2016

# Integration of lifetime-balancing schemes in wireless sensor networks

Rui Bai
*Iowa State University*

# Integration of lifetime-balancing schemes in wireless sensor networks

by

**Rui Bai**

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:

Wensheng Zhang, Major Professor

Daji Qiao

Leslie Miller

Iowa State University

Ames, Iowa

2016

# TABLE OF CONTENTS

# LIST OF TABLES

## LIST OF FIGURES

## ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, my major professor, Dr. Wensheng Zhang. For his continuous guidance, patience and support in all time of this research and the writing of this thesis. I would also like to thank my committee members, Dr. Hadji Ciao and Dr. Leslie Miller, for their suggestions and contributions to this work.

Finally, I must express my very profound gratitude to my family and friends without whose encouragement and support I would not have been able to complete this work.

# ABSTRACT

The wireless sensor network (WSN), once deployed, is usually expected to operate for months even years. In the WSN, each individual sensor node is typically equipped with small batteries which offer only a limited amount of energy. Hence, numerous energy-efficiency schemes have been developed for the WSN, aiming to help the WSN to achieve long lifetime under the stringent constraint of energy supply. These energy-efficiency schemes can reduce energy consumption, but they cannot address the issue of unbalanced energy consumption and energy replenishment in the WSN. In practice, different sensor nodes could have different initial energy supplies and/or play different roles in the network; consequently, some nodes may deplete their energy faster than the others. In such cases, the lifetime of the network becomes determined by the energy-bottleneck nodes who deplete their energy the fastest. To address this issue, the ideas of energy-balancing and lifetime-balancing have been proposed and energy-balancing and lifetime-balancing schemes have been developed for the MAC and routing layers of the WSN. However, there have been very few efforts on studying how to apply the idea to multiple layers of the WSN simultaneously and the effectiveness of such an integrated solution. To fill this gap, this thesis presents an integrated design, which includes lifetime-balancing schemes for the application, routing and MAC layers. Specifically, in the application layer where the sensing duty should be assigned to sensor nodes, we present a scheme in which leader nodes coordinate the sensing duty schedule among the sensor nodes based on their estimated nodal lifetime. In the routing layer, the routing metric is calculated based on both nodal lifetime and required end-to-end delay. In the MAC layer, we adopts the LB-MAC protocol, which balances nodal lifetime by adjusting the MAC parameters. Our design has been simulated in Network Simulator 2 (NS2) and compared with other solutions which do not apply lifetime-balancing idea or only applying the idea in some but not all of the three layers. The evaluation results have shown that our integrated design prolongs the network lifetime more effectively. Also, the integrated design achieves better performance in terms of data delivery ratio, end-to-end delay and wasted energy.

# CHAPTER 1. INTRODUCTION

## 1.1 Motivations

A wireless sensor network (WSN) is a network of numerous distributed, autonomous sensor nodes which collaborate in monitoring physical or environmental conditions such as temperature, humidity, etc. and gathering their sensory data through the network to a base station according to National Instruments (2012); Wikipedia (2016). Although initial WSNs were used for military purpose, they are now also deployed in industrial and consumer applications, for example, industrial process monitoring and control, and machine health monitoring according to Wikipedia (2016).

As sensor nodes are usually deployed in un-attended environments and are expected to operate limited battery energy for months even years, it is important to make sure the WSN to have a long lifetime. For this purpose, as discussed by Peng et al. (2010), energy (for example, solar and wind) harvesting technologies have been proposed to recharge the sensor nodes, but the effectiveness of these technologies is dependant on the environmental conditions. For example, sensor nodes deployed in shady area may not be recharged by solar energy effectively according to Peng et al. (2012).

Another approach to maintain long lifetime for a WSN is to reduce energy consumption. According to Moschitta and Neri (2016), the energy consumption of a sensor node is due to radio, computation, ADC (Analog to Digital Convertor) and DAC (Digital to Analog Convertor), but the radio costs more than the other parts. Therefore, the basic idea of energy saving is to lower the duty cycle of radio, which means the radio should sleep as long as possible while meeting a certain delay requirement according to Ye et al. (2002); Dam and Langendoen (2003).

Unfortunately, as the sensor nodes could have different levels of initial energy and play different roles in the network, some sensor nodes may deplete energy faster than the others. Energy-efficiency schemes extend the lifetime of senor nodes uniformly, so it is possible that the lifetime-bottleneck sensor nodes are out of energy while the others are still alive. According to Peng et al. (2012), this is likely to cause the whole network non-functional. For example, if a lost sensor node is responsible for data collection for a group of sensors, the sink may not retrieve data from the whole group. In such cases, as discussed by Peng et al. (2012), the network lifetime is defined by the time when the first sensor node runs out of energy.

Therefore, our research focus on extending the lifetime of the node which has the shortest lifetime. More specifically, we attempt to balance the nodal lifetime among all the sensor nodes of the whole network.

## 1.2 Our Research

The basic idea of lifetime-balancing is not completely new. Peng et al. (2010, 2012) have developed protocols based on lifetime-balancing in routing and MAC layers. There are also protocols by Ye et al. (2005); Qin and Zimmermann (2005) in the application layer to balance the workload for sensor nodes. However, applying a lifetime-balancing scheme for only one layer has limited the ability to balance the lifetime of sensor nodes. Thus, some recent proposed design I$^2$C by Peng et al. (2013) has attempted to combine routing and MAC layers and has accomplished better performance than prior efforts. In this thesis, we present a new scheme to apply the lifetime-balancing idea to all of the application, routing and MAC layers.

Specifically, in the scheme we propose for the application layer, the whole area is divided into subareas and each subarea can be monitored by a certain number of sensors that form a cluster. The deployment of sensor nodes should satisfy this requirement by having redundant sensor nodes in each subarea. In each cluster, a leader node is elected to coordinate the other sensors to switch among the states of leadership, active sensing and backup. Both the election and coordination algorithms are based on the distribution of lifetime among the sensor nodes.

In the scheme we propose for the routing layer, the goal is to establish a collection tree which enables each node to route its data packets to the sink through a path which has the longest lifetime.

For the MAC layer, we deploy LB-MAC proposed by Peng et al. (2012), which could balance the lifetime of all nodes in the transmission path by adjusting the four parameters: the wake up interval and the idle listening period for both the receiver and the sender. If the receiver has shorter lifetime than the sender, it increases the wake up interval and decreases the idle listening period. Otherwise, the receiver takes more communication overhead to save the energy for the sender by decreasing its wake up interval and increasing its idle listening period. After having finished the adjustment, the receiver informs the sender to update its parameters in order to meet the delay requirement.

To evaluate our proposed system of integrating the lifetime-balancing schemes for the three layers, we use Network Simulator 2 (NS2) to simulate our system and other systems in which lifetime-balancing schemes are deployed on at most two of these layers instead of all the three layers. The results of comparing all these simulated systems show that, our proposed system can effectively balance the lifetime among all the nodes in the network and achieves better performance than the other systems.

The rest of this thesis is organized as follows. Chapter 2 summarizes the related works regarding all of three layers. Chapter 3 introduces two important MAC protocols with more details. Chapter 4 presents the details of our proposed system. Chapter 5 reports the evaluation results obtained from the simulation. Finally, Chapter 6 concludes the thesis and discusses the future work.

# CHAPTER 2. RELATED WORK

In this chapter, we summarize the related works in the application, routing and MAC layers.

## 2.1 Application Layer Protocols

As sensor nodes have stringent power supply, the researchers have developed application protocols to achieve longer lifetime for a WSN. According to Ye et al. (2005), the data aggregation and clusters structure are the most popular technologies to be used. Heinzelman et al. (2000) proposed LEACH, which employs both of them to reduce the data redundancy and improves the network scalability. To form clusters, each node has to decide whether or not to be a head based on two factors. The first one is the number of heads needed in the network and the second factor is the times that the node itself has been a head. Once a node decides to be a head, it broadcasts advertisements to invite other nodes to join its cluster. When those non-head nodes receive the advertisements, they prefer to join the cluster which has the strongest signal. In terms of data transmission, the head is responsible to collect all data from its cluster and send them to the base station. Unfortunately, the heads may cost their energy much faster than non-head nodes, as they take more responsibilities. Thus, the clusters have to be reformed in every time interval. Although this design can prolong the network lifetime, the overhead introduced by cluster formation and re-formation may limit the performance of the network.

EECS proposed by Ye et al. (2005) is a design which also applies the ideas of cluster structure and data aggregation, but optimizes the cluster formation phase in LEACH. Instead of determining a head by non-energy related factors, EECS elects its heads based on residual energy level. At the beginning of the formation phase, every sensor node votes itself as a head and sends out advertisements until it is eliminated by a neighbor with higher energy. After election, all non-head nodes can reduce the energy consumption by joining the cluster with the shortest distance from itself to the sink.

Qin and Zimmermann (2005) proposed VCA. In this protocol, each sensor node votes the neighbor with highest energy as its head during the election stage. Later, in the cluster formation phase, the non-head sensor nodes prefer to join the cluster whose head has higher residual energy and fewer neighbors. Thus, the workload of a head with lower energy or more neighbors is balanced by other heads. As the distribution of energy level in the network changes after deployment, head re-election is necessary to protect the heads with heavier workload.

The experiments have demonstrated that all above three designs have prolonged the network lifetime effectively, but they ignored the potential redundant sensor nodes and the cost from sensing unit. Thus, it is possible to improve their performance by optimizing sensing duty schedule and utilizing the redundant sensors in the cluster.

## 2.2 Routing Layer Protocols

The most important factor for a routing layer protocol in WSN is energy-efficiency and the goal can be achieved by considering the nodal residual energy and the transmission energy requirement. According to Raghavendra et al. (2004), an energy-efficiency route can be selected based on one of following approaches:

- The maximum available power route attempts to find a route with the maximum available power route.

- The minimum energy requirement route can minimize the energy requirement of transmission.
- The minimum hop route regards the cost of each hop as one and aims to minimize the cost.
- The maximum minimum available power node route always picks the route which has higher residual energy in its bottleneck node.

### 2.2.1 Energy-efficiency Scheme

Li et al. (2001) proposed a routing algorithm named max-min zPmin, which uses the global energy information to select the route with minimum energy consumption to maximize the lifetime of network. In this routing algorithm, it first finds a route with minimum energy consumption and compares it with the lowest consumption route in the network. If no better path exists, the first path is the solution. On the other hand, the algorithm finds out the minimum consumption edge in the path and eliminates each of the edges which has a lower cost in the network. Although this design has a good performance in theoretical analysis, it is difficult to implement this scheme in a large network, since maintaining the global energy information is costly. Thus, the authors introduced hierarchy structure to reduce the problem to inter-zone and intra-zone routing problems. When doing inter-zone routing, each zone first estimates its metric by measuring the number of messages which can saturate at least one node in it. The intra-zone routing directly applies max-min zPmin, as the network scale is much smaller.

