

QMAC: A Quality of Service-Oriented Medium Access Control Protocol for Data Gathering in Wireless Sensor Networks

Winnie Louis Lee,* Amitava Datta,* Rachel Cardell Oliver*

10 August 2005

Abstract

We propose QMAC, a novel TDMA-based protocol for efficient data gathering in wireless sensor networks that delivers quality of service: predictable latency, predictable throughput, fair access, and robust self-healing, whilst also respecting the severe operating constraints of WSNs (energy and memory). QMAC achieves this balance through a flexible slot structure in which nodes in the network can build, modify, or extend their schedules based on their local information. This scheme allows QMAC to be strongly fault tolerant and also highly energy efficient. QMAC further minimizes energy by selecting optimum node transmission power for a given topology. QMAC is also scalable for large number of nodes because it allows communication slots to be reused by nodes outside each others' interference range, and its depth-first-search schedule minimizes buffering. This paper presents analysis and simulations to test QMAC against quality of service metrics, showing that QMAC ensures good quality of service while achieving energy efficiency and memory efficiency, for different network configurations (network topology and network density), providing an efficient solution for data gathering applications.

Index Terms : Sensor networks, power management, performance attributes, fault tolerance.

*School of Computer Science & Software Engineering, University of Western Australia, Perth, WA 6009, Australia. email:winnie,datta,rachel@csse.uwa.edu.au

1 Introduction

Wireless sensor networks are used to gather information in diverse settings including natural ecosystems, battlefields, and man made environments [1], [2], [3], [4]. The users of sensor networks need guarantees on the quality of service that the network can provide [5], [6], [7], [8]. In practice, however, it has proved difficult to achieve quality of service guarantees, because of the severe resource constraints (battery power and data memory) of sensor network nodes, and the hostile environments in which they must operate [1], [9].

This paper introduces the QMAC protocol: a self-healing protocol designed for periodic data gathering applications. QMAC is a schedule-based (TDMA) MAC protocol, in which each node owns time slots for transmitting and receiving data. Data gathering nodes sense their environment periodically at a fixed rate and deliver data to a base node [10]. Data gathering nodes serve as both gatherers and routers for the data. QMAC's flexible slot structure provides strong fault tolerance, whilst maintaining highly energy efficient operation.

The term *quality of service* (QoS) in wireless sensor networks is used to refer to guarantees with respect to the application (such as sensor coverage), reliability assurance (guaranteed delivery of critical messages), or traditional end-to-end guarantees on data delivery (such as latency and throughput) [6]. In this paper we focus on the end-to-end guarantees on data delivery. The QMAC protocol offers end-to-end QoS by guaranteeing:

- *predictable latency* in delivering periodic data from sensor nodes to a base station;
- *predictable throughput* for gathered data;
- *fair access* to the network for all data gathering nodes;
- *robust self-healing* of the network when nodes join or leave the network, and when communication conditions change [9], [11].

QMAC is designed for a static, ad hoc network, in which nodes may suffer temporary or permanent failure, and new nodes may be added to the network at any time but nodes otherwise remain in (nearly) the same location. QMAC adjusts to different node topologies and densities, to create an efficient, self-healing data delivery tree. The protocol scales for large numbers of nodes because after an initial one-off set up phase, all repair operations are local. QMAC nodes require minimal local buffering since the network schedule is chosen so that each node forwards messages immediately.

A wireless sensor network's energy budget is the sum of the battery energy at each of its nodes. QMAC is designed to share the energy load fairly between its homogeneous network nodes, and so to maximize the lifetime of the whole network. Buffering and retransmission are used to recover from a small number of lost packets, and averaging is used to aggregate information in the event of prolonged hostile communication conditions.

The central contribution of the QMAC protocol is its loose slot structure in which nodes in the network can build, modify, or extend their schedules on the fly. Nodes can claim or remove a slot based on the current information in their lookup table without exchanging information with any other nodes in the network prior to modifying their schedules. In QMAC, it is not necessary to specify the number of slots required for the network in advance and nodes can join or leave the network at any time. This flexible slot structure makes QMAC strongly fault tolerant and also highly energy efficient. QMAC provides QoS guarantees, whilst also respecting the severe operating constraints of wireless sensor networks. This balance is achieved as follows:

- *QMAC maximizes network lifetime* by 1) minimizing the energy used by each node to transmit its data, and 2) fairly distributing energy intensive operations across the homogeneous network,
- *QMAC is scalable for large numbers of network nodes* because 1) local repair is used to recover from node and communication failures, 2) nodes minimize buffered information and 3) communication slots are reused by nodes outside each others' interference range.
- *QMAC provides predictable delivery guarantees* because 1) scheduled communication is used to minimize communication errors from radio interference and to guarantee a communication path for each node, and 2) fault tolerant local repair is used to recover from node and communication faults.

Several researchers have already shown extensive solutions for achieving energy-efficient MAC protocols [12], [13], [14], [15], [16]. Their focus is to minimize sources of energy waste: idle listening, overhearing, collisions and protocol overhead. Either a slot-based approach [12] or TDMA-based approach [13], [14], [16] is used. This paper is distinguished from the previous ones for the initiative in delivering QoS within wireless sensor network constraints (i.e., energy and memory). The two protocols proposed in [17] and [18] that did address QoS issues in wireless sensor networks, were different from our approach as they used a two-radio architecture. Even though a direct comparison between the performance of QMAC

and these protocols has been performed, the different objectives between QMAC and these protocols makes this direct comparison difficult.

The rest of the paper is organized as follows. We discuss some related work in Section 2. We then describe our QMAC protocol approach in Section 3. Section 4 describes our experimental design including QoS metrics that are used to analyze energy efficiency, network performance, and fault tolerance scheme of QMAC. In Section 5, we show that QMAC achieves the above quality of service guarantees, under the given wireless network constraints through analytical studies and simulation results. We conclude with some discussion and present future work in Section 6.

2 Related Work

In recent work [17] and [18], QoS aware protocols are developed. Miller and Vaidya [17] propose a MAC protocol that addresses QoS in sensor networks in terms of energy efficiency and memory efficiency. Basically, the proposed protocol uses a second low power radio to allow senders to wake receivers if a specified number of packets are buffered. Therefore, a buffer overflow can be avoided. The protocol also determines an optimal period of the wakeup to minimize energy consumption. In [18], Yuan et al. use two different frequency bands for the intra-cluster communication and inter-cluster communication so that the cluster head can communicate with its members and other cluster heads at the same time. They propose a centralized off-line protocol to provide QoS provisioning. The basic idea of the protocol is to use the sink to adjust the modulation level (bits per symbol) of each cluster head based on the cluster head's feedback in order to make good trade-off between energy consumption and transmission quality. In each round, cluster heads in the network are required to send information about the shortest path to the sink, the number of members in their cluster, and the power density of noise. The sink uses this feedback to find the optimal modulation level and transmit power of each cluster head according to an optimization model. Cluster heads then adjust their modulation level accordingly. The major drawbacks of this protocol are the queuing process at cluster heads and the centralized decision making. The number of packets arriving at cluster heads is not controlled and so the packets will be dropped if the buffer overflows, resulting in latency and packet loss.

