LOCAL COORDINATION MEDIUM ACCESS CONTROL FOR WIRELESS SENSOR NETWORKS

By

MONIQUE SACHIE KOHAGURA

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

AUGUST 2008

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of MONIQUE SACHIE KOHAGURA find it satisfactory and recommend that it be accepted.

Chair

ACKNOWLEDGEMENT

I would like to thank Dr. Murali Medidi for being a wonderful advisor. His guidance and inspirational talks helped me to get to where I am today. I would also like to thank Dr. Sirisha Medidi and Dr. Roger Alexander for being on my thesis committee. Finally, I would like to thank the members of the SWAN Lab for their support: Ghayathri Garudapuram, Yuanyuan Zhou, Jiong Wang, Peter Cappetto, Lynsey Compton-Drake, and Christopher Mallery.

LOCAL COORDINATION MEDIUM ACCESS CONTROL FOR WIRELESS SENSOR NETWORKS

Abstract

by Monique Sachie Kohagura, M.S.
Washington State University
August 2008

Chair: Muralidhar Medidi

Energy-efficient medium access control (MAC) protocols have been at the forefront of research within wireless sensor networks (WSN) due to the unique resource-constraints of sensor nodes. With limited processing capabilities, memory space, and battery power, a node is unable to perform complicated tasks and computations nor can it accurately assess an event alone. Hence, the role of a base station (sink): the destination for all packet reports to be analyzed. With heavy contention at the event site and traffic becoming more concentrated towards the sink (known as *convergecast*), a new perspective in designing a MAC may be the key to better energy-efficient transmissions. Most MAC protocols have addressed the issue of contention and collision resolution at the traditional micro level, trying to access the channel to send a packet to a given destination, which is determined by the upper routing layer. However, with WSNs, one should look at communication from a macro perspective since most applications share the same traffic pattern. We propose an energy-efficient cross-layered MAC protocol which allows a node to locally coordinate amongst others for productive event reporting, named *LoC-MAC*. ECR-MAC [57] is used as a basis and we improve upon it by allowing each node to maintain a list of its multiple next-hop forwarders' duty cycle schedules in order to identify when it can ask a forwarder to perform data exchange. Performance evaluation has shown that LoC-MAC has comparable results with ECR-MAC, while consuming less energy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dedication

This thesis is dedicated to my parents Michiko and Yukihiro, brother Sean, and my love, Randall.

CHAPTER ONE

INTRODUCTION

1.1 Overview

Wireless sensor networks (WSN) consist of tens to thousands of nodes, each typically being battery-operated and equipped with a radio transmitter to communicate with other nodes, one or more sensors for gathering data, and a processor for carrying out tasks and refining data. Types of applications range from event-triggered (e.g. movements in a battlefield, a forest fire) to observational (e.g. periodically monitoring habitats or weather patterns). In either case, a sensor node has the simple task of collecting and reporting data to a *base station*, where the data is analyzed, interpreted, and acted on. Instead of the conventional network paradigm where each node is seen as an autonomous system, the collective work and collaboration between all the sensor nodes are what comprises a total system.

In order to deploy such large dense networks, both the manufacturing cost and size of each node should be minimal, resulting in affordable, but resource-constrained sensor nodes. Memory, battery life, and computational power are limited. Hence, there has been a shift in major focus from traditional network protocol design, such as latency, fairness, and reliability, to energy-efficiency and simplicity. The former requirements are still applicable, but are dependent on individual sensor applications while the latter is essential to the longevity of all sensor systems.

Radio transmitting operations are a major contributor to energy consumption, especially in cases of collision, overhearing, packet overhead, and idle listening [51]. Reducing the occurrence of these cases requires some form of coordination amongst nodes when a data transmission is to take place. For example, to avoid collisions, nodes within the same transmission range should not send messages at the same time. To avoid overhearing, nodes that are not involved in the current message exchange within their region should turn off their radio. These issues are addressed at the medium access control (MAC) layer, where a MAC protocol defines when a node should access the wireless medium for data communication. Besides handling contention and collision resolution, the notion of duty cycles (or sleep/wake schedules), where a node switches its radio transmitter on and off periodically to save energy, has been intertwined in most WSN MAC protocol. While this saves energy due to reduction in idle listening and overhearing, another layer of complexity has been added in that nodes are no longer free to transmit whenever they want since there may be no one available for message reception. Thus, there is a further need for a node to better coordinate transmissions with its neighbors.

To facilitate efficient communication, one could analyze the type of traffic that occurs most frequently. Data-gathering WSNs involve mainly source-to-sink traffic (convergecast [27]), which projects an image of the network as a tree in terms of routing (i.e. the sink is the root, with all paths leading to it). Most WSN MAC protocols assume this model, where a child node has one parent node that acts as its forwarder, scheduling sleep/wake periods in a way that would allow the child and parent to communicate. However, this presents an issue in event-triggered sensor applications. Nodes that sense the same event will want to transmit around the same time, resulting in severe *spatially-correlated contention* for those that share the same parent or are in the same tree level [7, 23]. Packet delay, throughput, and energy usage all suffer from this bottleneck. One method of relieving this is through the use of data aggregation, which summarizes an ongoing event by amalgamating collected data from several sources within the event region, but reduces the accuracy of the data itself [34]. In the case of a large event that spans several regions or multiple different events occurring around the same area, there is still the issue of major contention.

Recently, techniques employing the availability of multiple potential forwarders, and therefore having multiple dispersed paths, have been proposed. Keshavarzian *et al.* [25] and Liu *et al.* [30] has shown the benefits of selecting a forwarder from a subset of neighbors based on certain metrics, such as delay, link stability, etc. Zhou *et al.* [57] discusses and overcomes spatially-correlated contention by dynamically selecting a forwarder through their Energy-Efficient Contention Resilient

MAC (ECR-MAC). When a node wants to send a data packet, a stream of periodic wake-up packets are transmitted and the first potential forwarder to receive an uncorrupted wake-up will reply to the sender to notify that it can act as the forwarder.

We propose Local Coordination Medium Access Control *LoC-MAC*, a cross-layered MAC protocol which uses ECR-MAC as its basis. It is an improvement upon ECR-MAC in that a node obtains and maintains the wake-up time (i.e. channel sampling period) of its forwarders as a means for locally coordinating when and with whom a node should forward its data. This reduces the time in which a node is awake and the number of control packets that are sent, thus reducing energy consumption. Results show that LoC-MAC have comparable throughput and delay, while consuming less energy than ECR-MAC.

1.2 Thesis Organization

The rest of the thesis is organized as followed: Background information on wireless sensor networks and medium access control and Related Work are in Chapter 2, Motivation and Protocol Design are in Chapter 3, Performance Evaluation in Chapter 4, and finally, Conclusions in Chapter 5.

CHAPTER TWO

BACKGROUND AND RELATED WORK

2.1 Wireless Sensor Networks

Wireless sensor networks (WSN) differs from traditional networks in several aspects. Nodes are resource-constrained in terms of energy capacity, radio transmissions, processing capabilities, and memory storage in order for large deployments to be cost-effective. To ensure the longevity of a network, energy-efficient processes has become a huge focus in WSN protocols. Another difference is the type of traffic that takes place, the main one being convergecast in WSN where nodes that sense an event want to send their reports to the same destination, the base station. Forward-direction traffic is also possible, where the base-station may send queries or occasional updates to the rest of the nodes in the network. However, nodes communicating with other individual nodes that are not their direct one-hop neighbor (i.e. *local gossip*), are uncommon in event-triggered applications. While the amount of traffic is generally low (it all really depends on the frequency of an event), it tends to be bursty. Multi-hop communication is commonplace in WSNs that span a large region. Each node may increase its transmission range to reach the base station directly, but this requires a great amount of power, which quickly depletes its battery life. Not only that, it would take a significant amount of effort to coordinate amongst all the nodes to avoid network-wide collisions. By having nodes work locally in smaller regions, the throughput is higher and energy consumption is dispersed among all nodes. Figure 2.1 displays an example image of a densely deployed network with an event occurring in the bottom right corner and the base station in the upper right corner.

The unique nature and numerous uses of a WSN has spurred a multitude of research within the wireless networking area. Examples include redesigning traditional MAC, routing, and transport protocols and tackling more specific issues such as localization, coverage, and data aggregation.

Due to the limited resources of a sensor network, both in terms of capability and energy, researchers have taken into consideration the need for simple and energy-efficient designs in order to extend the lifetime of the network.

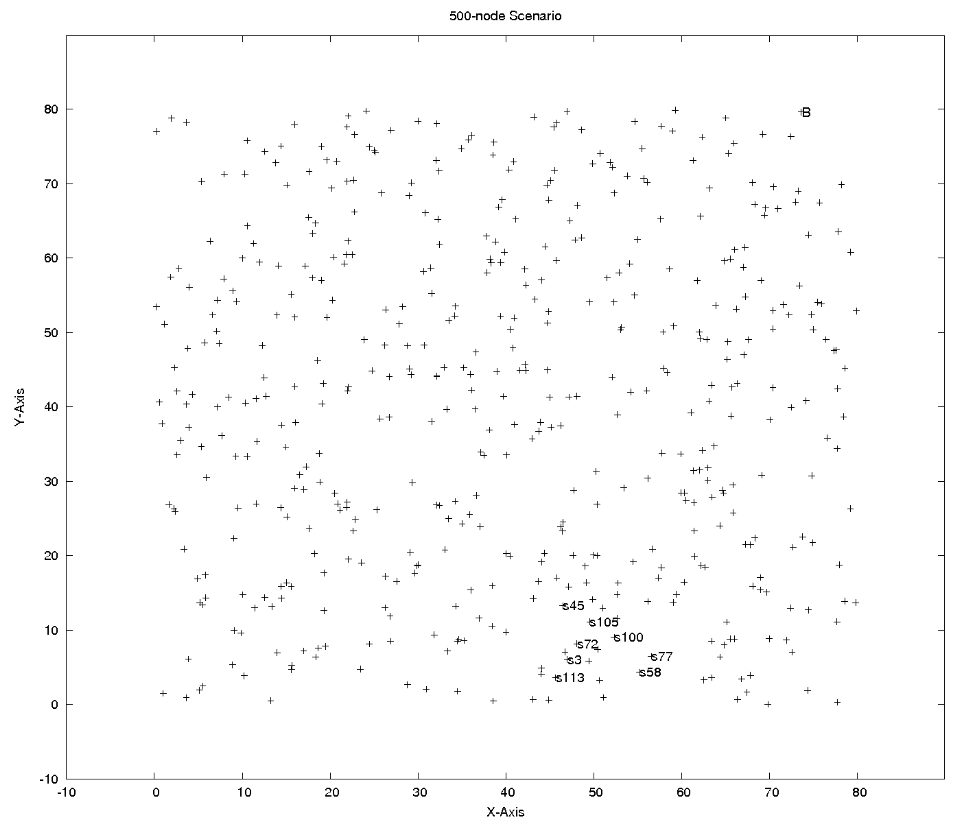


Figure 2.1: Example scenario of nodes sensing the same event

2.2 Energy Efficiency

In contrast to other wireless devices (e.g. laptops, personal digital assistants) where batteries are easily replaced or recharged by a user, a sensor node must work with whatever initial energy it is given. Since they are deployed in the hundreds to thousands range and sometimes in harsh environments, it would be infeasible to replace the battery of every single sensor node. While some [45] have suggested harvesting solar energy, the idea does not work in all environments such as those deployed underwater, in caves, or overcast areas. Battery technology will gradually improve over time, but energy-efficient designs will always be applicable in the present and in the future.

One of the major source of energy consumptions comes from the frequent use of the radio transmitter. The main cases where the transmitter is used are listed and explained in table 2.1. To address these energy issues, researchers have been working through the medium access layer for controlling node transmissions since contention and collision resolution are its major tasks.

Table 2.1: Sources of Energy Consumption

Collision	A collision occurs when two or more nodes transmit messages around the same time in an overlapping region. The receiving nodes within that region end up with corrupted packets that are discarded and the senders must retransmit their packets once more, resulting in energy wastage on both ends.
Overhearing	Occurs when a node receives a message that was not destined for it. Due to the broadcast nature of most wireless devices, a node may frequently end up overhearing more-so than receiving.
Packet Overhead	Control packets are used in accomplishing various tasks, but should not become a burden on the network by over-transmitting them and dominating the channel.
Idle Listening	Within sensor networks, the amount of traffic may be predictable (e.g. occasionally sending message on the current status of the environment) or unpredictable (e.g. event-based). Either way, it is not required for all nodes to have their radio transmitters on at all times, wasting energy when nothing is going on.
Traffic Fluctuations	With event-triggered applications, traffic can go from non-existent to overload. A sudden spike in traffic increases the probability of collisions and the amount of time spent carrier sensing (i.e. checking if the channel is idle).

The location and role of a sensor node are other factors in the amount of energy that is consumed. Those that are closer to the base station will deplete its energy more quickly since they have a heavier traffic load than those that are further away (termed the *energy hole problem* [28]). Several ways to alleviate this is by deploying more nodes or special nodes with more battery power near the base station, or aggregating data to reduce the traffic flow. In terms of roles, some nodes may have more responsibilities than others, requiring additional computations or transmissions.

For example, being the data aggregator or being a part of a communication backbone. To overcome this without having a detrimental effect to the network, these roles are either rotated amongst other neighboring nodes or replaced by another the moment it dies.

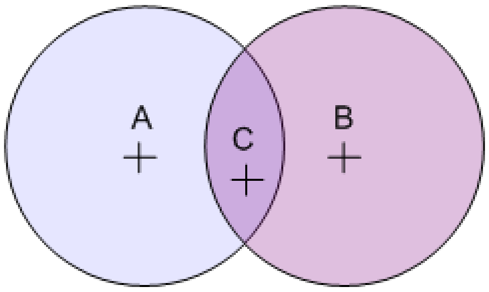
2.3 Medium Access Control

Research of the medium access control (MAC) layer is a flourishing area within wireless sensor networks. Typical protocols that work for both wired and wireless networks cannot easily translate over to sensor networks due to the resource constraints. For example, frequency-division multiple access (FDMA) would require a node to be equipped with a radio transmitter that will allow it to send and receive messages at different frequencies. With code-division multiple access (CDMA), a node would have to perform computational-intensive calculations to create and interpret signals, putting additional stress on the processor. In time-division multiple access (TDMA), nodes are required to be synchronized (e.g. all must have the same sense of time) in order for them to know exactly when they are allowed to transmit. Carrier-sense multiple access (CSMA) is the most simplest in design, where a node has to check if the medium is idle before it is allowed to transmit and if not, wait for some random time before checking again. FDMA and CDMA puts an additional strain on the node, both in terms of hardware costs and energy consumption, making them an infeasible solution for WSNs.

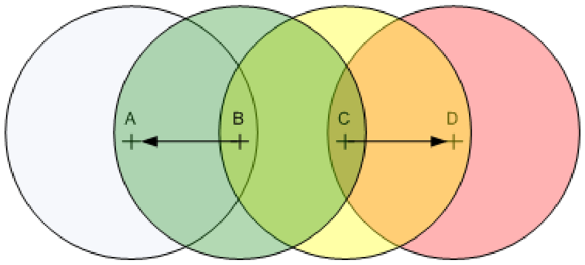
The two types of channel access methods that are most commonly proposed within WSN are CSMA-based and TDMA-based. Both have their pros and cons in terms of packet delays, bandwidth usage, and handling collision. Most MAC protocols have adopted both methods to some degree, using CSMA for handling contention and TDMA for determining when to transmit. To further reduce the energy wastage that occurs from transmitting needlessly, an important mechanism was introduced in [51]: the utilization of sleep/wake schedules, where each node follows a pattern of turning on and off its radio transmitter (and possibly other components such as the sensor) periodically. When a node is asleep, it is not allowed to transmit or receive any messages, thus decreasing idle listening and overhearing. Integrating this notion into the MAC layer has become a standard for saving energy and is assumed in most protocols.

A common base of comparison for WSN MACs is IEEE 802.11, which uses CSMA/CA (collision avoidance). It acts as the optimal example of delay, but worst case for energy consumptions since there is no duty cycle. In addition to regular carrier sense, *virtual carrier sense* is also performed to overcome the hidden terminal and exposed terminal problems (see Figure 2.2). When a node has data to send, it will first transmit a request-to-send (RTS) packet to the intended receiver, whom replies back with a clear-to-send (CTS) packet. Those that are not involved with the communication process, but overhears the RTS or CTS packet will know to refrain from transmitting anything for awhile, overcoming the hidden terminal problem. To conquer the exposed terminal problem, a node that overhears an RTS but not a CTS can confirm that it is safe to transmit. After receiving the CTS, the sender will know it is safe to transmit, sending its data packet. To confirm the reception, the receiver will send back an acknowledgment (ACK). Most WSN MACs have adopted this four-way handshake for reliable transmission of data. However, virtual carrier sense does not work well with randomly deployed duty cycles. For example, in the case with Figure 2.2(a) Node B might be sleeping when Node C sends a CTS back to Node A. During the middle of Node A's data transmission, Node B might awaken and find that the channel is idle, trying to send out its own data packet which results in Node C receiving a corrupted packet. Without the information of a node's two-hop neighbors' schedules and controlling the send time of each, both the hidden and exposed terminal problems continue to exist within sensor networks.

