = T_s - \eta_s \cdot \tau$, and $\phi = 20$ms. RI-MAC [10] can be obtained by setting $\eta_r = 1$, $\eta_s = 0$ (i.e., sender waits silently for receiver's beacon without sending a data packet), $T_s = \infty$, $\rho = T_s - \eta_s \cdot \tau = \infty$ (i.e., sender keeps listening idly as long as it has packets to send), and $\phi = 7$ms (a platform dependent value).

3.3.2 Analysis of Rendezvous Condition and Packet Delivery Delay

Though the rendezvous condition for existing MAC protocols has been analyzed in related works as discussed in Section 3.2, for the sake of completeness, we present the analysis of rendezvous condition based on the generic model given in Section 3.3.1.

To ensure that sender and receiver meet within T_r time to deliver a data packet, the MAC protocol parameters shall satisfy the following condition, called the *rendezvous condition*:

$$(\eta_r \cdot \tau + \phi) + (\eta_s \cdot \tau + \rho) > \min\{T_s, T_r\}, \tag{3.1}$$

[1]As the data packet transmission time is usually small and can be in the same fold as a probe in many sensor network applications, the LPL scheme in TinyOS 2.1 [43] uses data packets to replace the preambles. Similarly, in our design and analysis, we also let senders send data packets instead of probes.

Table 3.2 MAC protocol settings

	T_s	η_s	ρ	T_r	η_r	ϕ
RI-MAC	∞	0	∞	fixed	1	7ms
A-MAC	∞	0	∞	fixed	1	128μs
X-MAC	ϵ	1	$\epsilon - \tau$	fixed	0	20ms
SEESAW	$\phi/1.2$	1	$\epsilon - \tau$	fixed	0	dynamic
ZeroCal	ϵ	1	$\epsilon - \tau$	dynamic	0	fixed
GDSIC	∞	0	∞	dynamic	1	fixed
AutoSync	ϵ	1	$\epsilon - \tau$	dynamic	0	fixed
MaxMAC	ϵ	1	$\epsilon - \tau$	dynamic	0	fixed
LB-MAC	dynamic	1	dynamic	dynamic	1	dynamic

which can be summarized from the following cases:

- Case I: $0 < T_s \leqslant T_r$. In this case, as shown in Figure 3.3.2, if a sender fails in its first transmission attempt of a data packet (because the target receiver is asleep), it goes back to sleep and wakes up later. To ensure that sender and receiver meet within T_r time, one of the sender's future awake durations should overlap with the receiver's very next awake duration. That is, the following condition shall be satisfied:

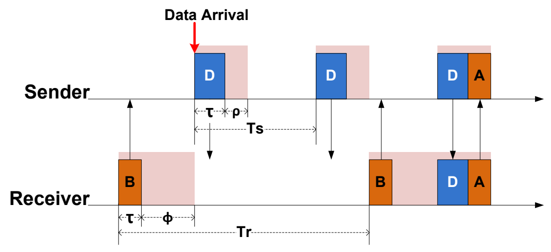
$$(\eta_r \cdot \tau + \phi) > T_s - (\eta_s \cdot \tau + \rho), \tag{3.2}$$

  which also means

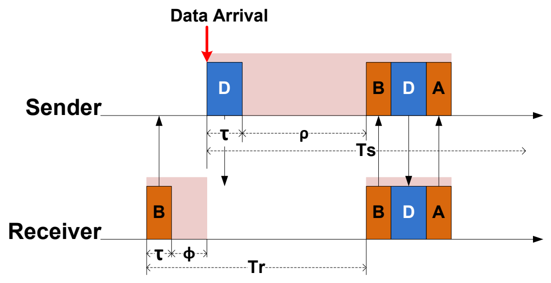
$$(\eta_r \cdot \tau + \phi) + (\eta_s \cdot \tau + \rho) > T_s = \min\{T_s, T_r\}. \tag{3.3}$$

- Case II: $T_s > T_r$. In this case, sender's data retry interval is longer than receiver's wakeup interval (e.g., in RI-MAC and A-MAC, $T_s = \infty$ as sender simply waits silently for receiver's beacon to start the data transmission). In order to deliver a data packet within T_r time, sender needs to keep listening the channel till the receiver's very next beacon is received, as illustrated in Figure 3.3.2. Therefore, the following condition shall be satisfied:

$$(\eta_s \cdot \tau + \rho) > T_r - (\eta_r \cdot \tau + \phi), \tag{3.4}$$



(a) Case I: $0 < T_s \leqslant T_r$.



(b) Case II: $T_s > T_r$.

Figure 3.3 Rendezvous between sender and receiver in duty cycle sensor networks.

which also means

$$(\eta_r \cdot \tau + \phi) + (\eta_s \cdot \tau + \rho) > T_r = \min\{T_s, T_r\}. \tag{3.5}$$

It is easy to verify that rendezvous condition (3.1) holds for all existing MAC protocols, including sender-initiated protocols such as X-MAC and SEESAW, and receiver-initiated protocols such as RI-MAC and A-MAC. When designing LB-MAC, we also require the condition to hold. In fact, we require a slightly more stringent rendezvous condition:

$$\phi + \rho \geqslant \min\{T_s, T_r\}, \tag{3.6}$$

which simplifies the design and analysis of the protocol by omitting the small value of τ.

When rendezvous condition is satisfied, the maximum one-hop packet delivery delay from node x to node y under a perfect channel condition is:

$$D_{x \to y} = T_r(y) - \phi(y). \tag{3.7}$$

3.3.3 Analysis of Nodal Lifetime

Based on the above analysis, the expected lifetime of sender x and receiver y, denoted as $L_s(x)$ and $L_r(y)$ respectively, can be estimated as follows:

$$L_s(x) = \frac{e(x)}{D_{x\to y} \cdot \frac{\rho(x)}{T_s(x)} \cdot R(x) \cdot P + g(x)} \tag{3.8}$$

and

$$L_r(y) = \frac{e(y)}{\frac{\phi(y)}{T_r(y)} \cdot P + g(y)}, \tag{3.9}$$

where (i) $e(x)$ and $e(y)$ are the amount of residual energy at sender and receiver, respectively, (ii) $R(x)$ is the sender's outgoing data rate, (iii) P is the amount of energy consumed when a node's radio is on for one unit of time (transmission and reception power are assumed to be the same [10,44,45]), and (iv) $g(x)$ and $g(y)$ are the energy consumption rates of sender and receiver, respectively, for other causes.

In the above estimation, the sender's outgoing data rate is assumed to be low so that there is no queueing at the sensor nodes, which is typical in low duty cycle sensor network applications [36,46,47]. Therefore, to send a data packet, sender x needs to wait for $D_{x\to y}$ time with a radio duty cycle of $\frac{\rho(x)}{T_s(x)}$. As a result, it consumes $D_{x\to y} \cdot \frac{\rho(x)}{T_s(x)} \cdot R(x) \cdot P$ power for data transmissions. For receiver y, it wakes up for $\phi(y)$ time every $T_r(y)$ interval. Hence, its energy consumption rate for receiving can be estimated as $\frac{\phi(y)}{T_r(y)} \cdot P$.

As a sensor node may act as both sender and receiver in the network, its expected lifetime shall be estimated by considering its power consumption for communicating with each of its senders and each of its receivers by combining Equations (3.8) and (3.9). From the equations, we can see that the nodal lifetime of sender and receiver can be balanced through tuning their MAC layer parameters (i.e., T_s, ρ, T_r, and ϕ) collaboratively.

3.3.4 Analysis of Cross Traffic Delay

Based on Equation (3.7), the cross-traffic delay [48] over node x for path $x' \to x \to y$ under a perfect channel condition, can be defined as

$$D_{x' \to x \to y} = D_{x' \to x} + D_{x \to y} \tag{3.10}$$

Equation (3.10) illustrates that, if the change of $D_{x' \to x}$ or $D_{x \to y}$ is within a certain range, such that the cross-traffic delay incurred after the change is no more than the value before the change, the original delay value can be preserved. This way, the end-to-end packet delivery delay may be preserved if the cross-traffic delay for all nodes from leaf to sink can be preserved.

Similar to the analysis in Sections 3.3.2 and 3.3.3 that the MAC layer parameters can affect rendezvous, one-hop packet delivery delay and nodal lifetime, the cross-traffic delay can also be affected. In particular,

- If a receiver node y increases its one-hop delay due to lifetime balancing (i.e, decrease ϕ and/or increase T_r), then a sender node may need to decrease its one-hop delay ((i.e, increase ϕ and/or decrease T_r)) in order to preserve $D_{x' \to x \to y}$.
- If a receiver node y decreases its one-hop delay due to lifetime balancing, then a sender node x can increase its one-hop delay without increasing $D_{x' \to x \to y}$. However, due to the existing rendezvous settings between node x and its own senders, node x may not be allowed to increase its one-hop delay to avoid possible rendezvous condition violations. In this case, node y may save this decreased delay value and use it to compensate future delay increases. In the following analysis and design, we refer to this delay savings at a receiver node as D_{credit}.

3.3.5 Problem Statement and Design Principle

To effectively prolong the sensor network lifetime, ideally, all sensor nodes shall work together to maximize the minimum nodal lifetime in the entire network. Unfortunately, it is impractical to solve this optimization problem in a realistic sensor network, because it requires each node to know the following information of every other node in the network: the residual nodal energy, the energy consumption rate, and the data arrival rate. Acquiring these information could incur very high communication overhead because of the potentially large network scale and the dynamic nature of the information. So instead, we design LB-MAC as a distributed, localized, and low-cost solution to approach the problem, such that for each node only needs to solve a localized problem.

Formally, the problem can be described as follows:

Objective: When communication happens between two nodes on any link $j \to i$,

- $\max \min\{L(i), L(j)\}$, where $L(i)$ and $L(j)$ are i's and j's nodal lifetime.

Subject to:

- *Rendezvous Condition:*
  $\phi(i,j) + \rho(j,i) \geqslant \min\{T_s(j,i), T_r(i,j)\}$.
- *Delay Preservation Requirement:*
  $D^{\text{new}}_{k\to j\to i} \leqslant D_{k\to j\to i} + D_{credit}(j) + D_{credit}(i)$, where $D^{\text{new}}_{k\to j\to i}$ is the cross-traffic delay according to the new MAC parameter settings.

Output:

- For node i,
  - its $T_r(i,j)$ and $\phi(i,j)$ parameters to communicate with its sender node j
- For node j,
  - its $T_s(j,i)$ and $\rho(j,i)$ parameters to communicate with its receiver node i
  - its $T_r(j,k)$ and $\phi(j,k)$ parameters to communicate with its sender node k

This way, each node only coordinates locally with its neighboring nodes to balance their lifetime, and the coordination occurs only when there are data communications between them.

- If a node as a receiver finds itself with a longer expected lifetime than its sender, it shall attempt to shift more communication overhead from the sender. According to Equations (3.8) and (3.9), this can be done by increasing ϕ and/or decreasing T_r at the receiver side, accompanied with increasing T_s and/or decreasing ρ at the sender side, as long as both delay preservation requirement and rendezvous condition are satisfied.

- On the other hand, if a receiver finds itself with a shorter expected lifetime than its sender, it shall attempt to shift more communication overhead to the sender via decreasing ϕ and/or increasing T_r at the receiver side, and decreasing T_s and/or increasing ρ at the sender side.

As a result, the minimal nodal lifetime between communicating neighbors can be extended, and the network lifetime may be prolonged.

3.4 LB-MAC Design

In LB-MAC, whenever there are data communications between a pair of sensor nodes, they adapt their MAC-layer behaviors together in a collaborative manner via piggybacking information in the data/ACK exchanged between them. For example, based on the information piggybacked in a data packet from a sender, the receiver decides its T_r and ϕ values and embeds them in an ACK to the sender. Upon reception of the ACK, the sender adjusts its T_s and ρ values accordingly to ensure that the rendezvous condition is satisfied. In LB-MAC, the receiver takes a leading role to coordinate the MAC behaviors of itself and each sender. This way, senders don't need to exchange information between themselves to adjust their behaviors, thus saving more energy. Receiver's and sender's behaviors are elaborated in Sections 3.4.1 and 3.4.2, respectively, where we use flow $x' \rightarrow x \rightarrow y$ as an illustrative example to explain the behavioral details.

3.4.1 Receiver's Behavior

The operational flowchart of an LB-MAC node as a receiver is shown in Figure 3.4. Every T_r interval (i.e., when the wakeup timer is fired), receiver y turns on radio, sends a beacon, and monitors the channel for ϕ time. During the monitoring period, if a data packet is received from sender x, the following information will be extracted from the data packet: *x's estimated nodal lifetime*, *one-hop* *communication delay from x's previous-hop node x' to x – denoted as* $D_{x' \to x}$, and *x's tuning credit – denoted as* $D_{credit}(x)$.

As analyzed in Section 3.3.4, node x's *tuning credit* refers to the cumulative delay savings (for one-hop communication from x' to x) generated by x's previous adjustments of T_r and ϕ values. For example, if x as a receiver increases its ϕ or decreases its T_r for 100ms, the tuning credit of x will be increased by 100ms; and if x keeps increasing ϕ or decreasing T_r, the tuning credit can be accumulatively increased over period. Initial value of the tuning credit is zero. In Section 3.4.2, we give examples in Figure 3.6 on how the tuning credit may be utilized by either a receiver or a sender.

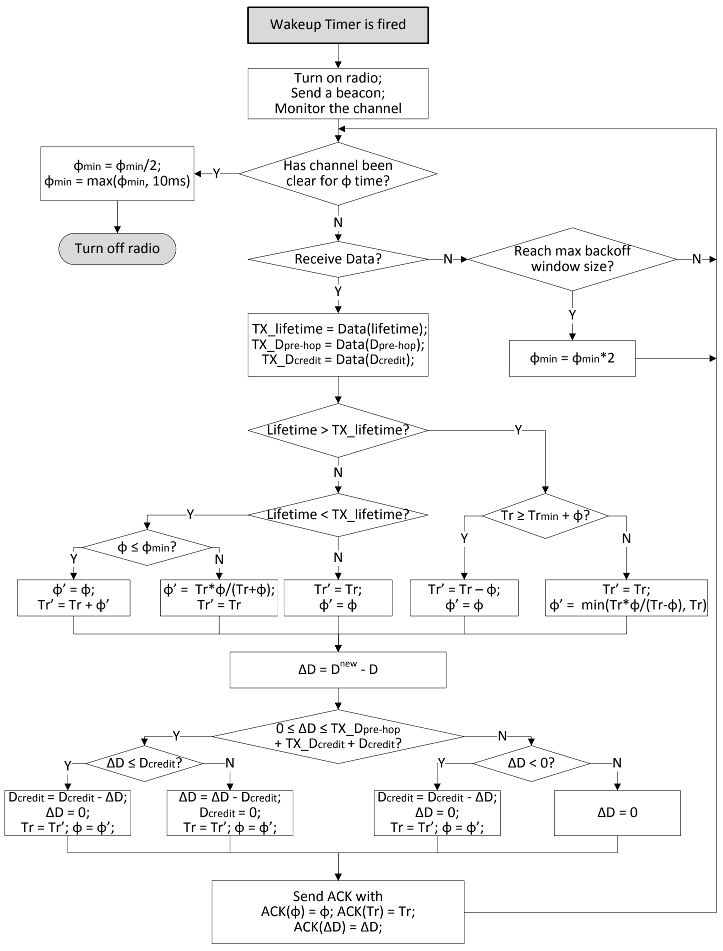

Wakeup Timer is fired
Turn on radio; Send a beacon; Monitor the channel
Has channel been clear for ϕ time?
ϕmin = ϕmin/2; ϕmin = max(ϕmin, 10ms)
Turn off radio
Receive Data?
Reach max backoff window size?
ϕmin = ϕmin*2
TX_lifetime = Data(lifetime); TX_Dpre-hop = Data(Dpre-hop); TX_Dcredit = Data(Dcredit);
Lifetime > TX_lifetime?
Lifetime < TX_lifetime?
Tr ≥ Trmin + ϕ?
ϕ ≤ ϕmin?
ϕ' = ϕ; Tr' = Tr + ϕ'
ϕ' = Tr*ϕ/(Tr+ϕ); Tr' = Tr
Tr' = Tr; ϕ' = ϕ
Tr' = Tr – ϕ; ϕ' = ϕ
Tr' = Tr; ϕ' = min(Tr*ϕ/(Tr-ϕ), Tr)
ΔD = D^new - D
0 ≤ ΔD ≤ TX_Dpre-hop + TX_Dcredit + Dcredit?
ΔD ≤ Dcredit?
ΔD < 0?
Dcredit = Dcredit - ΔD; ΔD = 0; Tr = Tr'; ϕ = ϕ';
ΔD = ΔD - Dcredit; Dcredit = 0; Tr = Tr'; ϕ = ϕ';
Dcredit = Dcredit - ΔD; ΔD = 0; Tr = Tr'; ϕ = ϕ';
ΔD = 0
Send ACK with ACK(ϕ) = ϕ; ACK(Tr) = Tr; ACK(ΔD) = ΔD;
Y
N

Figure 3.4 Receiver's behavior in LB-MAC.

When a receiver adjusts its operational parameters, the sender needs to adjust its own operational parameters accordingly to ensure that the rendezvous condition is satisfied. As a result, the nodal lifetime of both sender and receiver, as well as the one-hop communication delay between them, may be affected, which have been analyzed in Section 3.3. Therefore, receiver y is allowed to adjust its operational parameters (T_r and ϕ) only if the parameter adjustment does not violate the delay preservation requirement. This can be guaranteed as long as the following condition is satisfied:

$$\Delta D_{x\to y} \leqslant D_{x'\to x} + D_{credit}(x) + D_{credit}(y), \tag{3.11}$$

where $D_{credit}(x)$ and $D_{credit}(y)$ represent the tuning credits of nodes x and y respectively, and $\Delta D_{x\to y}$ is the increased one-hop communication delay from x to y as a result of y's parameter adjustment. $\Delta D_{x\to y}$ can be calculated as:

$$\Delta D_{x\to y} = D^{\text{new}}_{x\to y} - D_{x\to y}, \tag{3.12}$$

where $D^{\text{new}}_{x\to y}$ is the new one-hop delay after $T_r(y)$ and/or $\phi(y)$ has been changed.

Condition (3.11) implies that the maximum increment allowed in $D_{x\to y}$ (without violating the delay preservation requirement) is $\max\Delta D = D_{x'\to x} + D_{credit}(x) + D_{credit}(y)$. As shown in Equation (3.13) below, the maximum increment can be accommodated by (i) asking x to adjust its operational parameters to reduce $D_{x'\to x}$ to $D^{\text{new}}_{x'\to x} = 0$, and (ii) using up all the tuning credits saved for communication hops $x' \to x$ and $x \to y$.

$$\begin{aligned}
&D^{\text{new}}_{x'\to x\to y}\\
&= D^{\text{new}}_{x'\to x} + D^{\text{new}}_{x\to y}\\
&= 0 + [D_{x\to y} + \max\Delta D_{x\to y}]\\
&= 0 + [D_{x\to y} + D_{x'\to x} + D_{credit}(x) + D_{credit}(y)]\\
&= [D_{x'\to x} + D_{credit}(x)] + [D_{x\to y} + D_{credit}(y)]\\
&= D^{\text{currently\_allowed}}_{x'\to x\to y}.
\end{aligned} \tag{3.13}$$

As shown in the middle of Figure 3.4, receiver y attempts to adjust T_r and ϕ according to the following rules, and the adjustment takes effect only when the resulting $\Delta D_{x\to y}$ satisfies the delay preservation requirement (3.11).

- When the receiver has a longer expected lifetime than the sender, it decreases T_r gradually in steps of ϕ till T_r reaches a default minimal value then it starts to increase ϕ to $\frac{T_r}{T_r-\phi}\cdot\phi$ iteratively till $\phi = T_r$.

- When the receiver has a shorter expected lifetime, it decreases ϕ to $\frac{T_r}{T_r+\phi}\cdot\phi$ iteratively till reaching $\phi_{\min}$; then it starts to increase T_r in steps of ϕ. Here, $\phi_{\min}$ is an online parameter that we use to indicate the severity of the current channel contention; a larger $\phi_{\min}$ value corresponds to more severe channel contention. In Section 3.4.3.3, we will discuss in more detail how channel contention is handled in LB-MAC.

The reason for choosing such adjustment steps for T_r and ϕ is to ensure that T_r is always an integer multiple of ϕ, which simplifies the design, analysis, and implementation of LB-MAC.

After adjusting T_r and ϕ, y updates $\Delta D_{x\to y}$ and $D_{credit}(y)$, and the updated $\Delta D_{x\to y}$ value is embedded together with the new T_r and ϕ parameters in the ACK to the sender.

Upon receiving the ACK from receiver y, sender x adjusts its operational parameters to ensure that both rendezvous condition (3.6) and delay preservation requirement (3.11) are satisfied, which we discuss next.

3.4.2 Sender's Behavior

The operational flowchart of an LB-MAC node as a sender is shown in Figure 3.5. Every T_s interval (i.e., when the data retry timer is fired), sender x turns on radio, sends a data packet, and monitors the channel for ρ time. Within ρ time, if a beacon is received, node x retransmits the data packet; on the other hand, if an ACK is received from receiver y, x extracts the following information from the ACK: $T_r(y)$, $\phi(y)$, and $\Delta D_{x\to y}$, based on which to adjust its operational parameters as follows.