### 2.2.2 Energy-balancing Scheme

As previous algorithm is a energy-efficiency approach, some node may run out of energy while there are much energy remaining in the network. Thus, another kind of schemes based on energy-balancing have been presented. In EBRP proposed by Ren et al. (2011), the routing table is established by the hybrid of three virtual fields: depth, energy density, and residual energy. The depth allows packets to be transmitted along a shorter path. The energy density, which represents the energy level in a subarea, enables the packets to go through areas with higher energy. The last field, residual energy, protects the node with lower energy from accepting too many data flows. Thus, the packets can be always forwarded through a route with fewer hops and higher energy. As long as the network works, the energy level of network varies, so EBRP also introduces an updating mechanism. The updating can be triggered by any of these conditions:

- The maximum interval has expired;
- The node has costed at least 1% more energy since last updating;
- The depth has been changed and the minimum interval is expired.

### 2.2.3 Combination of Energy-efficiency and Energy-balancing Scheme

However, the pure energy-balancing scheme also has its drawback, as it may not utilize the energy effectively.

Chang and Tassiulas (2000) presented a two-stages routing protocol which selects the route based on the combination of energy-efficiency and energy-balancing. In its first stage, named as flow augmentation stage, a qualified node for augmentation defined as node with higher energy while lower transmission consumption rate. However, as these two factors cannot be optimized at the same time, their relative importance varies by time. At beginning, when all nodes have sufficient energy, the transmission consumption rate is more important. Later, as the nodes have less energy, the residual energy is emphasized. The second stage in the scheme is flow redirection, in which node with lower energy attempts to transfer its flows to other nodes with higher energy by following three steps:

1. In this step, the algorithm calculates the shortest length path from current node to the destination and the path which has the longest minimum lifetime in the network. By comparing the nodal lifetime with the shortest lifetime in both pathes, the algorithm decides which neighbor should accept the flow.

2. The second step decides a proper amount of flows to ensure that the redirection cannot decrease network lifetime.

3. The last step simply redirects the flows from old path to the new one.

However, as all three approaches are pure energy based, they may introduce extra end-to-end delay when updating the routing table.

## 2.3 MAC Layer Protocols

As the sensor nodes are expected to operate with small batteries for months and the radio consumes a high level of energy not only in data transmission and reception, but also a considerable amount of energy in the idle listening status. While the energy consumed for data transmission and reception is inevitable, the major purpose of radio power management is to reduce idle listening time. The intuitive solution is turning off radio when idle listening. However, since the node cannot send or receive data when radio is turned off, the radio needs to be turned on appropriately in order to avoid missing incoming data according to Huang et al. (2013). Thus, the MAC protocol focuses on the mechanism of duty cycling, which controls the nodes to switch between active and sleeping modes. There are two different kinds of MAC protocols: synchronous protocols and asynchronous protocols.

### 2.3.1 Synchronous MAC Protocols

In synchronous MAC protocols, neighbors wake up at the same time to ensure communication. Thus, as discussed by Huang et al. (2013); Sun et al. (2008), such protocols introduce a large overhead in synchronization. In S-MAC which is proposed by Ye et al. (2002), the wake up period is divided into two stages. The first one is used for synchronization in which all the nodes are required to broadcast their schedule information to their neighbors and only the second stage is available for real data transmission. Dam and Langendoen (2003) presented T-MAC, which is another synchronous protocol and optimizes S-MAC. In S-MAC, nodes have to stay awake for a certain period of time even without any data transmission. In stead of that, T-MAC adapts its idle listening time according to the amount of data traffic.

### 2.3.2 Asynchronous MAC Protocols

Although above synchronous protocols can reduce the idle listening time, they introduce high overhead, as the network has to synchronize in every time interval. Therefore, asynchronous protocols are developed to eliminate such overhead, but such kind of protocols have to provide efficient methods to establish communication between nodes according to Huang et al. (2013).

#### 2.3.2.1 Fixed Duty Cycle Protocol

**Sender Initiated Protocol** The first kind of asynchronous protocols are called Sender Initiated protocols. As a node spends most of its time in sleep and wakes up periodically to check whether there are packets ready, the sender needs to inform the receiver that there is packet for it. This idea is called Preamble Sampling according to Huang et al. (2013). In this protocol, the sender has to precede its data with a preamble which is long enough to be received by any receivers. If the receiver receives the preamble, it needs to stay awake and wait for the data. Otherwise, it may go back to sleep. This design works well when the traffic is low. However, as discussed by Huang et al. (2013), in some heavier traffic situations, the preambles shall influence the throughput of the network, as they capture more time. Another problem is that the non-target nodes cannot figure out that they are not the receiver until the beginning of data. Thus, there is an improvement to make the preambles shorter. Schurgers et al. (2002) developed a protocol named STEM, which uses a series of beacon packets instead of a long preamble and all the beacons contain the address of both sender and receiver. In this protocol, the non-target node is allowed to go sleep once any beacon indicates that it is not the destination. Furthermore, the receiver can reply an ACK to inform the sender to stop sending beacons and start data transmission. It is clear that this protocol shows a good performance when the data rate is low, but lower performance when data rate increases, because the setup latency becomes higher. To improve its performance, we can increase the data packet size to reduce the influence from setup overhead as discussed by Huang et al. (2013); Schurgers et al. (2002), but it increases the cost of collision and retransmission.

**Receiver Initiated Protocol** As the sender initiate protocols have above disadvantages, a protocol called Receiver Initiated Protocol (RI-MAC) is proposed by Sun et al. (2008). Different from the sender initiated protocols, it relies on the receiver's beacon message to initiate data transmission. During the process, the receiver broadcasts a beacon to declare that it is ready to receive data. If the sender is awake and receives the beacon, it is allowed to start data transmission. In this protocol, collision is more likely to happen, because we do not have the preamble to block other transmissions. Thus, if a collision is detected, the receiver needs to increase the backoff window size. Otherwise, the sender could send data immediately after the beacon. An interesting improvement on RI-MAC is using the ACK to inform which node is allowed to transmit next. Thus, the node with ACK could transmit immediately but the others have to go through backoff for collision avoidance. In this improvement, the receiver makes decision by the bandwidth demands from the sender and adjusts the transmitter schedule, so the fairness is ensured. Unfortunately, according to Peng et al. (2012), RI-MAC only benefits the receiver and the sender may have unnecessary idle listening.

#### 2.3.2.2 Dynamic Duty Cycle Protocol

Although above fixed duty cycle protocols can reduce the energy consumption and prolong network lifetime, they have limited ability to resolve the energy bottleneck problem. Thus, Braynard et al. (2006) proposed SEESAW to balance the energy between the sender and receiver. In SEESAW, every communication is initialized by the sender and the receiver is required to reply acceptance packet to trigger the data transmission. By this progress, the receiver needs to record the data retry interval and the sender learns the channel checking period. In next wake up interval, receiver can adjust its channel checking period and the sender is allowed adapt the data retrying interval to balance energy. However, performance of SEESAW is limited by two main aspects. Firstly, the setup progress introduces high overhead. Secondly, it does not involve the wake up interval in receiver and idle listening time in sender to optimize the lifetime-balancing. Another sender initiated protocol, called ZeroCal proposed by Meier et al. (2010), balances the energy consumption through wake up interval adaption; however, similar as SEESAW, it does not consider the other parameters according to Peng et al. (2012).

## 2.4 Lifetime-balancing in Multiple Layers

$I^2C$ proposed by Peng et al. (2013) is a design which balances nodal lifetime by the combination of routing and MAC layers while maintaining a certain end-to-end delay. For the MAC layer, this design follows a duty cycle similar to RI-MAC, but adjusts wake up interval to balance the lifetime as follows. When a node receives a data packet which indicates its lifetime is shorter than the sender, it increases the wake up interval. On other hand, if receiver has longer lifetime, it decreases interval. As the one hop delay in this protocol is defined by wake up interval, the MAC layer adjustment may increase the end-to-end delay by increasing interval. Thus, the receiver attaches new time interval in ACK and sender adjusts its own interval based on this new value. In the routing layer, the nodes can update routing table by the parent's lifetime and have to update wake up interval to absorb the increased delay. Although this design could dramatically extend the network lifetime, it does not involve the application layer and does not optimize the idle listening period in both sender and receiver.

# CHAPTER 3. PRELIMINARIES

In this chapter, we introduce RI-MAC and LB-MAC with more details.

## 3.1 RI-MAC

RI-MAC by Sun et al. (2008) relies on the receiver′s $BEACON$ message to initiate $DATA$ transmission. According to Figure 3.1, node $R$ wakes up periodically and broadcasts a $BEACON$ to claim that it is ready for receiving $DATA$. Shown as Figure 3.2 in which the shady fields are specific for RI-MAC, the $BEACON$ contains the source of $BEACON$, the destination of $BEACON$ and the backoff window size.

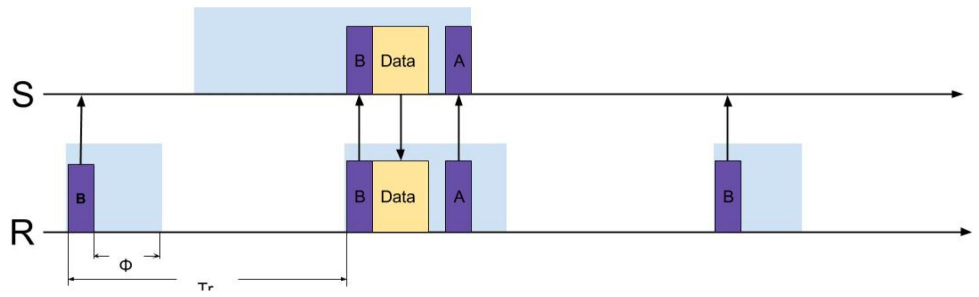
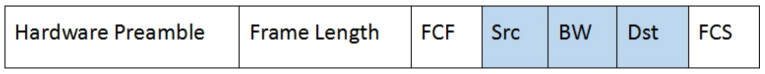


Figure 3.1 Duty Cycle of RI-MAC



| | | | | | | |
|---|---|---|---|---|---|---|
| Hardware Preamble | Frame Length | FCF | Src | BW | Dst | FCS |

Figure 3.2 Beacon in RI-MAC

If $R$ also has a $DATA$ packet buffered, it shall stay awake until it receives $BEACON$ from the potential receiver. Otherwise, $R$ turns off its radio after the backoff window size. On the other hand, if a node $S$, which has buffered $DATA$ for $R$, is awake and receives the $BEACON$ from $R$, it shall start the $DATA$ transmission after backoff. As $R$ may have multiple senders at same time and we do not have the preamble to block other $DATA$ transmissions, collision is more likely to happen. Thus, if a collision is detected, $R$ has to increase the backoff window size by Binary Exponential Backoff (BEB). Otherwise, the sender $S$ keeps awake for the acknowledgement. Once $R$ receives $DATA$, it broadcasts out $ACK$ beacon. The $ACK$ beacon in RI-MAC has two main functions. Firstly, it acknowledges the received $DATA$. Also, it can be used to invite more $DATA$ to the same receiver. Although $ACK$ beacon is broadcasted, $R$ can specify $S$ as its destination. In this case, once receives such $ACK$ beacon, the sender $S$ is allowed to transmit next $DATA$ to $R$ after SIFS while the other nodes regard the $ACK$ beacon as a normal $BEACON$.

## 3.2 LB-MAC

The RI-MAC is only able to benefit the receiver, because the sender has to keep radio on and wait for $BEACON$. The LB-MAC proposed by Peng et al. (2012) combines the sender initiate and receiver initiate approaches together. As a node could be both sender and receiver, LB-MAC maintains one duty cycle for each of them. The Figure 3.3 illustrates the behavior of LB-MAC.