To subdivide WSN MACs, we considered whether a node maintains a synchronized or independent sleep/wake schedule. With synchronized schedules, groups of nodes follow the same schedule of going to sleep and waking up. During the wake period, nodes generally contend for a short period and the winning node transmits its data packet to a neighboring node. With independent schedules, each node adopts a random wake/sleep schedule. The wake periods are generally used for channel sampling (i.e. checking of the channel is busy), allowing for lower duty cycles.



(a) Hidden Terminal Problem - Node A cannot hear Node B's transmissions (and vice versa) resulting in packet collisions at Node C if the other two transmits around the same time



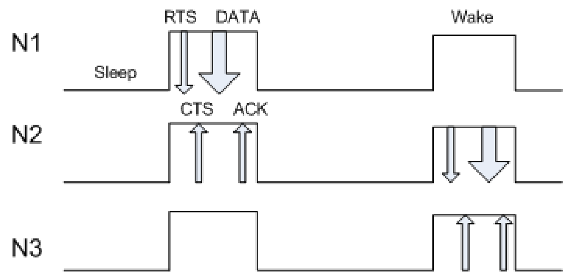
(b) Exposed Terminal Problem - Node B wants to transmit to A, while node C wants to transmit to D, but only one of them is allowed to transmit

Figure 2.2: Wireless Network Problems

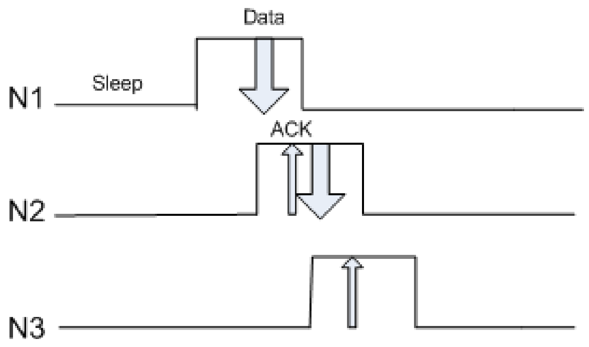
For further categorization, there is the notion of employing a single forwarder or multiple forwarders. Most WSN MACs fulfill the traditional role of not dealing with whom the forwarder is, since the selection is generally the routing layer's job, hence the assumption of a single forwarder for most MACs. However, recent MACs are utilizing multiple forwarders for better handling the type of traffic that is most frequent and special to WSNs, spatially-correlated convergecast. The rest of this section summarizes and discusses the differences in types of methods and the protocols that have been proposed for wireless sensor networks.

2.3.1 Synchronous Schedules

With synchronous schedules, nodes are assigned a sleep/wake schedule that is either shared with their neighbors or follows a topological pattern. These schedules require some form (either global or local) of synchronization, where nodes share the common sense of time and can therefore accurately identify when a neighbor is available for communication. Besides the initial set-up cost, synchronized schedules require additional maintenance to adapt to network changes (e.g. dead nodes and adding new nodes) and overcome clock drifts, a phenomenon where the internal clocks of each node starts to deviate since they do not run at the exact same speed. Figure 2.3 gives two examples of synchronous schedules, (a) shows three nodes all sharing the same schedule while (b) displays a staggered (or ladder) scheduling, where a node does not have to wait long before forwarding data to another neighbor.



(a) Same Schedules



(b) Staggered Schedules

Figure 2.3: Examples of Synchronous Scheduling

Single Forwarder

The general archetype of a MAC is to decide when to access the wireless medium for transmission. It already knows who the intended receiver is, assuming that the network layer's routing protocol is the one that has determined who the next-hop forwarder is. When multiple nodes share the same forwarder, which commonly occurs in tree-based routing protocols, they must compete for it if they all have data to send at the same time. Having such a bottleneck in the area where the event occurred is detrimental to the timely delivery of packets and will deplete the forwarder's energy quickly. Most of the following protocols have not explicitly inspect this issue of spatially-correlated contention, testing only one to two neighboring sources or multiple randomly scattered sources.

Ye *et al.* [51] introduced the concept of duty cycles which is employed in their MAC protocol ‘sensor MAC’ (S-MAC). Each node goes through a period of discovering their own sleep/wake schedule, either by randomly selecting a time to go to sleep and broadcasting the message or by receiving a schedule message from a neighbor and adjusting to it. Certain nodes will end up adopting two schedules, the one it first selected and the one it receives after from a neighbor. This will allow each node to be able to communicate with at least one neighbor during their wake period. To overcome clock drifts and adjust to topological changes, the schedules must be synchronized periodically. For collision avoidance, the authors adopted the RTS/CTS mechanism from CSMA/CA (collision avoidance). Nodes that overhear an RTS or CTS packet goes to sleep for the duration of the data packet transmission to avoid both transmitting and further overhearing. When it comes to forwarding long messages, they suggest fragmenting it into smaller packets and using only one RTS and CTS to reserve the channel. To test the capabilities of S-MAC, the protocol was implemented on a testbed of Rene Motes (a two-hop network with five nodes). IEEE 802.11 and a simpler S-MAC with only overhearing avoidance were also implemented as a comparison. Results have shown that S-MAC consumes half as much energy as 802.11, some due to the reduced idle listening and mostly due to avoiding overhearing and transmitting a long message efficiently. In [52], the authors modified S-MAC to include adaptive listening which decreases latency by as much as five times when the hop distance is long (e.g. 10 hops) and even some energy consumption compared to the original S-MAC. When a node overhears a RTS or CTS packet, it will wake up for awhile after the end of the current transmission which, in the case that it happens to be the next-hop node, allows it to forward a packet quickly instead of having the neighbor wait until the node wakes up again. *Key contribution: Introduced duty cycles where nodes have locally synchronized fixed sleep/wake schedules which are formed through virtual clusters.*

In [12], the authors have proposed Timeout-MAC (T-MAC) to improve upon S-MAC’s fixed active periods, the time used for nodes to communicate with each other. T-MAC shortens the active period if an event (such as receiving a packet) does not occur for a certain amount of time, which should be long enough to include the contention interval, receiving an RTS packet, and the time until the CTS packet is sent. To increase throughput in the case where several nodes are contending for the medium at the same time, the authors have proposed two solutions. One method involves allowing a node to send a 'future request-to-send' (FRTS) if it overhears a CTS packet that was destined elsewhere, notifying its own neighbor to wake up when the other transmission ends. At the same time the FRTS packet is transmitted, the other node that has won the medium will send out a 'data send' packet before transmitting any real data to avoid collisions between the FRTS and data packets. While this indeed increases throughput, the control packet overhead increases the energy consumed. The other method suggested is to give sending priority to nodes with a full buffer. If a node with a full buffer has lost contention at least twice (a threshold) and is asked to be a receiver, it will not send back a CTS and instead send its own RTS to a neighbor. This was shown to increase maximum throughput in unidirectional traffic without being detrimental to heavy loaded omnidirectional traffic. T-MAC was implemented in OMNeT++, a discrete event simulator, and compared against CSMA and S-MAC in terms of energy efficiency. In both local unicast, nodes-to-sink, and event-based scenarios, T-MAC is shown to consume less energy than both S-MAC and CSMA (which has no energy saving features). The authors briefly mentioned that the maximum throughput of T-MAC is at worst 70% of what S-MAC's is, suggesting that under heavy loads, methods such as data aggregation may be used. Implementing the FRTS and full buffer priority methods has shown that the combination of the two results in better throughput, but at the same time consumes more energy. After the simulations, T-MAC was implemented on the EYES wireless sensor nodes and results have showed that they saved about 96% in energy, for reducing idle time. *Key contribution: Shortening wake periods by going to sleep early if nothing is going on, resulting in less idle time.*

TRaffic-Adaptive Medium Access (TRAMA) was introduced in [37], using adaptive transmission schedules to avoid packet collisions and having nodes switch their radio to low power mode when there is no packet to send or receive to save energy. There are three main components to

TRAMA: the Neighbor Protocol (NP) detects a node's two-hop neighborhood; the Schedule Exchange Protocol (SEP) allows nodes to trade current information on its own traffic and the intended receivers; and the Adaptive Election Algorithm (AEA), which uses SEP data to determine which nodes can act as the transmitter and receiver during the current time slot while the rest switch to low power mode. The types of time slots consist of random access (i.e. contention-based) *signaling slots* and scheduled *transmission slots*. Signaling slots are used for gathering two-hop neighborhood information (used with NP) while transmission slots are used for exchanging data and traffic-based information (during SEP). TRAMA has been implemented in Qualnet (a simulation platform) and compared against CSMA, IEEE 802.11, S-MAC, and NAMA (Node Activation Multiple Access). Types of scenarios tested include having nodes transmit data to a sink that is in the corner and a sink that is in the center. TRAMA is shown to have higher delivery packet ratio than its contention-based competitors, but higher queueing delay. Its average sleep time varies upon the type of scenarios, but follows the general trend of increasing linearly as traffic decreases, whereas S-MAC has a constant trend of sleep time. *Key contribution: Traffic adaptive scheduling for higher throughput.*

The authors of [31] have identified three shortcomings of a fixed low-duty cycle MAC protocol. First, the accumulation of sleep latency - when a node has to wait for its neighbor to wake up before it can send a message; secondly, the inability to adapt to traffic variations such as low and high loads; and thirdly, the increase in probability of collision when nodes wake up at the same time and contend for the medium. To address these issues, Lu *et al.* proposed DMAC, a protocol which allows for continuous data forwarding through the usage of staggered active/sleep schedules. This is implemented through the use of a data gathering tree representing the network, where the general pattern is that the parent of a child node wakes up shortly after the child has woken up. To reduce collisions amongst nodes within the same tree level, nodes must backoff and wait for some random time before transmitting any packet. In the case that a node has multiple packets to send, it sets a *more data* flag in the MAC header to notify the forwarder that it should stay awake longer (the forwarder will decide whether it will stay awake longer by setting the *more data* flag in an ACK that is sent back). In the case where two nodes are within the same transmission range, but have separate parents, only one of them can send its data while the other will have to wait an entire cycle before it can transmit its own. To reduce the delay time, a node can send a *more-to-send* (MTS) packet to its parent to make it wake up more often. DMAC was implemented in ns-2, a network simulator, and compared against S-MAC with adaptive listening and CSMA/CA in terms of energy consumption, latency, and delivery ratio. In a multi-hop chain and tree-based scenario, DMAC consumes the least energy among the two and has less delay than S-MAC, but higher than CSMA/CA (which does not have a periodic sleep schedule). *Key contribution: Staggered scheduling for minimal delay.*

Zebra MAC (Z-MAC) [38] uses CSMA during low contention and TDMA during high contention. CSMA allows for better channel utilization since nodes can contend for the medium whenever they have data to send, while TDMA reduces the occurrence of collisions by scheduling when a node can send. During the initialization of the network, Z-MAC performs the following operations only once: neighbor discovery (to gather two-hop neighbor slot and frame information), slot assignment (using the DRAND [39] algorithm), local time frame exchange (to fit its own local neighborhood size), and global time synchronization. When there is low contention, a node does not have to follow its own time slot (although it has priority over it) and can use other slots which may or may not belong to another nodes. Using the interfaces of B-MAC, Z-MAC backs off for some random time, performs carrier sense, and transmits its data if the channel. If the channel is not clear, the node will wait until it is. In the case that there is high contention, a node can only transmit during its own time slot or a slot that is not owned by a two-hop neighbor. To find out if there is high contention, a node must receive an explicit contention notification (ECN) from a two-hop neighbor that has assessed the amount of collisions have been occurring. The high contention level times out after some time, going back to a low contention level, unless it continues to receive ECN messages. To assess the performance of Z-MAC, it has been implemented on both ns-2 and

Mica2 motes. Results are compared to PTDMA, Sift, and B-MAC in one-hop, two-hop, and multi-hop scenarios. Z-MAC is shown to have higher throughput than B-MAC as the number of sources increases on the Mica motes, but consumes more energy. *Key contribution: Combines CSMA and TDMA to benefit from both types to obtain high channel utilization during low contention and reduce collisions during high contention.*

In [53], the authors designed a MAC protocol called Scheduled Channel Polling (SCP-MAC), which can achieve ultra low duty cycles of 0.01% - 0.1%. Similar to low power listening (LPL) protocols (such as B-MAC and WiseMAC - later discussed in the independent schedules section) SCP-MAC performs channel polling periodically, checking if activity is going on rather than checking on the activity itself (which is done in S-MAC). However, SCP-MAC differs with other LPL protocols in that all neighboring nodes poll the channel in a synchronized manner. Nodes no longer need to send a long preamble to notify a neighbor when it has data to send and can instead send a quick tone during the channel polling period. A node will perform carrier sense (selects a random slot as to when this is done) to check if the channel is idle, before it sends out a tone. After sending it, the node will check the channel again and if it is idle, it will send the data packet. Additional mechanisms, such as RTS/CTS, can be added to SCP-MAC in order to deal with high network contention and other ordeals which may be specific to an application. To maintain these synchronized schedule, the authors have proposed two methods: periodic synchronization interval where nodes can exchange their schedules or a piggy-back method, where nodes that send data attach their schedule to it as well. SCP-MAC was implemented in TinyOS, using Mica2 Motes and compared against a generic LPL protocol. In a 10-node one-hop scenario, SCP-MAC is shown to consume less energy and has higher throughput. *Key contribution: Synchronized channel sampling to obtain ultra low duty cycles.*

Dozer is a data gathering protocol comprised of a MAC, topology control, and routing mechanism, stating that having a solid integrated network stack is the way to achieve minimum power consumption [9]. A tree structure is used as a representation of the physical network, where the root is the sink. With the use of local synchronization, each node fulfills two different roles and therefore has two different schedules, one as a parent to possibly multiple children and the other as a child itself to one parent. As a parent, the node will decide the sending schedule for each of its children, while as a child, it will only have one slot to send data to its own parent. Schedules are created, adjusted, and known through the use of beacons. During initial setup, all nodes tries to join the data gathering tree by listening to beacon messages transmitted by neighboring nodes (which are transmitted periodically to allow new nodes that have just joined the network). To create a TDMA schedule, a node uses a rating function, for example, using a node's distance to the sink or current traffic load. Network conditions may change over time, so each node keeps a list of potential parents in the case that its own parent is no longer working (e.g. consecutive transmissions have failed multiple times). If a node failed in finding a replacement parent, it will go back to scanning the channel for beacons periodically. When a node is allowed to send data, it will try to transmit all of its queued packets to its parent, dequeueing a packet only if it has received an ACK back. The ACK contains an additional message, notifying the child how many more packets it is willing to receive. Although the tree is mainly meant for traffic towards the sink, Dozer also allows the sink to send command messages to all the nodes through the use of sending it in beacons. For experimentation, Dozer was implemented over TinyOS, using the TinyNode 584 sensor platform, in a network of 40 nodes which ran for one month. On average, a node had a duty cycle of 1.67%. *Key contribution: Double schedules for each node.*

Application-Adaptive MAC (A^2-MAC) recognizes the differing needs among various WSN applications [56]. For control packets, slotted non-persistent CSMA is employed (i.e. if the channel's busy, backoff for some random time, else send). For data transmissions, a TDMA-like schedule is built based on the applications needs, for example, by giving more slots to nodes that generate packets at a high rate and less slots to those that send at a low rate. A^2-MAC divides the network into logical cells, where there is a root node (a position which is rotated occasionally) for each cell. The root node acquires information on the other nodes within the cell and determines their scheduled send time, while it remains awake most of the time for reception from the other nodes. To communicate with another cell, a root node transmits data to another root node. To evaluate the performance of A^2-MAC, analytical models were built with results showing that it saves more energy than CSMA. *Key contribution: Scheduling built upon packet sending rates of nodes.*

Multiple Forwarders

The concept of selecting a forwarder from a list of neighboring nodes generally resides in the network layer. It is the routing protocol's task to select a forwarder based on a certain metric, such as one that would result in the shortest path to the destination or another where the path is reliable in terms of link communication. However, this information can be used at the MAC layer also to better schedule wakeup/sleep periods.