Step 1: As shown in the middle of the flowchart, to satisfy rendezvous condition (3.6), x sets $\rho(x)$ to *tx-rx turnaround time*, and $T_s(x)$ to $\phi(y)$, except when $\phi(y)$ is less than $\phi_{\min}$. In the latter situation, x remains awake and keeps listening idly for beacon or ACK from receiver y by setting

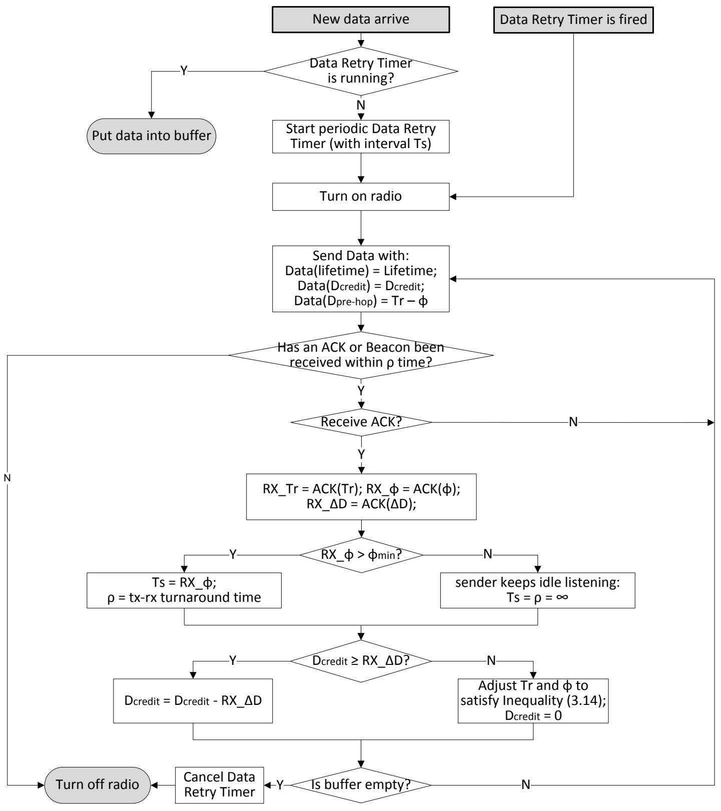

New data arrive
Data Retry Timer is fired
Data Retry Timer is running?
Y
N
Put data into buffer
Start periodic Data Retry Timer (with interval Ts)
Turn on radio
Send Data with: Data(lifetime) = Lifetime; Data(Dcredit) = Dcredit; Data(Dpre-hop) = Tr – ϕ
Has an ACK or Beacon been received within ρ time?
Receive ACK?
RX_Tr = ACK(Tr); RX_ϕ = ACK(ϕ); RX_ΔD = ACK(ΔD);
RX_ϕ > ϕmin?
Ts = RX_ϕ; ρ = tx-rx turnaround time
sender keeps idle listening: Ts = ρ = ∞
Dcredit ≥ RX_ΔD?
Dcredit = Dcredit - RX_ΔD
Adjust Tr and ϕ to satisfy Inequality (3.14); Dcredit = 0
Is buffer empty?
Cancel Data Retry Timer
Turn off radio

Figure 3.5 Sender's behavior in LB-MAC.

$T_s(x) = \rho(x) = \infty$ (similar to how RI-MAC operates), instead of retransmitting data every short $\phi(y)$ time.

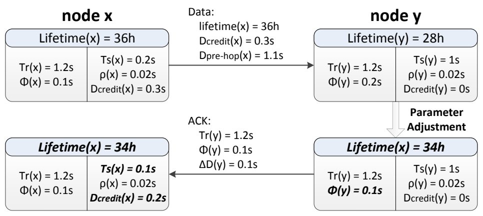
Step 2: As shown in the bottom right of the flowchart, if $D_{credit}(x) \leqslant \Delta D_{x \to y}$, this means that the saved tuning credit won't be able to pay off the remaining delay increment that receiver y demands. In this situation, x needs to adjust its own T_r and ϕ parameters (used to communicate with its own sender x') to satisfy delay preservation requirement (3.11). Specifically, x will first decrease T_r and then increase ϕ, if needed, till it finds the first pair of T_r and ϕ that satisfy the following inequality:

$$\Delta D_{x' \to x} \geqslant \Delta D_{x \to y} - D_{credit}(x), \tag{3.14}$$

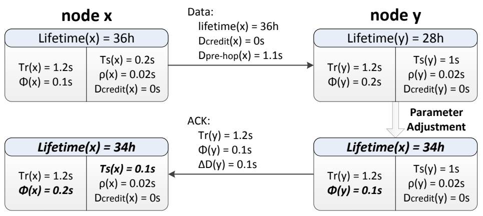
where

$$\Delta D_{x' \to x} = D_{x' \to x} - D^{new}_{x' \to x}. \tag{3.15}$$

Figure 3.6 gives two examples on how the sender adjusts its parameters under different scenarios.



(a) As receiver y has a shorter expected lifetime than sender x, it decreases ϕ which results in an increase in one-hop communication delay: $\Delta D_{x \to y} = (1.2 - 0.1) - (1.2 - 0.2) = 0.1$s. Since $D_{credit}(x) > \Delta D_{x \to y}$, sender x simply pays off $\Delta D_{x \to y}$ using the saved credit: $D_{credit}(x) = 0.3 - 0.1 = 0.2$s.



(b) Similar to (a), receiver y decreases ϕ which results in an increase of 0.1s in one-hop communication delay and $\Delta D_{x \to y}$ is updated to 0.1s. This time, however, since $D_{credit}(x) < \Delta D_{x \to y}$, sender x has to adjust its own ϕ value so that $D_{x' \to x}$ is decreased to offset $D_{credit}(y)$ and the end-to-end delay remains the same.

Figure 3.6 Parameter tuning examples in LB-MAC. Tuned parameters are shown in ***italic bold*** font.

Note that the above adjustment may only increase ϕ and/or decrease T_r; hence, the rendezvous condition remains valid after the adjustment.

3.4.3 Robustness of the LB-MAC Design

In order for LB-MAC to be practically useful, it is critical to ensure that LB-MAC functions properly in the presence of failed data packet transmissions, route changes, and multiple concurrent senders, all of which occur often in practical environments.

3.4.3.1 Failed Data Packet Transmission

A failed data packet transmission may be due to loss of data packet itself or loss of ACK, either of which can occur due to imperfect channel conditions in practice.

The loss of data packet has no effects on rendezvous between sender and receiver in LB-MAC; the loss of ACK may cause sender and receiver to lose synchronization of their MAC-layer behaviors, because the important decision on MAC behavior adaptation may be piggybacked in the ACK. For example, a receiver may decide to reduce ϕ and carry this decision in an ACK. Unfortunately, due to loss of ACK, the sender never gets notified of the change and continues to operate with a T_s value that is larger than the new ϕ. As a result, rendezvous condition (3.6) given in Section 3.3.2 may be violated. The loss of data packet or ACK may increase the one-hop delivery delay and violates the delay preservation requirement.

LB-MAC deals with these situations as follows.

- For each data packet transmission, a sender node may retransmit the packet up to a certain retry limit (i.e., 3 times in LB-MAC implementation); for each beacon packet transmission, a receiver node may retransmit the packet when the channel is not clear. The retries may help to tolerate imperfect channel conditions.

- A sender transmits the data packet with the previously-agreed upon MAC-layer operational parameters till either the packet is delivered successfully or when the packet has been retried for $(T_r - \phi)$ time.

- If a data packet cannot be delivered to the receiver after $(T_r - \phi)$ period, the sender node asks the routing layer to make a decision to either retransmit or discard the packet in MAC layer. For future packets sent to the same receiver, sender will listen idly till the receiver's beacon is received to reestablish the rendezvous. Notice that, the routing layer may decide to retransmit the data packet, which would increase the actual packet delivery delay; however, this does not violate the delay preservation requirement at the MAC layer.

3.4.3.2 Handling of Multiple Senders or Receivers

In LB-MAC, as the parameter tuning is made between a pair of sender and receiver, a node who serves as a common receiver to multiple senders may decrease ϕ or increase T_r for one sender and then lose the rendezvous with other senders. To address this problem, a receiver records the the tuning credit and the scheduled T_r and ϕ values with each sender, and chooses the smallest T_r as its wakeup interval and the largest ϕ as its channel checking period. This way, the rendezvous with all senders can be guaranteed.

LB-MAC can also work in mesh topology networks where each node may have multiple receivers rather than a single receiver node within every certain period of time. A sender nodes simply needs to transmit the data packet according to each receiver's parameter settings, such that the rendezvous with the target receiver can be guaranteed.

3.4.3.3 Handling of Channel Contention

Under the circumstances where the receiver has a shorter expected lifetime than all its senders, it will keep decreasing the ϕ value. However, when ϕ becomes too small, data packets will be transmitted frequently every $T_s = \phi$ time, which may cause severe contention to the channel and a large number of packet collisions. As a result, senders may waste lots of energy contending for the channel.

To deal with this situation, LB-MAC maintains an online parameter $\phi_{\min}$ as an indicator of the severity of the channel contention. A larger $\phi_{\min}$ corresponds to more severe channel contention. As shown at the top of Figure 3.4, $\phi_{\min}$ is doubled/halved when the receiver senses the channel busy/idle after it sends a beacon. The minimum value for $\phi_{\min}$ is set to 10ms. Then, when the intended new ϕ value is smaller than $\phi_{\min}$, the receiver will notify the sender to set T_s and ρ to ∞. This way, the sender will listen idly for the receiver's beacon to start a data transmission, instead of attempting a data transmission every $T_s = \phi$ time; hence, channel contention can be reduced and energy can be saved at both sender and receiver.

3.5 LB-MAC Implementation

We have implemented LB-MAC in TinyOS 2.1.0 [43]. Figure 3.7 shows its composition within the UPMA framework [49, 50], where the shaded parts are the main components of LB-MAC:

- *LBMACScheduler* is the core scheduling component. It resides atop the radio core layer and handles all operations of message processing and parameter tuning, based on the flow charts shown in Figures 3.4 and 3.5.

- *LBMAC Adaption Code* of the radio core layer provides a variety of low-level supports for the LBMACScheduler component. Particularly, it monitors channel after sending each beacon and estimates channel contention status based on it.

In the following, we first present the message formats used in LB-MAC and then discuss some implementation issues.

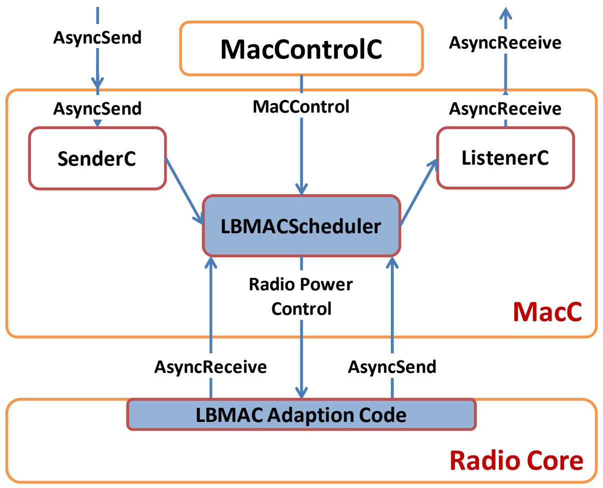


Figure 3.7 LB-MAC architecture.

3.5.1 Message Formats

Figure 3.8 shows the message formats used in LB-MAC, where the shaded fields are the ones added/modified for LB-MAC.

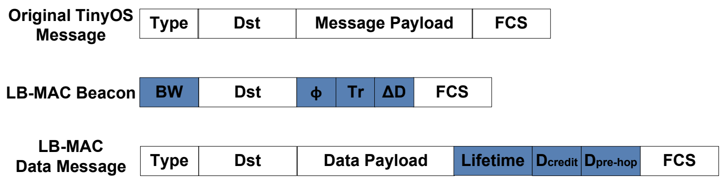


Figure 3.8 Message formats used in LB-MAC (shaded fields are added/modified in LB-MAC).

- The beacon message is used by a receiver either as a notification sent upon its wakeup or as a software ACK to acknowledge the reception of a data packet.

- Similar to RI-MAC, LB-MAC reuses the *type* field in the beacon message to carry the backoff window size that will be used by the sender to select its backoff value. Different from RI-MAC and A-MAC, LB-MAC adds 6-byte fields to each beacon message to carry ϕ, T_r and ΔD values.

- The sender piggybacks the following information in each data packet: the estimated nodal lifetime, the communication delay of the previous hop, and the tuning credit. These information will be used by the receiver to tune the MAC-layer parameters, as discussed in Section 3.4.1.

3.5.2 Residual Energy Estimation

In order for sensor nodes to make proper decisions on tuning their MAC-layer parameters, it is critical that they can measure/estimate the nodal residual energy and the nodal lifetime. We have designed and fabricated a TelosB power meter kit as shown in Figure 3.9 for this purpose. This kit measures the nodal power consumption rate, based on which a node can calculate the total energy consumed so far. The nodal residual energy is the difference between the battery energy capacity and the consumed energy.

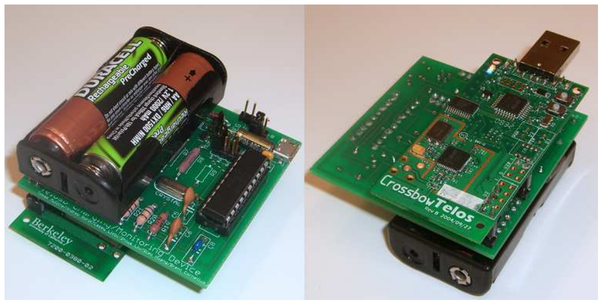


Figure 3.9 TelosB power meter kit used in LB-MAC. The working power consumption of this kit is 2.4μW which is small compared to radio power consumption.

3.6 Performance Evaluation

Experiments have been conducted to evaluate the performance of LB-MAC and compare it with X-MAC, RI-MAC, and SEESAW, in terms of *network lifetime*, *data delivery ratio*, *average nodal power* *consumption*, and *end-to-end data delivery delay*.

3.6.1 Experiment Setup

In the experiments, the testbed is composed of 37 TelosB motes, in which node 0 is connected to a computer, and its radio is kept on all the time to serve as the sink. CTP (Collection Tree Protocol) [51] is used to find the routes for data packet forwarding, and the network topology may vary over time in the experiments; the initial topology established by the routing protocol is shown in Figure 3.10. The end-to-end delay requirement D_{e2e} is 6 seconds in all experiments.

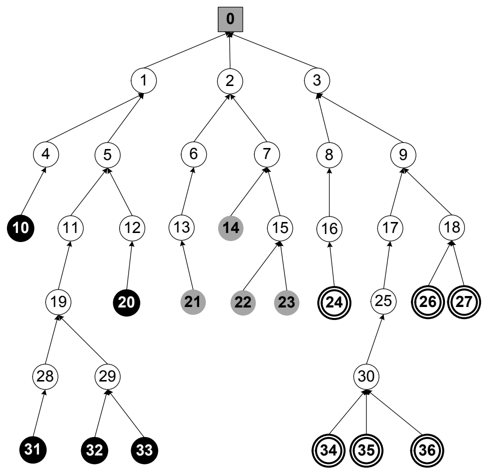


Figure 3.10 The initial network topology of the testbed determined by CTP. Sensors in black circles, gray circles and double circles form three different sensing areas.

For X-MAC, and RI-MAC, the ϕ, ρ, and T_s parameters are set according to Table 3.2, which reflects the settings in [9] and [10]. The value of T_r for X-MAC and RI-MAC are selected based on empirical results to achieve better network lifetime performances without violating the end-to-end delay requirement. Particularly, in each experiment, X-MAC and RI-MAC are evaluated with T_r value set at 0.4, 0.6, 0.8 and 1 second, and the measurements associated with the best network lifetime performances are plotted for X-MAC and RI-MAC in the following figures.

For SEESAW, the initial value of ϕ is set to 30ms and T_s is set to $\phi/1.2 = 25$ms [23]. To be comparable with SEESAW, the initial values of both ϕ and T_s in LB-MAC are set to 30ms. For both

SEESAW and LB-MAC, the initial T_r is calculated as $T_r = \frac{D_{e2e}}{\texttt{network diameter}}$, which is based on the end-to-end delay requirement and network diameter. As D_{e2e} is 6 seconds, and the monitored maximum network diameter is 6, the initial T_r value for both SEESAW and LB-MAC is 1 second.

In the following sections, experiment results are plotted with a 95% confidence interval, except snapshots and traces.

3.6.1.1 Lifetime Measurement

During the experiments, we notice that it may take weeks to completely drain fully-charged batteries of sensor nodes. In order to complete all the experiments within a reasonable amount of time while demonstrating the features and performances of evaluated protocols, we study how fast a sensor node consumes a designated small amount of energy, and evaluate its nodal lifetime as the time period during which this designated amount of energy is consumed[2]. This also allows us to start the experiments with nodes at different initial energy levels, which simplifies and speeds up the evaluation process significantly.

3.6.2 Static Network Settings

We first compare LB-MAC with other protocols under the scenario of static network settings, in which the sensing event detection pattern, network topology, and packet loss ratio are all fixed. Particularly, the setup is as follows:

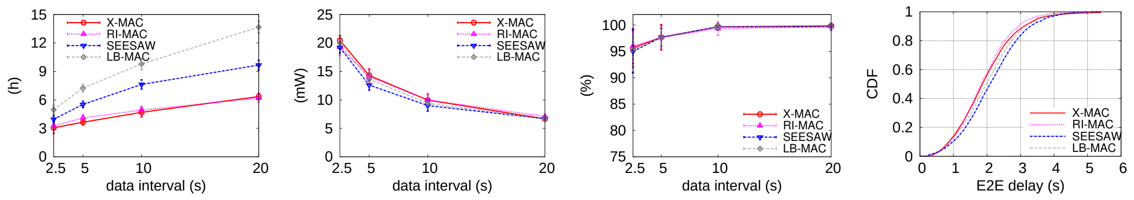
- *Static routing paths*, that is, the network topology is setup by CTP at the beginning of experiments and not changed thereafter (by disabling routing updates in CTP).
- *Static sensing events*, that is, sensing events are assumed to be detected by sensors 24, 26, 27, 34, 35, and 36 only. These sensors (i.e., source nodes) generate data packets at a certain fixed rate and forward them hop by hop to the sink.
- *Static packet loss ratio*, that is, the channel is under the regular lab condition and node software will not drop any packets on purpose; as we measured, the packet loss ratio is negligible.

[2]Based on the ratio between the full nodal energy capacity and this designated amount of nodal energy, the measured nodal lifetime can be scaled up to obtain the actual nodal lifetime. Specifically, if the full nodal energy is E_c, the designated nodal energy is E_m and the measured nodal lifetime using the designated energy is ℓ, the actual nodal lifetime is $L = \ell * \frac{E_c}{E_m}$.

3.6.2.1 Uniform Initial Nodal Energy

When the initial nodal energy is uniform, the designated amount of energy available at each sensor node is 400 Joules.

Figures 3.11 and 3.12 compare the performances of evaluated protocols with uniform initial nodal energy. As shown in Figure 3.11(a), LB-MAC yields a longer network lifetime than RI-MAC, X-MAC, and SEESAW under various data generation intervals: when the data generation interval is 2.5 seconds, LB-MAC extends the network lifetime by about 60% more than RI-MAC and X-MAC, and 30% more than SEESAW. When the data generation interval is 20 seconds (very low traffic in the network), the improvement of the network lifetime is about 100% over RI-MAC and X-MAC. This is due mainly to the following reasons. As RI-MAC and X-MAC fix the MAC-layer operational parameters, bottleneck nodes (such as node 9) have the heaviest workloads and consume more energy than others; thus, they yield a shorter nodal lifetime, which constrains the network lifetime as shown in Figure 3.12(a).



(a) Network lifetime.

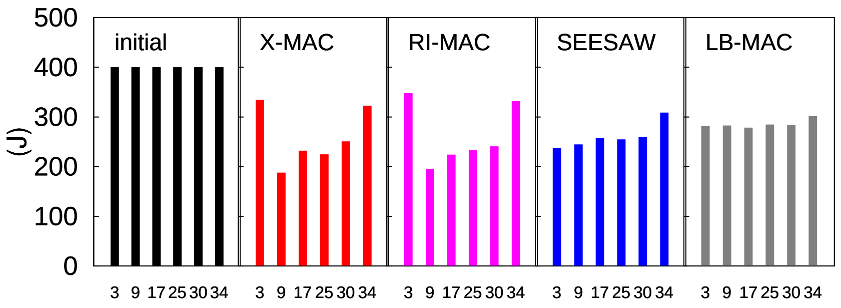
(b) Average nodal power consumption.

(c) Data delivery ratio.

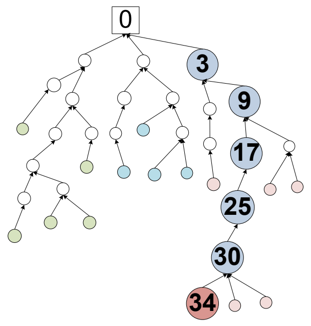
(d) CDF of end-to-end delay.

Figure 3.11 Performance comparison with uniform initial nodal energy. At data intervals 2.5s, 5s, 10s, and 20s, the best network lifetime performance for X-MAC is achieved under T_r values of 0.6, 0.8, 1, and 1 second, respectively; the best network lifetime for RI-MAC is obtained under T_r values of 0.8, 0.8, 1, and 1 second, respectively.

In comparison, LB-MAC dynamically adjusts the MAC-layer parameters to shift communication overhead away from the bottleneck nodes, thus increasing the network lifetime significantly. SEESAW also attempts to balance nodal lifetime by adjusting some of the MAC-layer parameters. However, the parameter adjustment in SEESAW is less effective than that in LB-MAC because SEESAW simply



(a) Residual energy of nodes 3, 9, 17, 25, 30, and 34 after 2 hours of network operation. Data generation interval at source nodes is 5 seconds.



(b) The studied route from node 34 to the sink node 0.

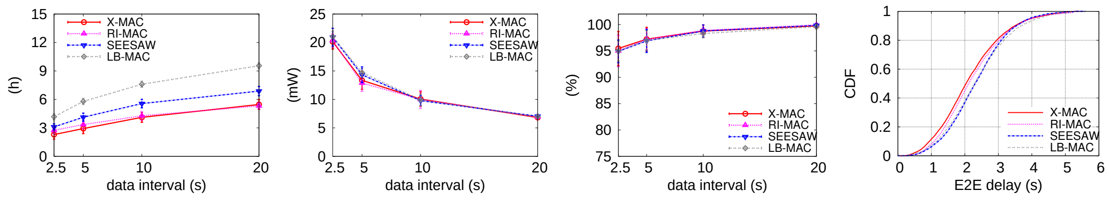
Figure 3.12 Snapshots of residual energy with uniform initial nodal energy.

adopts a set of fixed policies that are not adaptive to changes in network conditions. Besides, SEESAW always relies on senders to initiate communications and its performance is degraded in the presence of channel contention; in contrast, when the channel contention is high, LB-MAC switches from sender-initiated to receiver-initiated rendezvous so that channel contention can be alleviated and more energy can be saved.