MAC protocols can be categorized into two groups: contention-based and scheduled-based protocols. The contention-based protocols use CSMA for node transmission in which a node

that intends to transmit listens to the medium and it can transmit if no other node is transmitting. The main advantage of contention-based protocols is that they allocate resources on-demand and so they can adapt to traffic fluctuations and changes in node density more easily. The main challenge with contention-based protocols is to reduce the energy consumption caused by collision (hidden-terminal problem), overhearing, and idle-listening [15]. Sensor MAC (SMAC) [12] is the best known example of MAC protocols designed for WSNs that is built on contention-based protocols. TDMA is the scheduled-based approach that has attracted attentions of sensor network researchers. In contrast to contention-based protocols, TDMA-based protocols separate nodes in time, which means that nodes own a time slot for transmitting and receiving a message without contending for medium access, and nodes do not interfere with each others transmissions. Thus, TDMA-based protocols guarantee a collision-free communication and reduce idle listening and this results in significant energy savings. The main challenges of this scheme are determining the collision free slots to be assigned to nodes in multiple hop networks [19], the overheads to set up and distribute a schedule through the network [15], and accurate time synchronization so that nodes' time slots do not overlap [20], [21], [22]. Examples of TDMA-based MAC protocols are the traffic-adaptive medium access protocol (TRAMA) [13], Lightweight Medium ACcess (LMAC) [14], and Self-Stabilizing TDMA (SSTDMA) [16].

SMAC uses the CSMA technique and an RTS/CTS handshake in an attempt to avoid collisions [12] and uses a fixed duty cycle consisting of a short listen period of 300ms and sleep period. SMAC puts a node into a sleep mode for a period of time and then wakes it to allow the node to listen to its neighboring nodes. Periodic sleeping scheme reduces idle listening and so results in energy savings. However, when all nodes wake up, any node that wants to send a message must contend for the channel. This scheme increases the probability of collision. Furthermore, a drawback of SMAC fixed listen period is that it may not be long enough to handle a high traffic load. This can increase latency because when a sender wants to transmit a packet, it must wait until the receiver wakes up. In SMAC, a node that has more data to send can monopolize the wireless radio channel. This is unfair for other nodes which have short packets to send but need to wait for the completion of transmission of the long packet. Thus, SMAC trades performance for energy efficiency.

TRAMA organizes time into frames and uses a distributed election scheme based on traffic information at each node to determine which node can transmit at a particular slot [13]. Nodes regularly exchange information about traffic flows routed through them and the identities of

their one-hop neighbors, so that each node knows the demands of its one-hop neighbors and identity of its two-hop neighbors. This neighborhood information is then used to determine a collision-free slot assignment. This is done through a distributed hash function that computes the winner (i.e., sender) of each slot based on node identities and the slot number. Nodes adaptively adjust their communication mode by listening to the current traffic information. If a node is not selected to transmit and is not the intended receiver of the current slot, it can switch to the sleep mode. A drawback of TRAMA is that a node may assume that some of its neighbors are transmitting when the transmission does not actually happen. This can increase energy consumption because nodes may be in the receive mode unnecessarily. TRAMA allows nodes to adjust their schedule according to traffic conditions. If a node has few packets to send, it may release its slot for the remainder of the frame to other nodes that have many packets to send. This scheme results in a high bandwidth utilization, however TRAMA does so at the expense of high latency and high algorithmic complexity [20].

LMAC divides time into slots with a fixed frame consisting of a control message and a data unit [14]. In principle, when an active node controls a time slot, it can broadcast a control message section of its frame and transmit the data afterwards. Neighboring nodes listen and check the control message. If they are the intended receivers of the packet, they turn on the radio during the data transmission part of the frame; otherwise they switch off their radio and wake up at the next time slot. This approach reduces power consumption since all nodes that are not involved in the communication remain asleep. LMAC is also able to adjust to topology changes in a network. A new node can join the network by listening to all control message sections of neighboring nodes and then randomly selecting a slot that is free. The main drawback of LMAC scheme is that it increases idle-listening overhead since nodes must always listen to the control sections of all slots in a frame, to allow nodes to receive data and to allow new nodes to join the network anytime.

SSTDMA is designed to operate on a regular grid topology such as rectangular, hexagonal, and triangular grids [16]. SSTDMA assumes a 2-band model for communication. The communication range is the distance to which a node can communicate with other nodes with a high probability. The interference range is the distance to which a node can communicate with other nodes with a low probability. The collision group of a node is other nodes that are in the interference range of the node. SSTDMA then assigns a time slot to each node in the network based on the collision-group information, for example a node labeled *A* and a node labeled *B* can use the same time slot as long as *A* is not within the collision-group of *B*. The TDMA algorithm

proposed in SSTDMA is self-stabilizing. It tolerates faults such as nodes that are improperly initialized, slots assigned to corrupted nodes, and nodes with clock drifts. SSTDMA uses a diffusing computation to re-validate the slots assigned to a node. It freezes a node from transmitting messages if the revalidation message is not received. When the revalidation message is received, the node can continue to transmit messages. The main drawback of SSTDMA is its constraint on the location of the nodes, which is impractical for many WSN applications.

QMAC differs from the schemes detailed above as it proposes a novel scheme for scheduling data gathering in WSNs by utilizing a TDMA-approach and adaptively adjusting node transmission power for a given topology. This adaptive transmission range scheme also gives the advantages of obstacle avoidance and fault tolerance [23]. Nodes can increase their transmission range when they are repelled by an obstacle. Nodes can also increase their transmission range if their current range cannot reach any neighbor due to low network densities. QMAC provides superior energy efficiency to LMAC because nodes are only active in their own time slots and so no energy is wasted in idle listening. Unlike SSTDMA, QMAC caters to any network topology. Furthermore, QMAC has a simpler algorithm to establish a schedule than does TRAMA. TRAMA builds a schedule when a node has data to send. This random scheduling scheme increases queuing delays. In contrast, QMAC reduces queuing delays as a schedule is assigned to nodes at the time of initial network setup and nodes maintain this schedule throughout their lifetime. In this way QMAC also avoids the need for a centralized scheduler. QMAC properties make it well suited for applications that are not highly delay sensitive but require energy efficiency and high delivery guarantees such as periodic data collection and monitoring applications.

3 QMAC

3.1 Protocol Overview

3.1.1 QMAC Phases

QMAC has two main phases: an initial network setup and a data gathering cycle. Figure 1 shows the order of steps executed in a network. Some steps within phases are carried out in parallel by the nodes, which are the neighbor schedule exchange and the fault tolerant schedule exchange (depicted in dashed lines).

The QMAC initial network setup consists of two main steps: the data gathering tree con-



Figure 2: A node's transmission schedule structure. The slots are not contiguous in time.