Considering the type of traffic in WSN (source-to-sink and sink-to-all nodes), Keshavarzian *et. al* proposed several wakeup schedules, one of them involving a *multi-parent* method which allows a node to choose a parent forwarder that will allow the for the fastest delivery of its packet [25]. This combined with a staggered schedule can better handle contention without impairing packet delivery delay. However, their algorithm for assigning parents (basically a graph-coloring problem) was shown to be NP-complete, providing a heuristic instead for finding two parents per node. The algorithm is centralized, where the base station or another node with enough computational power determines the schedules, given information on the connectivity between nodes in the network. There are several different approximation phases, each involving determining what layer a node and its possible parents are in. To test the success rate of finding a valid coloring for each phase, the layering algorithm was implemented in Matlab and C++. *Key contribution: Heuristic for multi-parent scheduling.*

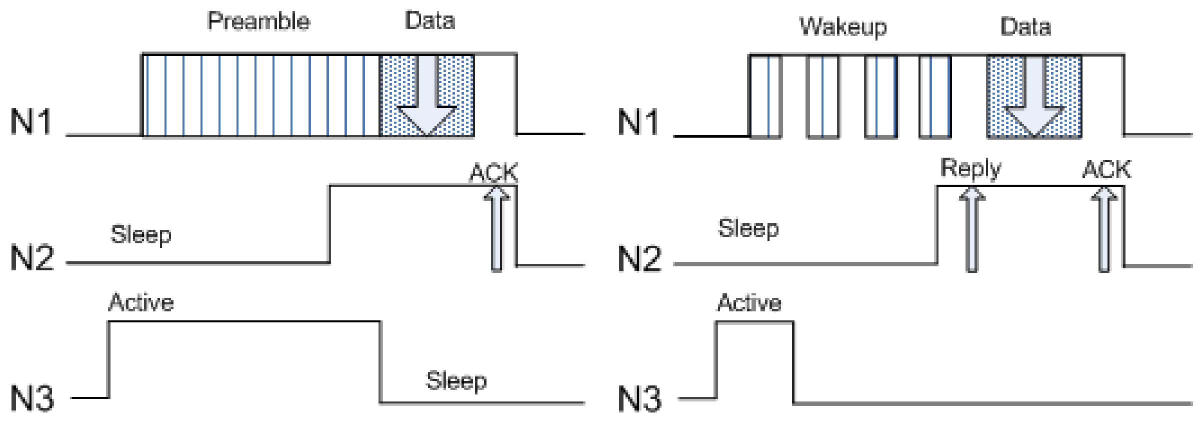
In [58], Zhou and Medidi proposed a topology control in the form of a MAC protocol: Multi-parent Staggered wakeup scheduling MAC (MS-MAC). The idea is to divide the network into concentric circles, with the base station in the center. A node is placed in a specific ring based on its distance from the base station and has multiple potential forwarders that reside in the next ring. Wake/sleep schedules are set in a staggering pattern, from ring to ring, to reduce packet delivery delay. In a set of potential forwarders and a given time frame, the wakeup time slots are evenly distributed. In other words, when a node has data to send, there will always be a forwarder that is fairly close to waking up instead of having to wait an entire frame (or cycle) like in most tree-based topologies. MS-MAC was implemented in ns-2 and compared against IEEE 802.11 and two other MACs that employ staggered scheduling: DMAC and LEEM. In a multisource scenario, MS-MAC has higher throughput than DMAC and LEEM, has close to optimal (802.11) delay, and consumes the least amount of energy. *Key contribution: View of the topology as concentric circles to allow for multi-parent staggered scheduling.*

2.3.2 Independent Schedules

Schedules that do not require synchronous wake and sleep periods, being randomly selected instead are what we dub as *independent schedules*. The wake periods are smaller than those in synchronized schedules, allowing for extremely low duty cycles (i.e. longer sleep time), since they are only used to sample the medium and check for activity. When a node has a data packet to send, it initially sends out a preamble or a wake-up packet to notify a forwarder that a data packet will be on the way. Figure 2.4 displays examples of methods of data transmission for independently scheduled nodes. In (a), nodes must remain awake until the end of the preamble to find out if the following data is meant for them, whereas in (b), a node can find out if it is the intended receiver or not through the short wakeup packets. Certain protocols allow nodes to keep track of the neighbors wake periods in order to reduce the preamble size or know when to send a wake-up message. However, this comes at the cost of local synchronization in order to identify the wake-up time.

Single Forwarder

WIreless SEnsor MAC (WiseMAC) [15, 17] employs preamble sampling for both uplink (nodes to base station) and downlink (base station to nodes) communication. Nodes periodically sample



(a) Preamble (b) Wakeup

Figure 2.4: Examples of Independent Scheduling

the medium to check for activity. If a node receives a signal (a preamble), it will remain awake until it has received a data packet or the channel is idle again. Instead of sending out a long preamble where multiple nodes will end up waking up and wasting energy, nodes employing WiseMAC learns the sampling schedules of their neighbors in order to send a shortened preamble right before the intended receiver wakes up. Although nodes exchange their schedules every time data is exchanged, the preamble must still be long enough to overcome clock drift between the sender and receiver. In a simulation, WiseMAC is shown to consume less energy than CSMA/CA and T-MAC, although consumption increases linearly as the traffic increases. For downlink traffic, WiseMAC was compared against IEEE 802.15.4 ZigBee protocol analytically, where it consumes less energy ZigBee, but shares the same delay. *Key contribution: Uses neighbors sampling schedule to know when to send a preamble.*

Berkeley Media Access Control (B-MAC) was proposed in [36], having in mind simplicity in implementation, reconfigurability, scalability, tolerance of changes in network condition, and other factors which previous MAC protocols have not thoroughly addressed. Core functionalities, which can be tuned to an application's needs through the use of interfaces, include clear channel assessment, packet backoffs, link layer acknowledgments, and low power listening. Solving the hidden terminal problem (for example, through RTS/CTS) and fragmenting large messages are not a part of B-MAC, but can be implemented separately on top of it. When a node has a packet to send, it will first send out a preamble that is as long as the interval in which a neighbor checks the channel for activity. Instead of going back to sleep, the node will remain awake to receive the data that comes after the preamble. To test the effectiveness of B-MAC and its variants, in terms of throughput, energy consumption, and latency, they were implemented in TinyOS and installed in Mica2 wireless sensor nodes. Results are compared to S-MAC (with the motes) and T-MAC (through Matlab simulations). In a test without duty-cycling, B-MAC (without any ACKs or RTS/CTS) was shown to achieve higher throughput than S-MAC. With duty-cycling, B-MAC consumes less energy than S-MAC as the throughput increases, and has less delay. *Key contribution: Simple, configurable MAC interface.*

To reduce lengthy preambles, [8] proposed X-MAC, which employs short strobed preambles. With B-MAC, nodes that are awakened by the long preamble must remain awake until the end of it to find out if they are the intended recepient, consuming more energy than necessary. In X-MAC, when a node receives a short preambles, it will know whether or not it is the intended receiver, going back to sleep if it is not. This is determined by the address of the target that is included in the stream of short preambles. When the target awakens and hears a preamble, it will reply with an early ACK message, allowing the sender to transmit the data packet right after receiving the ACK. Another reason for the usage of short preambles is that it can be used with the newer generation of sensor motes, which employs packetized radios instead of bit-streaming radios (which allows the transmission of a long preamble). The authors have also proposed an algorithm for adapting duty cycles based on optimizing an objective function, which reduce energy consumption and/or latency. To evaluate the performance of X-MAC, it was implemented on top of the Mantis Operating System (MOS) and deployed on an indoor testbed of TelosB motes. Results were compared with a low power listening (LPL) MAC that simulated a long preamble by rapidly sending a stream of packets followed by the data packet. In a single-source scenario, the duty cycle of X-MAC is 1-2% less than that of LPL. In a multi-source scenario, with varying sending rates (one packet per second and one packet every ten seconds), X-MAC is shown consuming less energy for the most part than LPL in all cases of varying sleep time. It is also shown that latency, reception rate, and fairness are shown to be better in X-MAC. *Key contribution: Breaks a long preamble into short strobed packets.*

O-MAC [10] is a receiver-centric protocol, stating that most of the power consumption occurs at the receiving node. Before discussing their protocol, the authors analyze maximal energy efficiency (conclusions are shown in parenthesis) for several MAC models: synchronous blinking (depends on the number of interfering nodes), long preamble (receiver's duty cycle must be almost equal to the sender's duty cycle), asynchronous wakeup (proportional to the total duty cycle), random time spreading (has the worst power efficiency due to time and space wastage), staggered-on (efficiency increases by the number of interfering nodes), and pseudo-random staggered (efficiency decreases with respect to the number of interfering nodes). The later two are receiver-centric methods introduced by the authors, with the pseudo-random staggered method being implemented in O-MAC. Using asynchronous neighbor discovery, each node keeps track of their neighbors and their next active slots. When a node has data to send to a particular neighbor, it knows when that neighbor will wake up and can transmit it unicastly. The receiving node will then send back an ACK using unicast. If broadcast is necessary, a node can keep another schedule where it knows when all its neighboring nodes will wake up. In an earlier comparison, staggered-on and pseudo-random theoretically has higher energy-efficiency than the other MAC methods and remains high as the number of interfering nodes increases. For O-MAC, the authors wrote a simulation of its performance and the results show that it reaches near-optimal energy efficiency at a certain message sending rate, but decreases as the rate increases. *Key contribution: A sender knows its neighbors' wake-up times and sends packets unicastly.*

In [47], *Vuran et.al.* performed a case study simulation to determine how distorted data representing an event can get, using a function of the number of representative nodes that sent data to the base station. Results have shown that while having all nodes that observed the same event send data achieves minimum distortion, this value can still be intact even if the number of nodes reporting decreases. Also, there are two factors that contribute to the amount of distortion, a node's distance from the event (the closer the more accurate) and a node's distance from its neighboring observers (the further apart, the less distortion). Knowing this, the authors proposed an iterative node selection algorithm to find the minimal number of effective representative nodes, given a distortion constraint, which in turn decreases energy consumption and latency due to the reduction of collisions and transmissions. Spatial Correlation-based Collaborative MAC (CC-MAC) is used to control the sensor node transmissions, allowing only representative nodes to send their event data. CC-MAC is comprised of two parts, Event MAC (E-MAC) data-generating nodes to filter out correlated data and Network MAC (N-MAC) for routing, giving send priority to received packets over their own. As a base, CSMA/CA is used for accessing the medium, where correlation-information is added to the RTS/CTS/DATA/ACK packets. Correlation nodes go to sleep while the representative node sends out data. CC-MAC was implemented in ns-2 and compared with S-MAC, T-MAC, TRAMA, IEEE 802.11, and CSMA. In terms of packet drop rate and energy consumption, CC-MAC performs slightly better than others. Its goodput is higher than others, but is outperformed by TRAMA as the reporting period increases. Delay is less than TRAMA's, but slightly higher than the others. *Key contribution: Resolves spatially-correlated contention by reducing the number of reports that are sent from a particular event.*

The main goals of Proper MAC (P-MAC) [26] is to minimize idle time, which maximizes sleep time, and ensures packet reliability. During an initial phase where all nodes are awake, a node will send out a beacon packet and gather one-hop neighbor information (i.e. node ID and frame offset - difference between node's own wake schedule and a neighbor's). Schedules are calculated so that nodes do not have overlapping wakeup times as other nodes within its neighborhood. When a node has data to send, it uses the frame offset to determine when the intended receiver will be awake for data reception. Once the receiver wakes up, the sender transmits the data, and the receiver will send back an ACK after it acquires the data. For collision resolution, carrier sense is employed and there is a back-off period in the case a packet has to be retransmited. To evaluate the performance of P-MAC, it has been implemented in Qualnet [3] and compared with S-MAC. In a scenario with eight nodes, transmitting 200 packets to each other randomly (rates ranging from 0.2 packets/second to 2 packets/second), P-MAC (with 3% duty cycle) has less idle time and is shown to save more energy than S-MAC (with 10% duty cycle). Latency is constantly low for P-MAC whereas S-MAC's latency increases as the transmission rate increases. In another scenario where there are multiple sources sending at the same time, P-MAC is shown to have better throughput and less latency than S-MAC as the number of sources increases. *Key contribution: A node keeps track of its neighbors' wake-up schedules, none of which overlaps each other, so it can directly send a data packet to the desired forwarder without the worry of others overhearing it.*

In reservation-MAC (R-MAC) [54], Nodes compete for the reservation of time slots (using the CSMA/CA mechanism for contention) for future transmissions, notifying neighboring nodes of who will be transmitting or receiving during those slots. During the reservation period, which is initiated by the first sender to win the channel, those that want to transmit data exchange RTS/CTS packets with the intended receiver, notifying the intent of when the data transmission/reception will take place. Nodes that are not participating during any reserved slot will send out a go-to-sleep (GTS) packet at the end of the reservation period before entering going to sleep. The listening and sleep periods are adaptable by observing the traffic load, allowing for longer listens in the case of high traffic. In the case of overlapping reservations, a node that overhears two or more reservations (from different node-initiated periods) will have to choose one of them to follow. R-MAC was implemented in OPNET (in a scenario of 32 sensor nodes and 1 sink) and compared with S-MAC and T-MAC in terms of average percentage of time a node goes to sleep, average energy consumption, and the occurrence rate of collisions. With low packet rate, R-MAC spends less time sleeping and consumes more energy than S-MAC and T-MAC. However, as the packet rate increases, R-MAC spends more time sleeping and consumes less energy than the other two protocols. Due to the reservations, R-MAC avoids collisions between data packets and hence has less collisions than S-MAC and T-MAC. *Key contribution: Reservation period allows nodes to notify intended receivers ahead of time that data transmissions need to take place, avoiding data packet collisions.*

Multiple Forwarders

Energy-efficient Contention Resilient MAC (ECR-MAC) [57] is able to overcome the spatially-correlted problem (where when an event occurs, nodes all sensing that event contend for the medium at around the same time), while saving energy, decreasing latency, and increasing overall network throughput. Previous MACs imply that a sender only has one receiver, where a node would have to wait for the intended receiver to wake up. Not only that, nodes that share the same forwarder (through synchronized scheduling) will result in more collisions and delay in the case they all have data to send. With ECR-MAC, all nodes have a random wake/sleep schedule and employs a Dynamic Forwarder Selection (DFS), where a sender has multiple potential forwarders instead of one. When a node has data to send, it will transmit wake up packets in a strobed pattern until one of the potential forwarders replies. Even if multiple nodes try to send around the same time, the path in which these packets follow are dispersed, away from the contention area. To avoid overhearing, if a node wakes up and senses that the channel is busy, it will go to sleep for some time. ECR-MAC was implemented in ns-2 and compared against three CSMA-type protocols which employ dual radios: STEM, PTW, and LEEM. In a single-source scenario, ECR-MAC consumes the least amount of energy and is fairly constant as the packet generation rate increases. Although ECR-MAC's end-to-end delay is less than STEM's, it is higher than PTW and LEEM since those two employ a pipeline (i.e. staggered) technique. However, in the case of eight sources (multi-source), ECR-MAC's end-to-end delay (for first 10% of reports) and energy consumption is less than STEM, PTW, and LEEM, and has the highest packet delivery ratio, even as the packet rate increases. *Key contribution: Utilizes multiple forwarders without any synchronized scheduling.*

Similar to X-MAC and ECR-MAC, Convergent MAC (CMAC) [30] reduces the length of a long preamble (like the one in B-MAC) by sending a burst of RTS packets separated by gaps that are long enough for a CTS to be sent back. The sending node will then transmit the data packet to the forwarder that has sent back a CTS. CMAC additionally employs a *double channel check* and *anycast based forwarding*, which is similar to ECR-MAC's dynamic forwarder selection. When a node wants to assess the channel, it will do so twice (sampling the channel up to five times during the assessment) in order to make sure it does not miss an RTS packet. Nodes that hear an RTS packet can send back a CTS if it is included in the sender's forwarding set. If there is more than one forwarder that wants to send back a CTS, the forwarder will use a routing metric to determine when the CTS should be sent, rather than having them all transmit at the same time. Forwarders that overhear a CTS can cancel their own CTS, to avoid causing unnecessary collisions. CMAC uses geographical distance as an example for a routing metric, where nodes that are closer to the main destination are allowed to send a CTS quicker as opposed to nodes that are further away. *Convergent packet forwarding* occurs when a node has a forwarder with a good routing metric, deciding to unicast to it. To reduce delay, the authors suggest that a synchronized staggered schedule (like in DMAC) can be implemented with CMAC. For experimental evaluation, CMAC was implemented and tested on XSM motes, over a Kansei testbed of 105 nodes, and compared with B-MAC. With a low duty cycle (1%), CMAC is shown to have better throughput, lower latency, and lower energy consumption per packet than B-MAC. For larger scenarios, CMAC (both regular and staggered) was implemented in ns2 and compared against CSMA/CA, Anycast (CMAC without convergence), GeRaF, and S-MAC. *Key contribution: Nodes can decide to anycast an RTS to multiple forwarders or unicast to an optimal forwarder.*

The technique Dynamic Switch-based Forwarding (DSF) [20] is a routing protocol that addresses the issue of both the unreliability of radio communication and sleep-latency in order to obtain optimal expected delivery ratio (EDR), expected end-to-end delay (EED), and/or expected energy consumption (EEC). A node keeps track of the times in which its neighbors wake up, allowing it to select a forwarder that is closest to waking up when it has data to send. If a transmission has failed, the node will select the next forwarder that will wake up soon. If the node has tried all forwarders, it will drop the packet. Each node calculates its own EDR, EED, and EEC values in order to optimize its forwarding sequence by selecting a subsequence that meets the desired criteria. If link quality exceeds a certain threshold, these calculated values are updated by exchanging information with neighboring nodes. DSF, using CSMA as its base MAC, has been implemented on TinyOS, using 20 MicaZ motes. ETX, expected transmission count metric for finding high-throughput paths, has also been implemented as a source of comparison. In an experiment where one source node sends 100 packets to the sink node, DSF (with EED forwarding sequence) is shown to have the least set of delays. For a larger-scaled experiment, the authors have used simulation to test a 250 node scenario and compared results with ETX, PRRxD (uses the product of packet reception rate and distance toward destination as a metric), and DESS. *Key contribution: Utilizes multiple forwarders to overcome unreliable communication links, selecting the next-hop forwarder based on an expected calculated metric.*

2.4 Summary

Figure 2.2 displays a list of summarized MAC protocols that were discussed in the previous section. All of them have acknowledged the need for energy-efficient access of the medium, trying to reduce idle listening and overhearing; most of them recognize the convergecast pattern; however, only a handful of them have addressed the issue of spatially-correlated contention [47, 57, 58]. With the placement of duty cycles, network performance such as latency and throughput tend to suffer. Most WSN MAC protocols tried to emphasize on the improvement of these metrics while continuing to suggest more energy-efficient schemes.