Figure 3.11(b) demonstrates that the longer network lifetime yielded by LB-MAC is achieved without increasing the overall energy consumption in the network. Indeed, LB-MAC maintains similar average nodal power consumption as RI-MAC, X-MAC, and SEESAW. Figures 3.11(c) and (d) show that LB-MAC satisfies the end-to-end delay requirement and achieves a high data delivery ratio.

3.6.2.2 Non-uniform Initial Nodal Energy

As the initial nodal energy may be different in practice, we also evaluate LB-MAC under *non-uniform initial nodal energy*. In this case, the designated amount of energy available at each sensor node varies between 200 and 400 Joules. Results plotted in Figure 3.13 show that LB-MAC is able to balance the energy consumption effectively and yield a longer network lifetime.



(a) Network lifetime. (b) Average nodal power consumption. (c) Data delivery ratio. (d) CDF of end-to-end delay.

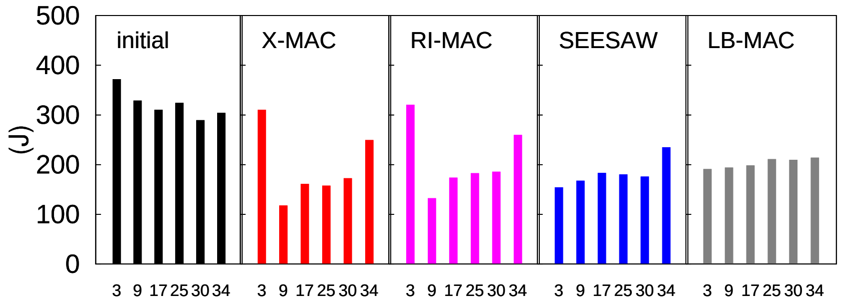
Figure 3.13 Performance comparison with non-uniform initial nodal energy. At data intervals 2.5s, 5s, 10s, and 20s, the best network lifetime performances for X-MAC are achieved under T_r values of 0.6, 0.8, 1, and 1 second, respectively; the best network lifetime performances for RI-MAC are obtained under T_r values of 0.8, 1, 1, and 1 second, respectively.

3.6.2.3 A Trace Study

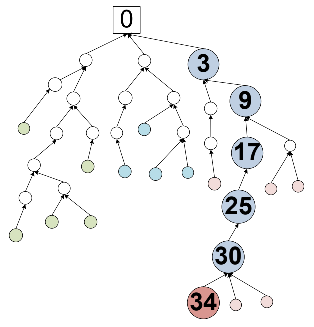
To further illustrate how LB-MAC adaptively tunes the MAC-layer operational parameters to balance the nodal lifetime between neighboring sensor nodes, we examine the experiment that we used to plot the residual energy snapshots in Figure 3.14 in more detail, and plot in Figure 3.15 the changing traces of the operational parameters of the nodes along the path $34 \to 30 \to 25$: T_s, T_r, and ϕ of node 30, and ϕ of node 25. We have the following observations:

- During the time period [0, 0.25h], as shown in Figure 3.15(a), node 30 has a shorter lifetime than both nodes 25 and 34. To balance the nodal lifetime between them, node 25 increases its ϕ to shift communication overhead from node 30 to itself. Correspondingly, node 30 increases its T_s to save energy on transmission and maintain the rendezvous condition. Meanwhile, node 30 also attempts to shift communication overhead to node 34 by first decreasing its ϕ and then increasing its T_r.

- At the time instance of 0.25h, nodes 25 and 30 have reached a similar nodal lifetime. However, as node 30 still has a shorter lifetime than node 34, it continues to shift communication overhead to node 34. As a result, its lifetime continues to increase, resulting in a lifetime imbalance between



(a) Residual energy of nodes 3, 9, 17, 25, 30, and 34 after 2 hours of network operation. Data generation interval at source nodes is 5 seconds.



(b) The studied route from node 34 to the sink node 0.

Figure 3.14 Snapshots of residual energy with non-uniform initial nodal energy.

itself and node 25. This is the reason why node 25 gradually decreases its ϕ during the time period [0.25h, 0.6h].

- Finally, during the time period [0.6h, 2h], as all three nodes have a similar nodal lifetime, both ϕ of node 25 and T_s of node 30 stabilize (to fluctuate within a small range around 20ms) to maintain the lifetime balance between them.

3.6.3 Dynamic Network Settings

In contract to static network settings, we also change the network environments to evaluate LB-MAC under more dynamic and time-varying conditions.

Specifically, the dynamic network environment settings are as follows:

- *Dynamic routing paths*, that is, the network topology is maintained by CTP protocol, and the topology may vary as experiments continue.

- *Dynamic sensing events*, that is, sensing events are assumed to be detected by sensors in one of three sensing areas as illustrated in Figure 3.10. Every certain period, a sensing area will be active and a sensor in that area will generate data packets and forward them hop by hop to the sink.

- *Dynamic packet loss ratios*, that is, the node software will randomly drop data packets at a certain ratio; this way, we emulate the effect of time-varying packet loss ratios caused by different channel conditions.

3.6.3.1 Time-varying Data Generation Rates

Figure 3.16 shows the comparison results when the data generation rates change over time and the packet loss ratio is not arbitrarily adjusted. In this scenario, LB-MAC also produces a significantly longer network lifetime than the state-of-the-art MAC protocols while maintaining similar end-to-end packet delivery delay, delivery ratio, and average nodal power consumption. The results well demonstrate the robustness and effectiveness of LB-MAC in practical scenarios where (i) the routing paths and traffic patterns are time-varying, and (ii) the data sources are temporally and spatially dynamic. In particular, the superiority of LB-MAC over SEESAW can be seen more clearly from the experiments as SEESAW's fixed and empirical policies (for MAC-layer parameter tuning) do not work well with dynamic events while LB-MAC adapts to network dynamics.

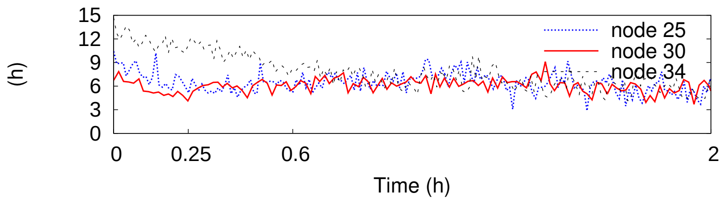
3.6.3.2 Time-varying Packet Loss Ratios

We also evaluate the performance of LB-MAC under time-varying packet loss ratios by letting each sensor node drop packets with certain arbitrary ratios; this way, we emulate the changes of communication conditions in a lab environment. The data generation intervals are 10 seconds in these experiments.

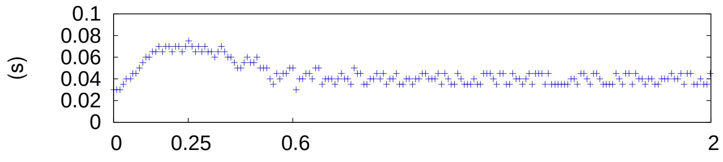
As shown in Figure 3.17(a), when the packet loss ratio is increased, the performance of all evaluated protocols degraded. However, LB-MAC can still yield noticeable lifetime improvement over other protocols.

3.7 Conclusions

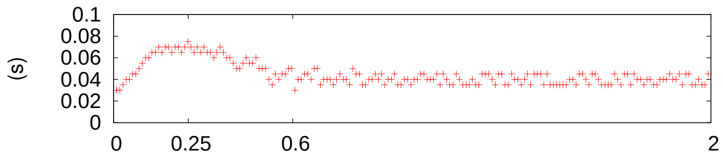
In this chapter, we present a new sensor network MAC protocol, called LB-MAC (Lifetime-Balancing MAC), which is designed from the perspective of network lifetime maximization. LB-MAC emphasizes collaboration between sensor nodes to benefit the network as a whole, even at the expense of a single node. The key idea is that communicating neighbors adjust their MAC-layer behaviors together in a collaborative manner to shift the communication overhead between them. As a result, nodal life-time can be balanced between neighbors and network lifetime can be extended. The effectiveness of the proposed scheme is demonstrated via in-depth experimental results.



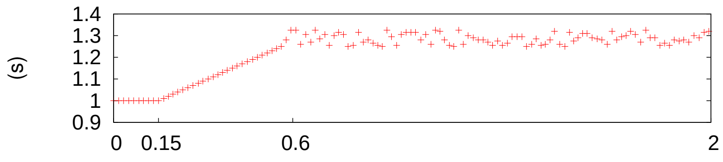
(a) Comparison of nodal lifetime.



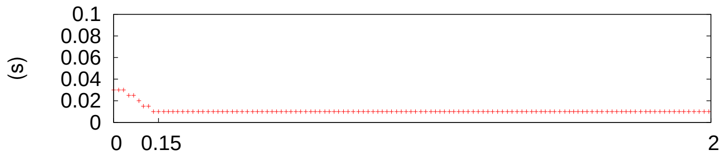
(b) ϕ of node 25.



(c) T_s of node 30.

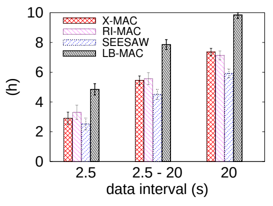


(d) T_r of node 30.

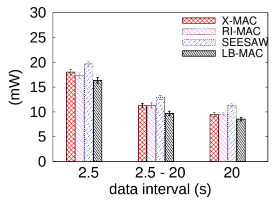


(e) ϕ of node 30.

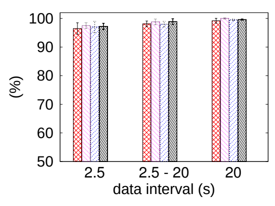
Figure 3.15 Changing traces of T_s, T_r, and ϕ of node 30, and ϕ of node 25 along the path $34 \rightarrow 30 \rightarrow 25$.



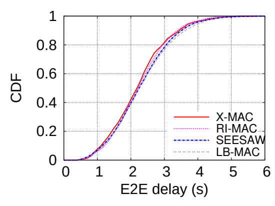
(a) Network lifetime.



(b) Average nodal power consumption.

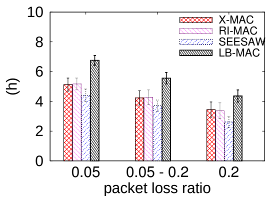


(c) Data delivery ratio.

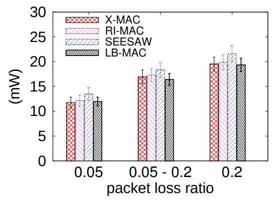


(d) CDF of end-to-end delay.

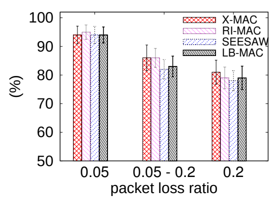
Figure 3.16 Performance comparison with non-uniform initial nodal energy and dynamic sensing events. Data interval “2.5-20” means that data packets are generated at an interval uniformly distributed in [2.5s, 20s]. Data intervals “2.5” and “20” mean that data packets are generated at an interval uniformly distributed with means 2.5s and 20s, respectively; and the deviations are 0.25s and 2s, respectively.



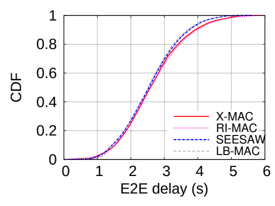
(a) Network lifetime.



(b) Average nodal power consumption.



(c) Data delivery ratio.



(d) CDF of end-to-end delay.

Figure 3.17 Performance comparison with non-uniform initial nodal energy and dynamic packet loss ratios. Packet loss ratio interval “0.05-0.2” means that packets are dropped at a ratio uniformly distributed in [0.05, 0.2]. Packet loss ratios “0.05” and “0.2” mean that packets are dropped at ratios uniformly distributed with means 0.05 and 0.2, respectively; and the deviations are 0.005 and 0.02, respectively.

CHAPTER 4. I^{2}C: A HOLISTIC APPROACH TO PROLONG SENSOR NETWORK LIFETIME

4.1 Introduction

When applying sensor networks for long-term applications such as continuous monitoring, how to prolong the network lifetime is of critical importance. For these applications, *network lifetime* is often defined as the minimal nodal lifetime among all nodes in the network [16–18]. In addition to operating sensor nodes at a low duty cycle to conserve energy, many works have been proposed to approach this goal via balancing the distribution of nodal lifetime in the network.

4.1.1 Motivations

Energy-aware routing and *intra-route coordination* are two nodal lifetime balancing techniques commonly used in sensor networks to prolong the network lifetime. The energy-aware routing schemes [34, 35] attempt to balance the nodal lifetime through distributing more communication workload to routes that contain nodes with longer nodal lifetime and/or higher residual energy. However, as these schemes balance the nodal lifetime through re-routing only, bottleneck nodes such as the nodes close to the sink may still consume more energy than others in the network and thus bound the network lifetime.

Different from energy-aware routing, the intra-route coordination schemes [23, 29, 52] attempt to balance the nodal lifetime of nodes along the same routing path such that the communication workload at the bottleneck nodes can be shifted to other nodes on the same route but with a higher nodal lifetime. Though intra-route coordination can overcome the bottleneck effects efficiently, it may not fully utilize the network energy resources, as it only attempts to balance nodal lifetime within a route but cannot balance the nodal lifetime of nodes belonging to different routes.

Therefore, it is necessary and beneficial to have an integrated scheme which can take advantage of both energy-aware routing and intra-route coordination and meanwhile avoid their limitations. However, without careful analysis and design, simply operating existing energy-aware routing and intra-route coordination schemes together may not provide an efficient solution. For example, as shown in Figure 4.1, the network lifetime achieved by a simple combination of energy-aware routing and intra-route coordination is comparable to that achieved by intra-route coordination alone.

number of nodes	IaC	EA+IaC	I^2C
25	47.1h	43h	60.2h
100	16.5h	18.5h	25.6h

Figure 4.1 Network lifetime comparison between intra-route coordination only (denoted as IaC), a simple combination of energy-aware routing and IaC (denoted as EA+IaC), and our proposed I^2C schemes. The data generation interval is 40 seconds and the number of nodes in the network varies from 25 to 100. These results are extracted from our ns2-based simulation results in Section 4.5.

4.1.2 Contributions

To remedy the deficiencies of either energy-aware routing or intra-route coordination, or a simple combination of the two, we propose a novel holistic approach, called I^2C (*Intra-route and Inter-route Coordination*), which leverages the two lifetime balancing techniques.

The proposed I^2C scheme is composed of two core modules: Intra-Route Coordination and Inter-Route Coordination, which are designed to work together in a collaborative manner. For example, with I^2C, the new parent node of a sensor node may not simply be the one with the highest nodal lifetime (among all potential parent nodes). Rather, it is the one with the maximal potential to increase the minimal nodal lifetime among the node's neighborhood. I^2C accomplishes this by predicting the nodal lifetimes after the potential route switch, via close collaboration between the two modules. Due to such a sophisticated design, I^2C is able to prolong the network lifetime more effectively and efficiently, as shown in Figure 4.1. The contributions of this work are summarized below.

- To the best of our knowledge, I^2C is the first holistic approach which leverages both inter-route (i.e., energy-aware routing) and intra-route lifetime balancing techniques for duty cycle sensor networks.

- I^{2}C is a distributed and lightweight solution. It works through limited control information exchange locally between neighbor nodes.
- I^{2}C has been implemented and evaluated, and it achieves significant improvement on network lifetime over the state-of-the-art solutions.

4.2 Related Work

Among the techniques to prolong the network lifetime, multiple energy-aware routing protocols have been proposed for ad hoc and sensor networks and [18, 34, 35] are representative ones among them. Recently, authors in [16, 36, 37] proposed specially-designed energy-aware routing schemes for duty cycle sensor networks. In all these works, the main idea is to route packets through nodes with a higher residual energy or a longer nodal lifetime such that nodes with a lower energy or a shorter lifetime can participate less in data transmission activities. As a result, the minimum nodal lifetime in the network may be extended and the network lifetime may be prolonged.

Intra-route lifetime balancing, as another approach to prolong the network lifetime, has also been studied in [23, 26, 29, 30, 52]. Particularly, SEESAW [23] was proposed to balance the energy consumption between sender and receiver through adapting the data retry interval at the sender side and the channel checking period at the receiver side. ZeroCal [26] targets at improving the fairness of energy utilization in duty cycle sensor networks by dynamically tuning the nodal wakeup interval. Different from ZeroCal, GDSIC [29] decides the individual nodal wakeup interval through solving distributed convex optimization problems. Though the network lifetime can be prolonged by these schemes, they do not guarantee the end-to-end delay bound. pTunes [30] is a recently proposed centralized solution which adjusts the MAC parameters dynamically for low-power sensor networks. It formalizes a multi-objective optimization problem, in which prolonging network lifetime and guaranteeing the end-to-end delay can be solved together.

In addition to the inter-route and intra-route lifetime balancing schemes, approaches to prolong the network lifetime through cross layer design are proposed in [38–42, 53, 54]. In these works, [38] attempts to maximize the network lifetime via joint routing and MAC design, [42] solves the problem via joint routing and congestion control, and [39] tackles the problem through joint optimal design of physical, MAC, and routing layers in time slotted networks. Though these works can prolong the network lifetime, they either impose high overhead to the system or are not designed in a collaborative manner. More importantly, most of these works are not suitable for duty cycle sensor networks.

4.3 System Model and Design Overview

4.3.1 System Model

We study the problem of prolonging the network lifetime of a sensor network that is configured for long-term monitoring applications. Each node in the network generates and reports sensory data periodically and all nodes form a data collection tree rooted at the sink. The data collection tree is maintained and updated through periodic routing update messages exchanged between neighbor nodes. We do not assume data aggregation in this work.

At the MAC layer, the design principle of our proposed scheme does not require a particular MAC protocol underneath the routing layer. In fact, it works fine with other duty cycle MAC protocols as well, as long as the node's MAC behavior and duty cycle are adjustable [10, 23, 52]. In this work, to simplify the presentation, we assume that each node runs an RI-MAC [10] like protocol as follows. As shown in Figure 4.2, in order to receive a data packet, a node wakes up every T_r interval to interact with potential senders. Upon wakeup, it sends out a beacon and then checks the channel activity for ϕ time for incoming data packets. If a data packet is received within ϕ time, it replies with an ACK; otherwise, it goes back to sleep. On the other hand, if a node has a packet to send, it remains awake and waits idly for the target receiver's beacon to start the data transmission (with a duration of τ). Different from the RI-MAC protocol which has a fixed T_r, we assume that T_r is a tunable MAC layer parameter in this work.

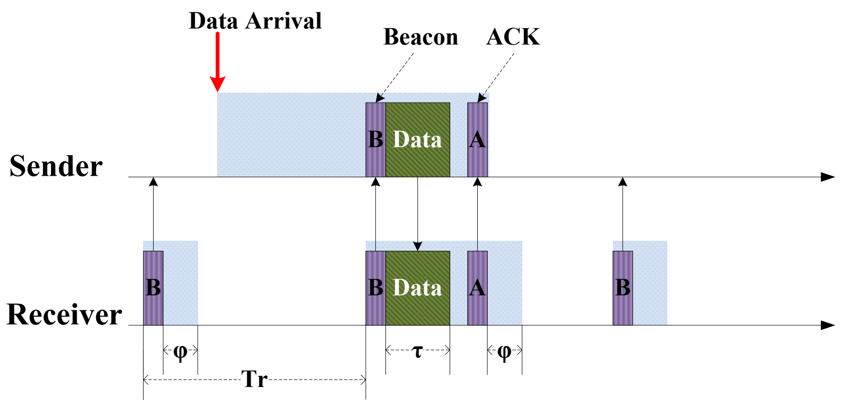


Figure 4.2 An RI-MAC like protocol but with a tunable T_r parameter.

4.3.2 Nodal Lifetime

With the MAC protocol described in the previous section, the nodal lifetime of node i can be estimated as follows:

$$L(i) = \frac{e(i)}{c(i)}, \tag{4.1}$$

where $e(i)$ is the residual energy and $c(i)$ is the energy consumption rate:

$$c(i) = \sum_{j \in \Omega(i)} f(i,j)\left(\tau + \frac{T_r(j)}{2}\right)P + \sum_{k \in \Omega(i)} f(k,i)\tau P + \frac{\phi(i)}{T_r(i)}P. \tag{4.2}$$

Here, $\Omega(i)$ is the set of i's neighbor nodes, $f(i,j)$ is the traffic rate from i to j, and P is the amount of energy consumed when the node's radio is on for one unit of time.

In the above estimation, the short beacon and ACK transmissions are omitted. Therefore, to send a data packet to j, i needs to wait for $\frac{T_r(j)}{2}$ time on average, and the data transmission duration is τ. As a result, it consumes $\sum_{j \in C(i)} f(i,j)\left(\tau + \frac{T_r(j)}{2}\right)P$ power on average for data transmissions. Similarly, the second term in Equation (4.2) represents the average power consumed for data receptions, and the third term is the average power consumed for monitoring the channel activity for ϕ time every T_r interval.

From Equations (4.1) and (4.2), it is interesting to see that nodal lifetime of node i is affected by two factors: (i) the routing behaviors of sensor nodes which decide the outgoing and incoming data rates to i, i.e., $f(i,j)$ and $f(k,i)$; and (ii) the T_r values of i and its receivers, i.e., their MAC behaviors.

4.3.3 End-to-End Delivery Delay

With the MAC protocol described in Section 4.3.1, the worst-case one-hop packet delivery delay from i to j is simply

$$D_{i \to j} = T_r(j). \tag{4.3}$$

Subsequently, the worst-case end-to-end packet delivery delay from a source node to the sink node is

$$\mathcal{D}_{\text{src}\to\text{sink}} = \sum_{\texttt{all hops from source to sink}} D_{i \to j}. \tag{4.4}$$

From Equation (4.4), we can see that, similar to nodal lifetime, the end-to-end packet delivery delay is also affected by two factors: (i) the routing behaviors of sensor nodes which decide the route from source to sink; and (ii) the MAC behaviors of sensor nodes which decide the T_r values.