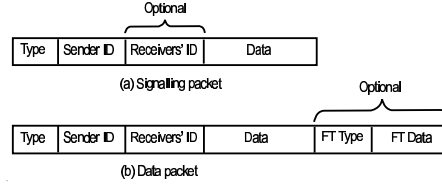


Figure 3: QMAC (a) signaling and (b) data packet structure.

of fault tolerant listening slot (FTS), receive slots list (RSL), and transmit slots list (TSL), as depicted in Figure 2. Note that the schedule only represents a list of slots when a node should be active and so the slots are not contiguous in time. In the node deployment stage, every node in a network is already assigned the same FTS. FTS is used to allow nodes to listen to the environment activities in order to allow other nodes to join the data gathering tree if they are able to get a parent that is attached to the tree. Furthermore, the duration of FTS is longer than the data packet routing and multi-function period in order to allow nodes to respond and self-configure themselves to network dynamics in their neighborhood. During the data packet routing period, nodes go through their RSL and TSL. Nodes switch to the receive mode if they expect to receive data in a slot and switch to the transmit mode if they have to send or to forward data. Nodes switch to the sleep mode if they are not scheduled to receive or transmit data. Finally, in the multi-function slot (MFS) a local synchronization is executed. The MFS in a node's TSL means that the node becomes the sender of a synchronization packet in that slot, whilst the receivers of the packet record the MFS in their RSL. Furthermore, a local repair can be executed concurrently with the time synchronization by piggybacking a fault tolerant data on the synchronization packet.

3.1.2 QMAC Packet Structure

There are two types of packets used in QMAC: signaling and data packets. Signaling packets are used in the initial network setup and the fault tolerant listening period. In the data routing period and the multi-function period, a data packet is sent in each time slot. Figure 3(a) shows the structure of a signaling packet. A signaling packet carries information about the type of signaling, the sender of the signal, the list of intended receivers of the packet (optional), and the signaling data. On the other hand, the data packet as shown in Figure 3(b) contains the type of data (sensing data or time synchronization data), the sender of the packet, the list of intended

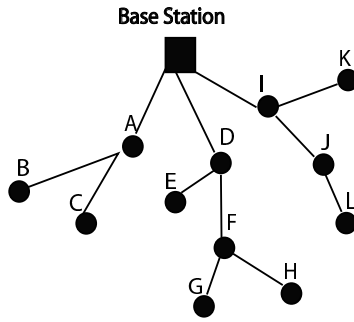


Figure 4: A data gathering tree.

receivers of the packet, the data, and a reserved space for piggybacking a fault tolerant data if exists.

3.1.3 Terminology

In this paper, the QMAC protocol will be described using the following terminology (illustrated in Figure 4):

- *Parent node* is an immediate node that forwards a source node's packet and which is closer to the base station (in terms of hop-count) compared to the node, for example *A* is the parent of both *B* and *C*.
- *Ancestor node* is a router that is more than one-hop away from a source node to which the source node's packet is forwarded to and which is closer to the base station (in terms of hop-count), for example *D* is the ancestor of *G* and *H*.
- *Child node* is an immediate node from which a node receives a packet and which is further from the base station (in terms of hop-count) compared to the node, for example *E* and *F* are children of *D*.
- *Descendant node* is a node that is more than one-hop away from a node and which is further from the base station (in terms of hop-count) compared to the node, for example *G* and *H* are descendants of *D*.
- *Sub-tree* is a branch of a tree that consists of a parent and one or many children, for example the base station and node *A*, *D*, and *I* make a sub-tree; Node *J* and *L* also make a sub-tree.

Symbol	Description
NID	Node ID
PID	Parent ID
TL	Tree level
CL	List of children
DL	List of descendants
NT	Neighborhood table
FTS	Common fault tolerance listening slot
RSL	List of receive slots
TSL	List of Transmit slots
CSL	List of conflict slots
MFS	Multi-function slot
GHS	Global highest slot
TR_parent	Transmission radius to reach the parent
TR_children	Transmission radius to reach the children

Table 1: Node lookup table.

- *Tree level* is the number of hops required by a node to reach the base station. For example, the base station has the lowest tree level, while node *G*, *H*, and *L* have the highest tree level as they are three hops away from the base station.

3.1.4 QMAC Lookup Table

Nodes in QMAC maintain a lookup table that contains scheduling information including when a node should listen to neighboring nodes, when a node should expect data, and when a node can transmit data (see Table 1). Receive slots of a node overlap with transmit slots of the node's children. The conflict slot list records slots that are used by a node's first-level and second-level neighbors. A first-level (direct) neighbor of the sensor node is any node within its communication range. Furthermore, any node that is within the communication range of the node's first-level neighbor(s) is referred to as a second-level neighbor. The global highest slot field contains the highest slot number known to a node. The value is always updated whenever a node is informed of a slot that is higher than any existing slot in its TSL, RSL, CSL, and GHS. The destination of data maybe the parent or children of a node and the transmission power required to send data is measured based on the strength of the signal received by the node from its parent or its children respectively. Information in the lookup table is gathered from two main phases: the data gathering tree construction and the time slot assignment.

3.2 Data Gathering Tree Construction

The fundamental property of QMAC data gathering tree construction is that nodes in a network adaptively adjust the power required to send data to their parent. Nodes minimize energy use by selecting the closest node within a default transmission radius of their parent and using the minimum transmission power needed to reach their parent. Shorter links are also chosen because they can prevent cross-path-connectivity and optimize time slot reuse. This reduces interference and improves scalability. A data gathering tree is constructed in three main phases: parent selection, network coverage optimization, and neighborhood data collection. The nodes use CSMA during these phases.

3.2.1 Parent Selection

The parent selection phase is performed within a predefined period of time. The base station acts as the root of a data gathering tree. The base station generates a connectivity token and initiates the data gathering tree construction using a simple flooding scheme. The connectivity token is distributed using a top-down sub-tree by sub-tree approach. The rule is that the connectivity token is passed from a node to the child with the smallest NID in the current sub-tree. When a node obtains the connectivity token, it is called as a CT_holder and it can broadcast a tree construction signal to find its prospective children. A node can only become the CT_holder once and so when it receives the connectivity token again, the node know that it needs to pass the token to another node. Furthermore, during this token passing, nodes update their children and descendants list.

The base station initiates the data gathering tree construction phase by broadcasting a tree construction signal (TC_signal) with the transmission power required to achieve the default transmission radius specified by the human manager in TR_children field. At this stage, the CT_holder is the base station. The TC_signal carries information about the CT_holder's ID and tree level. Nodes that are able to receive the TC_signal become the children of the CT_holder by sending a confirmation signal containing their NID to the CT_holder through the contention-based channel access. In the meantime, the parent records the strongest confirmation signal it receives from its children to be its TR_children. The children update their lookup table by retrieving the CT_holder's ID as the PID and assigning a tree level by incrementing the CT_holder's tree level. The children also determine the value of TR_parent from the strength of the TC_signal they receive. We assume that the signal strength corresponds to the distance

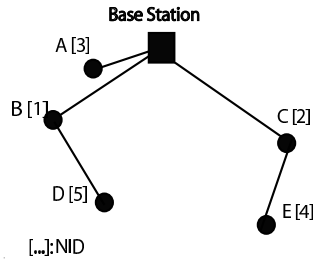


Figure 5: A tree built for gathering data in the network of five nodes.

between two nodes and the nodes can adjust the transmission power to control the transmission range.

If a CT_holder does not receive a confirmation signal from any node within a period of time, it increases its transmission power until the number of desired children is found (user-defined parameter). This scheme works under a condition that the transmission power is less or equal to the node's maximum power. The advantage of this approach is that the human manager has a degree of control over the breadth of a data gathering tree by specifying the desirable number of children a node should have.

The current CT_holder can release the connectivity token if one of the following conditions is met:

- the current CT_holder has found its children, or
- the current CT_holder is unable to find its children after using the maximum power.