Table 2.2: WSN Medium Access Control Protocols

		Protocol	Year	Keywords	Evaluation
Synchronized	**Single**	S-MAC [51]	*2002*	Synchronized group schedules	Sim, Testbed
		T-MAC [12]	*2003*	Adaptive wake periods	Sim
		TRAMA [37]	*2003*	Traffic-adaptive schedules	Sim
		DMAC [31]	*2004*	Staggered Schedule	Sim
		Z-MAC [38]	*2005*	CSMA for low, TDMA for high contention	Sim, Testbed
		SCP-MAC [53]	*2006*	Synchronized channel sensing	Testbed
		Dozer [9]	*2007*	Double schedules	Testbed
		A^2-MAC [56]	*2007*	Group schedules based on traffic rate	Anayltic
	Mult	Multi-parent [25]	*2006*	Synchronized group schedules, parent assignment	Analytic, Sim
		MS-MAC [58]	*2007*	Concentric circle topology, staggered	Sim
Independent	**Single**	WiseMAC [15]	*2003*	Short preamble, knows neighbors' schedules	Analytic, Sim
		B-MAC [36]	*2004*	Low-power listening, configurable interface	Testbed
		X-MAC [8]	*2006*	Strobed-shorten preambles	Testbed
		O-MAC [10]	*2006*	Unicastly transmits, knows neighbors' schedules	Sim
		CC-MAC [47]	*2006*	Reduce number of event reports	Sim
		P-MAC [26]	*2006*	Non-overlapping schedules, knows neighbors' schedules	Sim
		R-MAC [54]	*2007*	Reserve data transmission ahead of time	Sim
	Mult	ECR-MAC [57]	*2007*	Multiple forwarder selection	Analytic, Sim
		CMAC [30]	*2007*	Anycast or unicast forwarder selection	Sim, Testbed
		DSF [20]	*2007*	Metric-based forwarder selection	Sim, Testbed

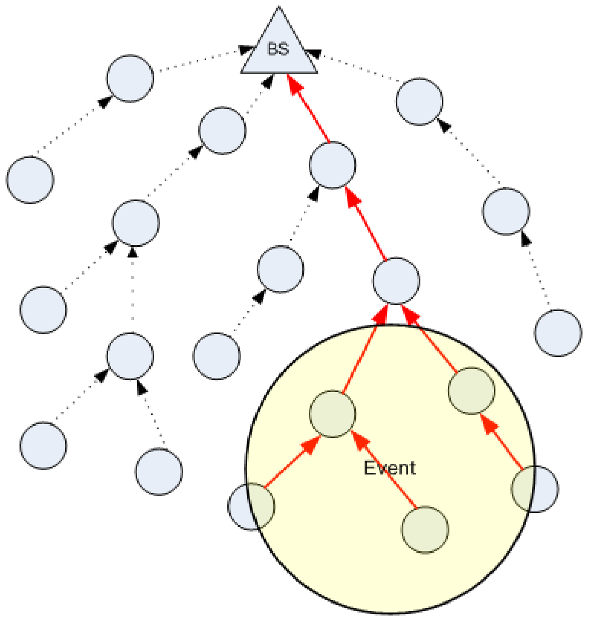
CHAPTER THREE

LOCAL COORDINATION MEDIUM ACCESS CONTROL

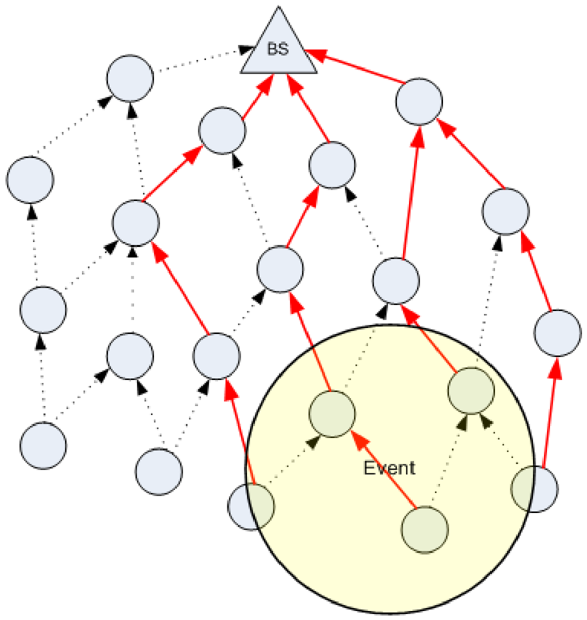
3.1 Motivation, Design Choices, and Assumptions

Deviating from the traditional view of designing layer-specific protocols (i.e. reliability in transport layer, routing in network layer, medium access in data-link layer), we propose a cross-layered protocol, providing medium access control additional flexibility through the use of routing information. Wireless sensor networks are unique in that while individual nodes are autonomous devices, the collaborative work of the nodes are what makes up a complete system. Protocols can either be designed from the individual node perspective or at a higher level, considering the overall network by taking into account traffic patterns and application needs. As mentioned before, the major pattern for all event-triggered WSNs is convergecast: where all nodes report their data to the nearest base station. Most MAC protocols have acknowledged this pattern and views the network topology as a tree, with the sink as the root and routes that represent the shortest path. Duty cycle schedules are set amongst the nodes in a manner that can reduce delay (staggered) or reduce collisions (allocated time slots). However, a tree structure can be problematic. Figure 3.1(a) shows an example of the paths (the dark, filled arrows) that are taken for the set of nodes that sense the same event. Nodes that share the same parent will have to take turns, either contending for the medium or waiting for their time slot in order to upload their data to the parent. Not only that, neighboring nodes that do not share the same parent may also have to contend for the medium. Having such high contention and being restrained to only one forwarder may result in packet drops, less throughput, longer delay due to the contending periods, and quick energy depletion amongst key forwarding nodes.

Selecting one among several forwarders is generally a task of the network layer, where the routing protocol selects the next-hop forwarder from a routing table toward the direction of the



(a) Single



(b) Multiple

Figure 3.1: Routing methods

intended receiver and/or based on a performance metric. With WSNs, there is only one major destination when an event occurs: the closest base station. Figure 3.1(b) shows the same example scenario as in (a), but instead has the option of choosing one among several forwarders (arrows with dotted-line represent other possible routes). The paths selected may not be shortest in distance, but this does not strongly correlate with the time it takes for a packet to reach the destination since duty cycles are employed. The use of multiple forwarders can overlay a MAC protocol (for example, [20] uses DSF with CSMA) or used as a means of setting up synchronized wake-up schedules [25]. Integrating forwarder selection with the functions of the MAC layer can ease heavily congested areas caused by spatially-correlated contention through disseminating traffic, reduce end-to-end delay, and better disperse energy consumption. Neighboring nodes that want to transmit can select an uncommon forwarder and will no longer have to wait an entire duty cycle frame to elapse for each data transmission that is to take place. Instead of having a few forwarders quickly drain their energy and requiring a new tree layout in the case one dies, traffic and hence energy usage is instead distributed among several forwarders.

The concept of designing network protocols based on abstract layers was first introduced in the Open Systems Interconnection Basic (OSI) Reference model and later adapted into the TCP/IP model for the Internet. Each layer concentrates on a set of services and can assume they are fulfilled when receiving a message from a lower layer. The higher levels apply to application needs while the lower levels deal with the actual next-hop transmission without having to know application details. With wireless sensor networks, layering can be upheld, but resource constraints can deter any adequate means of clearly designing energy-efficient, individual layered protocols at the lower levels. Others have already seen the advantage in designing cross-layered protocols [21, 42]. We believe that both the MAC and network layer can benefit from each other. Since duty cycles are employed in WSNs, a node would have to wait an entire frame to retransmit each time it fails to send a packet to the selected forwarder. If the routing layer is notified of this failure (caused by a collision, busy channel, or a bad link), it could select a new forwarder with the next best routing metric immediately. This dynamic hop-by-hop process of routing a packet along different paths is not detrimental to the end-to-end delivery of a packet. Paths do not have to be constructed ahead of time as done with source initiated and distance vector routing in wireless networks since a WSN is generally connectionless (i.e. a node and a base station does not have to establish a connection before any data is transmitted between the two) and the shortest path is not necessarily the best path to take, especially when there is high contention within an event area. Dynamically creating routes is instead advantageous, since paths with less delay or stable links (depends on the metric) are chosen and congested areas are relieved quicker.

The choice of employing independent scheduling over synchronized scheduling is very simple: creating synchronized schedules in a meaningful manner can be a complex task and requires a set of heavy assumptions. Lu *et al.* [32] has proved that creating delay efficient sleep schedules is NP-complete and Keshavarzian *et al.* [25] has shown assigning multiple parents that share different schedules is also NP-complete. Both have provided heuristics which require a centralized algorithm (e.g. a base station knows the location of each node in the network and computes a schedule for each one). The latter has proposed localized algorithms, but has shown that a centralized algorithm provides better results. Zhou *et al.* [58] proposed a staggered, multi-parent scheduling algorithm that is performed in a distributed manner. However, it assumes all nodes are globally synchronized and location information is available. Some applications may already require individual node location information or accurate packet timestamps due to the type of data that is collected, which can conveniently be used with the aforementioned MACs. Applications that do not utilize such information will suffer energy consumption needlessly in order to obtain the requirements of the MAC. One could argue that a protocol stack can be designed to fit the needs for each type of application. However, in a world where the idea of ubiquitous computing is indeed becoming a reality, the number of WSNs will increase and each application should not have to worry about the specific details of the lower levels in networking, which is the case for other computer networks today.

With duty cycles in place, there is a trade off between energy efficiency and performance metrics such as delay, throughput, packet in-delivery arrival, and fairness. Packet latency is important in surveillance applications that require a quick notification of a certain event in order to respond immediately (e.g. natural disaster warnings, medical response units). With throughput, the higher it is, the more accurate an event can be represented. Duty cycles have a detrimental effect on these metrics in that at each hop, a node must wait for its intended receiver to wake up, thus accumulating what is termed *sleep latency*. The performance of latency and throughput are fairly critical and we intend on improving them without compromising energy-efficiency in comparison to other MAC predecessors. As for in-delivery arrival and fairness, these do not play as important roles in WSNs since the aim is to report an event more-so than making sure packets arrive in the order they were generated and having each node get a chance at delivering its own packet.

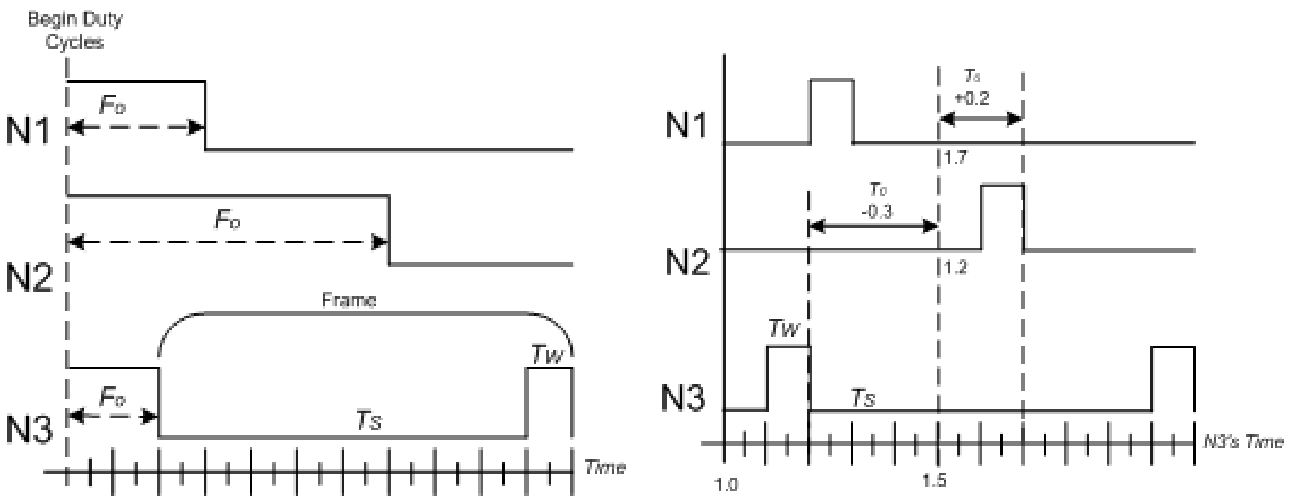
Our goal for protocol design is to use very little assumptions while still making a significant impact on energy savings. Besides simplicity of implementation, independent scheduling allows duty cycles to be very low since wake-up periods tend to be used for sampling the channel rather than being long enough to receive an actual data packet. Combining this with multiple forwarders becomes a powerful idea in that sleep latency is reduced since each forwarder should be waking up at different times. However, without a wake-up prediction scheme for independent schedules, nodes would have to use a preamble [36] or a stream of strobed packets [8, 57] in order to notify the intended forwarder that a data transmission needs to take place. These wake-up messages consume energy needlessly and dominates the wireless channel, preventing neighboring nodes from transmitting. With a simple design of hurling packets forward as fast as possible, ECR-MAC [57] has shown significant results in terms of energy consumption, end-to-end delay, throughput, and scalability in comparison to MACs that employ dual radios under scenarios with spatially-correlated contention. One natural progression towards further improving its performance would be for a node to reduce the busy wake-up mechanism that is used. To do so, we suggest that each node maintains the wake/sleep schedules of its forwarders so it knows when to send data. This can easily be done when a node is gathering and exchanging one-hop neighborhood information. There is the issue of being able to predict a forwarder's wake-up time accurately due to clock drifts, so we do require a loose local synchronization. Nodes do not need to change their internal clocks, but can instead take into account the differing time offsets of its neighbors when notifying a forwarder at its wake-up time.

Overall, our proposed Local Coordination MAC *LoC-MAC* will use ECR-MAC as its basis while utilizing information on its neighborhood to achieve additional energy-efficiency, without compromising delay and throughput. Unlike synchronized scheduling, which usually involves global coordination amongst the nodes, we localize coordination to a node and its neighbors. We assume that the network is densely deployed and static. There may be more than one base station throughout the network, each remaining awake at all times and therefore available for packet reception at any time, but for the sake of simplicity, LoC-MAC assumes there is one base station. It is still possible to accommodate multiple base stations in that a node keeps track of a table of forwarders that creates routes towards the closest base station. Acknowledging that convergecast is the main source of traffic in an event-triggered sensor application, LoC-MAC mainly assists communication in the source-to-sink direction. However, LoC-MAC can also work with other traffic patterns, such as sink-to-nodes (ex. for querying). To make such communication possible without maintaining a huge routing table that stores next-hop forwarders to all other nodes, geographic location information must be available such that a node can select a forwarder that is geographically closer to the destination.

3.2 Initial Setup

Before any event reporting can take place, there is a series of steps that a node must follow to ready itself for the data forwarding involved in LoC-MAC. While ECR-MAC only requires each node to find out the distance from the base station and how many forwarders it has that are one-hop closer, we require a more detail on a node's neighborhood to identify a forwarder's next wake-up period in real time. To determine these times, there are two offsets we need to account for: a frame offset for calculating the difference between the node's own wake schedule and its neighbor's, and a time offset to account for the clock differences. With these time identification schemes, a node no longer has to remain awake, actively searching for a node that can act as a forwarder as done in ECR-MAC. The time spent searching is now spent sleeping in LoC-MAC, waiting for the selected forwarder to wake-up.