When a node $R$ wakes up as a receiver every interval $Tr$, it sends out beacon to inform the senders that it is ready for receiving $DATA$ and stays awake for period of time $\Phi$. If $R$ receives $DATA$ from any other nodes, it shall send $ACK$. The $DATA$ packet is shown as Figure 3.4 and the shady fields are specific for LB-MAC. On the other hand, when a node $S$ wakes up as sender, it directly sends out its buffered $DATA$ and stays idle listening for $ACK$. If $S$ receives a $BEACON$ during the idle listening time, it needs to transmit buffered $DATA$ again. Otherwise, node $S$ turns off its radio. When

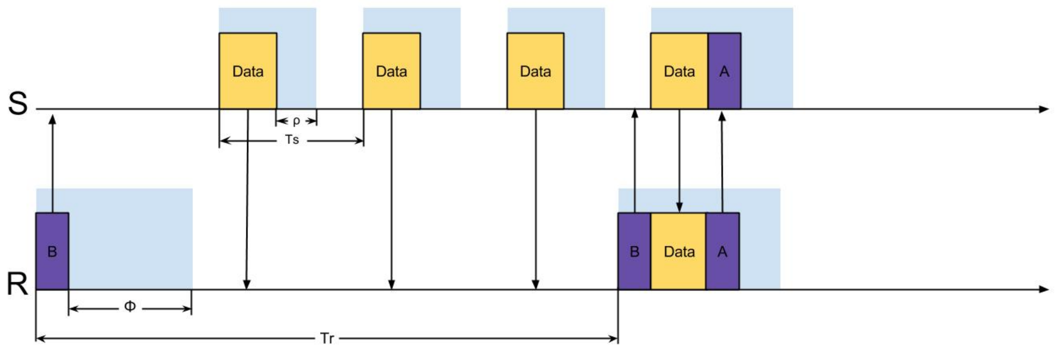


Figure 3.3 Duty Cycle of LB-MAC

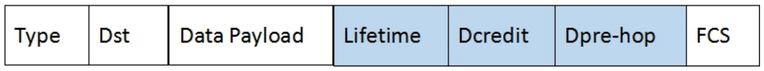
transmitting the $DATA$, $S$ shall estimate its lifetime ($L(S)$) by following formula:

$$L(S) = \frac{e(S)}{\frac{Tr}{2} \cdot \frac{\rho}{Ts} \cdot R_d \cdot P + g_S}, \tag{3.1}$$

where $e(S)$ is the residual energy of $S$, $Tr$ is the wake up interval of $R$, $\rho$ is the idle listening period of $S$, $Ts$ is the wake up interval of $S$, $R_d$ is the data rate, $P$ is the power consuming rate when radio is on and $g_S$ is the power consuming rate for other reasons.

Meanwhile, the LB-MAC satisfies the one-hop delay requirement by maintaining following rendezvous condition:

- Rendezvous Condition: $\rho + \Phi \geq min\{Ts, Tr\}$.



| | | | | | | |
|---|---|---|---|---|---|---|
| Type | Dst | Data Payload | Lifetime | Dcredit | Dpre-hop | FCS |

Figure 3.4 Data in LB-MAC

The most important work of LB-MAC is to balance the nodal lifetime between neighbors through $DATA$ transmission.

**Receiver Behavior** When node $R$ receives a $DATA$ packet from node $S$, it extracts the lifetime of $S$ ($L(S)$), the previous hop delay ($Dpre$) and sender's credit ($Credit(S)$)

from the packet. To ensure the increased delay can be absorbed within two hops, $R$ calculates the adjustment upper bound by following formula:

$$max\Delta \leq Dpre + Credit(S) + Credit(R),$$

where $max\Delta$ is the upper bound, $Dpre$ is the previous hop delay, $Credit(R)$ is the Credit of $R$ and $Credit(S)$ is the Credit of $S$.

Meanwhile, $R$ calculates its own lifetime $(L(R))$ by formula:

$$L(R) = \frac{e(R)}{(\frac{\Phi}{Tr}) \cdot P + g_R}, \tag{3.2}$$

where $e(R)$ is the residual energy of $R$, $\Phi$ is the idle listening period of $R$, $Tr$ is the wake up interval of $R$, $P$ is the power consuming rate when radio is on and $g_R$ is the power consuming rate for other reasons.

Then, it compares $L(S)$ with $L(R)$.

- Case 1: If $L(R) > L(S)$, $R$ shall first decrease the $Tr$ until $Tr{=}min\{Tr_{min}, \Phi\}$. For further adjustment, it shall attempt to increase the $\Phi$ until $\Phi{=}Tr$. Then, $R$ replies the updated $Tr$ and $\Phi$ in $ACK$ to the sender $S$.

- Case 2: If $L(R) < L(S)$, $R$ shall move more communication overhead to the sender $S$. Firstly, it shall decrease the $\Phi$ until $\Phi = \Phi_{min}$. If this is still not enough, $R$ shall attempt to increase the $Tr$.

- Case 3: If $L(R) = L(S)$, do nothing.

By adjusting the $Tr$ and $\Phi$, the one hop delay can be changed and the difference can be calculated by formula:

$$\Delta D_{S \to R} = (Tr_{new} - \Phi_{new}) - (Tr - \Phi).$$

Obviously, in case 1, the delay shall be decreased while increased in case 2. Thus, $R$ recalculates its credit by this formula:

$$Credit(R) = Credit(R)_{old} - \Delta D_{S \to R}.$$

Then, $R$ replies $S$ with its newest $Tr$, $\Phi$ and new credit in $ACK$.

**Sender Behavior** When sender $S$ receives the $ACK$, it is required to adjust its own parameters to satisfy the Rendezvous Condition and absorb the possible increased delay introduced by receiver $R$.

The first step is setting $\rho$ to $\varepsilon$ and setting the $Ts$ to $\Phi$ of the receiver $R$. Thus, the Rendezvous Condition can be satisfied. The second step is to absorb the possible extra delay introduced by the receiver $R$, since the end-to-end delay has to be maintained. Thus, $S$ shall attempt to absorb such extra delay. In the worst case, $S$ does not have sufficient credit to absorb the increased delay, so it needs to adjust its own $\Phi$ and $Tr$ and send $ACK$ to its sender for further adjustment.

# CHAPTER 4. SYSTEM DESIGN

In this chapter, we introduce our lifetime-balancing protocols in the application, routing and MAC layers.

## 4.1 Lifetime-balancing Sensing Protocol

The whole set of sensor nodes are divided into clusters, and each cluster is responsible for monitoring a subarea of the field where the sensor network is deployed. Within each cluster, there is one leader node and the rest nodes are called member nodes. The member nodes are further divided into two classes: the class of sensing nodes and the class of backup nodes. Thus, we have three types of nodes in each cluster:

- The leader node is elected by all nodes in the cluster, and responsible for coordinating its member nodes to switch their roles among leader, sensing nodes and backup nodes.
- The sensing nodes take the monitoring duty in the cluster and report their own residual energy to the leader.
- The backup nodes do not monitor but have to periodically report their residual energy to the leader.

The lifetime-balancing sensing protocol consists of an election stage and a sensing stage, which are explained as follows. In the election stage, all nodes in the cluster elect the leader node and decide their initial sensing duty schedule. After the election stage, the cluster enters the sensing stage. During this stage, the leader and the sensing nodes perform their assigned sensing duties; all member nodes are also required to report their residual energy to the leader once every interval. Meanwhile, the leader is responsible for adjusting the roles of member nodes based on their estimated lifetime.

### 4.1.1 Election Stage

The election stage includes two components, leader election and initial sensing duty scheduling.

**Leader Election** Once the sensors are deployed, they are assigned to clusters based on the locations they are deployed to. Then, each of them starts electing the cluster leader automatically. At the beginning of election, every node regards itself as leader and broadcasts its residual energy in an $ELECT$ packet. Meanwhile, it constructs three nodes lists. The first list, named as sensing nodes list, contains all the sensor nodes that should perform the sensing task and is sorted by nodal residual energy digressively. The second list contains all the backup nodes and sorts them based on their residual energy as well. The last list stores the leader itself.

If a node $R$ receives an $ELECT$ packet from some node $S$, it first extracts the information about the residual energy of $S$ from the packet, and compares it with the amount of its own residual energy. Here, we define $E(S)$ as the $S$'s residual energy and $E(R)$ as the residual energy of $R$.

- If $E(S) < E(R)$, node $R$ survives in the election. Thus, $R$ adds the information of node $S$ into its sensing nodes list if the list is not filled. Suppose each cluster can be monitored by $m$ sensors, the capacity of sensing nodes list is defined as $m - 1$. After having filled the sensing nodes list, the other nodal information will be added into the backup nodes list.

- If $E(S) > E(R)$, node $R$ has to quit the election. It stops sending ELECT packet, and cleans its three lists.

- If $E(S) = E(R)$, node $R$ compares its ID with that of $S$. If $R$ has smaller ID, it survives. Otherwise, $R$ quits the election as the previous case.

However, as a node which has quitted the election is not allowed to send $ELECT$ packet, it may be lost kept track of. Suppose the following scenario: Node $A$ has been eliminated by node $B$ without the engagement of node $C$. When node $C$ joins the election, $B$ quits election as it has lower residual energy. Then, $C$ loses the information of $A$.

To address this problem, $PREINFO$ packet is introduced to the design. If a node $S$ survives in the election, it wakes up periodically and broadcasts the its three lists in the payload of $PREINFO$, together with the information of its residual energy. The three lists are mapped into an integer array. As shown in Figure 4.1, the array includes all member nodes' IDs and uses ID '-1' as the separator of sensing nodes and backup nodes.

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | -1 | 4 | 5 |

Figure 4.1 Payload for PREINFO packet

For any node $R$ which receives the $PREINFO$ packet, it extracts energy $E(S)$ and the payload from $PREINFO$. If $R$ is a leader, it follows the same steps as it receives $ELECT$ packet. Otherwise, as a member node, $R$ searches in the payload and acts according to the rules below.

- If $R$ finds itself in the sensing nodes list, it sets itself as a sensing node.

- If $R$ finds itself in the backup nodes list, it sets itself as a backup node.

- If $R$ does not find itself in either list, it rejoins the election by sending an $ELECT$ again.

**Initial Sensing Duty Scheduling** The leader sets a timer. When such timer expires, the leader finishes the election and unicasts an $INFORM$ packet to each of its member nodes. In the $INFORM$ packet, the leader sets the sensing status field to indicate whether a member node should take the sensing duty. Once a member node receives $INFORM$, it sets its sensing status according to the value from $INFORM$ packet and enters the sensing stage.

### 4.1.2 Sensing Stage

After the election stage, the cluster moves on to the sensing stage which consists of the following components.

- Data and energy reporting: When the network starts the sensing stage, the leader and the sensing nodes report sensory $DATA$ to the sink. Also, all member nodes, include sensing and backup nodes, are required to report their residual energy and energy consumption rate to the leader.

- Sensing duty schedule updating: During the sensing stage, the leader can change the role of any node in its cluster, including transferring its leadership to another node.

- Leader reelection: When the current leader becomes unreachable, the cluster has to reelect a new leader.

These components are detailed as follows.