4.3.4 Problem Statement

From the above analysis, it is clear that, in order to effectively prolong the network lifetime of a sensor network under the end-to-end packet delivery delay constraint, it is critical to have a holistic approach that adjusts both routing and MAC behaviors of sensor nodes together, which is precisely the goal of this work. Formally, it can be described as follows:

Given:

- For each node i, its residual energy $e(i)$, data generation rate $\lambda(i)$, and set of neighbor nodes $\Omega(i)$.

Objective:

- $\max \min L(i)$, where $L(i)$ is the nodal lifetime of i and can be calculated using Equation (4.1).

Subject to:

- *Network Flow Constraint:* for each sensor node i, $\sum_{k \in \Omega(i)} f(k,i) + \lambda(i) = \sum_{j \in \Omega(i)} f(i,j)$.

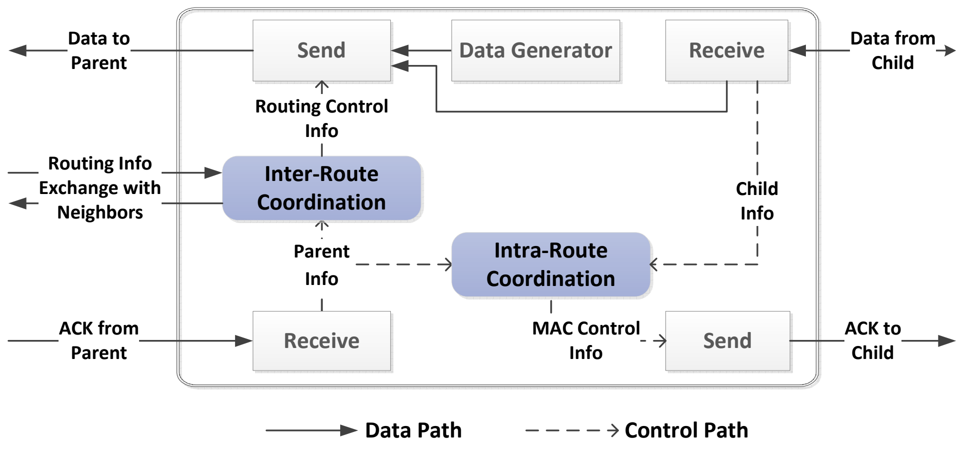


Figure 4.3 Overview of the I^{2}C scheme.

- *End-to-End Delay Requirement:* $\mathcal{D}_{\text{src}\to\text{sink}} \leqslant \mathcal{D}_{\text{e2e}}$ for all source nodes, where $\mathcal{D}_{\text{e2e}}$ is an application-specified delay bound.[1]

- $\forall i, j,\ f(i, j) \geqslant 0$.

- $\forall i,\ T_r(i) > 0$.

Output:

- For each node i in the network, its MAC behavior, i.e., $T_r(i)$, and its routing behavior, i.e., $f(i, j),\ \forall j \in \Omega(i)$.

4.3.5 Design Overview

Directly solving the above optimization problem by individual nodes is impractical because it requires each node to collect the following information from every other node in the network: residual nodal energy, data generation rate, and network topology. Acquiring these information could incur very high communication overhead because of potentially large network scale and dynamic nature of the information. So instead, we propose a distributed, localized, and low-cost solution, called I^{2}C (Intra-route and Inter-route Coordination).

[1]This value can be determined before deployment, or dynamically updated after deployment. In the latter case, the update can be disseminated through sink-to-node communications [46, 55] or piggybacked in a packet and disseminated hop by hop.

In I^{2}C, coordinations only take place between neighbor nodes which exchange lightweight control information and adjust their routing and MAC behaviors together in a collaborative manner. As shown in Figure 4.3, when a parent node receives a data packet from its child node, it extracts the control information (e.g., the expected nodal lifetime) embedded in the data packet and feeds them into the *Intra-Route Coordination* module, which decides how the node shall adjust its MAC behavior (i.e., T_r). It also decides how the child node shall adjust its T_r and piggybacks the decision into the ACK packet to the child node, based on which the child node adjusts its MAC behavior accordingly. This way, the shorter nodal lifetime between parent and child nodes can be extended (at the expense of the other one).

Moreover, a child node may also decide (via the *Inter-Route Coordination* module) to adjust its routing behavior by selecting a different parent node for future communications. With such inter-route coordination, the network lifetime may be extended further as the overall network resource may be utilized more efficiently. For example, the minimal nodal lifetime between the child node, the current parent node, and the new parent node may be extended more (at the expense of the other two). Both coordination modules operate under the condition that the end-to-end delay requirement shall be satisfied. Details of the modules will be elaborated in Sections 4.4.1 and 4.4.2.

4.4 The I^{2}C Scheme

In this section, we describe the details of the two core modules of the proposed I^{2}C scheme: *Intra-Route Coordination* and *Inter-Route Coordination*.

4.4.1 Intra-Route Coordination

The Intra-Route Coordination module coordinates between neighbor nodes on the same route of the current data collection tree. More specifically, it coordinates the MAC behaviors of a pair of parent-child nodes, and adjusts their MAC parameters (i.e., T_r) in a collaborative manner whenever there are data communications between them. I^{2}C achieves this goal by piggybacking lightweight control information in the data/ACK exchanged between parent-child nodes.

Table 4.1 Examples of intra-route coordination with the end-to-end delay requirement of 20 seconds

child node i with $T_r(i) = 1s$		parent node j with $T_r(j) = 1s$			T_r adjustment	
$L(i) = 20h$	$\mathcal{D}_{\text{leaf}\to i} = 10s$	$L(j) = 30h$	$\max_{x\in\Phi(j)-i} \mathcal{D}_{\text{leaf}\to x} = 10s$	$\mathcal{D}_{j\to\text{sink}} = 9s$	$T_r^{\text{new}}(i) = 1.02s$	$T_r^{\text{new}}(j) = 0.98s$
$L(i) = 30h$	$\mathcal{D}_{\text{leaf}\to i} = 10s$	$L(j) = 20h$	$\max_{x\in\Phi(j)-i} \mathcal{D}_{\text{leaf}\to x} = 8s$	$\mathcal{D}_{j\to\text{sink}} = 9s$	$T_r^{\text{new}}(i) = 0.98s$	$T_r^{\text{new}}(j) = 1.02s$
$L(i) = 30h$	$\mathcal{D}_{\text{leaf}\to i} = 8s$	$L(j) = 20h$	$\max_{x\in\Phi(j)-i} \mathcal{D}_{\text{leaf}\to x} = 10s$	$\mathcal{D}_{j\to\text{sink}} = 9s$	$T_r^{\text{new}}(i) = 1s$	$T_r^{\text{new}}(j) = 1s$

4.4.1.1 Parent Node's Behavior

Every T_r interval, a parent node j in the data collection tree turns on radio, sends a beacon, and monitors the channel for ϕ time. During the monitoring period, if a data packet is received from a child node i, the following information will be extracted from the data packet:

- $L(i)$ – *i's estimated nodal lifetime;*
- $T_r(i)$ – *i's MAC parameter;*
- $\mathcal{D}_{leaf\to i}$ – *the maximal delivery delay from the leaf nodes on the data collection subtree rooted at node i to node i.*

By comparing $L(i)$ with its own nodal lifetime $L(j)$, node j attempts to adjust its T_r differently in the two cases discussed below, and then embeds the updated T_r (denoted as T_r^{new}) in the ACK to node i. Note that, according to Equations (4.1) and (4.4), the adjustment of T_r not only affects the nodal lifetime of both parent and child nodes, but the end-to-end delivery delay as well. Therefore, j needs to make sure that the following conditions are satisfied after the T_r adjustment:

$$\begin{cases} \max\limits_{i\in\Phi(j)} \mathcal{D}_{\text{leaf}\to i} + T_r^{\text{new}}(j) + \mathcal{D}_{j\to\text{sink}} \leqslant \mathcal{D}_{\text{e2e}}, \\ T_r^{\text{new}}(i) > 0. \end{cases} \tag{4.5}$$

Here, $\Phi(j)$ is the set of j's children nodes, and $\mathcal{D}_{j\to\text{sink}}$ is the delivery delay from j to the sink, which is maintained locally by j and also embedded in the ACK to i.

Case 1: **$L(j) > L(i)$.** In this case, j decreases $T_r(j)$ by a small amount[2]. Correspondingly, i will increase $T_r(i)$ by the same small amount. This procedure repeats every time when a data packet

[2]In our implementation, we adjust T_r by 20 ms each time. The reason for choosing a small adjustment step is to avoid the potential thrashing effect that may be caused by the following factors: (i) the nodal lifetime estimation may be inaccurate; (ii)

is received, till $T_r(j)$ reaches a default minimal value (which is used to prevent excessive beacon transmissions that may cause severe channel contention). This way, according to Equation (4.1), the time that i waits before transmitting a data packet to j is reduced. As a result, i reduces its energy consumption and consequently increases its nodal lifetime, which is at the expense of node j spending more time on periodic channel checking. Note that, as $T_r^{\text{new}}(j) < T_r(j)$ and $T_r^{\text{new}}(i) > T_r(i)$, both conditions in Equation (4.5) are satisfied after the T_r adjustment.

Case 2: $L(j) < L(i)$**.** In this case, j may increase $T_r(j)$ to reduce its energy consumption for idle listening and increase its nodal lifetime, as long as the conditions in Equation (4.5) are satisfied. This can be guaranteed if $T_r^{\text{new}}(j)$ satisfies:

$$\begin{cases} \mathcal{D}_{j\to\text{sink}} + T_r^{\text{new}}(j) + \max\limits_{y\in\Phi(i)} \mathcal{D}_{\text{leaf}\to y} < \mathcal{D}_{\text{e2e}}, \\ \mathcal{D}_{j\to\text{sink}} + T_r^{\text{new}}(j) + \max\limits_{x\in\Phi(j)-i} \mathcal{D}_{\text{leaf}\to x} \leqslant \mathcal{D}_{\text{e2e}}. \end{cases} \tag{4.6}$$

This is because such $T_r^{\text{new}}(j)$ can always be accommodated by decreasing $T_r(i)$ to:

$$T_r^{\text{new}}(i) = \mathcal{D}_{\text{e2e}} - \mathcal{D}_{j\to\text{sink}} - T_r^{\text{new}}(j) - \max_{y\in\Phi(i)} \mathcal{D}_{\text{leaf}\to y}, \tag{4.7}$$

since we have $T_r^{\text{new}}(i) > 0$ by plugging the first condition in Equation (4.6) into Equation (4.7), and

$$\begin{aligned} & \mathcal{D}_{\text{leaf}\to i}^{\text{new}} + T_r^{\text{new}}(j) + \mathcal{D}_{j\to\text{sink}} \\ = \; & \max_{y\in\Phi(i)} \mathcal{D}_{\text{leaf}\to y} + T_r^{\text{new}}(i) + T_r^{\text{new}}(j) + \mathcal{D}_{j\to\text{sink}} \\ = \; & \mathcal{D}_{\text{e2e}}. \end{aligned} \tag{4.8}$$

Combining Equation (4.8) with the second condition in Equation (4.6), we can see that the end-to-end delivery delay requirement is guaranteed after the T_r adjustment.

Table 4.1 gives three examples to illustrate the parent node's behavior, where the first example corresponds to Case 1, and the second and third examples correspond to Case 2. Take the third example for instance. The parent node j intends to increase its T_r by $20ms$ since $L(j) = 20h < 30h = L(i)$.

multiple nodes may adjust T_r simultaneously; and (iii) the data collection tree varies over time as nodes may join and leave at any time.

Table 4.2 Decision making of the inter-route coordination module

Case	Description			T_r adjustment *if i switches to new parent p*		*Reason*
				$T_r(i)$	$T_r(p)$	
1	$L(p) \leqslant \min(L(i), L(j))$			*Node i shall not switch to new parent p.*		*Switching to p would add more workload to p, thus reducing $L(p)$ and $L_{\min}$.*
2	$L(p) > \min(L(i), L(j))$	$\Delta\mathcal{D} \geqslant 0$	$L(i) < L(p)$	$T_r^{\text{new}}(i) = T_r(i) + \Delta\mathcal{D}$	no change	*Increasing $T_r(i)$ would reduce energy consumed by i for channel checking, which may increase $L(i)$ and $L_{\min}$.*
3			$L(i) \geqslant L(p)$	$T_r^{\text{new}}(i) = T_r(i) + \Delta\mathcal{D}$	no change	*The end-to-end delay requirement prevents $T_r(p)$ from increasing.*
4		$\Delta\mathcal{D} < 0$	$L(i) < L(p)$	no change	$T_r^{\text{new}}(p) = T_r(p) + \Delta\mathcal{D}$	*Since p has a longer lifetime, it sacrifices its lifetime to satisfy the end-to-end delay requirement by reducing $T_r(p)$.*
5			$L(i) \geqslant L(p)$	$T_r^{\text{new}}(i) = T_r(i) + \Delta\mathcal{D}$	no change	*Since i has a longer lifetime, it sacrifices its lifetime to satisfy the end-to-end delay requirement by reducing $T_r(p)$.*

However, Equation (4.6) (more specifically, the second condition in Equation (4.6)) is not satisfied, meaning that the intended increment in $T_r(j)$ would result in a violation of the end-to-end delay requirement of $20s$. Therefore, j instead sticks with the current T_r till the arrival of the next data packet, which leaves the nodal lifetimes between itself and its child node i temporarily unbalanced.

4.4.1.2 Child Node's Behavior

When a child node i has a data packet to send, it turns on radio and waits idly for its parent node j's beacon to start the data transmission. After an ACK is received for the data packet, it extracts the $T_r^{\text{new}}(j)$ information carried in the ACK and simply adjusts its own T_r to:

$$T_r^{\text{new}}(i) = \mathcal{D}_{\text{e2e}} - \mathcal{D}_{j \to \text{sink}} - T_r^{\text{new}}(j) - \max_{y \in \Phi(i)} \mathcal{D}_{\text{leaf} \to y}. \tag{4.9}$$

4.4.2 Inter-Route Coordination

Complementary to the Intra-Route Coordination module, the Inter-Route Coordination module attempts to extend the network lifetime via dynamic adjustment of the data collection tree. Specifically, based on the control information carried in the routing update messages, each sensor node periodically selects the best neighbor as its parent node towards the sink, which maximizes the minimal nodal lifetime between the node, its current parent, and the new parent. This, essentially, decides how the node's communication workload shall be distributed among neighbors. Different distributions of workload may result in different energy consumption rates and hence different nodal lifetimes among neighbors. As such adjustment is conducted by every node in the network, the nodal lifetimes may be balanced gradually across the entire network.

The goal of inter-route coordination can be formally described as follows. Consider node i in the network. Let j denote its current parent. Let $p_1, \cdots, p_n$ denote the set of i's communication neighbors (excluding j). We denote the lifetimes of these nodes as $L(i)$, $L(j)$, and $L(p_1), \cdots, L(p_n)$, respectively. The goal is to find $p^* \in \{p_1, \cdots, p_n\}$ such that

$$\min(L'(i), L'(j), L'(p^*)) > \min(L(i), L(j), L(p^*)), \tag{4.10}$$

and

$$\begin{aligned} &\min(L'(i), L'(j), L'(p^*)) \\ &= \max_{p \in \{p_1, \cdots, p_n\}} \min(L'(i), L'(j), L'(p)), \end{aligned} \tag{4.11}$$

where $L'(i)$, $L'(j)$, and $L'(p)$ are the predicted nodal lifetimes of i, j, and p, assuming that (i) node i selects p as its new parent, and (ii) after the route switch, nodes i and p along the new route behave according to the intra-route coordination principle, which are summarized in Table 4.2 and details are discussed below. If such p^* can be found, i switches to p^* as its new parent; else, it sticks with the current parent j till the next round of routing update.

To aid the inter-route coordination, each node embeds the following control information in the routing update messages: *nodal residual energy (e)*, *nodal energy consumption rate (c)*, *T_r of node itself and its parent node*, and *delivery delay from the node to the sink ($\mathcal{D}_{\text{node}\to sink}$)*. Based on these information, i can predict the nodal lifetime for each of its potential new parent nodes $p \in \{p_1, \cdots, p_n\}$. As listed in Table 4.2, there are five possible cases.

Case 1: $L(p) \leqslant \min(L(i), L(j))$. Node i shall not choose any neighbor node that belongs to this case. This is because, if i switches to p, more workload would be added to p which will decrease the nodal lifetime of p. Therefore,

$$\begin{aligned}\min(L'(i), L'(j), L'(p)) \leqslant L'(p) < L(p) \\ = \min(L(i), L(j), L(p)),\end{aligned} \tag{4.12}$$

meaning that Condition (4.10) is not satisfied.

Case 2: $L(p) > L(i)$ (which implies $L(p) > \min(L(i), L(j))$) and $\Delta\mathcal{D}_{\text{leaf}\to i\to p\to\text{sink}} \geqslant 0$, where $\Delta\mathcal{D}_{\text{leaf}\to i\to p\to\text{sink}} = \mathcal{D}_{\text{e2e}} - \mathcal{D}_{\text{leaf}\to i} - T_r(p) - \mathcal{D}_{p\to\text{sink}}$. In this case, if i would select p as its new parent, its future data packets would be relayed towards the sink by p instead of j. Thus, j's nodal lifetime would be increased to:

$$L'(j) = \frac{e(j)}{c(j) - f(i,j)\left(2\tau + \frac{T_r(j\text{'s parent})}{2}\right)P}, \tag{4.13}$$

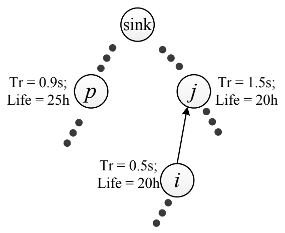
and p's nodal lifetime would be decreased to:

$$L'(p) = \frac{e(p)}{c(p) + f(i,j)\left(2\tau + \frac{T_r(p\text{'s parent})}{2}\right)P}. \tag{4.14}$$

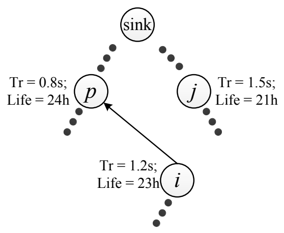
On the other hand, a positive $\Delta\mathcal{D}$ means that i would reach the sink via p with a smaller delay than the required delay bound. This would allow either i or p to increase its T_r (by $\Delta\mathcal{D}$) and consequently the nodal lifetime. As i has a shorter lifetime than p, the intra-route coordination principle would allocate $\Delta\mathcal{D}$ to $T_r(i)$. Therefore, we have

$$L'(i) = \frac{e(i)}{c(i) + \left(f(i,j)\frac{T_r(p)-T_r(j)}{2} - \frac{\Delta\mathcal{D}\cdot\phi}{T_r(i)\cdot(T_r(i)+\Delta\mathcal{D})}\right)P}. \tag{4.15}$$

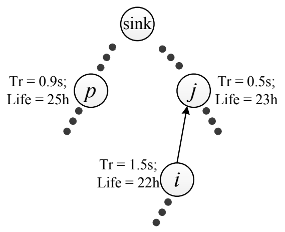
An example is given in Figures 4.4(a) and 4.4(b). In this example, the minimal nodal lifetime between i, j, and p is increased from 20h to 21h after the route switch. However, in general, as $L'(p)$ and $L'(i)$ depend on many factors, there is no definitive relation between $\min(L(i), L(j), L(p))$ and $\min(L'(i), L'(j), L'(p))$ when $L(p) > \min(L(i), L(j))$ (i.e., Cases 2, 3, 4, and 5). Node i would have to plug in the control information carried in the routing update messages from each potential parent, and check whether Condition (4.10) is satisfied.



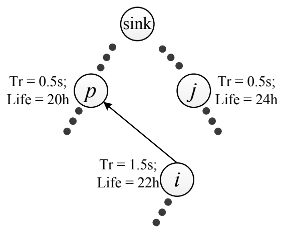
(a) Case 2: before route switch.



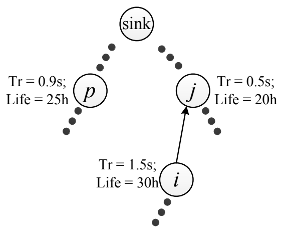
(b) Case 2: after route switch.



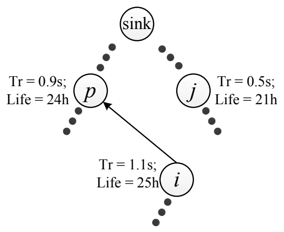
(c) Case 4: before route switch.



(d) Case 4: after route switch.



(e) Case 5: before route switch.



(f) Case 5: after route switch.

Figure 4.4 Examples of inter-route coordination in I^{2}C.

Case 3: $L(i) \geqslant L(p) > L(j)$ (which implies $L(p) > \min(L(i), L(j))$) and $\Delta\mathcal{D}_{\text{leaf}\rightarrow i\rightarrow p\rightarrow\text{sink}} \geqslant 0$. In this case, ideally, p would increase $T_r(p)$ and extend its nodal lifetime. However, as p may have other children nodes, an increase in $T_r(p)$ may result in a violation of the end-to-end delay requirement on other branches of the subtree rooted at p. As a result, we keep $T_r(p)$ unchanged, and allocate $\Delta\mathcal{D}$ to $T_r(i)$ instead. The calculations of the predicated nodal lifetimes are the same as in Case 2.