Prior to filling its forwarder table, a node should determine when it should first go to sleep, the time value we dub as the node's frame offset F_o. It selects a random time slot within a given frame f, which consists of an entire sleep period T_w and a wake period T_s. The amount of time a node sleeps or is awake depends on the length of the frame and the duty cycle percentage ($dc = \frac{T_w}{T_s+T_w}$), where the number of wake slots that are available within a frame can be calculated as $\frac{1}{dc}$. During a wake period T_w, a node scans the channel to see if a neighbor wants to use it as a forwarder, thus the period must at least be long enough for the neighbor to receive a notification (WAKEUP) message. Since a node does not know when its neighbors wake up in ECR-MAC, it requires the wake/active period to be long enough for the reception of two WAKEUP messages, some carrier sensing, and a REPLY message to ensure that a forwarder does not miss out on any WAKEUP messages. By reducing the wake period, we were able to reduce ECR-MAC's duty cycle of 1% to 0.8%, which results in more energy savings since less time is spent idling and the chances of overhearing have decreased.



(a) Frame Offset (b) Time Offset

Figure 3.2: Different Offsets

At some scheduled time, after a node is done gathering information on its neighbors and performing any other application tasks, all nodes will wait for their own F_o time before going to sleep for the first time. This "scheduled time" is relative to each node's clock as opposed to real time (e.g. all nodes will start its duty cycle after 10 minutes have passed, which may be at different points in real time), so in order to account for the clock differences, the time offset T_o must be determined between each node and its forwarder. Figure 3.2(a) and (b) shows an example of three nodes and their selected frame offsets and calculated time offsets. In (b), at time 1.5 for N3, N1's current time is 1.7 and N2's current time is 1.2. Knowing this, we can see that N3 is behind N1 by 0.2 and ahead of N2 by 0.3. In addition to storing these offsets for each neighbor, a node needs to differentiate amongst neighbors that are one-hop closer to the base station or the same hop distance. Since packets should always be moving towards the base station, a node does not have to keep track of its neighbors that are further away. We store the neighbor's hop distance H_n and the number of forwarders P_n it has to achieve a similar effect of ECR-MAC, but instead actively selects a neighbor that is closest to waking up if it is one hop closer or is in the same hop distance, but has 'more' one-hop closer forwarders. The latter case has been included to reduce end-to-end delay, since a same-hop neighbor with more one-hop forwarders may be able to route a packet faster than waiting for its own forwarder to wake up. Table 3.1 shows the entry needed for each potential forwarder N_{id}. To achieve comparable performance results as ECR-MAC, the size of the table should depend on the density of the network. The more forwarders a node has, the less it would have to wait to forward the packet.

Table 3.1: Potential Forwarders Table

N_{id}	H_n	F_o	T_o	P_n

Getting into the details of filling the table, H_n and F_o can both be obtained by an initial broadcast from the base station (view Algorithm 1), with a 'hop-distance' counter H_n initially set to zero. The nodes receiving this first broadcast message (line 1-2) will know that it is one-hop away from the base station. They will create a new message by adding their own node ID *myNodeID*, frame offset *myFrameOffset*, and an incremented 'hop-distance' counter *myHopDistance*, and then broadcast it (line 3). This will continue until every node in the network has received and sent out a broadcast. To prevent some collisions amongst these broadcasts, a node will randomly select a time to perform carrier sense, where if the channel is idle, it will broadcast, otherwise select another time to try again. To prevent a *broadcast storm* from occurring, a node will broadcast the hop/frame message only once. Neighbors that are one-hop closer to the base station are kept in the potential forwarders table. Those that do not have many of these neighbors or has enough room in their table should also keep neighbors that are in the same hop-distance away from the base station and has more potential forwarders than itself (*InsertIntoTable* function in line 6 will determine this).

Algorithm 1 ReceiveBroadcastInformation

```
Input: $H_n$, $F_o$, $N_{id}$
1: if myHopDistance is empty then {First time receiving this message, broadcast}
2:     myHopDistance $= H_n + 1$
3:     SendBroadcast(myNodeID, myHopDistance, myFrameOffset)
4: end if
5: if $H_n \leq$ myHopDistance then {This is a neighbor I want to keep}
6:     InsertIntoTable($N_{id}$, $F_o$, $H_n$, $P_n$)
7: end if
```

The time offset T_o and number of parents P_n can be retrieved after the table has been filled with who the potential forwarders are. A node can roughly calculate T_o through the exchange of time-stamped packets. Maroti *et. al* has taken into account all the delays that occur from sending to reception of a packet and introduced a flooding time synchronization protocol with accuracy from tens of microseconds to 1.4μs [33]. The one-hop version can be used for a node to get the time offsets of its potential forwarders. The value of T_o may change over time since the clocks between two nodes tend to drift. The periodicity of updating T_o is dependent on the frequency differences between a node and its neighbor's clock, which is beyond the scope of this thesis. P_n is only needed for neighbors in the same-hop distance from the base station and thus can be included in these time-exchange packets. After gathering this information and the 'scheduled time' has approached, the node will set a timer for F_o time units, where when it expires, the node will go to sleep by turning off its radio transmitter for the first time.

3.3 Data Forwarding

Once a node senses an event, it will put together a report that is to be interpreted and analyzed by the base station. In a network spanning a large area, packets from afar will have to travel multiple hops. With duty cycles employed, sleep latency starts to accumulate with each hop which can cripple event-critical applications where immediate action is needed to a reported and confirmed event. While synchronized schedules can be set such that there is a staggered pattern where forwarders wake up one after the other to facilitate quicker delivery, the overhead in maintaining such a system can be expensive and is not necessarily robust to spatially-correlated contention. Another approach most proposed protocols tend to take is assuming the usage of a data gathering tree based on the shortest path. In a network with low duty cycles, making the most of the next hop in terms of physical distance does not always result in the faster delivery of a packet. ECR-MAC has recognized such limitations and utilizes a simple greedy approach, where any neighbor that meets the requirement of being closer or has more forwarders that are closer to the base station can reply to a wake-up call. This in turn overruns the channel with a series of messages coming from several sources, consuming energy needlessly on both sending and receiving ends (which end up with corrupted messages). To mitigate this concern, LoC-MAC uses a light coordination such that nodes still have the opportunity to select the next best forwarder (in terms of delay) while providing courtesy towards neighboring senders in terms of channel usage.

3.3.1 Basic Scheme

Before going into the details of handling contention, let us go over the basics of data forwarding. The data exchange that takes place between two nodes utilizes a four-way handshake, where the sender transmits a WAKEUP message asking if the intended receiver can be the forwarder, and if so send back a REPLY. The DATA is sent and is followed by an ACK that notifies the sender of the successful reception. Since LoC-MAC can calculate the next wake-up periods of each of its potential forwarder, it knows when is the appropriate time to send a WAKEUP message, thus relieving the need for sending out periodic WAKEUP messages as in ECR-MAC. To ease the understanding of a node's state in terms of its radio transmitter and status, we have four main states: WAKE (radio transmitter is on to sense the channel for wakeup messages), SLEEP (transmitter is turned off to save energy), WAIT (transmitter is off until node's selected forwarder is awake), and ACTIVE (transmitter is on until data exchange is completed). The details of the four-way handshake for LoC-MAC between two nodes are described below (view Figure 3.3 for a graphical depiction of the steps):

1. A node N_{id} first selects a neighbor $FT_i.N_{id}$ that is closest to waking up and also obtains the amount of time that is left before the neighbor is scheduled to wake-up (see Algorithm 2 for selection method, which is described in detail later).

2. The node N_{id} goes to sleep and sets itself to **WAIT** mode, only waking up when its selected forwarder $FT_i.N_{id}$ also wakes (to perform its routine channel sensing)

3. When N_{id} wakes up, it goes into **ACTIVE** mode and sends a *WAKEUP* message to $FT_i.N_{id}$, which in turn

4. returns a *REPLY* message to N_{id} (if it is free to be the forwarder) and remains awake in an **ACTIVE** mode (overriding the duty cycle schedule) that is long enough to receive the data.

5. The *DATA* packet is sent to $FT_i.N_{id}$. If it has been received succesfully, the forwarder

6. sends an *ACK* back to N_{id}. $FT_i.N_{id}$ stores the DATA in its queue and selects the next forwarder (go back to step 1, this time assuming the role of a sender).

7. N_{id} ends **ACTIVE** mode after receiving an *ACK*. Remove the DATA packet that was just sent from its queue. If there is still DATA in its queue, go back to step 1, else go back to its original sleep schedule state (either **SLEEP** or **WAKE**).

While the data exchange is going on, both nodes will continue to keep track of their wake/sleep schedule by running a timer in the background, although the ACTIVE state overrides the status of the radio transmitter. Neighboring nodes rely on the periodicity of these schedules in order to accurately identify the forwarders' wake-up times, hence the need for the continual switch between

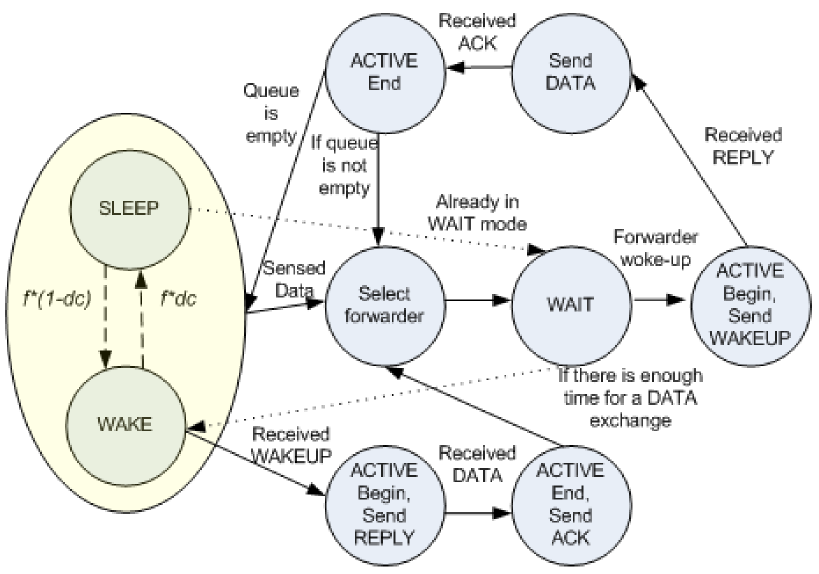


Figure 3.3: States a node goes through, both as a sender and a receiver

WAKE and SLEEP states in the background. To ensure the continuous flow of forwarding data quickly, once a node receives a data packet, it will assume the row of the sender by selecting the next-hop forwarder. Sticking with this theme of getting a packet out quickly, there is a point during the WAIT period where a node should wake up as normally scheduled (represented in Figure 3.3 as the dotted arrows). For example, a node A is in WAIT mode but node B wants to use A as a forwarder, relying on it to wake up. If there is enough time for node A to go through a data exchange before its own forwarder wakes up, it should wake up to show that it is available for reception. This requires the use of a queue (mentioned in steps 6 and 7), so that a node is allowed to act as a forwarder for more than one packet.

Several routing metrics may be used in the selection of a forwarder. For LoC-MAC, delay is chosen as the factor in order to emulate ECR-MAC's network performance and therefore select a forwarder that is closest to waking up and fits the criteria of being one-hop closer to the base station or a same-hop forwarder that has more forwarders than the sender. Neighbor selection in Algorithm 2 utilizes the fields within the potential forwarders table that was constructed during initial setup. Line 2 starts a loop that runs until all entries in the table have been observed. If the

Algorithm 2 ForwarderSelection

```
Input: $T_{nextWake}, F_{me}$
Output: $fID, T_{wait}$
 1: firstForwarder= $true$
 2: for $i = 0$ to $|FT|$ do
 3:     if $FT_i.ID$ =baseStationID then {no need to wait or select any other forwarder}
 4:         $T_{wait} = 0$
 5:         $fID = i$
 6:         Return $fID, T_{wait}$
 7:     else if $FT_i.H_n \leq H_{me}$ then {neighbor that is one-hop closer or same-hop to the BS}
 8:         off $= FT_i.F_o - F_{me}$
 9:         off $+= FT_i.T_o$
10:         if off $< 0$ then
11:             off $+= T_{frame}$
12:         end if
13:         if off$+T_{nextWake} > T_{frame}$ then
14:             off $-= T_{frame}$
15:         end if
16:         if firstForwarder or ((off + $T_{nextWake} < T_{wait}$) and ($FT_i.H_n < H_{me}$) or ($FT_i.P_n \geq P_{me}+$
            MAXDIFF)) then
17:             $fID = FT_i.ID$
18:             $T_{wait} =$ off $+T_{nextWake}$ {saving the forwarder that is closest to waking up so far}
19:             firstForwarder = $false$
20:         end if
21:     end if
22: end for
23: if firstForwarder == false then {a forwarder was not found}
24:     $fID = -1$
25: end if
26: Return $fID, T_{wait}$
```

node happens to have the base station as its next-hop neighbor, it will exit the function immediately and the node will know that it does not have to wait to send its data packet since the base station is awake at all times (lines 3-6). Line 7 determines whether the current neighbor has the same hop distance or less. If so, the predicted wake-up time is determined by lines 8-15. The calculations ensure that the wake-up time is within a valid time-span (e.g. wake-up slot is between the current time and an entire frame later: *CurrentTime* $< T_{wakeSlot} <$ *CurrentTime*+f). Lines 16-20 check if this is the first forwarder we have come across (*firstForwarder*) or if we have found a forwarder that wakes up sooner than the current one that is stored. If so, store the node ID and the predicted wake up time. Once a forwarder is found, the node goes to sleep and sets a timer for the scheduled wake up time of the intended forwarder.

3.3.2 Handling Contention and Collisions

Contention occurs when several nodes within the same area want to use the wireless medium at the same time. Nodes must 'fight' amongst each other, where the winning node is allowed to use the channel while the other nodes should hold back from sending anything to avoid collisions. In surveillance WSNs, spatially-correlated contention is very much an issue, but has only been addressed by a few. Jamieson *et al.* [23] explicitly brought up the concern of this contention and proposed an event-driven MAC called Sift which is a CSMA-based protocol with a fixed contention window to reduce delay, but does not employ any duty cycles which will deplete the network's life quickly. With CC-MAC [47], the authors have proposed reducing the number of nodes reporting their sensed event, by selecting a few representative nodes. While this reduces contention and collision, some applications may still require more reports for reliability. Akan *et al.* [6] discusses the importance of moving from the notion of end-to-end reliability to event-to-sink reliability as a new paradigm for transport in WSNs. ECR-MAC has focused on trying to subdue the effects of spatially-correlated contention on throughput and end-to-end delay, through the use of multiple forwarders. LoC-MAC follows similar methods as ECR-MAC to achieve the same performance, while lowering overall energy consumption.

Multiple neighboring nodes that want to send data at the same time may share several forwarders, and hence select the same forwarder that is closest to waking up. Senders within each others' transmission range can perform carrier sense to avoid colliding with each other, but those that are hidden from each other will end up sending corrupted messages to the intended receiver. Several methods are out there to prevent the hidden terminal problem, but they do not work with protocols that employ independent schedules. 802.11 utilizes a virtual carrier sense, where nodes that overhear an RTS or CTS packet will know that a transmission is going to take place, and should avoid sending anything until the data exchange is over. This would not work effectively with nodes that employ random duty cycles, since nodes can easily miss out on hearing the first two control packets and will not know that a transmission is occurring two hops away. Another solution is to schedule time slots in a manner that avoids two-hop neighbors from transmitting at the same time, but this results in a more complex mechanism to maintain. With multiple forwarders however, nodes are not stuck with waiting for one forwarder until it is available, but instead can move on to another available forwarder. ECR-MAC has pointed out how unlikely it is for two non-neighboring senders to share all the same forwarders and hence, these nodes will eventually select a forwarder that does not interfere with each other.

Since nodes know the wake-up times of their forwarders in LoC-MAC, it is possible for two neighboring nodes to select the same forwarder and send a WAKEUP message at the same time. To prevent such collisions, LoC-MAC allows for a small contention period where nodes perform carrier sense at a random time before sending a WAKEUP message (view Figure 3.4 for an example). The one that senses the channel to be idle first will get to send its message, while others can overhear the WAKEUP and immediately move on to selecting a new forwarder. The WAKE period is then extended to contain this contention period, which is still less than ECR-MAC's active period. A concern that exists in ECR-MAC, but not in LoC-MAC, due to allowing any potential forwarder to reply to a WAKEUP call, is the occurrence of REPLY collisions. Two or more non-neighboring forwarders may send back a REPLY at around the same time, which prompts the sender to broadcast another WAKEUP message (after receiving a certain number of corrupted messages) to notify them that they should reschedule their wake/sleep periods so that they do not overlap each other. LoC-MAC does not have to worry about this since a node explicitly selects a forwarder, inserting the forwarder's address into the WAKEUP message. If a node happens to overhear a WAKEUP that was not meant for it, it will simply go back to its own schedule. However, the idea of readjusting can still be used to better spread the wake-up times of a node's neighbors and notify senders that it may not be desirable to select it as a forwarder for the current event reporting period. If a forwarder receives several corrupted messages (perhaps WAKEUP messages sent by two non-neighboring nodes), it can readjust its own schedule to make itself available at later times. This not only reduces overhearing, but also sleep latency. While readjustment comes at no cost for ECR-MAC, LoC-MAC has to take care of notifying nodes that a change in schedule has occurred for a forwarder (further discussed in the Maintenance section).