The current CT_holder releases the token by passing the token to the other node with the lowest NID in the current sub-tree through the parent (the root of the tree). The parent then passes the token to the next child with the lowest ID. The winning child then becomes the CT_holder and executes the same procedure as described above. After all nodes in the current sub-tree have held and released the token i.e., all leaves of the sub-tree are reached and have become a CT_holder, the token is sent up to the root of the sub-tree and then the token is passed down again to the child with the lowest ID. Since the child has become the CT_holder previously, the child will act as the root of the new sub-tree. For example, Figure 5 shows that NID of first tree level nodes: A, B, and C is 3, 1, and 2 respectively. Furthermore, NID of second tree level nodes: D and E is 5 and 4 respectively. The connectivity token is passed in the following order: BS - B; B - BS; BS - C; C - BS; BS - A; A - BS; BS - B; B - D; D - B; B - BS; BS - C; C - E; E - C; C - BS. Thus, the order of CT_holder is: BS - B - C - A - D - E.

Nodes in the network may receive more than one TC_signal. Receivers re-evaluate their parent against a new prospective parent based on the signal strength of the TC_signal, tree level, and whether the new prospective parent has a connectivity path to the base station. The TC_holder that sends the strongest signal with a higher tree level than the receivers' tree level will be elected as the receivers' parent. This scheme ensures that nodes get the strongest link path to the base station, i.e., the shortest possible path. This phase ends when the token comes back to the base station finally, i.e., after all the children of the base station and their descendants have selected their parents.

3.2.2 Network Coverage Optimization

The network coverage optimization is the phase in which QMAC identifies nodes that are out of default transmission range of nodes in the parent selection phase. A node that is unable to get a TC_signal during the parent selection phase (called as an orphan node) will send a distress signal at the start of the network coverage optimization phase. Recall that nodes use CSMA/CA during this phase and so the orphan node backs off for a random period of time if it overhears another distress signal; if not, it gradually increases its radio power (until the maximum range) until it finds at least one neighboring node. Neighboring nodes that receive this signal send a reply and confirm their availability to accept the distressed node as their child. The distressed node then selects the nearest neighbor to become its parent. Note that the duration of this phase is fixed arbitrarily, which means that a sufficient length of time is allocated for orphan nodes to find a parent.

3.2.3 Neighborhood Data Collection

After the period of the network coverage optimization phase expires, nodes in the network can start to execute their neighborhood data collection. Nodes take turn in collecting their neighborhood information. The neighborhood data collection starts from the node with the smallest NID to the node with the highest NID in the network. Let T_{nco} be the expiry time of the network coverage optimization phase and t_{ndc} be the fixed period of neighborhood data collection. Nodes know the earliest time (T_{ndc}) for them to find their neighbors based on the value of NID, T_{nco} , and t_{ndc} that is formulated as follows:

$$T_{ndc} = T_{nco} + \text{NID} \cdot t_{ndc} \quad (1)$$

The period of the neighborhood data collection is designed to be long enough for a node to build its neighborhood information table. For example, a node with NID equal to one will be the first node to find its direct neighbors. Hence, in the network of N nodes, the node with NID of N will be the last node to perform the neighborhood data collection. Note that different nodes may enter this phase at different times (according to their local clocks) and there is no harm in that, i.e., all nodes do not enter this phase at the same time. In the worst case, if a node still overhears activities when it is about to start its neighborhood data collection, it will wait for a random period of time before it starts its neighborhood data collection period. The transmission power used by a node to find its neighbors is based on the highest value between the node's TR_parent and TR_children. Nodes that are within the communication range of the node will be recorded as the node's direct neighbors in its neighborhood table. Thus, this phase collects only direct neighbors (first-level neighbors) information.

3.3 Time Slot Assignment

QMAC uses a depth-first-search (DFS) schedule to achieve memory efficiency. The DFS schedule allows data sent by a source node to be forwarded by routers to the base station in an interleaving manner, eliminating the need of buffering. This enhances reliability by preventing the loss of packets due to buffer overflow. After a data gathering tree is constructed, the local topology is known to nodes in the network such that each node knows its parent and children (if any). The base station then generates a time slot assignment token (TSA_token) and initiates the time slot assignment phase. The TSA_token passing is done using the DFS technique. The token is passed to the child that has the lowest NID. As illustrated in Figure 5, node A , B , and C are the children of the base station (BS). Since B has the lowest NID, the token will be passed from BS to B . Afterwards, B will pass the token to its child that has the lowest NID, which is D . This token passing mechanism is replicated until all nodes in the network receive the token and the token is then returned to the base station permanently.

In QMAC, the slot selection phase always starts from slot number 2 because slot number 1 is dedicated for FTS (fault-tolerant slot). A node can claim a slot if the slot is not listed in the RSL (list of receive slots), TSL (list of transmit slots), and CSL (list of conflict slots) fields of the node's lookup table. Therefore, in selecting a slot, there is no need to know the true value of GHS. The GHS in a node's lookup table indicates the global highest slot number from that node's point of view. A node's RSL, TSL, and CSL fields are gathered and updated during the data transmission slot assignment, multi-function slot assignment, and neighbor schedule

exchange. Furthermore, in the QMAC time slot assignment phase, a node's GHS value is updated based on the current highest slot number of all node's RSL, TSL, and CSL entries, incremented by one. Every time a node claims a slot, the node waits for a period of time (user-parameter), t_{wait} , to allow the neighbor schedule exchange phase to be executed. The main idea behind the neighbor schedule exchange is to propagate a slot claimed by a node to the node's first-level and second-level neighbors. These neighbors list the informed slot in their CSL in order to prevent them from claiming the slot due to possible transmission interferences and hence collision-free traffic can be guaranteed. This slot selection scheme allows a slot to be reused by many nodes as long as nodes' transmissions are not within interference range of each other. In this way, the spatial slot re-use can be optimized. Next, we will describe the algorithm of the time slot assignment step. The algorithm will be illustrated through an example in Section 3.3.4.

3.3.1 Data Transmission Slot Assignment

Data transmission slots are the slots used in data packet routing. A data transmission slot is a transmit slot in which a node's parent is the intended receiver of the data sent in that transmission. When a node obtains the TSA_token, it declares itself as the TSA_holder. The TSA_holder allocates a transmit slot to send its own data by choosing the lowest slot number available i.e., one more than the current known GHS. Recall that a node selects a slot that is not listed in its RSL, TSL, and CSL fields. Once the TSA_holder selects a slot, it checks whether the slot number is higher than all entries in its RSL, TSL, and CSL fields. If that is the case, the TSA_holder will update its GHS value with the selected slot number. The TSA_holder then informs the claimed slot to its parent (one of its first-level neighbors) by broadcasting a *forward_slot_request* as well as piggybacking an *inform_slot_request*. In this way, slots to forward the TSA_holder's data to the base station are allocated. Furthermore, the *inform_slot_request* on the broadcast packet triggers the neighbor schedule exchange phase that will be described in Section 3.3.3. The *forward_slot_request* contains the claimed slot information: the slot number, the receiver's ID (parent ID), and the neighbor level counter. Nodes (TSA_holder's first-level neighbors) that receive the *forward_slot_request* checks the intended receiver of that packet. If a node is the intended receiver of the packet, it knows that it is the parent of the sender (TSA_holder); if not it will execute the neighbor schedule exchange only. In parallel to the current execution of the neighbor schedule exchange phase by the rest of the TSA_holder's first-level neighbors, the parent of the TSA_holder lists the received slot number in its RSL.