**Data and Energy Reporting** By receiving an $INFORM$ packet from the leader, a node enters the sensing stage. As mentioned in Section 4.1, there are three types of nodes and they have different kinds of packets to transmit according to their roles.

- A sensing node ought to send sensory $DATA$ to the sink and $ENERGY$ reports to the leader periodically.

- A backup node is required to report its residual energy and energy consumption rate to the leader once every interval.

- A leader needs to report sensory $DATA$ to the sink. Meanwhile, it has to receive and respond the $ENERGY$ reports from its member nodes.

Here, we define sensory $DATA$ packet as the packet which contains the sensory data of the cluster and needs to be routed to the sink through a collection tree. The details of routing algorithm and collection tree establishment have been discussed in Section 4.2. The $ENERGY$ packet contains the information about residual energy, energy consumption rate and role of the sender.

Obviously, as the leader and the sensing nodes have heavier workloads than the backup nodes, it is possible that some backup nodes may have longer lifetime than the sensing nodes or the leader after having been deployed for a while. In this case, we enable the leader to update sensing duty schedule by swapping the roles of sensing nodes and backup nodes or giving out its leadership.

**Sensing Duty Schedule Updating** The most important task of a leader node is to maintain the sensing duty schedule for its cluster. After having worked for a period of time, the distribution of nodal lifetime in the cluster is likely to be different. As our goal is to balance lifetime among all nodes in the network, we prefer the nodes with longer lifetime to take more responsibilities in sensing duties and leadership. Thus, when a leader receives the $ENERGY$ packets from member nodes, it is not only allowed to update the sensing duty schedule by switching some sensing nodes to backup and verse vice, but also give out its leadership to another node.

To initiate $ENERGY$ reporting, the leader sends out a $BEACON$ and waits for a time period of $\Phi$. When it receives an $ENERGY$ report from a member node $M$, it sends back an $ACK$. The $ACK$ indicates whether this member node $M$ needs to switch its sensing status or whether there is a leadership swapping.

The detail is as follows. Once the leader receives an $ENERGY$ report from member node $M$, it updates information of $M$ in its three lists. Then, the leader needs to estimate the lifetime of $M$ as $L(M)$. Meanwhile, it also estimates lifetime for the head node of backup nodes list ($L(B)$), the tail node of sensing nodes list ($L(S)$) and itself ($L(R)$). Here, the lifetime is estimated by following formula:

$$L(i) = \frac{E(i) - R(i) \times (c - pre)}{R(i)}, \text{ where i} \in \{\text{All nodes in the network}\},$$

$E(i)$ is the residual energy of node $i$, $R(i)$ is the average energy consumption rate of $i$, $c$ is the current system time, $pre$ is the time when leader receives the latest $ENERGY$ report from node $i$.

As the $ENERGY$ packet can be from either a sensing node or a backup node, we discuss these two cases, respectively.

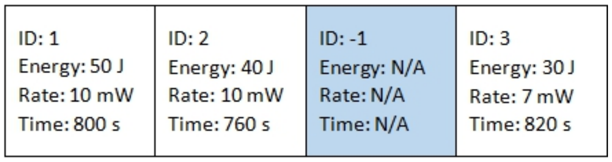
- Case I: $M$ is a sensing node, the leader compares $L(M)$, $L(R)$ and $L(B)$.
  - If $L(B) > L(M)$ and $L(R) > L(M)$, both the leader and the head node of backup nodes list have longer lifetime, $M$ should swap with the head node of backup nodes list to ensure the leader has the longest lifetime in the cluster.
  - If $L(M) > L(R)$, $M$ needs to take the leadership.
  - Otherwise, it is unnecessary to update the sensing duty schedule.
- Case II: $M$ is a backup node, the leader compares $L(M)$, $L(R)$ and $L(S)$.
  - If $L(R) > L(M) > L(S)$, $M$ swaps with the tail node of sensing nodes list.
  - If $L(M) > L(R)$, $M$ swaps with the leader.
  - Otherwise, there is no need to update the sensing duty schedule.

The details of swapping algorithms for both sensing status and leadership is discussed in following paragraphs.

**Sensing status swapping** If $M$ is a sensing node and it attempts to swap with the head node of backup nodes list, the leader moves $M$ from the sensing nodes list to the backup nodes list. Meanwhile, it moves the head node of backup nodes list to the sensing nodes list. As both lists are sorted in the descendant order of nodal residual energy, the leader has to insert these two nodes into appropriate positions to maintain the order. After having finished the swapping, the leader sends $ACK$ packets to both $M$ and the head node of backup nodes list to inform their new roles.

If $M$ is a backup node, the process is similar to the above. The only difference is that $M$ needs to swap with the tail node of sensing nodes list, instead of the head node.

**Leadership swapping** The leader is allowed to swap with either a sensing node or a backup node. However, as the leader maintains the sensing duty schedule for the whole cluster, the current leader has to share the sensing duty schedule with the new leader. Thus, the sensing nodes list and backup nodes list are mapped to an array which is sent as the payload of $ACK$. As shown in Figure 4.2, besides the nodal IDs, however, more information should be transmitted in the $ACK$, which includes the residual energy, the energy consumption rate, and the time when the latest $ENERGY$ report was received, for each node.



| | | | |
|---|---|---|---|
| ID: 1 | ID: 2 | ID: -1 | ID: 3 |
| Energy: 50 J | Energy: 40 J | Energy: N/A | Energy: 30 J |
| Rate: 10 mW | Rate: 10 mW | Rate: N/A | Rate: 7 mW |
| Time: 800 s | Time: 760 s | Time: N/A | Time: 820 s |

Figure 4.2 Payload for ACK packet

**Finish updating** When a member node receives an $ACK$, it first checks the leader field to decide whether it needs to switch to the role of leader. If it becomes a leader, the node has to rebuild its three nodes lists by the payload of $ACK$. Otherwise, it skips to the next step. At last, the node updates its sensing status according to the sensing status field in $ACK$.

**Leader Reelection** As a leader is responsible for maintaining the sensing duty schedule for the whole cluster, it has to be always reachable. For this purpose, we introduce a reelection mechanism into our design. When a member node sends out an $ENERGY$ report, it starts a timer for $ACK$ packet. If it receives the $ACK$ before the timer expires, the timer is cancelled. Otherwise, the node triggers the reelection and starts broadcasting an $ELECT$ packet. Once other nodes receive the $ELECT$ packet, they shall rejoin the election as well. Then, the process is the same as the leader election in Section 4.1.1.

## 4.2 Lifetime-balancing Routing Protocol

As each sensing or leader node generates a sensory $DATA$ packet every certain time interval and routes the packet to the sink, the node needs to create its own path to the sink. Thus, in our system, we propose an lifetime-balancing routing protocol to construct a collection tree. The rest of this section consists of two parts: first, we define the routing metric; second, we discuss the details of routing algorithm.

### 4.2.1 Routing Metric Definition

Firstly, we introduce the formula to calculate the routing metric. As mentioned earlier, our algorithm is based on the idea of balancing nodal lifetime, following which a node prefers to choose a neighboring node with the longest lifetime as its next hop.

According to Peng et al. (2010), an energy-balancing based cost of selecting node $i$ is defined as:

$$C_i = Tr_i \cdot u^{1-\frac{E_{residual}}{E_{initial}}}, \quad \text{where } i \in \{\text{All nodes in the network}\},$$

$Tr_i$ is the sum of energy consumption for packet transmission and reception at node $i$, $u$ is a system parameter and $u > 1$, $E_{residual}$ is the residual energy at node $i$, $E_{initial}$ is the initial energy of node $i$.

The above metric is based on the combination of residual energy and transmission consumption requirement, and it can balance the energy consumption rate among the network according to Peng et al. (2010). Unfortunately, it may not be sufficient for our design, as a node with lower energy consumption rate is not necessary to have longer lifetime. To construct a collection tree which balances the lifetime of the network, we introduce a factor which represents the residual energy into above metric. Thus, we update the metric to the following:

$$C_i = Tr_i \cdot u^{\alpha(\frac{E_{max}}{E_{residual}})+(1-\alpha)(1-\frac{E_{residual}}{E_{initial}})}, \quad \text{where } i \in \{\text{All nodes in the network}\},$$

$\alpha$ is a weight which use 0.5 as default value, $E_{max}$ is the battery capacity.

Above metric enables every node to find its next hop node, which has the longest lifetime.

### 4.2.2 Algorithm Description

With the metric defined as previous section, we design a routing algorithm, which works as follows. Once the network finishes deployment, the sink initiates the routing algorithm. As the sink initially has not recognized any of its neighbors, it sets the routing cost as 0 and broadcasts a routing $BEACON$ with the cost. For any other node $i$, at the first time it receives the routing $BEACON$, it adds an entry into its routing table. After having added the entry in routing table, instead of broadcasting the route immediately, node $i$ starts a timer. If node $i$ receives another routing $BEACON$ with lower cost before the timer expires, it replaces the current one with the new entry and restarts the timer. Otherwise, the existing route becomes the final decision. Then, node $i$ adds its own metric $C_i$ into the cost and broadcasts the routing $BEACON$ out. This way, each node can find a path to reach the sink.

As the nodal lifetime in the network may change after the network deployment, the collection tree has to be updated to protect the nodes whose residual energy drops fast. Thus, after having established the collection tree, the sink broadcasts out routing $BEACON$ periodically to update the collection tree. To enable a receiver to differentiate the new routing $BEACON$ with the old ones, each routing $BEACON$ carries a unique sequence number. When a node receives a routing $BEACON$ with newer sequence number, it should remove old entry from its routing table and recompute the table based on the new routing $BEACON$.

## 4.3 Lifetime-balancing MAC Protocol

In our system, we adopt LB-MAC, introduced in Section 3.2, as the MAC layer protocol. In LB-MAC, both receiver and sender can initiate $DATA$ transmission. Moreover, its most important work is to balance lifetime between neighbors through $DATA$ transmission. Once a receiver finds it has shorter lifetime than the sender, it shall increase its wake up interval and decrease idle listening period to save energy. Otherwise, the receiver needs to decrease wake up interval and increase idle listening period to protect the sender. After having adjusted its own parameters, the receiver informs the sender to adjust parameters to satisfy the delay requirement.

## 4.4 Integration of the Three Layer Protocols

When we deploy the application layer protocol in Section 4.1, the routing layer protocol in Section 4.2, and the LB-MAC protocol together, there are some necessary modifications for all of them.

### 4.4.1 Modification in Application Layer Protocol

In Section 4.1, a $ENERGY$ report transmission follows the duty cycle of RI-MAC. Thus, the leader wakes up as receiver and broadcasts $BEACON$ to invite $ENERGY$ reports. On the other hand, all member nodes follow sender's schedule to report their residual energy to the leader. However, as we have deployed LB-MAC in the MAC layer, the $ENERGY$ report transmission can be triggered by the $BEACON$ from MAC layer. Thus, the leader is not necessary to generate $BEACON$ from application layer any more; instead, it can create $BEACON$ from MAC layer. Furthermore, both leader and member nodes do not need to maintain the idle listening timer in application layer any longer, since LB-MAC has taken such responsibility.

Thus, the application layer protocol is updated as follows. The leader only sets a timer which enables it to send sensory $DATA$ to the sink periodically. On the other hand, the sensing nodes deploy with two timers. The first one is used for reporting residual energy and the second one maintains the schedule for sensory $DATA$. At last, The backup nodes report their residual energy based on their unique timer.