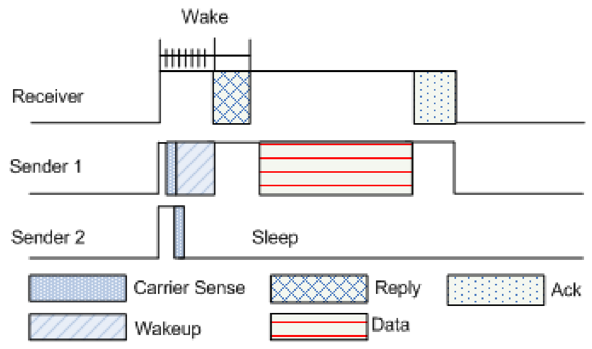


Figure 3.4: Two senders contending for the same forwarder

Understanding that collisions can still occur and knowing that the wireless medium is unreliable, we cannot assume that when a packet is sent out, it has been received successfully. Even if the DATA has been received correctly, the sender may receive a corrupted ACK packet and will end up assuming that the DATA transmission has failed. To be more persistent about the assurance of packet delivery, timers are employed at different points during the data exchange such that when one expires, it is assumed that a packet has been lost or corrupted and the node should try again. The periods at which a timer is set and the action taken is listed in Table 3.2 (in conjunction with other timers that have been mentioned in the past). Note that T_w, T_r, T_d, and T_a represent the time it takes to transmit/receive a WAKEUP, REPLY, DATA, and ACK packet respectively. α represents some leeway time that accounts for carrier sense and other time delay factors. Some timers are set differently according to a state the node is in.

If a node has been successful in receiving the desired packet, certain timers will be cancelled in order to avoid the execution of the expiration handle. For example, if a node has received a

Table 3.2: Timers

Timer	**Scheduled**	**Length**	**Expiration Action**
DutyCycle	SLEEP State: beginning of sleep period	$f \times (1 - dc)$	Set state to WAKE
	WAKE State: beginning of wake period	$f \times (dc)$	Set state to SLEEP
WaitMode	When the node has selected a forwarder	*Forwarder's wakeup time - current time*	Select a random slot to perform carrier sense (before WAKEUP packet is sent)
ActiveMode	Transmitter: beginning of data transmission	$T_w + T_r + T_d + T_a + \alpha$	Go back to regular schedule
	Receiver: beginning of data reception	$T_r + T_d + T_a + \alpha$	DATA was not received, go back to regular schedule
RetransmitWakeup	When the transmitter sends a WAKEUP packet	$T_w + T_r + \alpha$	REPLY was not received, select a new forwarder
RetransmitData	When the transmitter sends a DATA packet	$T_d + T_a + \alpha$	ACK was not received, select a new forwarder

wake-up REPLY, it will cancel the RetransmitWakeup timer. A modified diagram that is built upon Figure 3.3 by including contention periods and collision resolution with the use of timers is shown in Figure 3.5. If a sender was unable to transmit its DATA packet, it will try again by selecting another forwarder, whereas the intended receiver will go back to its own schedule if it has not received a DATA packet after some time. Retransmissions are necessary for reliable delivery of a DATA packet, but can cause an unwanted side effect of having duplicate DATA packets flowing throughout the network. While the base station can discount duplicates, the rest of the network will not be able to differentiate between an original and a duplicate. This occurs when DATA has been delivered successfully, but the ACK packet did not reach the sender. The sender will try to forward the data again, while the previous forwarder will continue to route the packet. By reducing the size of the ACK packet, we can reduce the creation of duplicate packets.

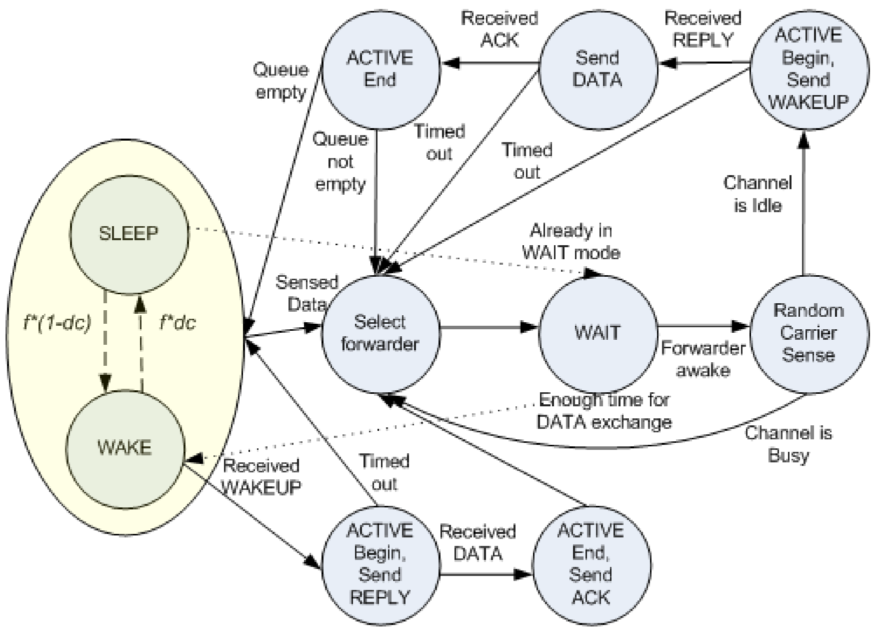


Figure 3.5: Complete State Diagram

In the case a node has been unable to send its packet for awhile (meets a certain threshold of retransmissions) due to high congestion, it will employ a back-off system similar to CSMA. It will select a random amount of time $T_{backoff}$ such that the next earliest forwarder that is selected must wake up at least after $T_{backoff}$ time units. This will allow the node to save energy by allowing it to wait and sleep for a longer time, and reduce the chances that another neighboring node will select the same forwarder again.

3.3.3 Multiple Base Stations

If multiple base stations are deployed within a wireless sensor network, nodes can maintain a list of potential forwarders that would direct packets towards the closest base station. Following the same steps as described in Initial Setup, each base station can broadcast a packet that contains a hop distance counter and its ID. A node can determine which one is the closest by the hop messages it receives, selecting the one that is closest so far, and only forwarding the messages containing the base station ID it has selected.

3.3.4 Base Station to Nodes Traffic

While convergecast is the predominant traffic pattern, base station to nodes (also known as *forward-direction traffic*) is another form of traffic that is possible in wireless sensor networks. The base may want to query a particular region of nodes for information or issue commands [25]. In such a case, we assume that localization has been performed such that each node knows the geographic location of its neighbors and its own location, whether it'd be relative or absolute. Readers can refer to [5, 22] for examples on localization techniques. The routing table should contain one-hop neighbors from all directions and their location in addition to the time offset and frame offset. When a packet is sent out from a base station, we assume that the destination's location is also embedded into the packet such that a router will know which direction the packet is heading and therefore who the best-suited forwarder is. To reduce delay, a node can select a forwarder that is closest to waking up and is closer to the destination than it is. An altered neighbor selection algorithm is shown in Algorithm 3 where $dstX$ and $dstY$ are the coordinates of the desired destination and $calcDistance$ calculates the distance between two nodes.

The path taken may not be the shortest, but each hop results in the packet moving closer to the destination in the fastest manner possible. Contention and collisions can be solved in the same way that was explained in the previous section, where the node can select a new forwarder that is also closer to the destination.

3.4 Maintenance and Scalability

With forwarders readjusting their duty cycles dynamically throughout the course of the network and clock drifts affecting the stored time offsets, LoC-MAC requires nodes to keep their table information fresh in order to continue accurately predicting their forwarders' wake periods. In ECR-MAC, nodes can easily adjust their duty cycles at no cost and has no worries about neighboring nodes' sense of time. For a longer lifetime of energy savings, LoC-MAC has a small price to pay of additionally maintaining its information.

Algorithm 3 ForwarderSelection

```
Input: $T_{nextWake}, F_{me}, dstX, dstY$
Output: $fID, T_{wait}$
1: $firstForwarder = true$
2: for $i = 0$ to $|FT|$ do
3:    if $FT_i.ID = baseStationID$ then {no need to wait or select any other forwarder}
4:       $T_{wait} = 0$
5:       $fID = i$
6:       Return $fID, T_{wait}$
7:    else
8:       $off = FT_i.F_o - F_{me}$
9:       $off += FT_i.T_o$
10:      if $off < 0$ then
11:         $off += T_{frame}$
12:      end if
13:      if $off + T_{nextWake} > T_{frame}$ then
14:         $off -= T_{frame}$
15:      end if
16:      $myDst = calcDistance(myX, myY, dstX, dstY)$
17:      $neiDst = calcDistance(FT_i.X, FT_i.Y, dstX, dstY)$
18:      if firstForwarder or (($off$ + $T_{nextWake} < T_{wait}$) and ($neiDst$ ¡ $myDst$)) then
19:         $fID = FT_i.ID$
20:         $T_{wait}$ = $off$ + $T_{nextWake}$ {saving the forwarder that is closest to waking up so far}
21:         firstForwarder = false
22:      end if
23:   end if
24: end for
25: if firstForwarder == false then {a forwarder was not found}
26:    $fID = -1$
27: end if
28: Return $fID, T_{wait}$
```

Since a node does not know who it acts as a forwarder for, the challenge lies in when and how update messages should be sent out. Further understanding when and why a node would want to change its schedule will give a better perspective on how to devise a solution. When there is no activity, nodes will go about their normal schedules. As soon as an event occurs, sources will be going after forwarders that are the closest to waking, some of which may be shared by non-neighboring nodes. If a forwarder is consistently receiving corrupted messages, it can assume that non-neighboring senders have been trying to reach it or it just happened to wake-up at a bad time where other transmissions are already taking place. By having a node select a new schedule, after reaching a threshold of collided or overheard packets, an immediate notification of the change can warn senders not to select this forwarder for awhile to avoid future collisions. Not only that, wake-up periods start to become more evenly dispersed in the eyes of a sender, which lessens sleep latency.

One extreme, yet complete, solution would be for a node to store information on all its one-hop neighbors. Knowing each of their wake schedules, a node can individually notify all its same-hop and one-hop further away neighbors with an update message. This would not scale well at all, since as node density increases, the larger the potential forwarders table would have to be and the more messages that would have to be sent out. Also, at a time of high contention and congestion, the node would be out of commission while senders may continue to select this node. To have an immediate affect on notifying at least some of the senders when a forwarder decides to change its offset, it can send out a broadcast (not unicast anymore) REPLY message with the new offset value included after receiving the last collided or overheard WAKEUP message. This message plays two roles, one is to notify the updated frame offset and the other as a warning that an unseen neighbor may also be utilizing it as a forwarder currently. Senders can then flag this node and avoid selecting it as a forwarder for the current event reporting period, thus saving energy by not wasting it on future collisions. There still lies the issue of notifying other neighboring nodes of the change in schedule, who may have been sleeping when it issued a broadcast REPLY. Once the channel has been idle for some time, indicating that the last event reporting is over, the node can broadcast small back-to-back UPDATE messages for each available wake-up slot within one frame ($T_{sleep} + T_{wake}$). To reduce the cost of sending out the message, it contains minimal information such as the node's ID and its new frame offset. A neighboring node that receives an UPDATE, will search through the table to see if the sender of the message is one of its forwarders, and if so, update the frame offset field.

For overcoming the effects of clock drifts, a node should periodically exchange time-stamped packets during down time with its forwarders using the method described by Maroti *et. al* [33]. A node should keep an additional field in the routing table which describes the last time a neighbor's time offset information has been updated. This can be used to calculate the estimated time before the clock drift has an effect on the time offset value, and thus when a node should update the time offset before it becomes too inaccurate in that it is no longer able to predict the wake-up period and cannot even start a time-exchange task. The periodocity of such needed updates depend on the difference between two nodes' crystal frequencies.

In terms of scalability, the more dense a network is, the more set of potential forwarders a node will have. Average latency will be reduced since a node will have less sleep latency at each hop due to the increase in number of forwarders waking up within a time frame. The more sparse a deployment, the less the number of potential forwarders a node has which increases overall latency. Instead of staying stagnant, performance can improve as the density increases.

3.5 Analysis

Here we provide an analysis on the average performance of LoC-MAC in terms of end-to-end delay and energy consumed for a single-source scenario. For simplicity, we assume the network has an even distribution of n nodes across an $l \times l$ area.

3.5.1 Delay

The total end-to-end delay is dependent on the source's distance from the base station in terms of hops H and the time it takes for a data exchange:

$$T_{delay} = H \times T_{exchange} = H \times (T_{setup} + T_{contention} + T_w + T_r + T_d + T_a) \tag{3.1}$$

Within $T_{exchange}$, T_{setup} represents the amount of time a node had to wait before its desired forwarder wakes up, $T_{contention}$ is the random period a node waits before checking if the channel is idle, and the others are the time it takes to transmit WAKEUP, REPLY, DATA, and ACK messages.

Here we follow ECR-MAC's method of calculating T_{setup}, which is first done by identifying the number of potential forwarders PF_{num} a node has on average. Let us first look at figure 3.6, which displays an example of three nodes (A, B, and C) and their transmission range. The set of rings represents the hop distance a packet would have to travel to reach the base station (which is at the center of all ther rings). The shaded regions represent forwarders that are one-hop closer to the base station. Node B has a large set, whereas node C has a smaller set and may need to rely on other neighbors that are in the same ring. Node A represents an average set, such that a packet travels half a hop closer to the base station, and will be used as the example for determining the average number of forwarders. Using rules of geometry and trigonometry, we can estimate the area where one-hop closer forwarders reside, and then multiply it by the network density $\left(\frac{n}{l^2}\right)$ to obtain an average number of forwarders.

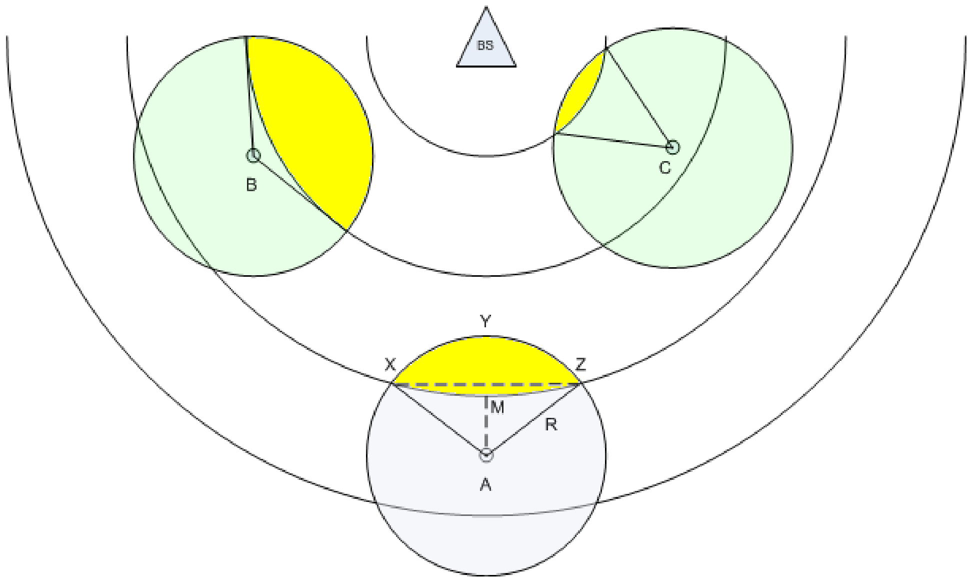


Figure 3.6: Area of Potential Forwarders

Below are the set of equations used to approximately calculate segment $\widehat{XMZ}$ (note that θ

represents the angle $\angle XAZ$ which is $\frac{2\pi}{3}$ radians).

$$\begin{aligned} A_{segment} &= A_{sector} - A_{triangle} \\ &= \frac{1}{2}R^2\left(\theta - sin\theta\right) \\ &= \frac{1}{2}R^2\left(\frac{2\pi}{3} - sin\frac{2\pi}{3}\right) \\ &\approx 0.614185 \times R^2 \end{aligned} \tag{3.2}$$

The average number of potential forwarders (that are one hop closer to the base station) is then:

$$PF_{num} = \left(0.614185 \times R^2\right) \times \frac{n}{l^2} \tag{3.3}$$

We assume that the randomly selected wake-up slots for each forwarder is evenly distributed among a frame ($f = T_{wake} + T_{sleep}$), such that the time between each slot is $T_{mxWait} = \frac{f - PF_{num}T_{wake}}{PF_{num}}$ (example in Figure 3.7). There are two different instances for T_{setup} which depend on the total length during a frame which is occupied by wake-up slots:

1. $f < PF_{num}T_{wake}$: $T_{setup} = 0$
2. $f \geq PF_{num}T_{wake}$: $T_{setup} = \frac{f - PF_{num}T_{wake}}{2PF_{num}}$

For the first instance, an entire frame is consumed by wake-up slots such that a node would not have to wait at all. For the second, there are still periods at which a forwarder is not available. In the best case, a node has a forwarder that is immediately available, having a wait time of zero; while in the worst case a node has to wait T_{mxWait}. The average should then be $\frac{T_{mxWait}}{2}$.