The parent then assigns the corresponding transmit slot, i.e., a slot to forward the TSA_holder's data. The corresponding transmit slot must be bigger than the received slot number and is not listed in the parent's RSL, TSL, and CSL. The parent then waits for a period of time, t_{wait} , before sending the *forward_slot_request* to the TSA_holder's ancestor (i.e., routers). The data transmission slot assignment as explained before is repeated until the base station receives the *forward_slot_request*.

3.3.2 Multi-function Slot Assignment

The multi-function slot is a slot in which a node's children are intended receivers of data sent in the transmission. From the sender's point of view, the multi-function slot (MFS) is regarded as the reserved transmit slot; whilst from the receiver's point of view, it is regarded as the reserved receive slot. The multi-function slot assignment is executed through the TSA_token passing. The basic idea is that if during the data transmission slot assignment phase a node has already assigned slots for forwarding data for all of its children and descendants, the node will hold the TSA_token again before passing it to other node. The node then selects a MFS and the claimed MFS must be bigger than any transmit slots in the node's TSL and must not be listed in the node's CSL. Again, the node updates its GHS if the claimed MFS slot is higher than all entries in its RSL, TSL, and CSL fields. The node then informs the claimed MFS to all of its children by sending an *inform_slot_request* packet consisting of the slot number, the receivers' ID (children IDs), and the neighbor level counter. The children then assign the MFS as the reserved receive slot in their RSL. Furthermore, the neighbor schedule exchange is also performed.

3.3.3 Neighbor Schedule Exchange

In the previous section we noted that the neighbor schedule exchange phase is used to propagate a node's claimed slot to its first and second-level neighbors. This is achieved through CSMA/CA communication in which a node broadcasts its claimed slot to its direct neighbors (first-level neighbors) and then each of these direct neighbors propagate the slot information again to their direct neighbors (second-level neighbors). Thus, the first-level and second-level neighbors are not necessarily along the tree. For example, in Figure 4, node *F*'s neighbor schedule exchange phase do not only cover one-hop and two-hop neighbors: *BS*, *D*, *E*, *G*, and *H* but also other nodes that may interfere with its transmission: *A*, *I*, and *J*.

If during the data transmission slot assignment or the multi-function slot assignment phase a

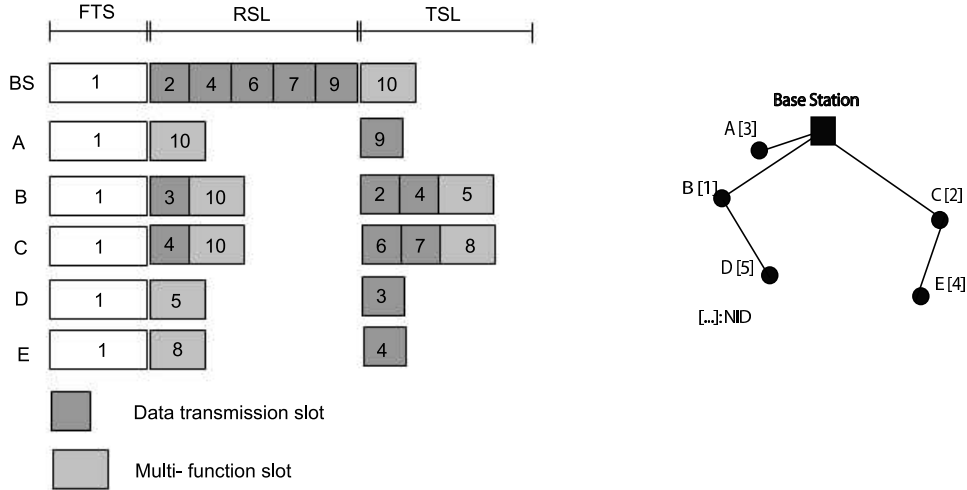


Figure 6: Nodes' schedule.

node receives the *inform_slot_request* in a packet, it means that the neighbor schedule exchange phase needs to be executed. The neighbor slot exchange is controlled by the neighbor level counter information in the packet, which indicates the status of a neighbor, whether it is a first-level neighbor (the neighbor level counter is equal to one) or a second-level neighbor (the neighbor level counter is equal to two). If a node receives a packet with the level counter less or equal to two, it will include the slot in its CSL. Furthermore, if the current level counter is less than two, the node increments the level counter value and broadcasts the packet to its direct neighbors; if not it stops the neighbor schedule exchange. Referring to the previous example, node *F*'s first level neighbors are *D*, *E*, *G*, and *H*; node *F*'s second-level neighbors are *BS*, *A*, *I*, and *J* (note that node *L* is not within the transmission range of *F*'s first level neighbors).

3.3.4 Example

We now illustrate the time slot assignment algorithms described previously by using the data gathering tree shown in Figure 6. Figure 6 also presents the schedule list of nodes in that data gathering tree. The base station will pass the TSA token to node *B* first and *B* becomes the TSA holder. At this stage, no node has claimed slot number 2, so *B* claims slot number 2 as its transmit slot and records this slot in its TSL. *B* then broadcasts a signaling packet and waits for a period of time (t_{wait}) to allow the neighbor schedule exchange phase to be executed before it passes the TSA token to its child with the lowest NID, which is node *D*. *B*'s

signalling packet contains the *forward_slot_request* and *inform_slot_request* with the following information: the sender is B , the base station is the intended receiver, the neighbor level counter is equal to one, and the data is slot number 2. Direct neighbors of B , which are node A , D , and BS overhear this transmission and receive the packet. Since BS is the intended receiver, it now knows that slot number 2 is used by B to send data to it and so BS records slot number 2 in its RSL. The *forward_slot_request* is stopped at BS because it is the base station. Thus, node B 's data transmission slot assignment phase ends. In the meantime, A , D , and BS also process the *inform_slot_request* by retrieving the neighbor level counter value of the packet. The packet's neighbor level counter is equal to one and hence they all record slot number 2 in their CSL. Afterwards, they increment the neighbor level counter and broadcast the *inform_slot_request* packet with the updated neighbor level counter to their direct neighbors through CSMA/CA scheme. Then, direct neighbors of A , D , and BS also record slot number 2 in their CSL, i.e., node C now know that it is unable to claim slot number 2. Furthermore, the neighbor schedule exchange phase stops at these nodes because the received neighbor level counter is already equal to two. Therefore, at the end of the neighbor schedule exchange, first and second-level neighbors of B are informed of its claimed slot; they are node BS , A , C , and D . After t_{wait} expires, B passes the TSA token to D .