### 4.4.2 Modification in Routing Layer Protocol

As we have put the application layer protocol above the routing layer, the routing layer protocol has to process different types of packets from application layer. The application layer protocol is allowed to generate six types of packets: $ELECT$, $PREINFO$, $INFORM$, $ENERGY$, $DATA$ and $ACK$, as introduced in Section 4.1. The first two, $ELECT$ and $PREINFO$, are both broadcasted packets, while the other four are all unicasted packets.

When the routing layer protocol receives a packet from the application layer, it checks the type of the packet and acts as follows.

- If the packet is $ELECT$ or $PREINFO$, both its destination and next hop fields are set as broadcast. Furthermore, as these two kinds of packets are transmitted for only one hop, the TTL (Time To Live) field is set to 1.
- If the packet is $INFORM$, $ENERGY$ or $ACK$, the destination and next hop fields are set as the ones set by the application layer protocol. Meanwhile, TTL is set to 1.
- If the packet is $DATA$, the routing layer protocol first sets the destination as the ID of the sink and checks the routing table. If there is an entry in the routing table, it sets the next hop according to the entry. Otherwise, the routing layer protocol discards the packet. At last, the TTL is set to the number of nodes in the network.

On the other hand, if the routing layer protocol receives the packet from the MAC layer, it first decrements the TTL of packet. If the TTL is less than zero, the protocol discards the packet. Otherwise, it checks the destination $D$ of packet and acts as follows.

- If $D$ is broadcast, the protocol needs to check the origin of this packet. If the origin is not this node itself, the packet should be delivered to the application layer. Otherwise, it is a routing loop and the packet should be discarded.
- If $D$ is unicast and $D$ is this node, the protocol delivers the packet to the application layer.
- If $D$ is unicast but $D$ is not this node, the protocol needs to check the routing table. If there is an entry matches $D$, it resets the next hop field and sends the packet to the MAC layer. Otherwise, the packet should be discarded.

The routing algorithm should be further updated to be compatible with the MAC layer protocol. As we know, LB-MAC aims to balance the lifetime of neighboring nodes while maintaining the end-to-end delay. However, the updating of collection tree may change the end-to-end delay requirement of LB-MAC. Thus, the routing algorithm is updated as follows.

Firstly, a routing $BEACON$ should carry not only routing cost and sequence number, but accumulated delay as well. At the beginning of the collection tree establishment stage, the sink sets cost, sequence number and delay to 0 in a routing $BEACON$ and broadcasts it out. For any other node, once it receives the routing $BEACON$, it first checks the routing table. If the routing table is empty, the node follows the steps of collection tree establishment as described in Section 4.2.2. Otherwise, the node should process it by following updating approach.

When the node attempts to update the collection tree, it first compares the sequence number from routing $BEACON$ with the routing table entry. If the routing $BEACON$ is not older than the routing table entry, the node will check the cost, delay and the previous hop fields. The node is allowed to update routing table when either of following conditions holds:

- Both cost and delay from routing $BEACON$ are smaller than the values from routing entry.

- The value of previous hop field from routing $BEACON$ equals the next hop field in routing entry.

The first condition maintains the end-to-end delay when updating the collection tree. In the second condition, as the collection tree remains the same, LB-MAC is able to adjust its parameters to maintain the end-to-end delay requirement.

### 4.4.3 Modifications in LB-MAC

The modifications to LB-MAC include the following.

#### 4.4.3.1 Radio Control

In Section 4.4.2, we have introduced several broadcasted packets, which include $ELECT$, $PREINFO$ and routing $BEACON$, from both the election stage and the collection tree establishment stage. As these two stages both start right after the network deployment, we combine them as the setup stage. To guarantee the delivery of broadcasted packets, the node keeps radio on during this stage. By the end of setup stage, each cluster should have had its elected leader. Meanwhile, each node should have found its own route to the sink and have recognized all its neighbors. After the setup stage, the sensor nodes enter the sensing stage. For the purpose of stage transition, the leader should unicast $INFORM$ packets and the member nodes finish the transition after receiving the $INFORM$ packets. Once a node is in the sensing stage, it is allowed to turn off radio according to its duty cycle. As LB-MAC is an asynchronous protocol, all packets in the sensing stage must be unicasted due to delivery guarantee. Even the broadcasted packets are delivered to all sender's neighbors one by one through unicasting, except the MAC layer $BEACON$.

#### 4.4.3.2 Beacon Behavior

As discussed in Section 4.4.1, we have combined the $BEACON$ in application protocol with the $BEACON$ in MAC protocol. However, the $BEACON$ in MAC layer is used to initiate $DATA$ transmission and all packets from upper layers are regarded as $DATA$ packets in the MAC layer. In this case, a node is likely to receive the $BEACON$ from other clusters, which may introduce some unnecessary transmissions. Thus, in our design, if a node $S$ receives a $BEACON$ from node $R$, it shall extract the following two fields: the location of $R$ and the role of $R$. The first field indicates whether $R$ is from the same cluster. The second one represents whether $R$ is the leader or not. By these two fields, there are four possible scenarios: $R$ is the leader from the same cluster, $R$ is a member node from the same cluster, $R$ is a leader from a different cluster, or $R$ is a member node from a different cluster. Thus, $S$ decides whether or not to transmit buffered $DATA$ packet based on these cases.

- If $R$ is the leader from the same cluster, $S$ needs to check its buffered packet $P$. If $P$ is an $ENERGY$ report, $S$ shall reset the destination as $R$ and start transmission. Otherwise, $S$ checks the destination $D(P)$ of $P$. If $D(P) = R$ or $D(P)$ is broadcast, $S$ is allowed to send out $P$ as well. However, if $P$ cannot match any of the above scenarios, $S$ ignores the $BEACON$.
- If $R$ is a member node from the same cluster, $S$ shall check the destination $D(P)$ in the buffered packet $P$. It is allowed to send out $P$ if and only if $D(P) = R$ or $D(P)$ is broadcast.
- If $R$ is a leader from a different cluster, $S$ checks the destination $D(P)$ of $P$ and sends out $P$ if either of the following conditions holds: $D(P) = R$ or $P$ is a broadcasted routing $BEACON$. Otherwise, $S$ discards the $BEACON$.
- If $R$ is a member node from a different cluster, the scenario is the same as previous one.

#### 4.4.3.3 Modification in Lifetime-balancing Strategy

As mentioned in Section 4.4.3.2, all packets from both application and routing layers are treated as $DATA$ in the MAC layer and LB-MAC triggers lifetime-balancing mechanism through $DATA$ transmission. Thus, if all the packets' paths overlap, the network topology becomes a cycle instead of a tree. Furthermore, since LB-MAC adjusts MAC layer parameters based on the credit in a $BEACON$ as described in Section 3.2, the cycle can increase the credit to $+\infty$. Thus, we update the lifetime-balancing mechanism to be triggered only by sensory $DATA$ packets from application layer. As the sensory $DATA$ packets are from sensing and leader nodes to the sink, we can eliminate reverse transmission in the network.

Unfortunately, it is still possible to accumulate the credit to an incorrect large value, which causes LB-MAC to adjust its parameters almost unlimited. Thus, we introduce a field $Credit(pre)$ to the MAC layer for maintaining the credit in previous sensory $DATA$ transmission. When the node receives a sensory $DATA$ packet, it first decreases its credit by $Credit(pre)$ and recalculates the local credit.

#### 4.4.3.4 Modification in the Transmission Schedules

With LB-MAC, a node maintains two schedules for its receiver and sender roles respectively. If a node wakes up as a receiver, it sends out $BEACON$ to invite packets. If it wakes up as a sender, it is allowed to send out buffered packet immediately. Furthermore, when a node receives a packet, it has to send an $ACK$ to acknowledge the packet as well. Unfortunately, since we have to go through a backoff before any transmission, it is possible that, when buffered packet is experiencing the backoff, the $ACK$ or $BEACON$ begins backoff as well. Moreover, as the MAC layer only has one backoff timer, a newly backoffed packet may block the transmission of an older packet. To address this issue, we update the MAC protocol when the backoff timer expires. Instead of only sending out the newly backoffed packet, the node should send all the backoffed packets. The only exception happens when the buffered packet is an $ENERGY$ report. In this case, the node has to check the destination of packet. If the destination has been set, which means the node has received $BEACON$ from the leader, the node is able to transmit the packet. Otherwise, it keeps the $ENERGY$ report in the buffer.

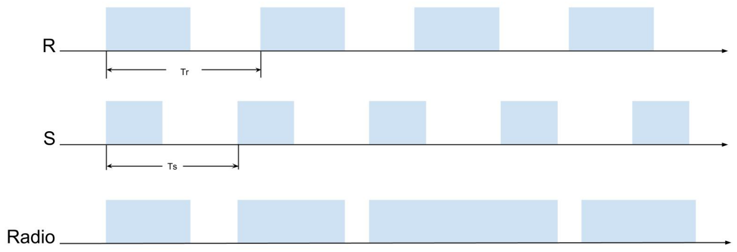


Figure 4.3 Radio Duty Cycle in LB-MAC

#### 4.4.3.5 Lifetime Estimation

In the original LB-MAC, the nodal lifetime is estimated by either formula 3.1 or formula 3.2. However, according to Figure 4.3 in which the shady parts represent that the radio is on, the actual radio duty cycle is formed by the combination of sender and receiver. Thus, the real energy consumption rate is hardly to be estimated by wake up interval and idle listening period. Furthermore, both formula 3.1 and formula 3.2 do not consider the energy consumption from sensing unit. Thus, we refine the lifetime estimation formula as follows.

$$L(i) = \frac{E(i)}{P_t(i)}, \text{ where i} \in \{\text{All nodes in the network}\},$$

$E(i)$ is the residual energy of node $i$, $P_t(i)$ is the the energy consumption rate for recent $t$ seconds.

Thus, the MAC protocol has to calculate energy consumption rate every $t$ seconds by

$$P_t(i) = \frac{E_t(i) - E(i)}{t}, \text{ where i} \in \{\text{All nodes in the network}\},$$

$E_t(i)$ is the residual energy before $t$ seconds, $E(i)$ is the current residual energy of node $i$, $P_t(i)$ is the the energy consumption rate for recent period of time $t$.

# CHAPTER 5. EXPERIMENT RESULTS

In order to measure the performance of our design, we have conducted simulation in Network Simulator 2 (NS2). The experiment compared our design which applies the lifetime-balancing idea to all of the application, routing and MAC layers, with other designs which apply the lifetime-balancing idea to at most two of the layers. For the sake of comparison, we have implemented one non-lifetime-balancing protocol for each of the application, routing and MAC layers. As we have introduced the RI-MAC protocol in Section 3.1, we next only describe the application layer protocol and the routing layer protocol in this chapter.

## 5.1 Application Layer Protocol: Average Sensing Protocol

The average sensing protocol for the application layer has the same assumptions and stages as the lifetime-balancing sensing protocol presented in Section 4.1. After being deployed, all the nodes go through the setup stage and then start the sensing stage. However, instead of coordinating sensing duty schedule based on the member nodes' lifetime, all nodes take the sensing duty and leadership by round-robin. More specifically, by every five sensory $DATA$ intervals, the leader updates the sensing duty schedule by following steps:

1. The leader moves the head node of sensing nodes list to the tail of backup nodes list.

2. The leader moves the head node of backup nodes list to the tail of sensing nodes list.

3. The leader sends a $SCHED$ packet to each of both nodes to update their sensing status.

Although we have introduced a new $SCHED$ packet in this design, it could work as the $ACK$ in lifetime-balancing sensing protocol. Thus, both the routing and MAC protocols process it as an $ACK$ as well. After having executed the above procedure for $n-1$ times, where $n$ is the number of member nodes, the leader gives out its leadership by following steps.