Case 4: $L(p) > L(i)$ and $\Delta\mathcal{D}_{\text{leaf}\rightarrow i\rightarrow p\rightarrow\text{sink}} < 0$. A negative $\Delta\mathcal{D}$ means that the new route via p towards the sink would incur a higher delay than the desired delay bound. In order to reduce the end-to-end delay to be under the bound, $\Delta\mathcal{D}$ has to be absorbed by either i or p. In this case, as p has a longer nodal lifetime, it would sacrifice its nodal lifetime to accommodate the extra delay by reducing $T_r(p)$. The calculation of $L'(j)$ is the same as in Case 2, while $L'(i)$ and $L'(p)$ may be estimated as follows:

$$\begin{cases} L'(i) = \frac{e(i)}{c(i)+f(i,j)\frac{T_r(p)+\Delta\mathcal{D}-T_r(j)}{2}P} \\ L'(p) = \frac{e(p)}{c(p)+\left(f(i,j)\left(2\tau+\frac{T_r(p\text{'s parent})}{2}\right)-\frac{\Delta\mathcal{D}\cdot\phi}{T_r(p)(T_r(p)+\Delta\mathcal{D})}\right)P} \end{cases} \tag{4.16}$$

An example is given in Figures 4.4(c) and 4.4(d), where $\Delta\mathcal{D} = -0.4s$ is accommodated by p through reducing $T_r(p)$ from $0.9s$ to $0.5s$. As a result, the minimal nodal lifetime between i, j, and p is actually decreased after the route switch. Therefore, i shall not change its parent node in this example.

Case 5: $L(i) \geqslant L(p) > L(j)$ and $\Delta\mathcal{D}_{\text{leaf}\to i\to p\to\text{sink}} < 0$. In this case, as i has a longer nodal lifetime, it will sacrifice its nodal lifetime to accommodate the extra delay by reducing $T_r(i)$. The calculations of the predicted nodal lifetimes are the same as in Case 2. An example is given in Figures 4.4(f) and 4.4(g). In this example, as i has a relatively long nodal lifetime, it successfully accommodates the extra delay incurred by the new route, and improves the minimal nodal lifetime between i, j, and p from 20h to 21h.

4.4.3 Design Discussion

4.4.3.1 Handling of Packet Losses

When the channel condition deteriorates, data or ACK packets may get lost, and the sensor node may need to retransmit multiple times before the data packet can be delivered successfully. As a result, the end-to-end delivery delay may exceed the delay bound. This issue can be dealt with by extending the I^{2}C scheme by including ETX(i, j) – the expected number of transmission attempts to deliver a data packet successfully from i to j – in the design and analysis of the scheme. For example, the end-to-end delivery delay in Equation (4.4) would become

$$\mathcal{D}_{\text{src}\to\text{sink}} = \sum_{\texttt{all hops from source to sink}} T_r(j) \cdot \text{ETX}(i, j). \tag{4.17}$$

This way, a deteriorated channel condition with an increased ETX can be accommodated by reducing the corresponding T_r. Similarly, the lifetime estimation in Equations (4.1) and (4.2) can also be modified to include the ETX information. The value of ETX(i, j) can be estimated based on the periodical exchanges of beacons between neighbors for the routing purpose, as has been implemented in the CTP [51] protocol.

4.4.3.2 Handling of Routing Loops

The Inter-Route Coordination module of the I^2C scheme handles the routing loops as follows. Firstly, when a node chooses a routing parent, any node that currently uses the node as its parent will not be considered. Secondly, when a node detects that the sum of delay from itself to the sink and delay from leaf to itself is larger than the end-to-end delay bound, while these reported delay values keep increasing but with a fixed T_r at its parent node, it considers that a routing loop has been detected; subsequently, the node's current parent node will be blacklisted for several rounds of data transmissions, and a new parent node is selected instead.

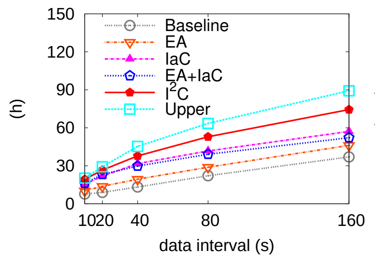
4.4.3.3 Handling of Child Leaving and Joining

After a child node has switched to a different parent node, its previous parent node may keep using the old T_r value that was selected to work with this child node. If this T_r value is small, the parent node wastes energy due to unnecessary short wake up intervals; if this value is large, it may take longer time for a newly joined child node to transmit data packets. In I^2C, each node checks its children nodes periodically to evict stale ones from its children set. When a node becomes a leaf node, it will reset its T_r to the default value.

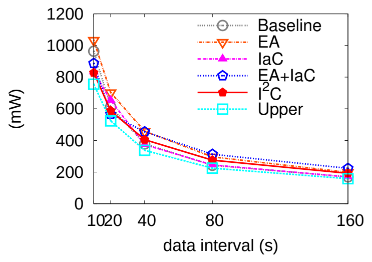
4.5 Performance Evaluation

NS-2 based simulations and TinyOS based testbed experiments have been conducted to evaluate the performance of the proposed I^2C scheme terms of *network lifetime*, *network power consumption*, and *end-to-end delivery delay*. Here, network power consumption is defined as the total amount of energy consumed by the entire network of sensor nodes divided by the network lifetime. We compare the performance of I^2C with the following representative combinations of energy-aware routing and intra-route coordination schemes.

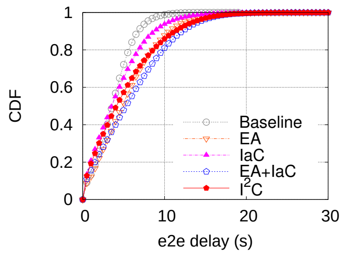
- *CTP + RI-MAC (denoted as "Baseline" in figures):* The routing protocol is a customized CTP (Collection Tree Protocol) [51] which is modified to work in duty cycle networks and is able



(a) Network lifetime.



(b) Network power consumption.



(c) e2e delay.

Figure 4.5 Performance comparison under different data generation intervals with uniform initial nodal energy distribution. The e2e delay requirement is 30 seconds and the total number of nodes in the network is 50.

to satisfy the end-to-end delay requirement when selecting routing paths. The underlying MAC protocol is RI-MAC [10], and in the evaluation, T_r is 2 seconds and ϕ is 25 ms. This combination serves as the baseline scheme in the evaluation.

- *CTP + Intra-route Coordination (denoted as "IaC" in figures):* The routing protocol is the same modified CTP as in the baseline scheme. Intra-route coordination refers to the Intra-Route Coordination module presented in Section 4.4.1 where the MAC parameter T_r is adjusted to balance nodal lifetime between neighbor nodes. This combination evaluates the effectiveness of intra-route coordination only.

- *Energy-Aware Routing + RI-MAC (denoted as "EA" in figures):* In this combination, the energy-aware routing is adopted in the routing layer where each node selects the parent node that has the longest nodal lifetime from its neighbor set. In addition, only the routing paths that satisfy the end-to-end delay requirement may be selected. This combination evaluates the effectiveness of energy-aware routing only.

- *Energy-Aware Routing + Intra-route Coordination (denoted as "EA+IaC" in figures):* This is a simple combination of energy-aware routing and the Intra-Route Coordination module presented in Section 4.4.1. Different from our proposed holistic I^{2}C scheme, energy-aware routing and intra-route coordination simply co-exist in this combination without collaborating with or even being aware of each other.

- *Upper Bound (denoted as "Upper" in figures):* This is the upper bound solution obtained from an NLP solver [56] of the formulation in Section 4.3.4.

4.5.1 Simulation Experiments

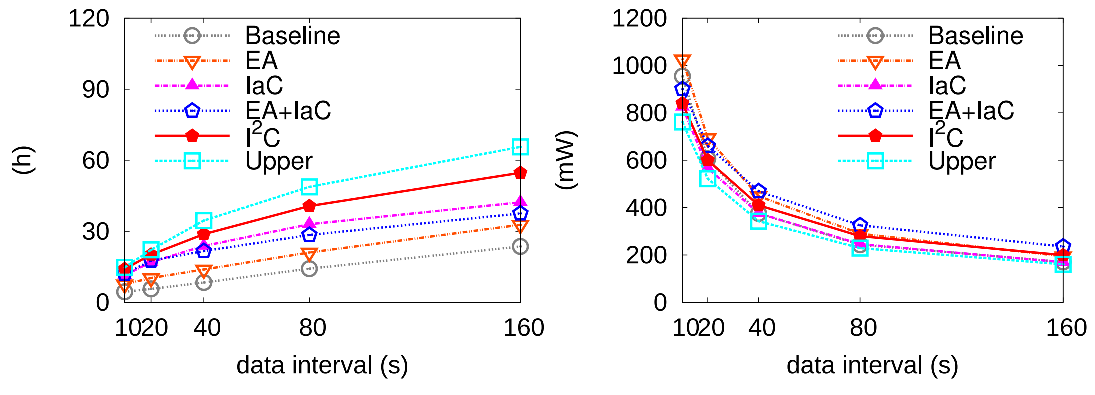
In the simulation, source nodes are randomly deployed in a 500m$\times$500m area and the sink is located at the center of the area. The evaluation results are averaged over results obtained in ten different random topologies.

We vary the data generation interval, the end-to-end delay requirement and the network density under different initial energy distributions. When the initial energy distribution is uniform, the initial nodal energy is full at 1000 Joules; when the distribution is non-uniform, the initial nodal energy is between 500 Joules and 1000 Joules at random. The maximal communication range is 70 meters and the power consumption is 69 mW when radio is on. In both simulations and testbed experiments, the default value of T_r is 2 seconds, the minimal value of T_r is 500 ms, and the routing update interval adopts the default setting in CTP.

4.5.1.1 Performance under Different Data Generation Intervals

Figures 4.5 and 4.6 compare the performances of all the evaluated schemes when the data generation interval at source nodes varies from 10 to 160 seconds.

As shown in Figure 4.5(a), I^{2}C always yields a longer network lifetime than other schemes. Particularly, when the data generation interval is 10 seconds (i.e., heavy workload scenario), I^{2}C extends the network lifetime by about 20% longer than the EA+IaC scheme, and 90% longer than the baseline scheme. When the data generation interval is 160 seconds (i.e., light workload scenario), the improvement on the network lifetime is about 40% over the EA+IaC scheme. The reasons behind the phenomena are explained as follows. The energy-aware routing allows nodes to choose routes of higher level of residual energy, but it may not be able to reduce workload for the bottleneck nodes on selected routes (for example, due to certain topology constraint) and therefore the network lifetime is bounded



(a) Network lifetime.

(b) Network power consumption.

Figure 4.6 Performance comparison with different data generation intervals under non-uniform initial nodal energy distribution. The e2e delay requirement is 30 seconds and the total number of nodes in the network is 50.

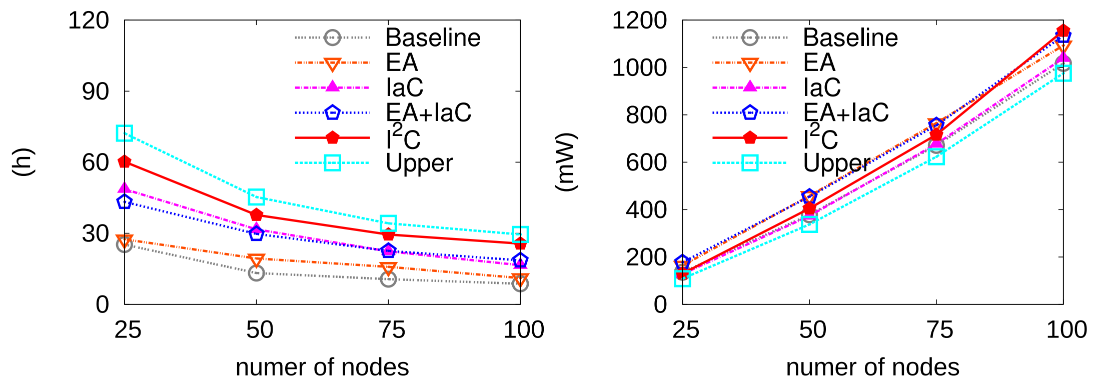
by these nodes. The intra-route coordination, on the other hand, can reduce the workload on the bottleneck nodes through shifting the workload to other nodes on the same route that have a longer nodal lifetime; however, it cannot coordinate the usage of nodes across routes, which constrains its capability in network lifetime prolonging. The above phenomena make it evident the necessity of integrating the two approaches.

A simple combination of the two approaches (i.e., EA+IaC), however, is shown to yield even a lower network lifetime than IaC under certain scenarios. This is because, without the awareness of intra-route coordination, the energy-aware routing protocol simply directs a sensor node to switch to a new parent node with a higher nodal lifetime. This may result in a lower network lifetime after intra-route coordination takes effect between the sensor node and its new parent node. Figure 4.4(c) and (d) in Section 4.4.2 show an example of such scenarios, and explanation can be found in Section 4.4.2, Case 4. On the contrary, the intra-route coordination module of I^{2}C works with an inter-route coordination module that is well aware of intra-route coordination. As a result, I^{2}C inherits the advantages of both approaches and meanwhile mitigates their drawbacks, and therefore is shown to yield a significantly longer network lifetime than other schemes.

Figures 4.5(b) and 4.5(c) demonstrate that I^{2}C does not compromise its performance in other aspects, such as the end-to-end delay and the network power consumption. Due to space limitation, we omit the results of the end-to-end delay for other evaluation scenarios, where all the evaluated schemes satisfy the delay requirement – similar to what has been shown in Figure 4.5(c). Moreover, Figure 4.6 show that I^{2}C also performs consistently better than other schemes under the non-uniform initial nodal energy distribution as well.

4.5.1.2 Performance under Different Network Densities

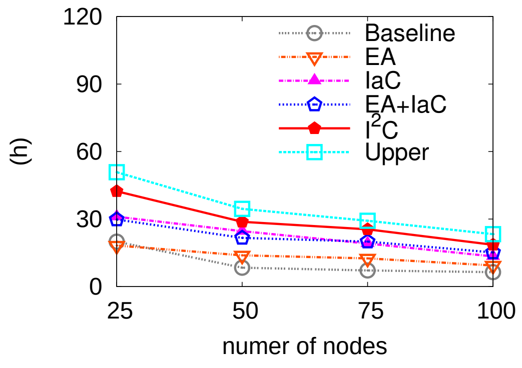
The performance when the network density varies is demonstrated in Figures 4.7 and 4.8. As we can see from these figures, when the network density varies (i.e., the number of nodes in the network changes from 25 to 100), I^{2}C always yields a significantly longer network lifetime than other schemes while maintaining a similar level of network power consumption.



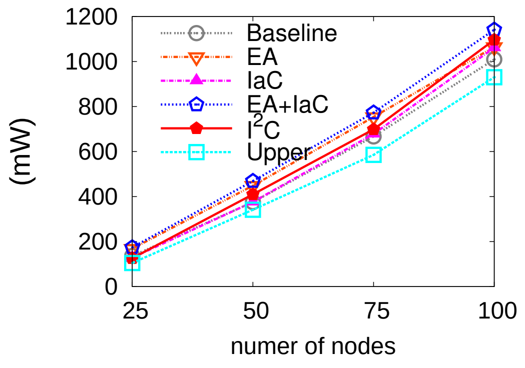
(a) Network lifetime.

(b) Network power consumption.

Figure 4.7 Performance comparison with different network densities under uniform initial nodal energy distribution. The e2e delay requirement is 30 seconds and the data generation interval is 40 seconds.



(a) Network lifetime.



(b) Network power consumption.

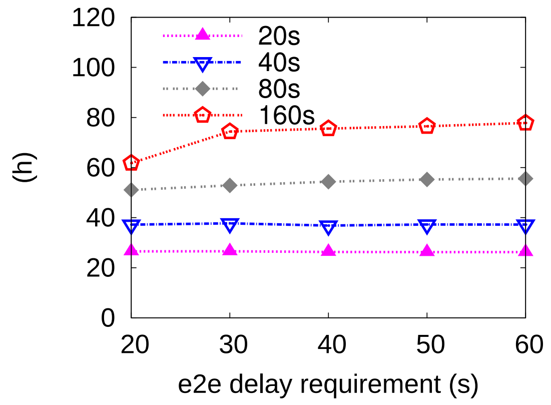
Figure 4.8 Performance comparison with different network densities under non-uniform initial nodal energy distribution. The e2e delay requirement is 30 seconds and the data generation interval is 40 seconds.

4.5.1.3 Performance under Different e2e Delay Requirements

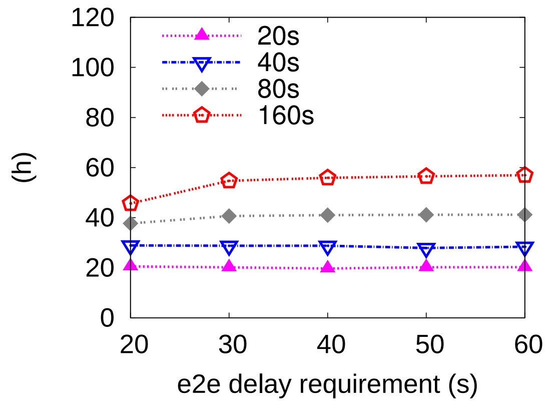
We also evaluate the performance of I^{2}C when both the data generation interval and the end-to-end delay requirement vary.

From Figure 4.9 we can see that, when the data generation interval is short (i.e., 20 seconds), the achieved network lifetime does not change much as the delay requirement increases. This is because, when the network workload is heavy, the energy consumption on data transmissions, rather than the cost on periodic wakeup for data receptions, dominates the nodal energy consumption. In this case, a node can only increase its wakeup interval T_r to a certain value, as too large a T_r value may cause considerably more energy consumption for its children nodes according to the analysis in Equations (4.1) and (4.2) in Section 4.3.2. Consequently, even with a relaxed end-to-end delay requirement, the change of T_r remains small; that is, the opportunity for nodal lifetime balancing brought by the relaxation of delay requirement may not be fully utilized.

On the other hand, when the data generation interval is long (i.e., 160 seconds), the attained network lifetime increases when the end-to-end delay requirement is relaxed. This is because, when the network workload is light, the periodic wakeup and channel checking activities (i.e., $\frac{\phi}{T_r}$ in Equation (4.2)) becomes the dominant factor in nodal energy consumption. Therefore, a node can adjust its T_r in a larger range without causing much overhead on its children nodes' energy cost for data transmissions. This way, the lifetime balancing between parent and children nodes can be conducted more efficiently.



(a) Network lifetime under uniform initial energy distribution.



(b) Network lifetime under non-uniform initial energy distribution.

Figure 4.9 Performance comparison with different e2e delay requirements. The total number of nodes in the network is 50. Different curves correspond to different data generation intervals.

To summarize, ns-2 simulation results clearly demonstrate the consistent performance improvement of I^{2}C over the state-of-the-art solutions on prolonging the network lifetime under various network conditions.

4.5.2 Testbed Experiments

4.5.2.1 Implementation

We have implemented I^{2}C in TinyOS 2.1.0. In our implementation, we modify the following sensor network messages to embed the needed control information. (i) Each *data message* carries a node's lifetime and the longest delivery delay from its leaf nodes to the node itself. (ii) Each *ACK message* carries a node's T_r value and the delivery delay from the node to the sink. (iii) Each *periodic routing update message* carries a node's residual energy and energy consumption rate, the T_r values of the node itself and its parent node, as well as the delivery delay from the node to the sink.

4.5.2.2 Testbed Setup and Evaluation Results

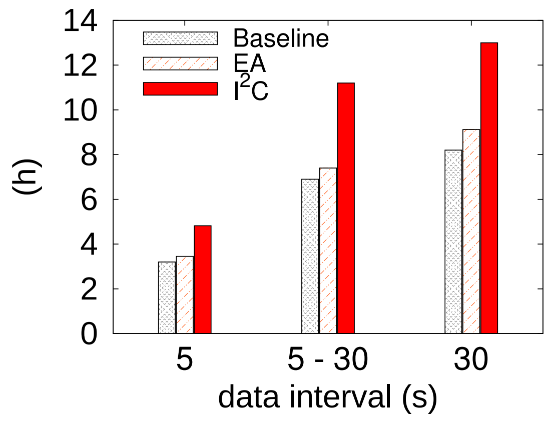
We set up a testbed network of 37 TelosB motes to evaluate the performance of the proposed scheme. In the testbed network, 36 nodes are placed in a 6×6 grid topology where the distance between two adjacent nodes is about 2 meters. All these nodes are source nodes and produce sensory data periodically. An extra node is placed near the upper left corner of the grid; it is connected to a PC and keeps its radio on all the time to serve as the sink. In the experiments, we compare the performance of I^2C with the the Baseline and EA schemes. The end-to-end delivery delay requirement is 30 seconds.

In order to complete the experiments within a reasonable amount of time, we study how fast a node consumes a small designated amount of energy, and evaluate its nodal lifetime as the time period during which the designated amount of energy is consumed. The network lifetime is the minimal nodal lifetime among all sensor nodes. At the beginning of each experiment, the initial nodal energy distribution is uniform or non-uniform. When the distribution is uniform, the initial available energy at an individual node is designated to 400 Joules; when it is non-uniform, the initial available energy at an individual node is designated to a random value between 250 Joules and 400 Joules.

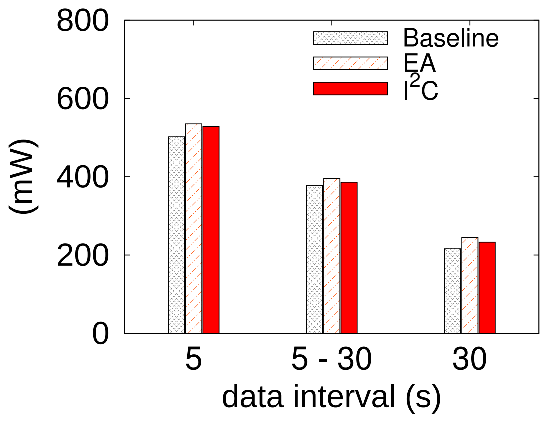
As can be seen from Figures 4.10 and 4.11, in the testbed network, the performance improvement achieved by the EA scheme over the Baseline scheme is limited due to the bottleneck effect. However, I^2C still yields a significant longer network lifetime than both EA and Baseline schemes under different network traffic loads and initial energy distributions.