When node D becomes the TSA_holder, it checks TSL, RSL, and CSL entries in its lookup table. Since slot number 2 is listed in its CSL (as the result from B 's neighbor schedule exchange described above), it claims the next lowest available slot that is slot number 3. Since D does not have a child, it skips the multi-function slot assignment. D then broadcasts the *forward_slot_request* as well as piggybacks the *inform_slot_request* with B as the intended receiver. The *forward_slot_request* and *inform_slot_request* will be propagated up to the base station. As a result, B selects the next available slot to forward D 's data, which is slot number 4. After D 's data transmission slot assignment finishes, B holds the TSA token again because it has assigned data transmission slots for all of its descendants. B then claims slot number 5 as its MFS and broadcasts the *inform_slot_request* packet to its neighbors. Since the child of B is the intended receiver of the packet, D records slot number 5 as its reserved received slot in its RSL and executes the neighbor schedule exchange. Meanwhile other neighbors of B that receive this packet also execute the neighbor schedule exchange phase. After t_{wait} for the neighbor schedule exchange phase expires, B sends the token back to the base station and the base station passes it down to node C . This TSA token passing scheme is used in the entire network. Note that node E can reuse slot number 4 because this slot does not appear in E 's CSL.

3.4 Time Synchronization Scheme

Time synchronization is critical in TDMA-based MAC protocols because nodes that are involved in a scheduled communication must wake up at the same time to exchange information. According to Miller and Vaidya [17], the times required for a MICA2 Mote radio to transit from the sleep state to idle ($T_{\text{tran_on}}$) and from the idle state to sleep ($T_{\text{tran_off}}$) are 2.45 ms and 0.25 ms respectively; these switching times are non-negligible. QMAC addresses the issue of getting nodes who are involved in a communication to wake up at the same time (i.e., nodes have overlapped active periods) as outlined in [24] by adjusting the time slot length (T_{slot}) to include $T_{\text{tran_on}}$ and $T_{\text{tran_off}}$, which is as follows:

$$T_{\text{slot}} \geq T_{\text{PacketTransfer}} + T_{\text{tran_on}} + T_{\text{tran_off}} \quad (2)$$

$T_{\text{PacketTransfer}}$ is the time required to send a datum. For MICA2 motes, $T_{\text{PacketTransfer}}$ for a maximum data size of 56 bytes with data radio rate of 19.2 kbps, is equal to 23.3 ms as used by [11]. Based on equation 2, the QMAC slot length is 26 ms. The biggest advantage of this scheme is the fact that when the sender wants to transmit it is guaranteed that the intended receiver(s) are awake and listening.

In QMAC, a hierarchical structure is constructed in the data gathering tree construction phase. Each node in the tree knows its parent, its children and its level in a tree. QMAC performs time synchronization locally, with each parent synchronizing its children. In the multi-function slot of a node, the node broadcasts its clock and the current global highest slot (GHS) number known to that node to the children. The children then synchronize their clock to the parent's clock and also update their GHS field if the received GHS value is greater than their existing GHS value. This time synchronization scheme is desirable as children only need to have the same clocks as their parent to ensure that a parent is in the receive mode when a child sends data to it and vice versa. QMAC local synchronization is simple but effective because clock drift is minimized by synchronizing nodes during each data gathering cycle as well as guaranteeing nodes that are involved in a synchronization are awake at the same time. Furthermore, it incurs low overheads since the synchronization message is piggybacked to the multi-function slot's packet. In addition, the value of global highest slot can be propagated to all nodes in a network through the local synchronization scheme. In this way, local synchronization and GHS update are achieved almost for free.

3.5 Fault Tolerance Scheme

Wireless sensor networks are prone to network dynamics such as dropped packets, nodes dying, being disconnected, powering on or off, and new nodes joining the network and so the networks need to be able to self-configure without knowing anything of the network topology in advance [25], [26], [27]. In QMAC, nodes listen to environment activities during the common fault tolerance listening slot. If there is a node that sends a distress signal during this slot, nodes in the network that receive this signal will reply to it and perform a local self repair.

3.5.1 Transmission Error

In QMAC, collision-free traffic can be guaranteed and so we assume that the transmission is perfect. In a real world implementation, there would be environmental factors such as noise, wind, and humidity that could affect a transmission. The fault tolerance action to manage this type of transmission error is beyond the scope of this paper, but there are few statements that can be made at this point. Depending on the application, a different recovery mechanism may be employed to tailor to the application's individual needs. One solution is to allow nodes to retransmit data if the data is lost or corrupted. Say if a node does not receive a packet when it expects data from its child, it assumes that the expected packet is lost. It then requests a data retransmission from the child during the multi-function slot. The child may append the requested data onto the data packet that will be sent in the immediate data transmission slot in the next data gathering cycle. Alternatively, the child may just send the average value. The last method minimizes the transmission cost and improves the transmission reliability. Furthermore, if a node receives the data retransmission request repeatedly, it may decide that the current link connectivity is too noisy and so it finds a new parent in the fault tolerant listening slot.

3.5.2 New Node

During the fault tolerant listening slot, a new node may send a fault tolerant signal (FT_signal). Nodes that are within the communication range of the new node become neighbors and prospective parents of the new node. The new node then selects one of them as its parent, based on the signal strength and the tree level. A prospective parent that sends the strongest signal and has the lowest tree level becomes the parent of the new node. The new node is then assigned the fault tolerant listening slot, data transmission slot, and multi-function slot. The selected parent allocates two slots to the new node; one for the new node's data transmission and one for the

new node's reserved receive slot. The new node gets the parent's GHS value incremented by one as its data transmission slot. Furthermore, the new node gets the parent's reserved transmit slot as its reserved receive slot. If the parent does not have a reserved transmit slot because it does not have a child before, it will assign a MFS to itself first, which is equal to its GHS incremented by two before allocating its MFS to the new node.

The data transmission slot assignment for a new node is performed using the *forward_slot_request*. The procedure of the *forward_slot_request* is similar to the time slot assignment phase except that slots can be claimed after they have been approved by a node's parent. A node that wants to propose a transmit slot to the parent is called as the *slot_requestor*. The proposed slot is piggybacked on the data packet sent in the immediate data transmission slot in order to get a slot approval and to allocate receive and forward slots for the *slot_requestor*'s data. The proposed slot will be checked against RSL, TSL, and CSL of the *slot_requestor*'s parent. If the proposed slot exists in the parent's table, a next available slot will be chosen to replace it. Afterwards, the approved slot will be listed in the parent node's RSL and a corresponding slot to forward the data is proposed to the *slot_requestor*'s ancestor (i.e., *slot_requestor*'s routers). In the meantime, the approved slot is sent to the *slot_requestor* in the multi-function slot. The *slot_requestor* then claims the approved slot and lists the slot in its TSL. This scheme is executed by traversing the path to the base station, so that data transmission slots for the new node are allocated. Note that every time a node claims a new slot, it triggers the fault tolerance schedule exchange, which is described in Section 3.5.4.

3.5.3 Dead Nodes

The term 'dead node' refers to one of the following conditions: a node whose battery is weak or dead, or a node that is unreachable or unable to communicate due to environmental factors such as fog, smoke, and obstacles [23]. If a node does not receive the expected data during all scheduled receive slots of a child after two data gathering cycles, it assumes that the child is dead. It then removes the child from its children list. It also performs a slot garbage collection in which all slots that are associated to that child are deleted from its RSL, TSL, and CSL and so these slots are not wasted on idle listening. On the other hand, if a node does not receive data when it is supposed to forward one of its descendants' data; it assumes that the descendant is dead. For example, when node *D* in Figure 5 fails to send data to its parent, node *B*, *B* do not forward *D*'s data to *BS*. If *BS* and *B* do not receive *D*'s data after two data gathering cycles, they can assume that *D* is dead. Furthermore, *BS* and *B* also executes a slot garbage collection by

removing the wasted slots from their RSL, TSL, and CSL.