1. The leader updates sensing duty schedule according to the above three steps as usual.

2. The leader picks the head node of sensing nodes list as the leader and moves the head node of backup nodes list to the tail of sensing nodes list.

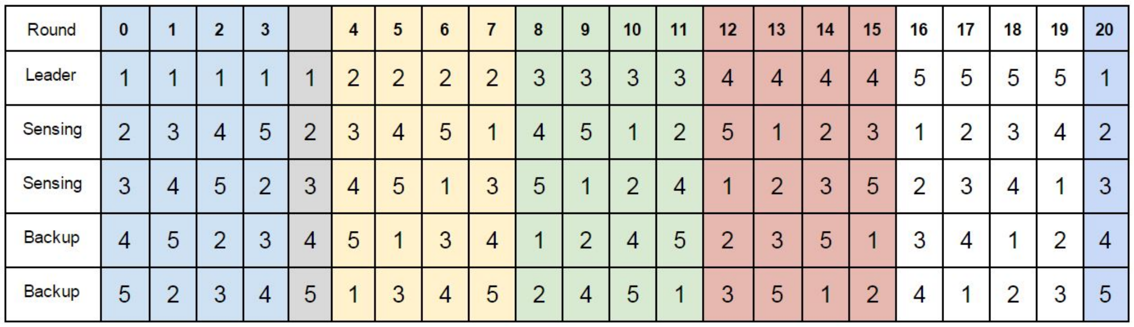
3. The leader puts itself to the tail of backup nodes list.

4. The leader sends $SCHED$ packets to the new leader and the tail node of sensing nodes list for the updates.

Upon receiving the $SCHED$ packet, a node updates its role the same as receiving the $ACK$ in Section 4.1.2.

Next, we demonstrate an example of this protocol. Suppose there are five nodes denoted as node 1 to node 5. By election, node 1 becomes the leader, node 2 and node 3 become the sensing nodes, and node 4 and node 5 become the backup nodes.

As shown in Figure 5.1, at the end of the round 0, node 2 is moved to the tail of backup nodes list and node 4 is moved to the tail of sensing nodes list. By the end of the round 3, the leader shall update the leadership. Thus, it firstly moves node 3 to the sensing nodes list and changes node 5 to a backup node. Then, the leader picks node 2 as the new leader and puts itself into the backup nodes list. At last, we shall have a sensing duty schedule as Figure 5.1.



| Round | 0 | 1 | 2 | 3 |  | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Leader | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 1 |
| Sensing | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 1 | 4 | 5 | 1 | 2 | 5 | 1 | 2 | 3 | 1 | 2 | 3 | 4 | 2 |
| Sensing | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 1 | 3 | 5 | 1 | 2 | 4 | 1 | 2 | 3 | 5 | 2 | 3 | 4 | 1 | 3 |
| Backup | 4 | 5 | 2 | 3 | 4 | 5 | 1 | 3 | 4 | 1 | 2 | 4 | 5 | 2 | 3 | 5 | 1 | 3 | 4 | 1 | 2 | 4 |
| Backup | 5 | 2 | 3 | 4 | 5 | 1 | 3 | 4 | 5 | 2 | 4 | 5 | 1 | 3 | 5 | 1 | 2 | 4 | 1 | 2 | 3 | 5 |

Figure 5.1 Sensing Duty Schedule for Average Sensing Protocol

## 5.2 Routing Layer Protocol: Shortest Distance Routing Protocol

The non-lifetime-balancing routing protocol is implemented as the shortest distance routing protocol. The routing algorithm is the same as the lifetime-balancing routing protocol presented in Section 4.2, except that the routing metric is changed to purely distance based. Suppose the coordinate of node $i$ is $(x_i,\ y_i)$ and it receives a routing $BEACON$ from node $j$ located at $(x_j,\ y_j)$. The routing metric calculated by node $i$ is as follows:

$$C_i = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},\ \text{ where } i,\, j \in \{\text{All nodes in the network}\}.$$

Then, node $i$ establishes or updates its routing table by the same steps in lifetime-balancing routing protocol.

## 5.3 Evaluation Results

### 5.3.1 Experiment Settings

**Combinations** We have introduced the lifetime-balancing sensing protocol in Section 4.1, the lifetime-balancing routing protocol in Section 4.2, and the LB-MAC protocol in Section 4.3. In this chapter, we also have presented the average sensing protocol in Section 5.1, the shortest distance routing protocol in Section 5.2, and the RI-MAC in Section 3.1. Using these six protocols, we have the following eight combinations for performance evaluation.

Table 5.1 Combinations

| Combination | Application Layer Protocol | Routing Layer Protocol | MAC Layer Protocol |
|---|---|---|---|
| ASR | Average Sensing Protocol | Shortest Distance Routing Protocol | RI-MAC |
| LSR | Lifetime-balancing Sensing Protocol | Shortest Distance Routing Protocol | RI-MAC |
| ALR | Average Sensing Protocol | Lifetime-balancing Routing Protocol | RI-MAC |
| LLR | Lifetime-balancing Sensing Protocol | Lifetime-balancing Routing Protocol | RI-MAC |
| ASL | Average Sensing Protocol | Shortest Distance Routing Protocol | LB-MAC |
| LSL | Lifetime-balancing Sensing Protocol | Shortest Distance Routing Protocol | LB-MAC |
| ALL | Average Sensing Protocol | Lifetime-balancing Routing Protocol | LB-MAC |
| LLL | Lifetime-balancing Sensing Protocol | Lifetime-balancing Routing Protocol | LB-MAC |

**Parameters** In following table, we list the values of important parameters in our simulation.

Table 5.2 Parameters in Simulation

| | | | |
|---|---|---|---|
| Tr | 1s | Radio Consumption Rate | 69mW |
| Ts | 0.1s | Sensing Consumption Rate | 5mW |
| $\phi$ | 0.1s | | |
| $\rho$ | 0.01s | | |

### 5.3.2 Evaluation Results

#### 5.3.2.1 Simulation Result for 32 Nodes

**Network Topology** The experiment is composed by 33 nodes and we divide them into 9 clusters as shown in Figure 5.2. Except for node 0, which acts as the sink and has infinite energy, all nodes are randomly deployed with an initial energy between 80 J and 400 J.

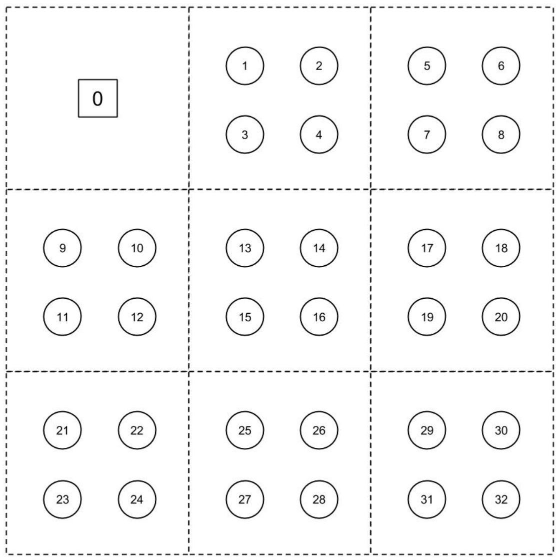


Figure 5.2 Network Topology for Simulation

**Simulation Result** We evaluated the the performance of the combinations in terms of end-to-end delay, data delivery ratio, network lifetime, and wasted energy. Here, we define the end-to-end delay as the average delay for sensory $DATA$ packets, and the delay is measured from when a packet is generated to the moment when the packet is delivered at the sink. The data delivery ratio is the fraction of delivered sensory $DATA$ packets in the whole network during the network lifetime. The network lifetime is defined as the time when any of the nodes is out of energy. The wasted energy is measured as the total residual energy for the whole network at the end of the network lifetime.

In Figure 5.3, we show the simulation result for the network with 32 sensor nodes and one sink. Meanwhile, we treat the results of all the four metrics (i.e., the end-to-end delay, the data delivery ratio, the network lifetime and the wasted energy) from

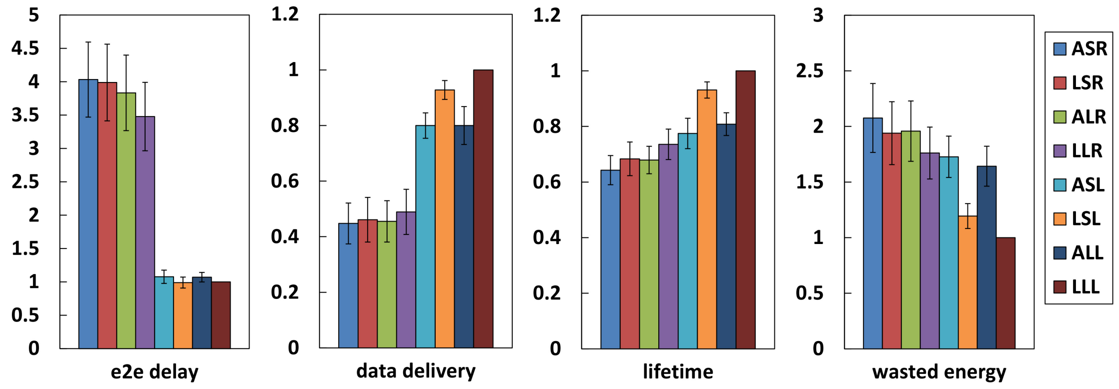


Figure 5.3 Simulation Result for 32 Nodes

combination LLL, which deploys lifetime-balancing solutions in all of the three layers, as the references, represented as values 1. The results from the other combinations are compared against the references and we present these values as ratios against the references. For example, by having end-to-end delay for ASR as 4.0, we demonstrate that LLL reduces the end-to-end delay by three times compared to ASR. The figure also shows the 90% confidence intervals for the metrics.

Among the combinations, ASR does not contain lifetime-balancing technique in any of the three layers. By Comparing LSR with ASR, we first evaluate the benefits from applying lifetime-balancing technique in the application layer. From Figure 5.3, we can see that LSR has longer life time and less wasted energy than ASR, as lifetime-balancing sensing protocol enables the nodes to take sensing duty based on their lifetime. Thus, the nodes with shorter lifetime are unlikely to take the sensing duty, which enables them to save more energy. Meanwhile, LSR also demonstrates shorter end-to-end delay and higher data delivery ratio. This is because the average sensing protocol updates the sensing duty in each time interval, and so it introduces extra overhead which may aggravate the workload of network and increase the queue delay in packet transmission.

Second, we evaluate the benefits from deploying a lifetime-balancing routing protocol by comparing the performance of ASR and ALR. With fixed MAC and application protocols, the lifetime-balancing routing protocol (LB-Routing) introduces shorter end-to-end delay, higher data delivery ratio, longer network lifetime and less wasted energy, compared to the shortest distance routing protocol. This is because LB-Routing establishes the collection tree based on the nodal lifetime, and thus can reduce the data flows through the nodes with shorter lifetime. As the nodes with shorter lifetime will have fewer data flows, they are able to save more energy and prolong lifetime.