4.6 Conclusions

In this chapter, we present I^2C – a new holistic approach to prolong the sensor network lifetime. I^2C is composed of two collaborative modules: intra-route coordination and inter-route coordination modules. Different from most of the existing works which conduct either intra-route or inter-route lifetime balancing alone, I^2C leverages and integrates the advantages of both approaches and therefore can prolong the network lifetime more efficiently. In addition, I^2C can also meet the end-to-end delay requirement specified by the applications. Extensive simulation and testbed experiments have been



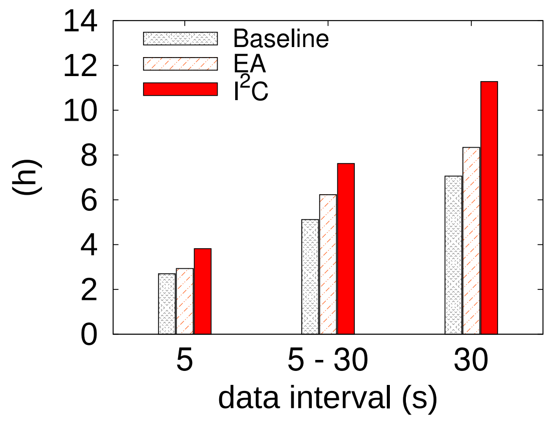
(a) Network lifetime.



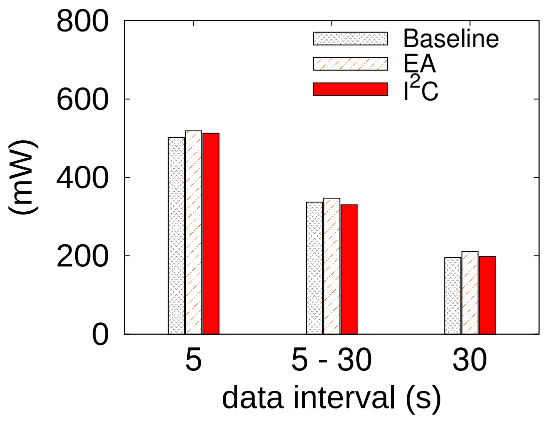
(b) Network power consumption.

Figure 4.10 Experiment results with different data generation intervals under uniform initial nodal energy distribution. Data interval "5-30" means that data packets are generated at an interval uniformly distributed in [5s, 30s].

conducted, and the evaluation results show that I^{2}C can significantly prolong the network lifetime than the state-of-the-art solutions.



(a) Network lifetime.



(b) Network power consumption.

Figure 4.11 Experiment results with different data generation intervals under non-uniform initial nodal energy distribution. Data interval “5-30” means that data packets are generated at an interval uniformly distributed in [5s, 30s].

CHAPTER 5. J-ROS: A JOINT ROUTING AND SENSING SCHEME TO PROLONG SENSOR NETWORK LIFETIME

5.1 Introduction

When a wireless sensor network is deployed for long-term continuous monitoring, it is essential to keep its lifetime as long as possible. Hence, extending network lifetime has been an important research topic.

There have been various definitions of network lifetime proposed in the literature [57–62]. Defining it as the earliest time when any one node of the network dies [57, 58] is simple but widely adopted. However, the definition is not realistic because sensor nodes are usually deployed with high level of redundancy in practice. Therefore, a network lifetime ends only when the death of sensor nodes cannot guarantee a certain level of application-required sensing coverage or the connectivity of all nodes assigned with sensing duties [61, 62].

A large number of schemes [23, 26, 29, 34, 35, 40–42, 63, 64] have been proposed to extend the network lifetime in terms of the above simple definition. Balancing nodal residual energy, lifetime, or energy consumption rates are the common techniques adopted by these schemes. However, very few works have been reported on how to effectively extend the network lifetime in terms of the more practical definition. This study aims to fill this blank.

Specifically, we first formulate the problem and develop a centralized solution to find the upper bound of network lifetime. As a centralized scheme is infeasible for large-scale sensor networks, we further develop a distributed scheme, called J-RoS, to jointly schedule both routing and sensing activities in the network. In a nutshell, the distributed scheme works as follows. Initially, a routing tree is constructed to connect all nodes for sensory data collection, and nodes are assigned with sensing duties to meet sensing coverage requirements. The tree construction and sensing duty assignment are conducted in an energy-aware manner to make nodes with higher levels of residual energy to take more communication and sensing workloads than other nodes. After the initialization completes, the sensing duty assignment and the routing tree are continuously adjusted in a local and gradual manner during the rest of the network lifetime. The purpose of the adjustments is to dynamically adapt the sensing and routing activities to the changes in system conditions (e.g., distribution of nodal residual energy and lifetime), so as to maintain as long network lifetime as possible. Also, the locality nature of adjustments introduces only low communication overhead. The periodical adjustments of sensing duties and collection tree structure are based the following heuristics: First, nodes that are not critical to meet the sensing and connectivity requirements should be scheduled to take more sensing and communication duties even at the cost of depleting their energy supplies quickly, in order to reduce the workloads of nodes that are critical to meet the sensing and connectivity requirements. Second, nodal lifetime should be balanced among the critical nodes to avoid the scenarios where the network lifetime ends because of the death of a small number of critical nodes while other critical nodes still have plenty of residual energy.

Extensive simulations have been conducted to evaluate the performance of the proposed distributed scheme, and compare it with that of the ideal upper bound solution, a nodal lifetime-balancing scheme and a naive scheme. The results show that, our scheme can significantly outperform the balancing and naive schemes, and it achieves a performance close to the upper bound.

5.2 Related Work

With different definitions of network lifetime, there has been a large variety of different techniques proposed to prolong network lifetime. In this section, we first summarize the related works on network lifetime definitions and then those on lifetime extension techniques.

Definitions of network lifetime Among the definitions comprehensively discussed in [65], the most widely used one is “the time until the first sensor is drained of its energy” [57, 58], which assumes all nodes in the network to be equally critical. Taking sensing coverage as the major criterion, network lifetime can be defined as the first time when a monitored target or area cannot be sensed with a certain required fidelity, such as k-coverage [59] and α-coverage [60]. Taking also network connectivity into consideration, network lifetime [61] can be defined as the first time when either the network connectivity or the coverage ratio drops below a certain threshold. Similarly, the network lifetime definition adopted by J-RoS is also based on the requirements of both network connectivity and "quality of monitoring" [62].

Techniques to prolonging network lifetime Numerous schemes have been proposed to prolong the lifetime of sensor networks. Among them, energy-aware routing protocols [16, 18, 34–37] route packets through nodes with higher residual energy or longer nodal lifetimes such that nodes with lower residual energy or shorter nodal lifetimes can live longer by participating less in data transmission. MAC layer techniques [23, 26, 29, 30, 52] dynamically tune parameters such as channel checking period, data retransmission interval, etc., under application-specified constraints, to adjust the distribution of communication overhead over different nodes with the purpose of prolonging the whole network lifetime. Besides routing or MAC layer protocols, cross-layer solutions [38–42, 53, 54, 63, 64] have also been proposed. For example, [38–40] attempt to maximize the network lifetime via joint routing and MAC, joint routing and congestion control, and joint optimal design of physical, MAC and routing, respectively. Recently, Peng et al. [63, 64] propose new cross-layer protocols, namely, I^2C - joint routing and MAC protocol and JAM - joint data aggregation and MAC protocol, enabling neighboring nodes to collaborate locally to extend the lifetime of duty cycle sensor networks.

All the afore-discussed schemes are proposed for networks in which all nodes are evenly critical and network lifetime is defined as the first time a node dies. Hence, balancing nodal residual energy, lifetime, or energy consumption rate is one of the essential techniques of all these works. Differently, J-RoS is unique in that it is designed with the awareness of node redundancy in network and with the more general network lifetime definition as the first time a required level of sensing coverage or connectivity fails. Therefore, novel techniques different from balancing have been developed.

5.3 Analysis

5.3.1 System Model

We consider a sensor network of one sink and N sensor nodes. The network is deployed to monitor M non-overlapped areas, where there are n_i sensor nodes within each area i ($i = 1, \cdots, M$), and each node is assumed to know the area it is deployed to. Required by a certain application, each area i should be α_i-covered; that is,

$$\sum_{j=1}^{n_i} S_j \geqslant \alpha_i, \tag{5.1}$$

where S_j ($0 \leqslant S_j \leqslant 1$) is the sensing duty cycle assigned to a node j of area i (Note: Here we assume nodes $1,\ 2,\ \cdots,\ n_i$ are in area i without loss of generality.), and it means that node j should be active in sensing for time period S_j every time unit. In order to deliver sensory data to the sink, at any time, all the alive nodes in network shall form a tree rooted at the sink to pass sensory data upwards from leafs to the root. Hence, the *network lifetime is defined as the earliest time when the sensing coverage requirement cannot be satisfied in any individual area, or any sensor node j with $S_j > 0$ does not have a path to forward its sensory data to the sink.*

Notations used in this chapter are summarized in Table 5.1.

5.3.2 Problem Statement

Formally, the problem studied in this chapter can be presented in a time-discrete manner as follows.

Objective:

- $\max\{T\}$

Given:

- θ_{rx}, θ_{tx}, θ_s, ϵ, and β
- For each area i: α_i
- For each node k: e_k, $\mathcal{C}_k$ and $\mathcal{P}_k$

Subject to:

Table 5.1 Notation summary

notation	meaning
S_i	sensing duty cycle of node i
α_i	sensing coverage requirement of area i
β	number of sensing samples generated per time unit by a node with 100% sensing duty cycle
θ_{rx}	energy consumed to receive one sensing sample
θ_{tx}	energy consumed to transmit one sensing sample
θ_s	energy consumed to collect one sensing sample
ϵ	energy consumed per time unit when a node is alive without performing sensing or communication duties
$f_{i\to j}$	number of sensing samples transmitted from node i to node j
$\mathcal{C}_i$	set of possible child nodes of node i
$\mathcal{P}_i$	set of parent candidate nodes of node i
$\mathcal{T}_i$	subtree rooted at node i
e_i	current residual energy of node i
c_i	current energy consumption rate of node i
L_i	current lifetime of node i
$\grave{\mathcal{L}}_i$	the lowest nodal lifetime in node i's subtree
$\acute{\mathcal{L}}_i$	the lowest nodal lifetime on the path from i to sink
$\mathcal{W}_i$	estimated energy waste in subtree rooted at node i
$\mathcal{R}_i$	ratio of wasted energy out of total consumed energy in subtree rooted at node i
$\mathcal{E}_i$	current residual energy in the subtree rooted at node i
λ_i	current energy consumption rate in the subtree rooted at node i

- Sensing Coverage Constraint:
  $\sum_{j=1}^{n_i} S_j(t) \geqslant \alpha_i$, for each area i, $t \in \{0, \dots, T\}$
  $1 \geqslant S_j(t) \geqslant 0$, $t \in \{0, \dots, T\}$

- Network Flow Constraint:

  $$\sum_{j\in\mathcal{P}_i} f_{i\to j}(t) = \sum_{k\in\mathcal{C}_i} f_{k\to i}(t) + S_i(t)\cdot\beta,\ t \in \{0, \dots, T\}$$

- Connectivity Constraint:

$$e_i(t) \geqslant S_i(t)\cdot\beta\cdot\theta_s + \sum_{j\in\mathcal{P}_i} f_{i\to j}(t)\cdot\theta_{tx} + \sum_{k\in\mathcal{C}_i} f_{k\to i}(t)\cdot\theta_{rx} + \epsilon,$$

$t \in \{0, \ldots, T\}$

Outputs:

- For each node i,
  - $S_i(t)$: its sensing duty, $t \in \{0, \ldots, T\}$
  - $f_{i\to j}(t)$: its outgoing traffic rate to any parent node j, $t \in \{0, \ldots, T\}$

Directly solving the above problem is difficult, as a large set of information about each child that might change dynamically needs to be collected. In addition, to distribute the solution to individual nodes throughout the network will impose a high communication cost. Hence, we analyze the upper bound performance of the problem and design a distributed heuristic scheme to solve the problem.

5.3.3 Upper Bound Performance Analysis

If ϵ is ignored in the connectivity constraint, it can be relaxed to:

$$e_i(t) \geqslant S_i(t) \cdot \beta \cdot \theta_s + \sum_{j \in \mathcal{P}_j} f_{i\to j}(t) \cdot \theta_{tx} + \sum_{k \in \mathcal{C}_i} f_{k\to i}(t) \cdot \theta_{rx},$$

and the upper bound value of T can be calculated using an Non-Linear Problem solver. However, because the number of variables and constraints might be significantly large as the increase of T and the number of nodes or links in network, an NLP solver [56] might not be able to obtain a solution within a reasonable period of time.

To further reduce the size of variable and constraint sets, let $\overline{S}_i = \frac{\sum_{t=0}^{T-1} S_i(t)}{T}$, $\overline{f}_{i\to j} = \frac{\sum_{t=0}^{T-1} f_{i\to j}(t)}{T}$, and $\overline{f}_{k\to i} = \frac{\sum_{t=0}^{T-1} f_{k\to i}(t)}{T}$, we can get an amortized version of the problem without changing the given inputs as follows:

Objective:

- $\max\{T\}$

Subject to:

- $\sum_{j=1}^{n_i} \overline{S}_j \geqslant \alpha_i, 1 \geqslant \overline{S}_j \geqslant 0$

- $\sum_{j \in \mathcal{P}_i} \overline{f}_{i \to j} = \sum_{k \in \mathcal{C}_i} \overline{f}_{k \to i} + \overline{S}_i \cdot \beta$
- $T \leqslant \frac{e_i}{\overline{S}_i \cdot \beta \cdot \theta_s + \sum_{j \in \mathcal{P}_i} \overline{f}_{i \to j} \cdot \theta_{tx} + \sum_{k \in \mathcal{C}_i} \overline{f}_{k \to i} \cdot \theta_{rx}}$

Output:

- For each node i,
  - $\overline{S}_i$: its average sensing duty cycle
  - $\overline{f}_{i \to j}(t)$: its average outgoing traffic rate to any parent node j

Till now, we have reduced the variable and constraint sets and changed the problem from time-discrete to time-continuous formulation, and the problem can be solved using an NLP solver easily. This upper bound value is used for comparison when evaluating J-RoS in Section 5.5.

5.4 J-RoS Design

In this section, we present J-RoS (Joint Routing and Sensing), a distributed and low-cost solution to jointly schedule routing and sensing activities in sensor networks.

5.4.1 Design Overview

J-RoS is designed to prolong the network lifetime, which is defined as the earliest time when the sensing coverage requirement cannot be satisfied in an area or a node assigned with sensing duty is disconnected from the network and cannot forward its sensory data to the sink. In general, the scheme works as follows:

- Initially, a routing tree rooted at the sink is constructed to connect all nodes for sensory data collection, and nodes are assigned with sensing duties to meet sensing coverage requirements in every monitoring area. Here, the tree can be constructed using an energy-aware routing protocol [18,34], such that nodes with higher residual energy take more communication workload than those with lower residual energy. The assignment of sensing duties also follows an energy-aware approach to make nodes with higher residual energy to take more sensing duties than others.

- After the initialization completes, the sensing duty assignment and the routing tree should be continuously adjusted in a local and gradual manner during the rest of the network lifetime. The purpose of the adjustments is to dynamically adapt the sensing and routing activities to the changes in system conditions (e.g., distribution of nodal residual energy and lifetime), so as to maintain as long network lifetime as possible. Also, the locality nature of adjustments introduces only low communication overhead.

The key ideas of the dynamic adjustments are further explained in the following.

5.4.1.1 Dynamic Adjustment of Sensing Duties

Every time when communication occurs between a pair of parent-child nodes, the parent needs to check whether there is an adjustment of the sensing duties assigned to its children and itself that can help extend the network lifetime. If such an opportunity is found, the adjustment is carried out.

More specifically, the checking starts with identifying *critical nodes*, which are defined as the nodes whose death or disconnection from the current routing tree can cause:

- violation of sensing coverage requirement in an area, or
- disconnection of nodes needing to perform sensing duties from the routing tree.

In other words, the network lifetime terminates as one critical node dies or gets disconnected. As shown in Figure 5.1, nodes 1, 2, 5, 8, 9 and 17 are critical nodes if 2-sensing coverage is required for every area. Then, the opportunity is sought to adjust the sensing duty assignment to:

- prolong the minimal nodal lifetime of critical nodes, or
- improve energy utilization efficiency of non-critical nodes.

Note that, prolonging the minimal nodal lifetime of critical nodes can be accomplished through shifting workload from critical nodes to non-critical nodes or from minimal-lifetime critical nodes to longer-lifetime critical nodes. Improving the energy utilization efficiency of non-critical nodes may not immediately extend the network lifetime (as the minimal nodal lifetime of critical nodes is not extended

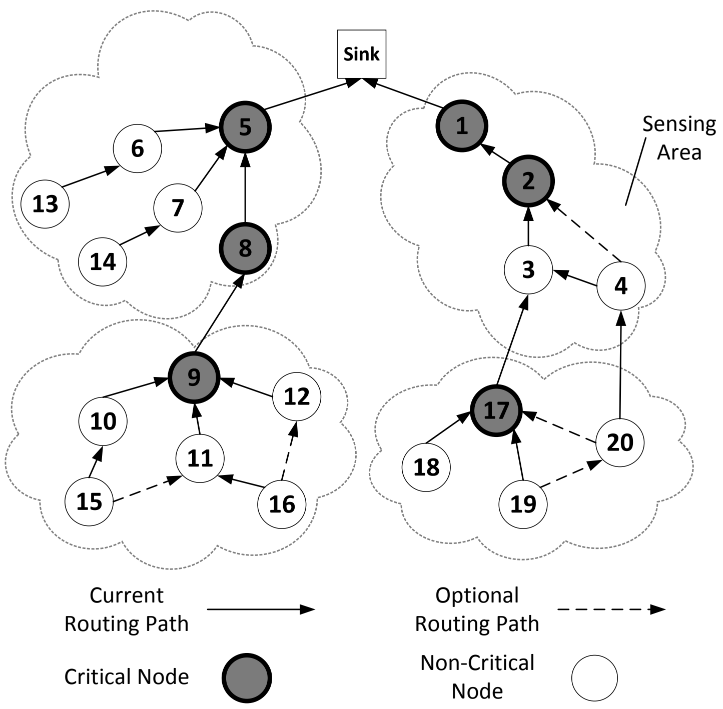


Figure 5.1 A network with critical nodes. If 2-sensing coverage is required for every area, nodes 1, 2, 5, 8, 9 and 17 are critical. Particularly, the death of node 1 or 2 can either violate the 2-sensing coverage in the area or fail the forwarding of sensory data from other nodes of the same area; nodes 5, 8 and 9 are critical because depletion of any of them can fail the forwarding of sensory data generated in the area of nodes 9, 10, 11, 12, 15 and 16.

immediately), but it can extend the overall lifetime of all non-critical nodes and therefore has the potential to delay the moment when the non-critical nodes die and hence their workload has to be completely shifted to critical nodes.

5.4.1.2 Periodical Update of Routing Tree

Every certain time interval, each alive node also needs to check whether its change of parent node can help extend the network lifetime; if such an opportunity is found, the routing adjustment is carried out. Similar to the adjustment of sensing duties, routing is adjusted only if the adjustment can prolong the minimal nodal lifetime of critical nodes or improve energy utilization efficiency of non-critical nodes.

In the rest of this section, we will elaborate the updating of sensing schedules and routing, respectively.

5.4.2 Dynamic Updating of Sensing Duties

After the system initialization has completed, each node maintains the assigned sensing duties of itself and all its child nodes. These sensing schedules may be changed when the node receives a data packet from its child node, as detailed in the following.

5.4.2.1 Parent Node Behavior

Upon receiving a packet from its child node, the receiver acts based on its collected information about all its child nodes that are in the same area as itself. Particularly, the following information is extracted from each data packet received from each child node i: (1) total residual energy in node i's subtree (denoted as $\mathcal{E}_i$), (2) total energy consumption rates in node i's subtree (denoted as λ_i), (3) the shortest nodal lifetime in node i's subtree (denoted as $\grave{\mathcal{L}}_i$), (4) the estimated amount of energy that will be wasted in node i's subtree (denoted as $\mathcal{W}_i$), and (5) node i's lifetime L_i.

Here, supposing node j is the shortest-lifetime node in the subtree rooted at i, $\mathcal{W}_i$ refers to the total residual energy in the subtree rooted at j when j dies. For example, in Figure 5.2(b), node 9 has the shortest lifetime in node 8's subtree; therefore, $\mathcal{W}_8$ is computed as the amount of residual energy when node 8 has used up its energy. Formally, if node i is the shortest-lifetime node in its subtree:

$$\grave{\mathcal{L}}_i = L_i \tag{5.2}$$

and

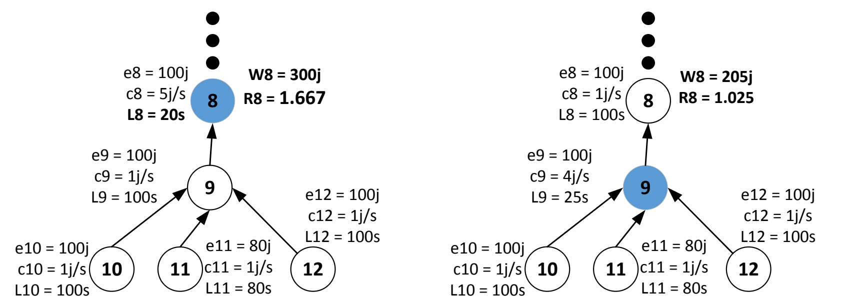
$$\mathcal{W}_i = \mathcal{E}_i - \grave{\mathcal{L}}_i \cdot \lambda_i; \tag{5.3}$$

otherwise,

$$\grave{\mathcal{L}}_i = \min\{L_j\}, j \in \mathcal{T}_i \tag{5.4}$$

and

$$\mathcal{W}_i = \mathcal{W}_x, \tag{5.5}$$



(a) Node 8's lifetime is the shoftest in the subtree rooted at it. E8 = e8+e9+e10+e11+e12 = 480j, λ8 = c8+c9+c10+c11+c12 = 9j/s, W8 = E8 – L8*λ8 = 300j, and R8 = W8/(L8*λ8) = 1.667.