$T_{contention}$ is essentially the sum of the contention window and T_w which all occurs within a wake-up period T_{wake}. We have set the contention window to half of T_{wake}, while the other half is the time it takes to send a WAKEUP packet. On average, a node will then select a carrier sense slot at $\frac{\text{contention window}}{2} = \frac{T_{wake}}{4}$.

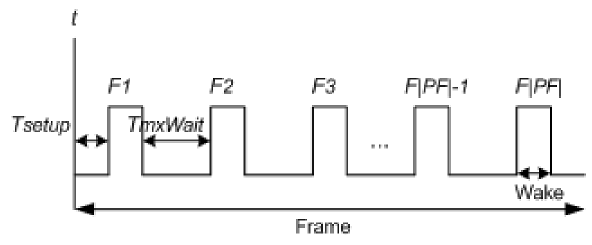


Figure 3.7: Distributed Wakeup Periods of Potential Forwarders

3.5.2 Energy Consumption

To calculate the overall energy consumption of the entire network (summing energy consumed by all n nodes), we take into account energy spent during idle time E_{idle}, transmission E_{tx}, reception E_{rx}, and sleep E_{sp}. Given a time T_{run}, the total energy consumed since the network started running up to T_{run} is:

$$E_{total} = E_{wake} + \left(\sum_{1}^{P} E_{exchange} \right) + E_{sleep} \tag{3.4}$$

E_{wake} approximates the total energy consumed while the node was in its wake period, $E_{exchange}$ is the sum of energy consumed for each packet that is reported (P being the total number of reports), and E_{sleep} is the approximated energy consumed during total sleep time.

$$E_{wake} \approx dc \times T_{run} \times E_{idle} \times n \tag{3.5}$$

$$E_{exchange} = [E_{idle}T_{contention} + E_{sleep}T_{setup} + T_{exchange}\left(E_{tx} + E_{rx}\right)] \times H \tag{3.6}$$

$$E_{sleep} \approx E_{sp} \times [n \times T_{run} - [H \times (T_{contention} + T_{setup} + 2T_{exchange}) + (n \times dc \times T_{run})]] \tag{3.7}$$

Note that $T_{exchange} = T_w + T_r + T_d + T_a$. E_{wake} uses the duty cycle ratio dc to determine the percentage of time in T_{run} where nodes are in WAKE mode. This may overlap the time spent during message exchange (in ACTIVE mode) and hence is an over-approximation. $E_{exchange}$ is calculated by taking into account the distance the data has to travel (H hops) and the time spent in contention, setup, and data exchange. In the case that the entire network is active (has their radio transmitter on) at all times during T_{run}, $E_{sleep} = 0$. Otherwise, E_{sleep} is calculated by subtracting the total time spent awake and during data exchange from the total run time.

CHAPTER FOUR

PERFORMANCE EVALUATION

To evaluate the performance of LoC-MAC, we have implemented and tested the protocol in ns-2, a discrete network event simulator [1], against varying parameters and scenarios. CSMA and ECR-MAC [57] are used as experimental comparisons and their code were also run on ns-2. CSMA is a fully active MAC (i.e. all nodes have their radio on at all times), which is used as a baseline in measuring delay since it does not suffer from any sleep latency. ECR-MAC is the basis for our work, making it a necessary comparison to see if the implemented changes has improved its performance. Simulation parameters are shown in Table 4.1 and experimental procedures are the same as those that were done with ECR-MAC in [57] in order to assess LoC-MAC's performance in a similar manner. With a single-source scenario, we can take a close look at how LoC-MAC performs in end-to-end delay without the concern of congestion and how much energy is consumed as packet generation rate increases. To emulate a more realistic situation in a dense network where spatially-correlated contention is indeed prevalent, we had a multiple-source scenario where several nodes sense the same 'event' and report data around the same time. Scalability has also been tested, where throughput and energy consumption were evaluated across the average number of neighbors a node has.

In all experiments (except for scalability), 500 nodes are randomly spread across an $80m \times 80m$ field and each node has a transmission range of $15m$. For ECR-MAC, its duty cycle is 1$\%$ and the wake period is 88ms (as reported in its experiment). LoC-MAC's duty cycle is 0.8$\%$ and has a wake period of 70.4ms, which is sufficiently long enough for a small contention period and the reception of a WAKEUP packet. In terms of routing for CSMA, we used a data gathering tree for forwarder selection, where the next hop for a node is the closest neighbor to the base station. Simulations were run for 150 seconds which is long enough for an event to be reported, and each reported data point is an average of 20 simulation runs with different seeds.

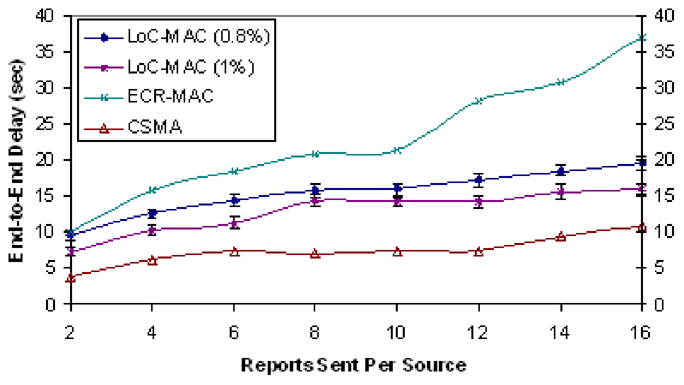
Table 4.1: Simulation Parameters

Parameter	Value
# of Nodes	500
Area	$80m \times 80m$
Node Density	0.078 nodes/m^2
Transmission Range	15 m
Data Packet Size	64 bytes
Bandwidth	2.4 Kbps
Transmit Power	14.88 mW
Receive Power	12.50 mW
Idle Power	12.36 mW
Sleep Power	0.016 mW

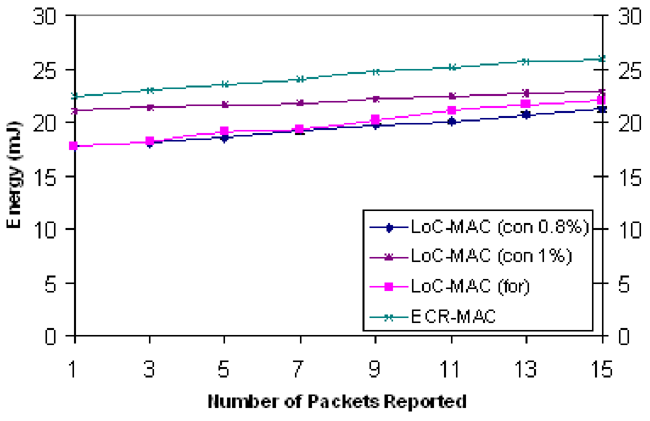
4.1 Single-Source Scenario

With a single-source scenario, we examined the end-to-end delay depending on a node's distance from the base station and the amount of energy consumed over the number of packets reported. For each simulation run, a source node is randomly selected from the scene with the base station remaining in the upper right corner of the topology. 95% confidence intervals are plotted for LoC-MAC results, to show the effect of randomly selected duty cycles at each run. Figure 4.1 displays the results for three types of LoC-MAC, two for convergecast traffic (both 0.8% and 1% duty cycle) and the other being forward direction (with 0.8% duty cycle), ECR-MAC, and CSMA (for delay only).

In the end-to-end delay test (Figure 4.1(a)), one packet was sent from a randomly selected source under different hop distances from the base station. For forward-direction LoC-MAC, the base station was the source and the destination was a random node in the network. The X-axis is labeled 'ideal' in that even if we select a node that is, for example, six hops away from the base station, ECR-MAC and LoC-MAC will not necessarily take a path that is six hops long. CSMA has the best delay since all nodes are active at all times, thus a packet can be delivered without any concern of sleep latency. We first expected LoC-MAC (convergecast) to have similar results



(a) End-to-end Delay vs. Hop Distance



(b) Energy Consumed vs. Packets Reported

Figure 4.1: Single Source Scenario

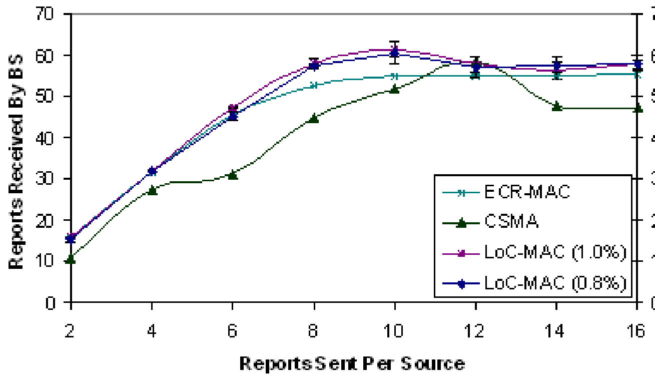
with ECR-MAC in terms of delay since both end up with a forwarder that is closest to waking up. However, we must keep in mind that there is a trade-off between delay and energy efficiency. By having a lower duty cycle, the sleep latency at each hop has increased. While the occurrence of the number of wake periods within a frame (wake+sleep time) should be around the same as ECR-MAC since we both use a frame of 8.8 seconds and the same node density, the length of the overall wake periods has decreased. Thus, a node ends up having to wait longer for a forwarder to wake-up in LoC-MAC than in ECR-MAC. A LoC-MAC with 1% duty cycle (changed frame to 7.04 seconds) has been plotted to show this difference, which has similar results as ECR-MAC. Due to the variation of duty cycle schedules with each simulation run, the error bars in LoC-MAC increase with each hop distance, since there are cases where the randomly selected schedules either work in favor or against a sender. With the forward direction LoC-MAC, the delay is slightly higher since the path taken is not determinant on trying to find a forwarder that is "one-hop closer," but rather finding any forwarder that is closer to the destination than the sending node itself and is the closest to waking up.

For energy consumption (Figure 4.1(b)), a randomly selected node that is six hops away from the base station sends out packets with differing intervals. The energy consumed is an average among all nodes. CSMA has not been plotted since the numbers are too high, which is expected due to nodes being awake at all times, wasting energy over idle listening and overhearing. With the reduction in its duty cycle, both convergecast and forward-direction versions of LoC-MAC consume less energy than ECR-MAC. However, it is not only the change in duty cycle, but the method itself that is energy-saving. LoC-MAC with 1% duty cycle still saves more energy than ECR-MAC, due to the decrease in control packet transmissions. Forward-direction traffic consumes a little more in energy than its counterpart due to the longer end-to-end delay. Although LoC-MAC was designed with convergecast in mind, its forwarding mechanism does not have any detrimental effect on forward-direction traffic.

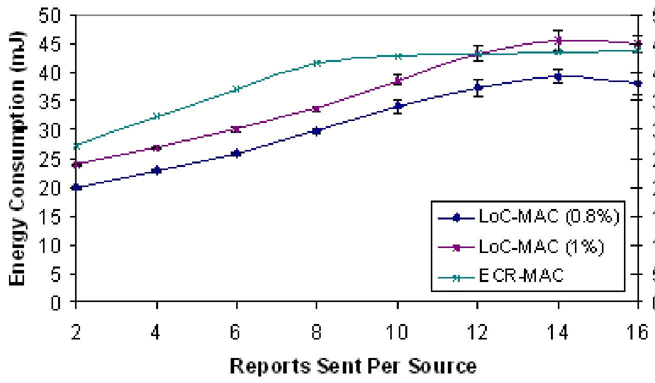
4.2 Multiple Sources Scenario

In a densely deployed network, an event can trigger several nodes into generating and sending out reports. To better simulate an event-reporting activity, we randomly selected an event site that is six hops away from the base station with eight sources that are within ten meters of the event. Each of them send a packet around the same time, having some variance to emulate differences in the time spent sensing and generating reports. Figure 4.2 has the results from our experiments which test throughput, energy consumption, and the end-to-end delay of the first 10% of generated reports that arrive.

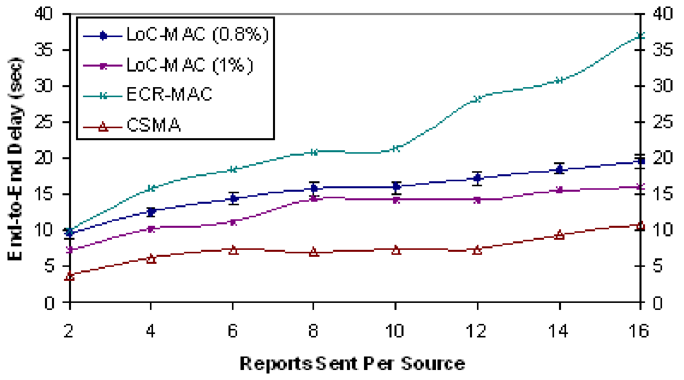
As packet rate increases, the probability of collision increases due to the hidden node problem, which in turn results in more packet retransmissions. For ECR-MAC and LoC-MAC, with every ACK that is lost, a duplicate packet is created, injecting more packets into the network. For throughput, we only count the number of original packets that reach the base station, discounting duplicate packets. Regular CSMA, without a virtual carrier sense, cannot avoid collisions that occur from a node transmitting at the same time a hidden node is, which puts it in the same position as LoC-MAC and ECR-MAC. However, the difference in results lies in the fact that the data gathering tree used causes high contention and the backoff period increases with each failed attempt



(a) Reports Received by the Base Station



(b) Energy Consumption

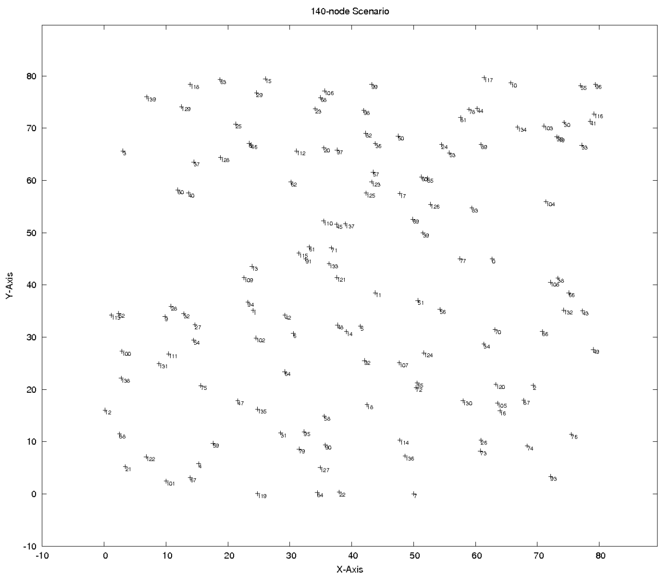


(c) End-to-end Delay of First 10% Reports

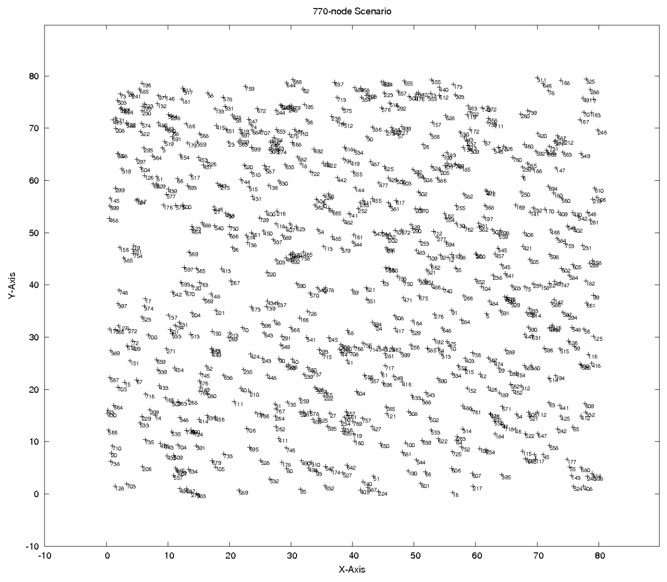
Figure 4.2: Multiple Sources Scenario

(i.e. the contention window increases by two), which is why the throughput is the least among the three. LoC-MAC, for both 0.8% and 1%, has around the same throughput in the beginning (receiving all packets that are generated), but overtakes ECR-MAC as packet generation rate increases, showing the effectiveness of the technique as contention increases. The channel is not dominated by WAKEUP packets as in ECR-MAC, but is instead more available to allow other data to flow.