Third, we compare the performance between LB-MAC and RI-MAC by comparing ASR with ASL. By the simulation result, we can see that LB-MAC has better performance in both end-to-end delay and data delivery ratio. This is because RI-MAC requires the sender to wait for the receiver to initiate packet transmission, while LB-MAC allows the node to start packet transmission immediately once it wakes up as sender. Also, LB-MAC extends network lifetime, as it prolongs the lifetime of bottleneck node.

By above three comparisons, we can get the conclusion that all lifetime-balancing sensing, routing and MAC protocols could improve the network performance independently.

Furthermore, if we compare the performance among LSR, ALR and ASL, the result shows that ASL achieves the lowest end-to-end delay, highest data delivery ratio, longest lifetime and smallest wasted energy among these three combinations. This is caused by two main reasons. First, as we have discussed, LB-MAC improves the transmission efficiency from RI-MAC, as we do not rely on the receiver to initiate the transmission any longer. Second, as discussed by Moschitta and Neri (2016), radio has higher energy consumption rate than any other components and LB-MAC optimizes radio control parameters to prolong the lifetime of the bottleneck node.

Next, we evaluate the combinations where two lifetime-balancing protocols are deployed, and so there are three possible combinations: application and routing (LLR), application with MAC (LSL), and routing and MAC (ALL). Comparing these three combinations (LLR, LSL and ALL) with the previous three combinations (LSR, ALR and ASL) which adopt only one lifetime-balancing protocol, we have the following observations:

- Due to the lifetime-balancing sensing protocol, LLR and LSL outperform ALR and ASL.
- Due to the lifetime-balancing routing protocol, LLR and ALL outperform LSR and ASL.
- Due to LB-MAC, LSL and ALL outperform LSR and ALR.

From above observations, we get the conclusion that having lifetime-balancing protocol deployed on one more layer is helpful to improve the network performance. Also, by comparing the performance among LLR, LSL and ALL, we find that the the later two combinations, which deploy LB-MAC as their MAC protocols, outperform the LLR. Hence, it indicates that lifetime-balancing in MAC layer is more effective than in other layers.

At last, we evaluate the combination LLL, which deploy lifetime-balancing protocols in all the three layers. It is obvious from Figure 5.3, LLL has the best performance among all the combinations. However, the performance improvements contributed by the three lifetime-balancing protocols have different significance. According to the simulation result, we can see that LB-MAC achieves the most significant improvement, but the lifetime-balancing routing protocol has the least significant improvement. LB-MAC achieves better performance improvement due to following reasons: Firstly, as LB-MAC does not depend on the receiver to initiate packet transmission any longer, the transmission efficiency gets improved. Thus, the combinations with LB-MAC deployed can always achieve shorter end-to-end delay and higher data delivery ratio. Secondly, as radio costs the energy much faster than the other components in sensor according to

Moschitta and Neri (2016) and the most direct way to implement radio control is to go through the MAC layer, lifetime-balancing in MAC layer prolongs the network lifetime and reduces the wasted energy more effectively than other layers. To analyze the limited performance improvement introduced by the lifetime-balancing routing protocol (LB-Routing), please refer to Section 4.2 where we present LB-Routing. With LB-Routing, a node is able to update its routing table if and only if the new parent has longer lifetime and will not introduce extra end-to-end delay. Unfortunately, when LB-MAC balances lifetime in network, it may increase the end-to-end delay. In this case, LB-Routing tends to just maintain a fixed collection tree and its lifetime-balancing has been accomplished by the protocols at the application and MAC layers.

After having evaluated performance from all the eight combinations, we can observe that any of lifetime-balancing protocols is able to improve network performance independently. Meanwhile, the lifetime-balancing technique applied in the MAC layer plays a more important role than the others. We can also observe that lifetime-balancing protocols in these two layers are helpful to improve the network performance when combining with others.

#### 5.3.2.2 Simulation Results for 16 Nodes and 64 Nodes

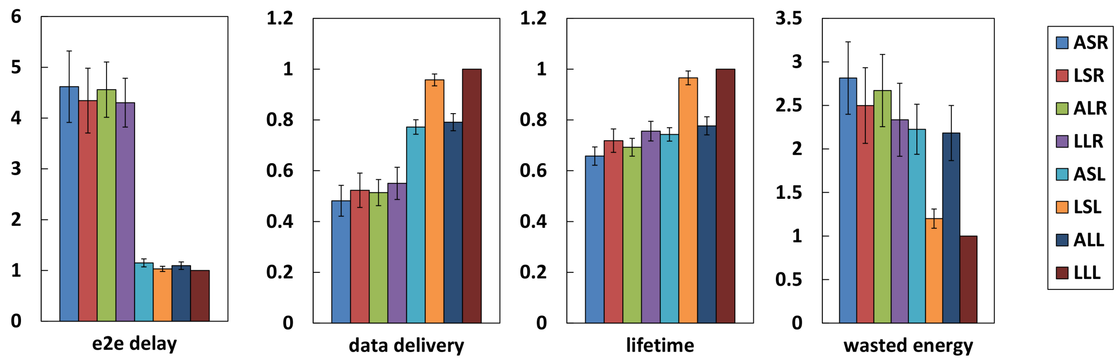


Figure 5.4 Simulation Result for 16 Nodes

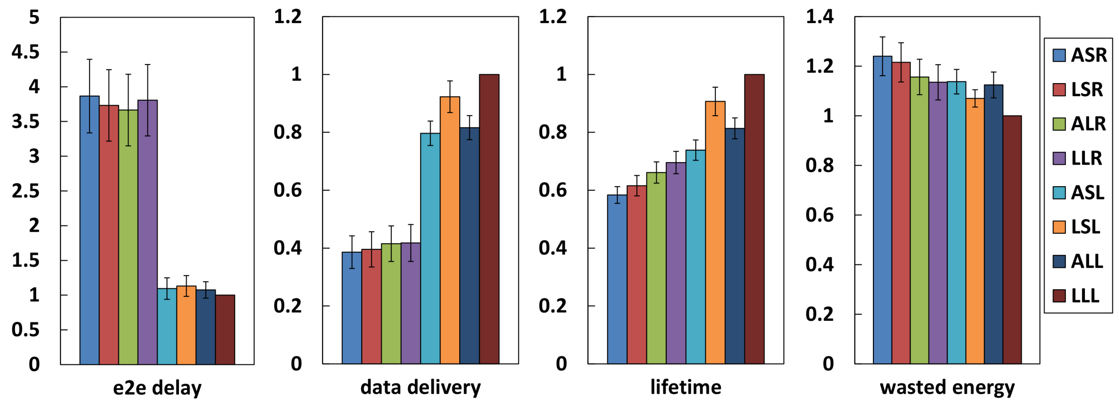


Figure 5.5 Simulation Result for 64 Nodes

In Figure 5.4 and Figure 5.5, we present the the simulation results for the settings where 16 nodes and 64 nodes are deployed respectively. From these two figures, we can get similar observations as we can from the simulations with 32 nodes deployed in Section 5.3.2.1.

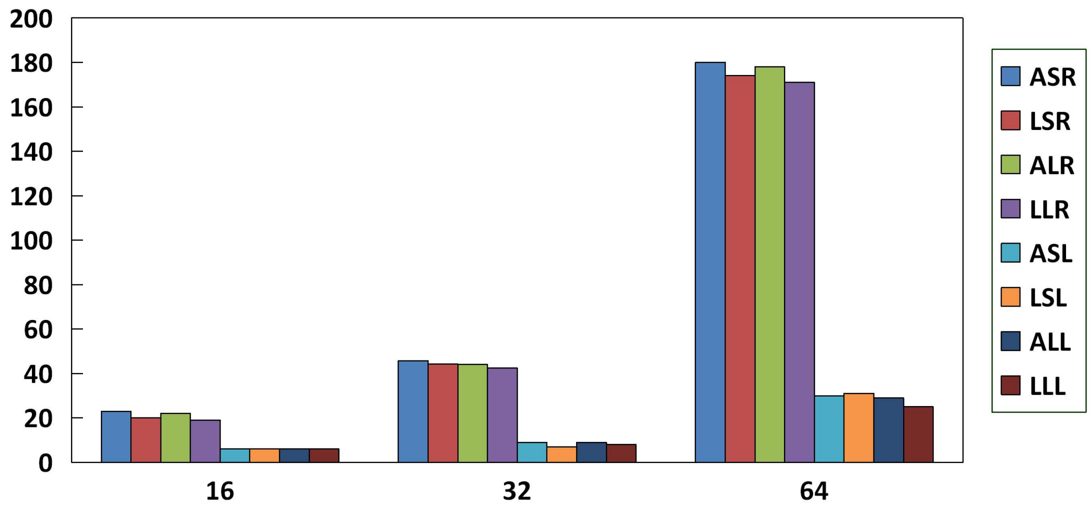


Figure 5.6 End-to-end Delay as Network Scale Varying

Figure 5.6 demonstrates that the end-to-end delay with all the eight combinations increases when the network scale gets larger. It is mainly due to the following reasons:

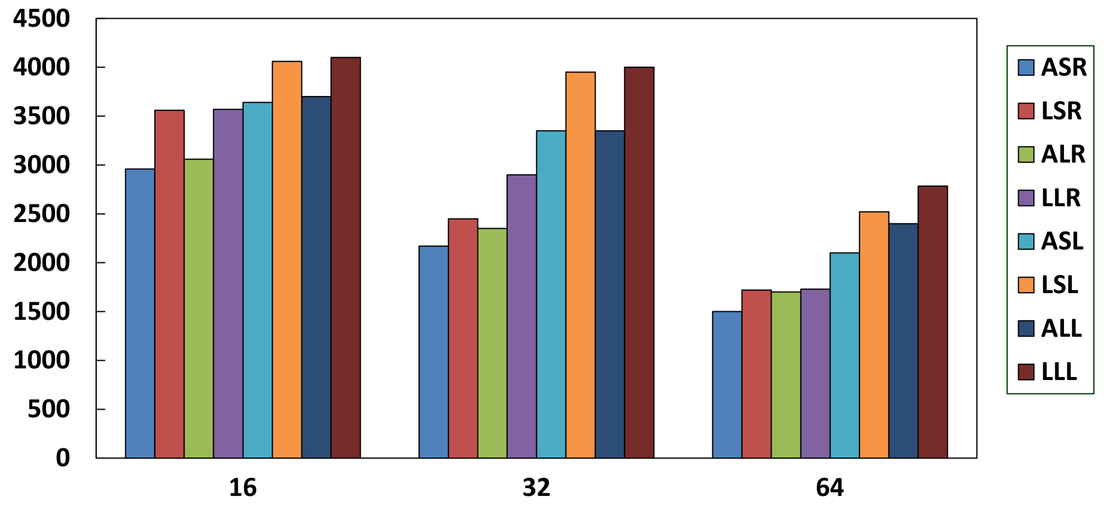


Figure 5.7 Network Lifetime as Network Scale Varying

As there are more sensor nodes in the network, more data packets are generated and transmitted. This increases the average hops between any node and the sink. Thus, the end-to-end delay is increased.