(b) Node 9 is the lowest lifetime node in node 8's subtree: E9 = e9+e10+e11+e12 = 380j, λ9 = c9+c10+c11+c12 = 7j/s, and W9 = E9 – L9*λ9 = 205j. Node 8's lifetime is not the shortest in its subtree: E8 = e8+e9+e10+e11+e12 = 480j, λ8 = c8+c9+c10+c11+c12 = 8j/s, W8 = W9 = 205j, R8 = W8/(L9*λ8) = 1.025.

Figure 5.2 Examples of how $\mathcal{W}$ and $\mathcal{R}$ are computed.

where $x = \arg\min_{j \in \mathcal{T}_i}\{L_j\}$.

$\grave{\mathcal{L}}_i$ and $\mathcal{W}_i$ are used to compute the ratio of wasted energy over total consumed energy (denoted as $\mathcal{R}$), which serves as an indicator of energy utilization efficiency for nodes in node i's subtree to perform sensing duties:

$$\mathcal{R}_i = \frac{\mathcal{W}_i}{\grave{\mathcal{L}}_i \cdot \lambda_i}. \tag{5.6}$$

Figure 5.2 uses two examples to show how a node computes its $\mathcal{W}$ and $\mathcal{R}$.

Knowing $\mathcal{R}_i$, lifetime L_i and criticality of each child node i, parent node can thus re-schedule sensing duties as follows:

- Select two nodes, n_s and n_d, out of the parent node itself and all child nodes in its area.
- Move sensing duties from source node n_s to destination node n_d.

More specifically, Table 5.2 shows how nodes n_s and n_d should be selected.

- Case 1: All nodes are critical. In this case, all child nodes are equally important for sensing coverage. Therefore, J-RoS employs the lifetime balancing strategy by moving sensing duties from the node who has the shortest lifetime to the one with the highest lifetime. This way, the shortest nodal lifetime can be improved and the sensing coverage period can be maintained longer.

Table 5.2 Selection table in sensing scheduling

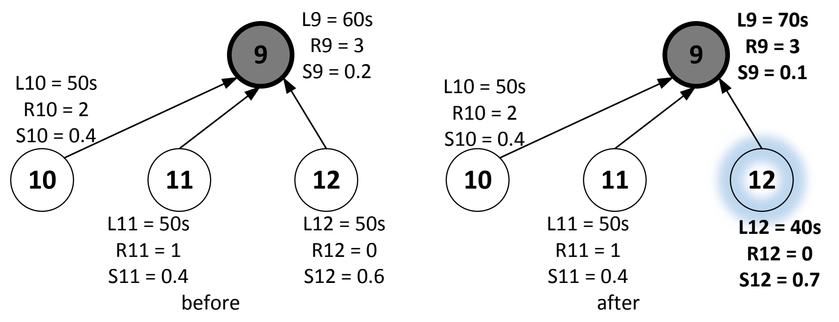
Case	Condition	Selection
1	All nodes are critical	n_s and n_d are the shortest-lifetime and the longest-lifetime ones among all the nodes, respectively
2	All nodes are non-critical	n_s and n_d are nodes who have the highest and the lowest values of $\mathcal{R}$, respectively
3	There exist both critical and non-critical nodes	n_s is the critical node who has the shortest lifetime and n_d is the non-critical node who has the lowest value $\mathcal{R}$

- Case 2: All nodes are non-critical. In this case, it is unnecessary to protect some node from depletion, as the sensing coverage won't be affected even if some node runs out of energy earlier than others. According to the design principles, how to improve energy utilization efficiency becomes the top priority of sensing activity scheduling. Therefore, sensing duties are moved from the node who has the highest value of $\mathcal{R}$ (i.e., lowest efficiency) to the node who has the lowest value of $\mathcal{R}$ (i.e., highest efficiency).

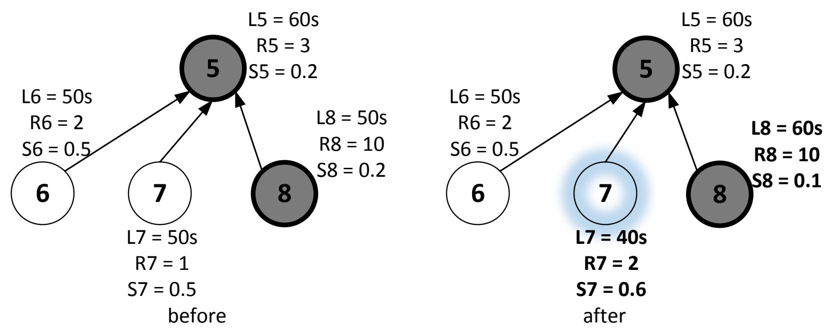
- Case 3: There exist both critical and non-critical nodes. In this case, sensing duties of those critical nodes should be reduced and some non-critical node may have increased sensing duties as a result. Similar to case 2, J-RoS first selects the non-critical node who has the lowest value of $\mathcal{R}$, and then moves sensing duties from the shortest-lifetime critical node to this non-critical node. After performing this change, the sensing coverage period can be prolonged and the energy utilization efficiency can be improved at the same time.

Note that, in both cases 2 and 3, if there are more than one node with the same lowest value $\mathcal{R}$, sensing duties should be shifted to the one with the shortest nodal lifetime to let the shortest-lifetime node die earlier and hence consumes less fixed energy cost. Figure 5.3 uses two examples to show how a parent node adjusts sensing duties between itself and child nodes.

After updating sensing duties for each child in its area, the parent node will update the sensing schedules that it maintains for each child, and an updated sensing schedule will be embedded into the ACK sent to a child node when the parent node communicates with the child next time.



(a) α=2, nodes 10, 11, and 12 are non-critical, and node 9 is critical. Sensing duty is moved from node 9 to node 12 (lowest *R* value).



(b) α=2, nodes 6 and7 are non-critical and nodes 5 and 8 are critical. Sensing duty is moved from node 8 (critical with shortest-lifetime) to node 7 (non-critical with lowest *R* value).

Figure 5.3 Examples of sensing duty adjustment between parent and child nodes.

5.4.2.2 Child Node Behavior

To facilitate a parent node's scheduling behavior, each child node i needs to compute the value of $\mathcal{E}_i$, λ_i, $\mathcal{W}_i$ and $\grave{\mathcal{L}}_i$ before sending a data packet. In particular, $\mathcal{E}_i$ and λ_i are computed as:

$$\mathcal{E}_i = \sum_{j \in \mathcal{C}_i} \mathcal{E}_j + e_i \tag{5.7}$$

and

$$\lambda_i = \sum_{j \in \mathcal{C}_i} \lambda_j + c_i, \tag{5.8}$$

where e_i is the current residual energy of node i and c_i is the node's current energy consumption rate. $\mathcal{W}_i$ and $\grave{\mathcal{L}}_i$ are computed as in Equations (5.2) to (5.5).

After receiving an ACK from parent, a child node needs to adjust its own and/or subtree's sensing duties according to the sensing requirement information embedded in ACK. The adjustment steps are as follows:

- A child node first updates its sensing duty to the value specified in ACK if possible.
- If its sensing duty cannot be changed anymore (reach 1 or 0), this child node will propagate the value of difference between its current sensing duty and the requirement in ACK to the child node selected using Table 5.2.

5.4.3 Periodical Updating of Routing Tree

The routing tree updating scheme in J-RoS is designed with the awareness of sensing schedules, such that the effort made by the sensing duty updating scheme can be further boosted or at least not be jeopardized. Particularly, the route updating scheme follows the same principle adopted by the sensing duty updating scheme, through directing more traffics to non-critical nodes who are working in energy efficient way and meanwhile directing less traffics to critical nodes to prolong their nodal lifetime. Periodically, the route updating scheme runs in two steps: information collection and route updating.

5.4.3.1 Information Collection

Every certain period of time - routing update interval, each node (as a parent candidate for its neighboring nodes) broadcasts a routing beacon message containing the following information:

- lifetime of the bottleneck node - $\acute{\mathcal{L}}$ - on the path from itself to the sink,
- value $\mathcal{R}$ of the bottleneck node, and
- criticality of the bottleneck node.

Figure 5.4 shows how to estimate the above information when sending a routing beacon message.

5.4.3.2 Route Updating

With the information of $\acute{\mathcal{L}}$, $\mathcal{R}$ and criticality of each parent candidate's bottleneck node, a node selects its parent from the parent candidates as follows.

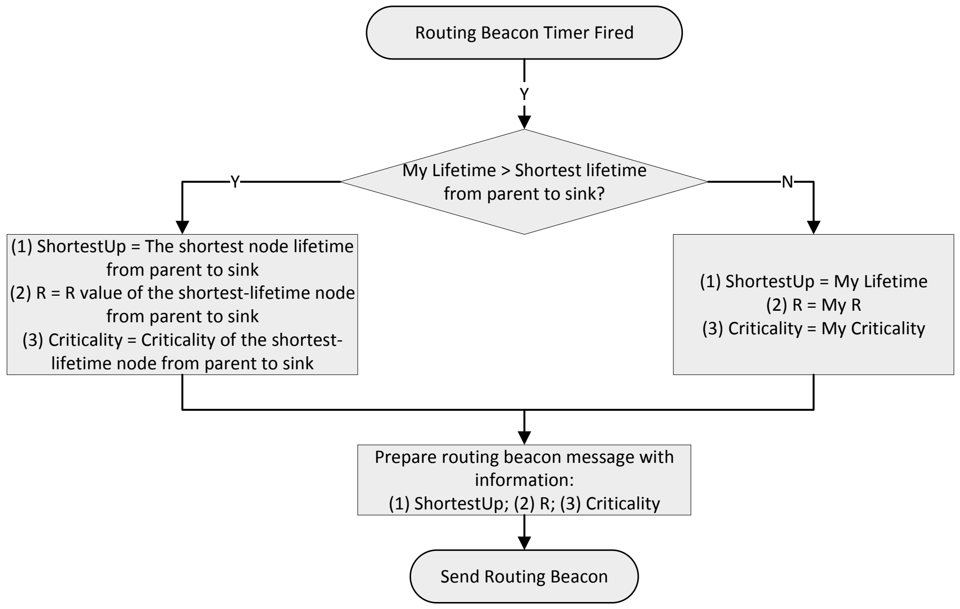

Routing Beacon Timer Fired
Y
My Lifetime > Shortest lifetime from parent to sink?
Y
N
(1) ShortestUp = The shortest node lifetime from parent to sink
(2) R = R value of the shortest-lifetime node from parent to sink
(3) Criticality = Criticality of the shortest-lifetime node from parent to sink
(1) ShortestUp = My Lifetime
(2) R = My R
(3) Criticality = My Criticality
Prepare routing beacon message with information:
(1) ShortestUp; (2) R; (3) Criticality
Send Routing Beacon

Figure 5.4 Procedure to send a routing beacon message.

- If all parent candidates contain critical bottleneck nodes on their paths to sink (if each of the candidates is chosen as the parent of the node under consideration), select the candidate with the shortest-lifetime bottleneck node.
- Otherwise, among the parent candidates with non-critical bottleneck nodes, select the one with the lowest value of $\mathcal{R}$ as parent to reduce the waste; if there is a tie, select the one with the shortest-lifetime bottleneck node.

Note that, the route updating scheme in J-RoS is not energy-balanced routing. Instead, J-RoS may direct traffic to a node who has lower lifetime if this node is non-critical or has smaller value of $\mathcal{R}$, which is similar to the behavior of scheduling sensing duties.

5.4.4 Other J-RoS Design Issues

5.4.4.1 Identification of Areas

The partitioning of monitoring areas in the network is determined by application. However, if the partition changes in runtime, the sink node will broadcast a message with “Area ID” and “Sensing

Coverage Requirement” to nodes in the affected areas. As each node knows the area it belongs to, it compares the IDs of its own and parent node’s area: if the two are different, a node would identify itself as root of an area and use its node ID as the area ID; otherwise, it uses its parent’s area ID. The area ID information is embedded into routing beacon messages, and each node knows the area it belongs to when it selects the parent node.

5.4.4.2 Identification of Critical Node

In order to determine whether a node is critical for sensing coverage, following information needs to be collected and available for each node: (i) the sensing coverage requirement for the area a node belongs to; (ii) the total number of nodes in the area a node belongs to; and (iii) the number of nodes in a node’s subtree that are in the same area as the node itself. Information (i) and (ii) can be embedded into notification messages broadcasted when a monitoring area changes; and (iii) can be obtained by letting each node embed its area ID in data packet, and monitor this information when transmitting data packets. With these information, a node knows whether it is critical for its own area with a minimal overhead.

5.4.4.3 Handling of Disconnection

As a node might be disconnected from its current parent due to energy depletion or route changes, it is important to monitor the total sensing duties in each area and take proper handing when sensing coverage of an area is violated. In J-RoS, a sink node sends a notification message to the root nodes of each sub-area where the sensing coverage violation is detected. When a node receives the notification message, it adjusts its own and/or subtree’s sensing duties as the same as receiving an ACK from a parent node.

5.5 Performance Evaluation

We have evaluated the performance of J-RoS in terms of network lifetime through ns-2 simulations, and J-RoS is compared to the following solutions:

- Upper: the upper bound solution obtained from an NLP solver [56] of the formulation in Section 5.3.3.

- Balance: a combination of energy-aware routing and sensing scheduling scheme, in which the routing scheme directs more traffic to nodes with higher residual energy and the sensing scheduling scheme also allocates more sensing duties to such nodes, to maximize the minimal nodal lifetime in the network.

- Even: a combination of energy-aware routing and a naive sensing scheduling scheme which allocates equal sensing duties to all nodes in the same area.

5.5.1 Simulation Setup

In the simulations, RI-MAC [10] is employed as the underlying MAC protocol, where nodal wakeup interval is 1 second and channel checking period is 7 ms, both being default setting of RI-MAC [10]. When the radio is on, the power consumption per node is 69 mW [44]. The power consumption for an actively sensing node is 2 mW when the node is in 100% sensing duty cycle. The power consumption of an idle node (i.e., not sensing or turning on radio), denoted as ϵ, is 80 μW. Every 20 seconds, a node sends out a routing beacon message and performs routing update if necessary.

J-RoS is evaluated in networks with line, star and random topologies, respectively. Figure 5.5 shows an example of random topology network with nine monitoring areas, which has been used in the simulations.

5.5.2 Simulation Results

5.5.2.1 Network Lifetime in Networks with Line Topology

With line topology, each node only has one parent node and one child node, and hence routing schemes do not affect the performance.

Figures 5.6 and 5.7 show the performance when nodes have the equal or different levels of initial nodal energy. As we can see, the "even" sensing scheduling scheme yields the shortest network lifetime, and the performances of both J-RoS and the "balance" strategies are close to the upper bound. This is

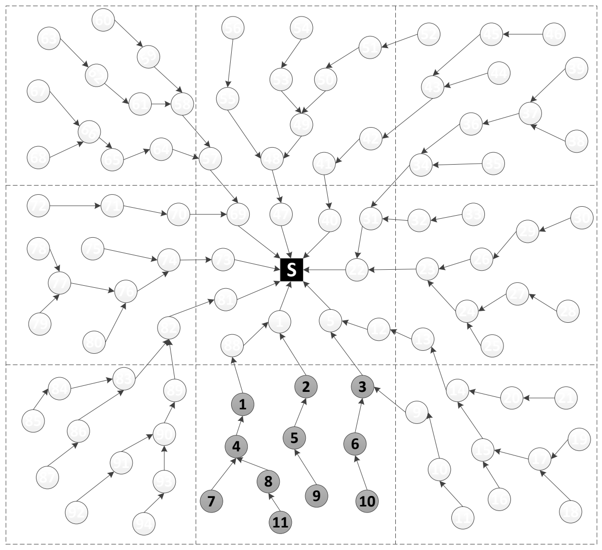


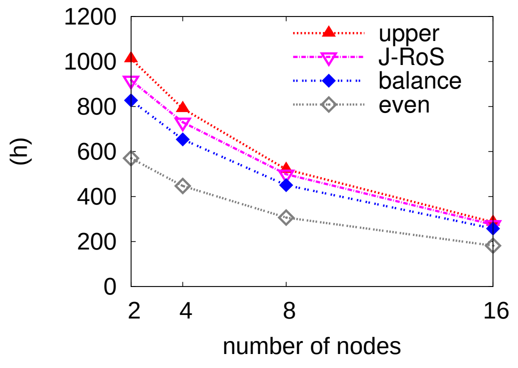
Figure 5.5 Example of a random topology network with 9 areas, and the sink node is at the center. The gray nodes are used for trace study.

a result of bottleneck effect caused by nodes which are close to sink and may consume more energy on communication (i.e., forwarding). The "even" strategy assigns higher but fixed sensing duties to bottleneck nodes; however, both J-RoS and the "balance" scheme can reduce the bottleneck nodes' sensing duties and therefore can extend the network lifetime. When the bottleneck effects are severe (i.e., there are more nodes on a line), the performance improved by J-RoS over the "balance" scheme is only about 5% to 10%. However, as demonstrated in Figures 5.8∼5.10, J-RoS outperforms the "balance" strategy with a ratio up to 40% when there are multiple branches in the network and the bottleneck effects are diminished (i.e., there are less nodes on a line).

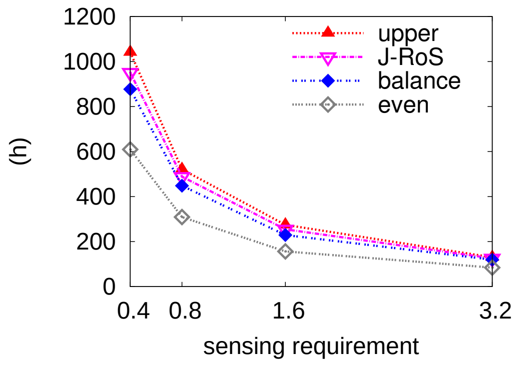
5.5.2.2 Network Lifetime in Networks with Star Topology

With star topology, all nodes are only one hop away from the sink, and hence routing schemes do not affect the performance as the sink node would always be selected as the only parent. The "balance" and "even" schemes obtained similar performance as they allocate similar levels of sensing duties to each node. The results are shown in Figures 5.8 and 5.9.

Compared to the performance achieved in networks with line topology, J-RoS achieves a significant improvement over the "balance" strategy in networks with start topology. In addition, as the number



(a) α is 0.2, 0.4, 0.8 and 1.6 when the number of nodes is 2, 4, 8 and 16, respectively.



(b) Number of nodes is 8.

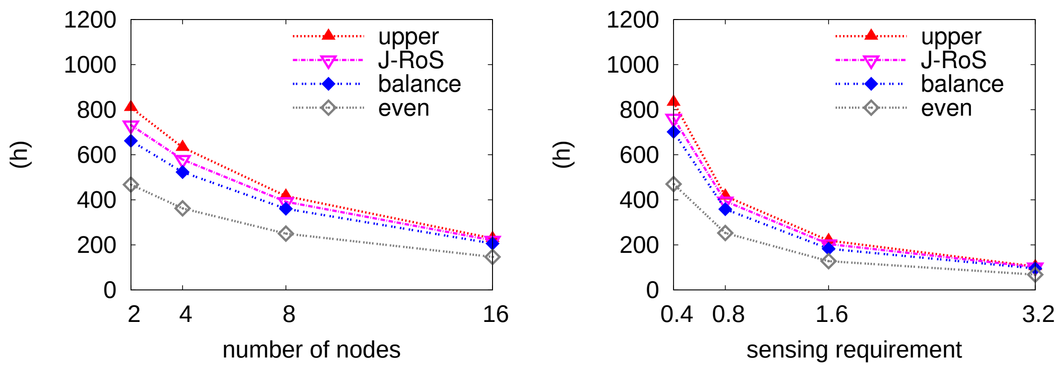
Figure 5.6 Network lifetime achieved in networks with line topology, where all nodes have the same initial energy 1000 J. There is one area in the network.

of nodes increases, the performance of J-RoS is further improved; particularly, the improvement ratio is about 40% when there are 16 nodes in network. This is because J-RoS can schedule more sensing duties to non-critical nodes for sensing coverage. As a result, these non-critical nodes can work energy-efficiently by consuming more energy on sensing and communication. This behavior delays the moment when critical nodes need to perform a high level of sensing duty to satisfy the sensing coverage requirement. Differently, the "balance" strategy would keep balancing nodal lifetime during the whole network lifetime, which may cause all nodes to work with lower energy efficiency and therefor may lower the network lifetime.

5.5.2.3 Network Lifetime in Multi-area Networks with Random Topology

We also evaluate the performance of J-RoS in networks with random topology, where all nodes are deployed to a 500 m $\times$ 500 m field randomly. The field is divided into grid areas, and the sink node is located at the center. The maximal communication range of each node is 100 meters. Figure 5.5 shows an example of the network topology in the simulation.

Figures 5.10(a) and 5.10(b) show the performance when nodes have the equal and different initial



(a) α is 0.2, 0.4, 0.8 and 1.6 when the number of nodes is 2, 4, 8 and 16, respectively.

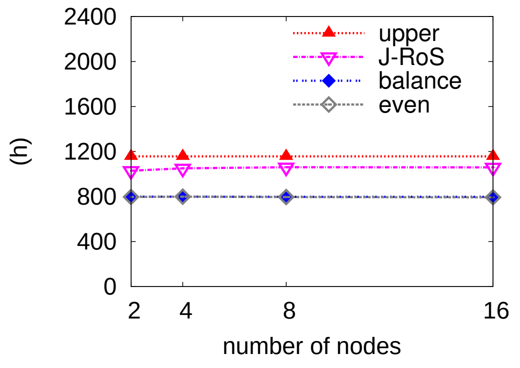
(b) Number of nodes is 8.

Figure 5.7 Network lifetime achieved in networks with line topology, where initial nodal energy is uniformly distributed between 500 J and 1000 J. There is one area in each network.

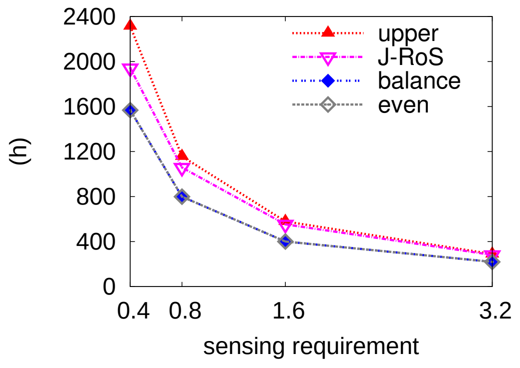
energy, respectively. The results are averaged over those for ten different random topologies.