Energy consumption (Figure 4.2(b)) is again, lower for LoC-MAC due to the low duty cycle which reduces idle time and overhearing, and the reduction in control packets used. There is also less time spent in an active/wake mode since a node can go to sleep and wait until its selected forwarder wakes up as opposed to ECR-MAC, which remain awake for as long as necessary until



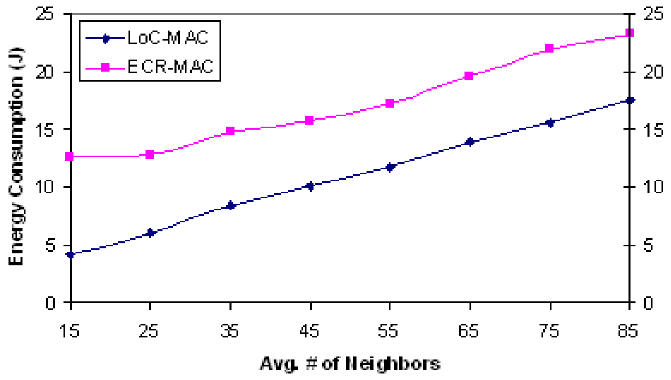
(a) 140-Node Scenario



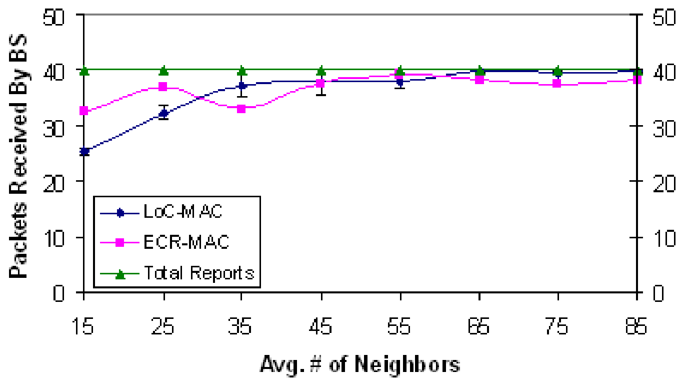
(b) 770-Node Scenario

Figure 4.3: Sparse vs. Dense Network

a forwarder replies to one of its wake-up messages. For 1% LoC-MAC, the same trend is followed as 0.8%, but the energy consumed rises slightly higher than ECR-MAC as packet generation rate increases. For end-to-end delay (Figure 4.2(c)), we average the delay of the first 10% of packets that have been reported (e.g. for 120 reports that are generated, we calculate the delay of the first 12 packets that have been reported) as done in ECR-MAC since there are applications that require a base station to respond to an event immediately [23]. CSMA still has the lowest delay since there is no sleep latency and packets have not yet backlogged. Unlike in the single source scenario, LoC-MAC has much lower delay than ECR-MAC as packet rate increases due to the availability of the channel, since a node goes to sleep while it is waiting for its forwarder to wake-up. This allows other neighboring nodes to communicate in the mean time, allowing for a faster method of moving data packets away from the event site. While CSMA is the lowest among delays, LoC-MAC has shown that it follows a similar trend, whereas ECR-MAC starts to increase in delay at a higher rate.



(a) Density vs. Energy Consumption



(b) Density vs. Packets Reported

Figure 4.4: Scalability Testing

4.3 Scalability

To test scalability performance, we varied the number of deployed nodes across an $80m \times 80m$ field. Figure 4.3 displays what a sparse versus a dense network would look like. Similar to the multi-source experiments, a random 'event' site is detected six hops away from the base station with eight sources generating reports on it. Packets are sent every 30 seconds, which result in a total of 40 packets that should arrive at the base station.

In Figure 4.4(a) and (b), we have the results for the total energy that is consumed amongst all nodes (as opposed to the average that was calculated in the prior experiments) and the throughput. The average neighbor of nodes was determined by multiplying a node's transmission area (πr^2) with the network density $(\frac{n}{l^2})$. LoC-MAC follows the general trend of ECR-MAC in terms of energy consumption, but is lower due to the smaller duty cycle. For throughput, both LoC-MAC and ECR-MAC suffers at a lower density since nodes have to wait a longer time for one of its forwarders to wake up. ECR-MAC suffers from the occurrence of REPLY collisions. The more neighbors it has, the higher the possibility that two non-neighboring forwarders will reply to a WAKEUP message around the same time. Whereas with LoC-MAC, there are no REPLY collisions and there is less sleep latency since a node has more neighbors to select from as the density increases.

4.4 Performance Summary

Overall, we have seen the tradeoff between energy and delay in the single source scenario, where LoC-MAC has higher delay than ECR-MAC, but consumes less energy. A fully active MAC (CSMA) always has better delay since it does not suffer from any sleep latency, but at the cost of the entire network depleting its energy quickly. LoC-MAC has around the same performance with ECR-MAC in terms of throughput for multiple source scenarios and even better in delay, handling spatially-correlated contention quicker. As for differences in duty cycles (1% and 0.8%), the lower duty cycle is shown to have similar performance as the higher one, while still saving more energy overall. Scalability-wise, LoC-MAC performs better in throughput as the network density increases since it explicitly selects the forwarder that is closest to waking up, as opposed to ECR-MAC which allows any forwarder that has woken up to respond, resulting in REPLY collisions. All in the meanwhile, LoC-MAC expended the least amount of energy by reducing the time a node is spent awake and the amount of control packets that are sent out.

CHAPTER FIVE

CONCLUSIONS

Energy-efficiency has been a focal point of discussion within medium access control in wireless sensor networks in order to ensure its longevity. Communication is a draining task and having a node listen to the medium at all times depletes energy quickly, not only due to idle time, but overhearing. With duty cycles in place, nodes can switch their radio transmitters on and off to save energy. Several MACs have been proposed which either synchronizes schedules to allow nodes to be awake around the same time to communicate or uses independent schedules, where preambles and wake-up packets have been used to notify a neighboring node that data needs to be transmitted. ECR-MAC came up with a novel approach of utilizing multiple forwarders to facilitate faster communication in convergecast traffic and overcome spatially-correlated contention. Predecessors assumed there is only one forwarder that it is allowed to transmit to (selected by the routing layer), not looking at contention from a higher perspective.

Our proposed LoC-MAC uses neighborhood information to improve upon ECR-MAC's energy consumption, with the goal of still maintaining the same performance. Since ECR-MAC does not know about its neighbors, it sends a stream of periodic messages to notify any potential forwarder that data is to be sent, consuming unnecessary energy. With LoC-MAC, a node maintains information on its neighbors wake and sleep schedules to predict when a potential forwarder is going to wake up next. When there is data to send, a node searches through its table of forwarder information for the neighbor that is closest to waking up. By employing this mechanism, we were able to reduce the duty cycle, since less time is required for a node to scan the channel for a wake-up message, and the time that is spent in active mode, searching for a neighbor that can act as a forwarder as done in ECR-MAC.

Performance evaluation has shown LoC-MAC has better results than ECR-MAC in terms of throughput and has better delay in a multiple source scenario, while consuming less energy overall.

In a single-source scenario, we have seen the trade-off between energy and delay, where LoC-MAC suffers from higher sleep latency at each hop, despite the higher energy savings. Future work can include employing a more intelligent forwarder selection method which can take into account other performance metrics or analyzing the surrounding traffic to better handling coordination.

BIBLIOGRAPHY

[1] The Network Simulator (ns-2), a discrete event simulator, *http://www.isi.edu/nsnam/ns*.

[2] OPNET, network simulator, *http://www.opnet.com*.

[3] QualNet, software virtual network, *http://www.scalable-networks.com/*

[4] TinyOS, an open-source OS for the network sensor regime, *http://www.tinyos.net/*

[5] A. Ahmed, H. Shi, and Y. Shang, "Sharp: A new approach to relative localization in wireless sensor network," in *IEEE International Conference on Distributed Computing Systems Workshop (ICDCSW)*, pp. 892-898, 2005.

[6] O.B. Akan and I.F. Akyildiz, "Event-to-sink reliable transport in wireless sensor networks," in *IEEE Infocom*, 2005.

[7] M. Ali et al., "Medium access control issues in sensor networks," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, April 2006.

[8] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006, pp.307-320.

[9] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: ultra-low power data gathering in sensor networks," in *International Conference on Information Processing in Sensor Networks.*, April 2007.

[10] H. Cao, K. Parker, and A. Arora, "O-MAC: A Receiver Centric Power Management Protocol," in *14th IEEE International Conference on Network Protocols*, November 2006.

[11] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *Information Processing in Sensor Networks*, 2005.

[12] T.V. Dam and K. Langendoen, “An adaptive energy-efficient MAC protocol for wireless sensor networks,” in *1st ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[13] I. Demirkol, C. Ersoy, and F. Alagoz, “MAC protocols for wireless sensor networks: a survey,” in *IEEE Comunications Magazine*, vol. 44, 2006, pp. 115-121.

[14] M. Dhanaraj, B. S. Manoj, and C.S.R. Murthy, “A new energy efficient protocol for minimizing multi-hop latency in wireless sensor networks,” in *Pervasive Computing and Communications*, 2005, pp. 117-126.

[15] A. El-Hoiydi and J.D. Decotignie, “WiseMAC: An Ultra Low Power MAC Protocol for the Downlink of Infrastructure Wireless Sensor Networks,” in *IEEE Symposium on Computers and Communication*, June 2004, pp. 244-251.

[16] J. Elson, L. Girod, and D. Estrin, “Fine-Grained network Time Synchronization using Reference Broadcasts,” in *Proceedings of the fifth symposium on Operating System Design and Implementation (OSDI)*, December 2002.

[17] C.C. Enz et al., “WiseNET: An ultralow-power wireless sensor network solution,” in *IEEE Comp.*, vol. 37, no. 8, August 2004.

[18] S. Ganeriwal, R. Kumar, and M.B. Srivastava, “Timing-Sync Protocol for Sensor Networks,” in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 138-149, November 2003.

[19] M. Gastpar and M. Vertterli, “Source-channel communication in sensor networks,” in *Proc. 2nd Int. Workshop on Information Processing in Sensor Networks (IPSN’03)*, vol.219, pp.162-177, 2003.

[20] Y. Gu and T. He, “Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links,” in *International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2007.

[21] I. Hakala and M. Tikkakoski, “From vertical to horizontal architecture: a cross-layer implementation in a sensor network node,” in *ACM International Conference on Integrated Internet Ad Hoc and Sensor Networks*, 2006.

[22] J. Hightower and G. Borrilello, “Location systems for ubiquitous computing,” in *IEEE Computer*, 2001.

[23] K. Jamieson, H. Balakrishnan, and Y. Tay, “Sift: A mac protocol for event-driven wireless sensor networks,” in *Third European Workshop on Wireless Sensor Networks (EWSN)*, Februaru 2006.

[24] I. Joe and H. Ryu, “A Patterned Preamble MAC Protocol for Wireless Sensor Networks,” in *International Conference on Computer Communications and Networks*, August 2007.

[25] A. Keshavarzian, H. Lee, and L. Venkatraman, “Wakeup scheduling in wireless sensor networks,” in *International Symposium on Mobile Ad Hoc Networking and Computing.*, 2006, pp. 322-333.

[26] N.P. Khan and C. Boncelet, “PMAC: Energy Efficient Medium Access Control Protocol for Wireless Sensor Networks,” in *Military Communications Conference*, October 2006.

[27] K. Langendoen and G. Halkes, “Energy-Efficient Medium Access Control” in *The Embedded Systems Handbook*, CRC Press, 2005.

[28] J. Li and P. Mohapatra, “An analytical model for the energy hole problem in many-to-one sensor networks,” in *IEEE Vehicular Technology Conference*, pp. 2721-2725, September 2005.

[29] A. Liu, L. Li, H. Yu, and D. Zhang, “An Energy-efficient MAC Protocol Based on Routing Information for Wireless Sensor Networks,” in *IEEE Wireless Communications & Networking Conference (WCNC)*, 2007.

[30] S. Liu, K. Fan, and P. Sinha, “CMAC: An Energy Efficient MAC Layer Protocol Using Convergent Packet Forwarding for Wireless Sensor Networks,” in *IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON)*, June 2007.

[31] G. Lu, B. Krishnamachari, and C.S. Raghavendra, “An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks,” in *Parallel and Distributed Processing Symposium.*, April 2004, pp. 224.

[32] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, ”Delay efficient sleep scheduling in wireless sensor networks,” in *IEEE INFOCOM*, 2005, pp. 2470-2481.

[33] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, “The Flooding Time Synchronization Protocol,” in *International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[34] E. Nakamura, A. Loureiro, and A. Frery, “Information fusion for wireless sensor networks: Methods, models, and classifications,” in *ACM Computing Surveys (CSUR)*, vol. 39, no. 3, September 2007.

[35] S. Pack, J. Choi, T. Kwon, and Y. Choi, “TA-MAC: Task Aware MAC Protocol for Wireless Sensor Networks,” in *Vehicular Technology Conference (VTC)*, vol. 1, pp. 294-298, 2006.

[36] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in *International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[37] V. Rajendran, K. Obraczka, J.J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," in *International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[38] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a hybrid MAC for wireless sensor networks," in *International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[39] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed Randomized TDMA Schedulilng For Wireless Ad-hoc Networks," in *MobiHoc*, May 2006.

[40] X. Shi and G. Stromberg, "SyncWUF: An Ultra Low-Power MAC Protocol for Wireless Sensor Networks," in *IEEE Transactions on Mobile Computing*, vol. 6, January 2007.

[41] S. Singh and C.S. Raghavendra, "PAMAS- power aware multi-access protocol with signalling for ad hoc networks," in *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 3, July 1998.

[42] L. Song and D. Hatzinakos, "A cross-layer architecture of wireless sensor networks for target tracking," in *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 1, February 2007.

[43] F. Stann, J. Heidemann, R. Shroff, and M. Murtaza, "RBP: robust broadcast propagation in wireless networks," in *International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006, pp. 85-98.

[44] C. Suh and Y. Ko, "A Traffic Aware, Energy Efficient MAC Protocol for Wireless Sensor Networks," in *IEEE International Symposium on Circuits and Systems*, 2005.

[45] C. Vigorito, D. Ganesan, and A. Barto, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," in *IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON)*, 2007.

[46] M.C. Vuran, O.B. Akan, and I.F. Akyildiz, “Spatio-temporal correlation: theory and applications for wireless sensor networks,” *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 45, no. 3, pp. 245-259, June 2004.

[47] M. Vuran and I. Akyildiz, “Spatial Correlation-based Collaborative Medium Access Control in Wireless Sensor Networks,” in *IEEE/ACM Transactions on Networking*, vol. 14, pp. 316-329, April 2006.

[48] L. Wang and K. Liu, “An Energy-Efficient and Low-Latency MAC Protocol for Wireless Sensor Networks,” in *IEEE International Symposium on Microwave, Antenna, Propagation, and EMC Technologies For Wireless Communications*, August 2007.

[49] G. Xing, and et. al., “Minimum power configuration for wireless communication in sensor networks,” in *ACM Transactions on Sensor Networks*, vol. 3, no. 2, 2007.

[50] X. Yang and N. Vaidya, “A wakeup scheme for sensor networks: Achieving balance between energy saving and end-to-end delay,” in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2004.

[51] W. Ye, J. Heidemann, and D. Estrin, “An Energy-Efficient MAC Protocol for Wireless Sensor Networks,” in *INFOCOM*, 2002.

[52] W. Ye, J. Heidemann, and D. Estrin, “Medium access control with coordinated, adaptive sleeping for wireless sensor networks,” in *IEEE/ACM Trans. Net.*, vol. 12, no. 3, pp.493-506, June 2004.

[53] W. Ye, F. Silva, and J. Heidemann, “Ultra-low duty cycle MAC with scheduled channel polling,” in *International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[54] S. Yessad, F. Nait-Abdesselam, T. Taleb, and B. Bensaou, “R-MAC: Reservation Medium Access Control Protocol for Wireless Sensor Networks,” in *IEEE Conference on Local Computer Networks*, pp. 719-724, October 2007.

[55] M. Zhao, Z. Chen, Z. Ge., and L. Zhang, “HS-Sift: a Hybrid Spatial Correlation-based MAC for Event-driven Wireless Sensor Networks,” in *First International Conference on Communications and Networking in China*, October 2006.

[56] S. Zhou, R. Liu, D. Everitt, and J. Zic, “A^2-MAC: an application adaptive medium access control protocol for data collections in wireless sensor networks,” in *International Symposium on Communications and Information Technologies (ISCIT)*, October 2007.

[57] Y. Zhou and M. Medidi, “Energy-efficient contention-resilient medium access for wireless sensor networks,” in *IEEE International Conference on Communications (ICC)*, 2007.

[58] Y. Zhou and M. Medidi, “Sleep-based topology control for wakeup scheduling in wireless sensor networks,” in *IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON)*, June 2007.

[59] M. Zorzi and R.R. Rao, “Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance,” in *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, October-December 2003.