Figure 5.7 compares the network lifetime as network scale differs. As we have mentioned when analyzing the end-to-end delay, a larger network generates more packets and introduces more hops in sensory $DATA$ transmission, the network has to cost more energy in transmission.

From both figures, we can see those combinations with LB-MAC demonstrate good performance and the combination LLL, which deploys lifetime-balancing protocols in all three layers, achieves the best performance.

#### 5.3.2.3 Simulation Results as Data Interval Varying

In this section, we compare the network performance with different data intervals. Similar observations can also be obtained.

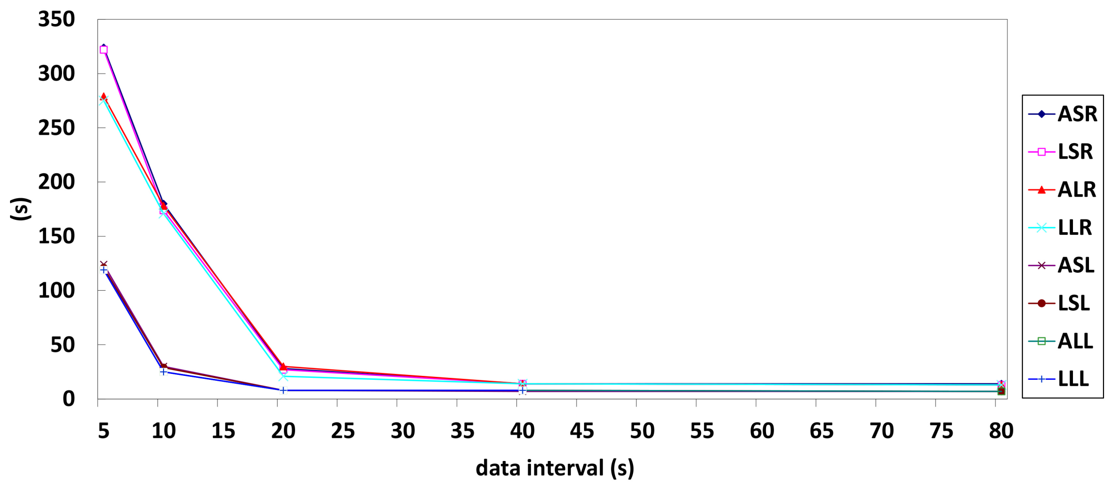


Figure 5.8 End-to-end Delay as Data Interval Varying

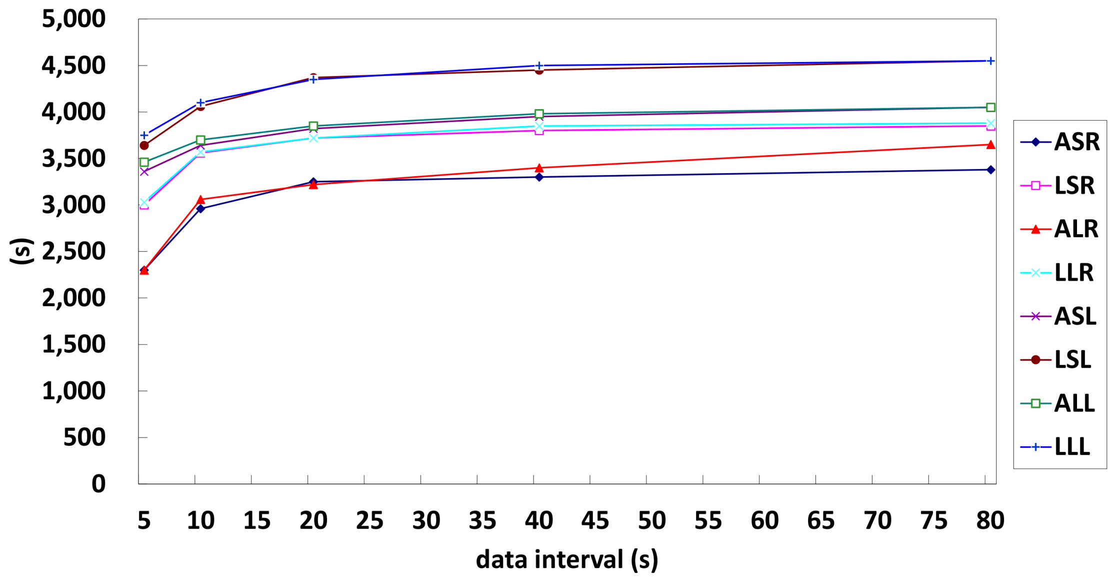


Figure 5.9 Network Lifetime as Data Interval Varying

# CHAPTER 6. CONCLUSION AND FUTURE WORK

Wireless sensor networks (WSNs) are expected to operate for months even years with limited power supply. Thus, many energy-efficiency solutions have been proposed to prolong the network lifetime. Unfortunately, as many applications define their network lifetime as the shortest nodal lifetime and energy-efficiency solutions can only reduce the energy consumption uniformly, such solutions are not able to resolve the bottleneck problem. To fill this gap, lifetime-balancing protocols from different layers are proposed to prolong the bottleneck nodal lifetime.

In this thesis work, we propose a solution with lifetime-balancing protocols in all of the application, routing and MAC layers. In the application layer protocol, we divide the network into clusters. In each cluster, a leader is elected to coordinate other nodes to switch among leading, sensing and backup modes. In the routing layer, we proposed a protocol, which calculates the routing metric as the combination of end-to-end delay and nodal lifetime to establish a collection tree, enables each node to transmit its sensory $DATA$ packets to the sink in a route with the longest lifetime. In the MAC layer, we modify LB-MAC to make it more compatible with the other layers.

Simulation has been conducted to evaluate the performance of our proposed scheme. The performance is evaluated in terms of end-to-end delay, data delivery ratio, network lifetime, and wasted energy. Based on the simulation results, we obtained following observations. Firstly, all the lifetime-balancing protocols in different layers are able to enhance the network performance independently, but the LB-MAC achieves the most significant improvement. Furthermore, if we combine any two of these lifetime-balancing protocols together, the system performance will be further improved; the combinations with LB-MAC deployed have better performance than the others. At last, our proposed system with all the lifetime-balancing schemes deployed achieves the best performance among all combinations. Meanwhile, we observe that the improvement introduced by the lifetime-balancing routing protocol is not significant. That is mainly because the routing algorithm treats end-to-end delay as a higher priority than balancing the lifetime.

In the future, we will improve the performance of our design on the routing layer. As current routing protocol is not allowed to update the collection tree if the new parent will increase the delay, we may not fully utilize the nodes with longer lifetime. Thus, the next step is to enable the routing protocol to adjust the MAC parameters to absorb the increased delay similar as discussed by Peng et al. (2013).

# BIBLIOGRAPHY

Peng, Y, Li, Z., Zhang, W., and Qiao, D. (2012). LB-MAC: A Lifetime-Balanced MAC Protocol for Sensor Networks. In *International Conference on Wireless Algorithms Systems and Applications (WASA)*, pages 272–291.

Lin, L., Shroff N., and Srikant, R. (2007). Asymptotically Optimal Energy-Aware Routing for Multihop Wireless Networks With Renewable Energy Sources. In *IEEE/ACM Transaction on Networking*, *15*(5), pages 1021–1033.

De, D., Song, W., Tang, S., and Cook, D. (2012). EAR: An Energy and Activity Aware Routing Protocol for Wireless Sensor Networks in Smart Environments. In *The Computer Journal*, *55*(12), pages 1492–1505.

Ren, F., Zhang, J., He, T., Lin, C., and Das, S. K. (2011). EBRP: Energy-Balanced Routing Protocol for Data Gathering in Wireless Sensor Networks. In *IEEE Transactions on Parallel and Distributed Systems*, *22*(12), pages 2108–2125.

Chang, J. and Tassiulas, L. (2000). Energy Conserving Routing in Wireless Ad-hoc Networks. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, *1*(1), pages 22–31.

Ye, M., Li, C., Chen, G., and Wu, J. (2005). EECS: An Energy Efficient Clustering Scheme in Wireless Sensor Networks. In *PCCC 2005. 24th IEEE International Performance, Computing, and Communications Conference, 2005.*, pages 535–540.

Raghavendra, C. S., Sivalingam, K. M., and Znati, Z. (2004). Communication Protocols in Sensor Networks. In *Wireless Sensor Networks Conference*, pages 33–35.

Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*. volume *8*, pages 8020–8029.

Li, Q., Aslam, J., and Rus, D. (2001). Online Poweraware Routing in Wireless Adhoc Networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 97–107.

Peng, Y., Li, Z., Qiao, D., and Zhang, W. (2013). $I^2C$: A Holistic Approach to Prolong the Sensor Network Lifetime. In *INFOCOM, 2013 Proceedings IEEE*, pages 2670–2678.

Sun, Y., Gurewitz, O., and Johnson, D. B. (2008). RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 1–14.

Peng, Y., Li, Z., Zhang, W., and Qiao, D. (2010). Prolonging Sensor Network Lifetime ThroughWireless Charging. In *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, pages 129–139.

Braynard, R., Silberstein, A., and Ellis, C. (2006). Extending Network Lifetime Using an Automatically Tuned Energy-Aware MAC Protocol. In *Proceeding EWSN'06 Proceedings of the Third European conference on Wireless Sensor Networks*, pages 224–259.

Ye, W., Heidemann, J., and Estrin, D. (2002). An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume *3*, pages 1567–1576.

Schurgers, C., Tsiatsis, V., Ganeriwal, S., and Srivastava, M. (2002). Optimizing Sensor Networks in the Energy-Latency-Density Design Space. In *IEEE Transactions on Mobile Computing*, *1*(1), pages 70–80.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). A Survey on Sensor Networks. In *IEEE Communications Magazine*, *40*(8), pages 102–114.

Huang, P., Xiao, L., Soltani, S., Mutka, M. W., and Xi, N. (2013). The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey. *IEEE Communications Surveys & Tutorials*, *15*(1), pages 101–120.

Dam, T. V. and Langendoen, K. (2003). An Adaptive EnergyEfficient MAC Protocol for Wireless Sensor Networks. In *Proceeding SenSys '03 Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180.

Qin, M. and Zimmermann, R. (2005). An Energy-Efficient Voting-Based Clustering Algorithm for Sensor Networks. In *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pages 444–451.

Meier, A., Woehrle, M., Zimmerling, M., and Thiele, L. (2010). ZeroCal: Automatic MAC Protocol Calibration. In *DCOSS'10 Proceedings of the 6th IEEE international conference on Distributed Computing in Sensor Systems*, pages 31–44.

Moschitta, A. and Neri, I. (2016). Power consumption Assessment in Wireless Sensor Networks. In *ICT-Energy-Concepts Towards Zero-Power Information and Communication Technology*, pages 203–224.

Crossbow Technology Inc (2016). TelosB Mote Datasheet.

*www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf*.

Monnit Corporation (2016). OEM RF Wireless Temperature Sensor Datasheet.

*resources.monnit.com/content/documents/datasheets/commercial/MD001-Temperature-Sensor-Data-Sheet.pdf*.

National Instruments (2012). What Is a Wireless sensor network?.

*www.ni.com/white-paper/7142/en/*.

Wikipedia (2016). Wireless sensor network.

*https : //en.wikipedia.org/wiki/Wireless_sensor_network*.