In networks of random topology, the improvement ratios of J-RoS over “balance” and “even” strategies are about 20-30% and 80-100%, respectively. This is contributed by the the routing scheme in J-RoS, which has the following two major differences from the energy-aware routing scheme used by the other two strategies. First, the routing scheme in J-RoS directs more traffic to non-critical nodes which have lower ratio of wasted energy or critical-nodes which have longer lifetime. This way, sensing coverage can be maintained for a longer period of time. Second, the J-RoS routing scheme works in the similar way as the sensing scheme, which can further prolong the network lifetime through overcoming the limitations of the sensing scheme, for example, the sensing scheme cannot schedule workload across areas.

Figure 5.11 shows some snapshots of the sensing duty cycles of the gray nodes in Figure 5.5 taken in one of the simulations. At the beginning, all the nodes are assigned with the same sensing duty cycle which is a result of default sensing duty assignment. As the system runs, node 11 which is a non-critical leaf node, is assigned with the highest sensing duty after 20 hours, and most of other nodes have none or lower sensing duties. After about 60 hours, all the nodes are depleted in the branch where



(a) α is 0.2, 0.4, 0.8 and 1.6 when the number of nodes is 2, 4, 8 and 16, respectively.



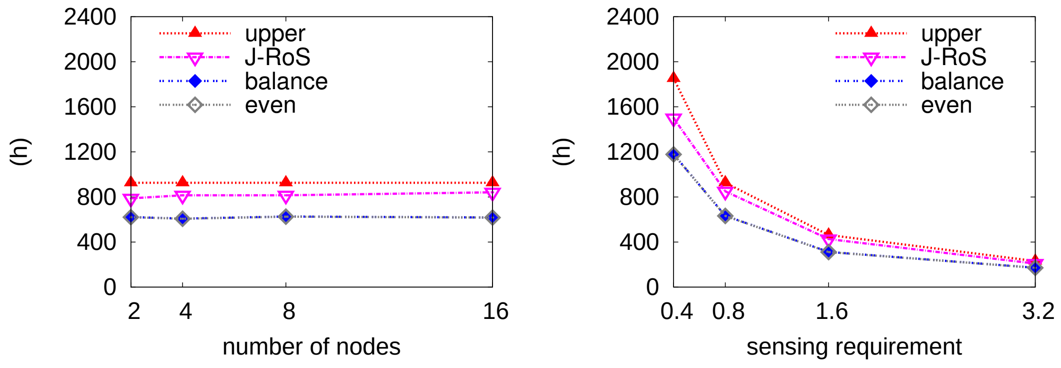
(b) Number of nodes is 8.

Figure 5.8 Network lifetime achieved in networks with star topology, where all nodes have the same initial energy 1000 J. There is one area in the network.

nodes 1, 4, 7, 8 and 11 belong to, and all sensing duties are shifted to nodes 2 and 10 due to their non-criticality and high energy utilization efficiency. Node 10 uses up its energy much sooner because of increased sensing duties after the change and only nodes 2 and 3 are alive after 80 hours. These snapshots illustrate the feature of how J-RoS utilizes the energy in the network: non-critical nodes (e.g., leaf nodes) are assigned with more sensing duties, while the energy of critical nodes is saved for as long as possible.

5.6 Conclusions

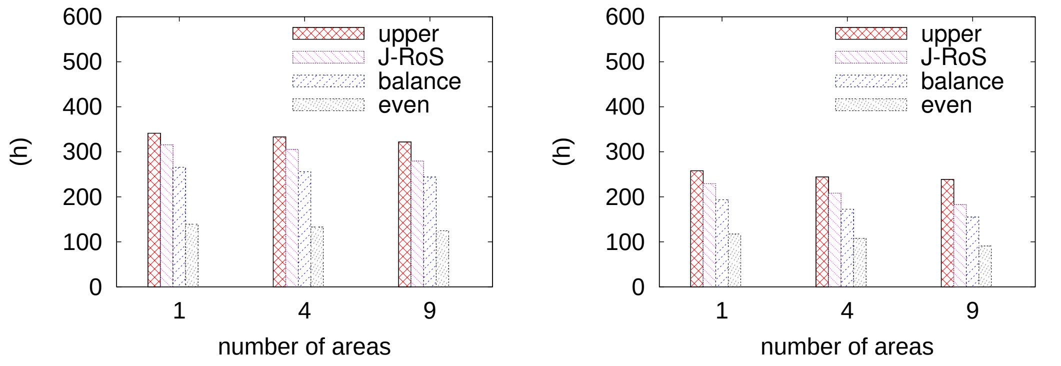
In this chapter, we proposed a distributed and low-cost joint routing and sensing scheme, called J-RoS, to prolong the lifetime of a sensor network under certain sensing coverage requirements. Different from lifetime-balancing schemes, J-RoS is unique in that it schedules less sensing and communication duties to nodes that are critical for sensing coverage, but more to non-critical nodes even at the cost of losing these nodes quickly. As the sensing and connectivity requirements can be satisfied for a longer period of time, the network lifetime can be prolonged. The effectiveness and advantages of J-RoS have been proved via extensive ns-2 simulations.



(a) α is 0.2, 0.4, 0.8 and 1.6 when the number of nodes is 2, 4, 8 and 16, respectively.

(b) Number of nodes is 8.

Figure 5.9 Network lifetime achieved in networks with star topology, where the initial nodal energy is uniformly distributed between 500 J and 1000 J. There is one area in the network.



(a) All nodes have the same initial nodal energy 1000J.

(b) The initial nodal energy is a random value between 500J and 1000J.

Figure 5.10 Network lifetime achieved in networks with random topology, where there are multiple monitoring areas. α is 1 for each area, and number of nodes is 100.

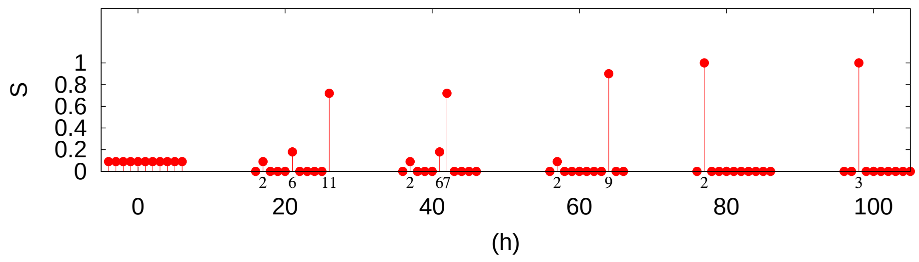


Figure 5.11 Snapshots of sensing duty assignments to gray nodes in Figure 5.5, which are taken in one of the simulations for random topology networks.

CHAPTER 6. CONCLUSIONS AND FUTURE WORKS

6.1 Research Contributions

In this dissertation, we have proposed several practical solutions to build a more sustainable sensor networks. We have demonstrated their effectiveness via extensive experimental and simulation studies. The main contributions of our work are:

- *Delay-bounded MAC protocol*

  In Chapter 2, we study how to reduce nodal idle listening time under a relative delay bound requirement. We propose a practical receiver-initiated MAC protocol, called CyMAC aiming at prolonging individual nodal lifetime. Different from existing schemes, CyMAC's design is based on the relative end-to-end delay requirement. We have implemented CyMAC on micaZ sensor motes and the effectiveness of CyMAC is demonstrated in different network settings via experiments and simulations.

- *Lifetime-Balanced MAC protocol*

  In Chapter 3, we study how to prolong the network lifetime. We present LB-MAC, a distributed and lightweight lifetime-balanced MAC protocol, which is designed from the perspective of network lifetime maximization. The key idea of LB-MAC is that communicating neighbors adjust their MAC-layer behaviors together in a collaborative manner to shift the communication overhead between them. As a result, nodal lifetime can be balanced between neighbors and network lifetime can be extended. The effectiveness of LB-MAC is demonstrated via in-depth experimental results.

- *Joint MAC and routing protocol*

  In Chapter 4, we present I^2C – a new holistic approach to prolong the sensor network lifetime.

I^2C leverages and integrates the advantages of both intra-route and inter-route coordinations and therefore can prolong the network lifetime more efficiently. In addition, I^2C can also meet the end-to-end delay requirement specified by the applications. Extensive simulation and testbed experiments have been conducted, and the evaluation results show that I^2C can significantly prolong the network lifetime than the state-of-the-art solutions.

- *Joint routing and sensing protocol*

  In Chapter 5, we propose a practical and efficient joint routing and sensing scheduling scheme, called J-RoS, to maximize the network lifetime while ensuring sensing coverage requirement. We present the design of J-RoS scheme and show its effectiveness in prolonging network via ns-2 simulations, under various configurations.

6.2 Future Research Topics

The past research experiences greatly help us understand how to design effective and practical protocols to increase the network sustainability. In this section, we share some of our opinions on these problems and discuss several potential research topics that are essential for future research towards building sustainable sensor networks.

- First of all, how to support broadcast or multicast data services in sustainable networks is of particular interesting. For sensor networks in which broadcast or multicast takes the majority of communications, network protocols shall be designed with the consideration of the unique communication patterns to prolong the network lifetime.
- Secondly, more cross-layer design shall be further investigated. Besides joint MAC and routing design, or joint routing and sensing design, lifetime elongation techniques in middle layer or application layer such as data aggregation, congestion control, etc., may also be jointly designed with MAC or routing layer protocols. This way, the energy heterogeneity problem may be better handled and network lifetime may be further prolonged.
- Finally, the network protocols shall be designed jointly with energy replenishment techniques such as solar energy harvesting and wireless charging. By predicting how energy replenishment

would happen or explicitly control the way energy is delivered to individual nodes, more sophisticated network protocols can be designed.

BIBLIOGRAPHY

[1] W. Ye, J. Heidemann, and D. Estrin, “An Energy-Efficient MAC protocol for Wireless Sensor Networks,” in *IEEE INFOCOM ’02*, 2002, pp. 1567–1576.

[2] L. M. Feeney and M. Nilsson, “Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment,” in *IEEE INFOCOM ’01*, 2001.

[3] E. Shih, P. Bahl, and M. J. Sinclair, “Wake on Wireless: An Event Driven Energy Saving. Strategy for Battery Operated Devices,” in *ACM MobiCom ’02*, 2002.

[4] T. van Dam and K. Langendoen, “An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks,” in *ACM SenSys ’03*, 2003, pp. 171–180.

[5] S. Du, A. K. Saha, and D. B. Johnson, “RMAC: A Routing-Enhanced Duty-Cycle MAC Protocol for Wireless Sensor Networks,” in *IEEE INFOCOM ’07*, 2007, pp. 1478–1486.

[6] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, “DW-MAC: A Low Latency, Energy Efficient Demand-Wakeup. MAC Protocol for Wireless Sensor Networks,” in *ACM MobiHoc ’08*, 2008, pp. 53–62.

[7] J. Polastre, J. Hill, and D. Culler, “Versatile Low Power Media Access for Wireless Sensor Networks,” in *ACM SenSys ’04*, 2004, pp. 95–107.

[8] A. El-Hoiydi and J.-D. Decotignie, “WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks,” in *LECTURE NOTES IN COMPUTER SCIENCE*, 2004.

[9] M. Buettner, G. V. Yee, E. Anderson, and R. Han, “X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks,” in *ACM SenSys ’06*, 2006, pp. 307–320.

[10] Y. Sun, O. Gurewitz, and D. B. Johnson, “RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks,” in *ACM SenSys ’08*, 2008, pp. 1–14.

[11] H. Zhang, A. Arora, Y.-r. Choi, and M. G. Gouda, “Reliable Bursty Convergecast in Wireless Sensor Networks,” in *ACM MobiHoc ’05*, 2005.

[12] B. Hull, K. Jamieson, and H. Balakrishnan, “Mitigating Congestion in Wireless Sensor Networks,” in *ACM SenSys ’04*, 2004.

[13] D. Estrin, R, Govindan, J. Heidemann and S.Kumar, “Next Century Challenges: Scalable Coordination in Sensor Networks,” in *MobiCom*, 1999.

[14] W. Ye, F. Silva, and J. Heidemann, “Ultra-Low Duty Cycle MAC with Scheduled. Channel Polling,” in *ACM SenSys ’06*, 2006.

[15] R. Krashinsky and H. Balakrishnan, “Minimizing Energy for Wireless Web Access with Bounded Slowdown,” in *ACM MobiCom ’02*, 2002.

[16] G. W. Challen, J. Waterman, and M. Welsh, “IDEA: Integrated Distributed Energy Awareness for Wireless Sensor Networks,” in *ACM MobiSys ’10*, 2010, pp. 35–48.

[17] J.-h. Chang and R. Tassiulas, “Energy Conserving Routing in Wireless Ad-hoc Networks,” in *IEEE INFOCOM ’00*, 2000, pp. 22–31.

[18] J. Park and S. Sahni, “An Online Heuristic for Maximum Lifetime Routing in Wireless Sensor Networks,” *IEEE Trans. Comput.*, vol. 55, pp. 1048–1056, August 2006.

[19] W. Wang, V. Srinivasan, and K.-C. Chua, “Using Mobile Relays to Prolong the Lifetime of Wireless Sensor Networks,” in *ACM MobiCom ’05*, 2005, pp. 270–283.

[20] K. Lin, J. Yu, J. Hsu, S. Zahedi, D. Lee, J. Friedman, A. Kansal, V. Raghunathan, and M. Srivastava, “Heliomote: Enabling Long-lived Sensor Networks Through Solar Energy Harvesting,” in *ACM SenSys ’05*, 2005, pp. 309–309.

[21] X. Jiang, J. Polastre, and D. Culler, “Perpetual Environmentally Powered Sensor Networks,” in *ACM IPSN ’05*, 2005, pp. 463–468.

[22] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, “Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless,” in *ACM SenSys ’10*, 2010, pp. 1–14.

[23] R. Braynard, A. Silberstein, and C. S. Ellis, “Extending Network Lifetime Using an Automatically Tuned Energy-Aware MAC Protocol,” in *Springer-Verlag EWSN ’06*, 2006, pp. 244–259.

[24] R. Jurdak, P. Baldi, and C. Videira Lopes, “Adaptive Low Power Listening for Wireless Sensor Networks,” *IEEE Transactions on Mobile Computing*, vol. 6, pp. 988–1004, August 2007.

[25] C. M. Vigorito, D. Ganesan, and A. G. Barto, “Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks,” in *IEEE SECON ’07*, 2007, pp. 21–30.

[26] A. Meier, M. Woehrle, M. Zimmerling, and L. Thiele, “ZeroCal: Automatic MAC Protocol Calibration,” in *IEEE DCOSS ’10*, 2010, pp. 31–44.

[27] C. J. Merlin and W. B. Heinzelman, “Duty Cycle Control for Low-Power-Listening MAC Protocols,” *IEEE Transactions on Mobile Computing*, vol. 9, pp. 1508–1521, November 2010.

[28] Y. Peng, Z. Li, D. Qiao, and W. Zhang, “Delay-Bounded MAC with Minimal Idle Listening for Sensor Networks,” in *IEEE INFOCOM ’11*, 2011, pp. 1314–1322.

[29] Z. Li, M. Li, and Y. Liu, “Towards Energy-Fairness in Asynchronous Duty-Cycling Sensor Networks,” in *IEEE INFOCOM ’12*, 2012, pp. 801–809.

[30] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiel, “pTunes: Runtime Parameter Adaptation for Low-power MAC Protocols,” in *ACM IPSN ’12*, 2012, pp. 173–184.

[31] M. T. Hansen, B. Kusy, R. Jurdak, and K. Langendoen, “AutoSync: Automatic duty-cycle control for synchronous low-power listening,” in *IEEE SECON ’12*, 2012, pp. 139–147.

[32] P. Hurni and T. Braun, “MaxMAC: A Maximally Traffic-Adaptive MAC Protocol for Wireless Sensor Networks,” in *EWSN ’10*, 2010.

[33] M. Sha, G. Hackmann, and C. Lu, “Energy-efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments,” in *ACM IPSN ’13*, 2013, pp. 277–288.

[34] Q. Li, J. Aslam, and D. Rus, “Online Power-aware Routing in Wireless Ad-hoc Networks,” in *ACM MobiCom ’01*, 2001, pp. 97–107.

[35] L. Lin, N. B. Shroff, and R. Srikant, “Asymptotically Optimal Power-Aware Routing for. Multihop Wireless Networks with Renewable Energy Sources,” *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1021–1034, October 2007.

[36] Y. Gu and T. He, “Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links,” in *ACM SenSys ’07*, 2007, pp. 321–334.

[37] ——, “Dynamic Switching-Based Data Forwarding for Low-Duty-Cycle Wireless Sensor Networks,” *IEEE Transactions on Mobile Computing*, vol. 10, pp. 1741–1754, 2011.

[38] S. Cui, R. Madan, A. Goldsmith, and S. Lall, “Joint Routing, MAC, and Link Layer Optimization in Sensor Networks with Energy Constraints,” in *IEEE ICC ’05*, 2005, pp. 725–729.

[39] R. Madan, S. Cui, S. Lall, and N. Goldsmith, “Cross-Layer Design for Lifetime Maximization in Interference-Limited Wireless Sensor Networks,” *IEEE Transactions on Wireless Communications*, vol. 5, no. 10, pp. 3142–3152, 2006.

[40] H. Wang, X. Zhang, F. Nait-Abdesselam, and A. Khokhar, “Cross-Layer Optimized MAC to Support Multihop QoS Routing for Wireless Sensor Networks,” *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2556–2563, 2010.

[41] S. Ehsan, B. Hamdaoui, and M. Guizani, “Cross-Layer Aware Routing Approaches for Lifetime Maximization in Rate-Constrained Wireless Sensor Networks,” in *IEEE GLOBECOM ’10*, 2010, pp. 1–5.

[42] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, “Cross-Layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks,” in *IEEE INFOCOM ’06*, 2006, pp. 1–13.

[43] TinyOS Community, "TinyOS Documentation Wiki," http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page.

[44] Texas Instruments, "CC2420: 2.4Ghz IEEE802.15.4/Zigbee-ready RF Transceiver," http://focus.ti.com/lit/ds/symlink/cc2420.pdf, 2006.

[45] Willow Technologies, "TelosB Mote Datasheet," http://www.willow.co.uk/TelosB_Datasheet.pdf.

[46] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links," in *ACM MobiCom '09*, 2009, pp. 133–144.

[47] X. Wang, X. Wang, G. Xing, and Y. Yao, "Dynamic Duty Cycle Control for End-to-End Delay Guarantees in Wireless Sensor Networks," in *IEEE IWQoS '10*, 2010, pp. 1–9.

[48] Y. Gu, T. Zhu, and T. He, "ESC: Energy Synchronized Communication in Sustainable Sensor Networks," in *IEEE ICNP '09*, 2009.

[49] K. Klues, G. Hackmann, O. Chipara, and C. Lu, "A Component Based Architecture for Power-Efficient Media Access Control in Wireless Sensor Networks," in *ACM SenSys '07*, 2007, pp. 59–72.

[50] UPMA - Washington University St. Louis, "UPMA Package: Unified Power Management Architecture for Wireless Sensor Networks," http://tinos.cvs.sourceforge.net/tinyos/tinyos-2.x/contrib/wustl/upma/.

[51] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *ACM SenSys '09*, 2009, pp. 1–14.

[52] Y. Peng, Z. Li, W. Zhang, and D. Qiao, "LB-MAC: A Lifetime-Balanced MAC Protocol for Sensor Networks," in *International Conference on Wireless Algorithms, Systems, and Applications (WASA '12)*, 2012.

[53] M. Conti, G. Maselli, G. Turi, and S. Giordano, "Cross-Layering in Mobile Ad Hoc Network Design," *IEEE Computer*, vol. 37, pp. 48–51, 2004.

[54] V. Kawadia and P. R. Kumar, “A Cautionary Perspective on Cross Layer Design,” *IEEE Wireless Communications*, vol. 12, pp. 3–11, 2005.

[55] Y. Gu, T. He, M. Lin, and J. Xu, “Spatiotemporal Delay Control for Low-Duty-Cycle Sensor Networks,” in *IEEE RTSS ’09*, 2009.

[56] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, pp. 25–57, March 2006.

[57] E. J. Duarte-Melo and M. Liu, “Analysis of Energy Consumption and Lifetime of Heterogeneous Wireless Sensor Networks,” in *IEEE GLOBALCOM ’02*, 2002.

[58] A. Giridhar and P. R. Kumar, “Maximizing the Functional Lifetime of Sensor Networks,” in *ACM IPSN ’05*, 2005.

[59] W. Mo, D. Qiao, and Z. Wang, “Mostly-Sleeping Wireless Sensor Networks: Connectivity, k-Coverage, and -Lifetime,” in *The 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[60] K. Wu, Y. Gao, F. Li, and Y. Xiao, “Lightweight Deployment-aware Scheduling for Wireless Sensor Networks,” *Mob. Netw. Appl.*, vol. 10, no. 6, pp. 837–852, Dec. 2005.

[61] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, “Integrated Coverage and Connectivity Configuration for Energy Conservation in Sensor Networks,” *ACM Trans. Sen. Netw.*, vol. 1, no. 1, pp. 36–72, Aug. 2005.

[62] D. Tian and N. D. Georganas, “A Coverage-preserving Node Scheduling Scheme for Large Wireless Sensor Networks,” in *ACM WSNA ’02*, 2002.

[63] Y. Peng, Z. Li, D. Qiao, and W. Zhang, “I2C: A Holistic Approach to Prolong the Sensor Network Lifetime,” in *IEEE INFOCOM ’13*, 2013.

[64] Z. Li, Y. Peng, D. Qiao, and W. Zhang, “Joint Aggregation and MAC Design to Prolong Sensor Network Lifetime,” in *IEEE ICNP ’13*, 2013.

[65] I. Dietrich and F. Dressler, “On the Lifetime of Wireless Sensor Networks,” *ACM Trans. Sen. Netw.*, vol. 5, no. 1, pp. 5:1–5:39, Feb. 2009.