A node becomes an orphan if it does not receive a synchronization message in its MFS slot from its parent after two data gathering cycles. The orphan node then sends the FT_signal in the fault tolerant listening period of the next data gathering cycle to find a new parent. During the fault tolerant listening period, nodes that are within the communication range of the orphan node become neighbors of the orphan node and will be considered as prospective parents if they meet the following criteria:

- a prospective parent's tree level is lower or equal to the orphan node's tree level in order to prevent descendants of the orphan node to become the parent, and
- a prospective parent's multi-function slot is not listed in the orphan node's RSL in order to avoid collisions with the orphan node's existing schedule.

The orphan node then selects a prospective parent that has sent the strongest reply signal as its new parent. The data transmission slot assignment for the orphan node follows the same procedures as the new node's data transmission slot assignment except that there may be many proposed slots. This is because the nodes that belong to the same sub-tree rooted at the orphan node are children and descendants of the orphan node. Therefore, the orphan node can propose all slots in its current TSL to the new parent. This method allows the orphan node to maintain its RSL so that children and descendants of the orphan node do not need to rebuild their schedules. The new parent sends approved slots and its MFS to the orphan node. If the parent does not have a reserved transmit slot, it will assign a MFS to itself first, which is equal to its current GHS incremented by one. The orphan node then claims the approved slots and assigns the new parent's MFS to replace its previous reserved receive slot. Furthermore, the approved slots are propagated up by the new parent to the base station.

3.5.4 Fault Tolerant Schedule Exchange

The purpose of the QMAC fault tolerant schedule exchange is to inform a node's claimed slot to other nodes that may interfere with its transmissions. In contrast to the neighbor schedule exchange phase described in Section 3.3.3 that uses CSMA/CA for communication, the fault tolerant schedule exchange phase uses the existing TDMA schedule: data transmission slots and multi-function slots. Thus, the term "neighbor level counter" used in this section does not correspond to the actual neighbor level.

Every time a node claims one or more transmit slots, the fault tolerant neighbor schedule exchange phase is triggered. A list of claimed slots is called the *inform_slots* and the slot information (slot IDs and the neighbor level counter) is piggybacked as the fault tolerant data (FT_data) on the node's data packet. The packet is sent in the node's immediate data transmission slot. The node that receives the *inform_slots* in the current data packet will list all slots in the *inform_slots* in their CSL. The node then retrieves the neighbor level counter value. If the neighbor level counter is less or equal to three, the node increments the neighbor level counter. Afterwards, the node updates the *inform_slots* with the new neighbor level counter value. The node then forwards the *inform_slots* to its parent in the immediate data transmission slot to the parent and also broadcasts the *inform_slots* to its children in its multi-function slot. This fault tolerant schedule exchange stops at nodes that receive the *inform_slots* with the neighbor level counter greater than three. This fault tolerance schedule exchange scheme ensures that first-level, second-level, and some of third-level neighbors of a node are informed of a new claimed transmit slot. Since the scheme utilizes existing TDMA schedule, a collision-free schedule exchange is guaranteed. Furthermore, there is no need to designate extra slots to do the fault tolerant schedule exchange. Thus, fault tolerance can be achieved almost cost-free.

The fault tolerant scheme described above shows that the multi-function slot can be used to: verify the connectivity between parents and their children; to send an acknowledgment of the *forward_slot_request* by piggybacking approved slot/s to a synchronization packet; and to trigger the fault tolerance schedule exchange by piggybacking the *inform_slots* to a synchronization packet.

4 Experimental Design

We have developed a C# simulator to model the behavior of QMAC data gathering sensor networks. This section presents the experimental and analytical assumptions of our simulations. All of the simulation result graphs are based on the average value of ten different network topologies involving 50, 75, and 100 nodes located randomly in a network dimension of 300 m x 300 m. The error bars shown in the graphs are the standard deviation of the mean. We excluded the error bars in several graphs to improve readability of the graphs. The location of the base station was fixed at the top-center of the network map. The base station was programmed to find at least three children in order to spread energy dissipation among nodes. A default transmission radius for data gathering tree construction was defined before simulating a WSN.

Tx Power (dBm)	Transmission Range (m)	Current Consumption (mA)
-20	5	8.6
-10	18	10.1
0	50	16.8
5	68	25.4

Table 2: Energy consumption models.

Transmission radii stored in nodes' lookup table were within the maximum transmission radius and varied depending on the distance between nodes and their parent as well as the distance between nodes and their children.

When applicable, the values of our simulation parameters were based on Mica2 Mote hardware [28] and empirical data [11], [17], [29]. In simulations, we assumed that communication between nodes follows the 2-band behavior observed in experiments [11] in which nodes within communication range receive each others messages with high probability, nodes in the outer band receive messages with low probability, and outside the outer band, there is no reception, and no interference between nodes. Furthermore, it has been shown [11] that the maximum transmission radius of Berkeley MICA2 motes under normal weather condition is 70 meters (m). Based on that, the maximum transmission radius allowed in our experiments was 70 m. However, the same research also showed that sensors' transmission radius can fall to 10 m in bad weather conditions. Moreover, field trials of sensor network for environmental monitoring of soil moisture in [9] showed that nodes may lose connectivity for extended periods. QMAC addresses this issue by allowing nodes to adjust their default transmission radius to find a new parent if the existing connectivity link is weak.

4.1 Energy Dissipation Models

In our simulations, we used the empirical energy models based on MICA2 Motes discussed in [11], [17], and [29]. Table 2 presents MICA2 platform transmission ranges and current drawn measured with a 3V power supply, which were obtained by Anastasi et al. in [11]. We then used the Lagrange Interpolating Polynomial [30] based on the known four points to find current consumptions for other transmission ranges. We set MICA2 Mote's listen/idle energy cost to be equal to the receive energy cost, which was 30 mW [17], [28]. The energy consumed for a MICA2 Mote radio to transit from the sleep state to idle ($P_{\text{tran_on}}$) and from the idle state to sleep ($P_{\text{tran_off}}$) is equivalent to the idle listening energy consumption (P_{idle}), which was set to 30 mW [17]. The sleep state energy cost was set to 0.003 mW (P_{sleep}) [17]. The time required

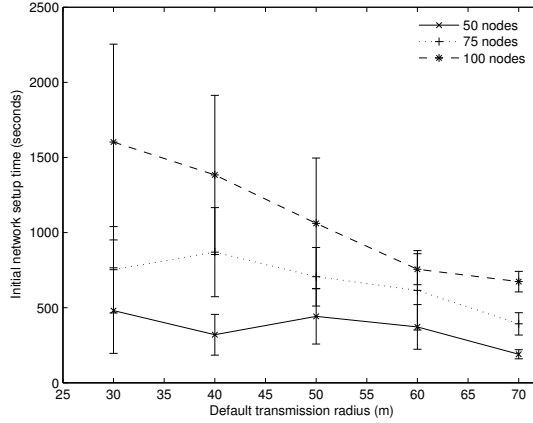


Figure 7: Average initial network setup time versus default transmission radius.

for a MICA2 Mote radio to transit from the sleep state to idle ($T_{\text{tran_on}}$) and from the idle state to sleep ($T_{\text{tran_off}}$) were set to 2.45 ms and 0.25 ms, respectively. We set the length of QMAC slot (T_{slot}) and fault tolerance slot (FTS) to be 26 ms and 1000 ms, respectively.

Although the switching energy cost is non-negligible [29], using the QMAC TDMA approach is still better than not putting nodes into the sleep state if they are inactive. For example, suppose a node is inactive for three slots. By putting the node into the sleep state during these three slots, the amount of energy spent is: $(T_{\text{tran_on}} * P_{\text{tran_on}}) + (T_{\text{tran_off}} * P_{\text{tran_off}}) + (3 * T_{\text{slot}} * P_{\text{sleep}}) = 0.08 \text{ mJ}$. This value is smaller than keeping the node idle that consumes: $(3 * T_{\text{slot}} * P_{\text{idle}}) = 2.34 \text{ mJ}$.

We performed simulations to observe the minimum period required for the initial network setup. Figure 7 shows the average of initial network setup time required by nodes in the networks to build a data gathering tree, to collect their neighborhood information, and to build a schedule as well as informing the schedule to their first-level and second-level neighbors. Based on the upper bounds of standard deviation in Figure 7, we derived conservative initial network setup periods for different network densities. The initial network setup time for a network of 50, 75, and 100 nodes was 1200, 1800, and 2700 seconds respectively. Furthermore, the initial energy for each node was 15,390 J [31]. For simplicity, we excluded energy consumptions for computation and sleeping in our simulations because their values are negligible compared to transmit, receive, listen, and switch energy costs.

4.2 Quality of Service Metrics

Three types of metrics that we consider important for measuring the QoS of sensor network operations are: energy efficiency, network performance, and fault tolerance.

Energy efficiency

- *Percentage of setup cost* is the rate of total energy consumption for setting up a data gathering tree and a network schedule to the total initial energy assigned to nodes, averaged over the entire network.
- *Node Energy Dissipation per Data Gathering Cycle* is the rate of total energy consumed for listening, switching, transmitting, and receiving during a data gathering cycle, averaged over the entire network.
- *Energy cost per datum* is the energy consumed in transmitting and receiving a datum from a source node to the base station, averaged over the entire network.
- *Node lifetime* measures the lifetime of nodes in terms of data gathering cycle, averaged over the entire network.
- *Percentage sleep time* is the rate of total sleeping time to the total network active time, averaged over the entire network.

Network performance

- *Network connectivity* is used to measure the number of nodes that are connected by multi-hop to the base station, i.e., connected to a sensor network data-gathering tree.
- *End-to-end latency* is the number of hops required to transmit data from source nodes in a network to the base station, averaged over the entire network.
- *Network throughput* is the number of data packets received at the base station to the total number of packets sent, averaged over the entire network.
- *Collision rate* is the number of transmission errors due to collisions to the total number of transmissions, averaged over the entire network.
- *Percentage of slot reuse* is the rate of total number of slot reuse to the total number of slots used in a network, averaged over the entire network.

Fault Tolerance

- *Local repair* that measures the average latency for new nodes to get a parent and measures the network connectivity reestablishment after node failures.

5 Simulation Results

5.1 Energy Efficiency

Recall that the initial network setup cost is the energy cost for building a data gathering tree and nodes' schedule. The schedule is then maintained by nodes throughout their lifetime in the network. Thus, this setup cost is a one-off cost. Table 3 shows that the average node energy overhead in the initial network setup is only less than 0.54% of the node's total initial energy across various network densities. Generally, as transmission radii increases, the average percentage of setup cost decreases. In the cases of high transmission radii, nodes have shorter hops to the base station and hence the number of token passing events is reduced, which leads to less energy dissipations. In contrast, low transmission radii result in nodes with short links requiring a large number of token passing in the data gathering tree construction phase and time slot assignment phase.

High default transmission radii make the network topology denser and nodes can reach the base station in fewer hops so that fewer nodes become routers. This results in a lower average node energy dissipation in each data gathering cycle and lower average energy consumed per datum, as confirmed by our simulation results in Figure 8 and 9 respectively. For low transmission radii, the network topology is sparse and there are a higher number of nodes with short links, which lead to a higher number of routers. This can increase the average node energy dissipation per data gathering cycle and increase the average energy cost per datum. For example, two routers consume twice the receive energy cost than one router only. Thus, smaller transmission radii are more expensive for the distance than larger transmission radii.

An important goal in developing algorithms for WSNs is to minimize the energy consumption of sensor nodes' operations in order to increase the lifetime of the network. In QMAC, nodes adjust their transmission power according to the transmission radius between nodes and their parent or children. If a node's parent is nearer than the default transmission radius, the node reduces its transmission power. Figure 10 shows that as the network density increases, the average node lifetime increases (measured in the number of data gathering cycles). It also

No. of nodes	30 m	40 m	50 m	60 m	70 m
50	0.2377%	0.2366%	0.2377%	0.2372%	0.2358%
75	0.3548%	0.3557%	0.3549%	0.3541%	0.3533%
100	0.5325%	0.5317%	0.5305%	0.5296%	0.5297%

Table 3: Average percentage of initial network setup cost versus default transmission radius.

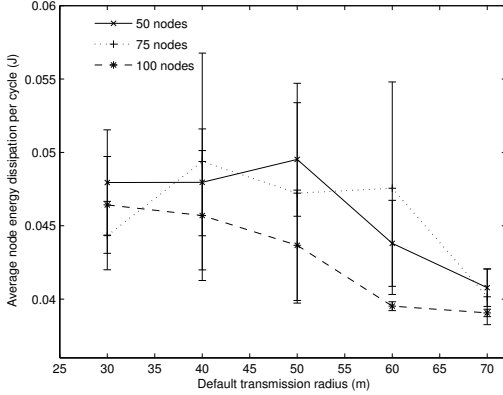


Figure 8: Average energy dissipation per data gathering cycle versus default transmission radius.

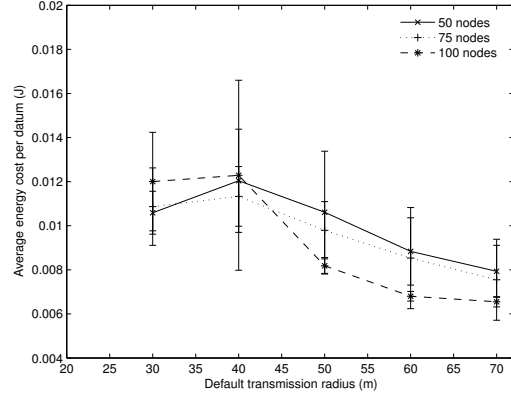


Figure 9: Average energy cost per datum per data gathering cycle versus default transmission radius.

shows that the default transmission radius of 70 m gives the highest average node lifetime across various network densities. The simulation results are consistent with that of the average node energy dissipation per data gathering cycle metric shown in Figure 8.

In QMAC, nodes switch to the low power sleep mode if they are not scheduled to receive or transmit data. Thus, nodes sleep most of the time in each data gathering cycle. Consequently, QMAC provides a high percentage of sleep time in each data gathering cycle over various network densities, as presented in Table 4. This simulation results prove that QMAC provides a significant energy saving. As the network density increases, the average percentage of sleep time increases. Furthermore, the average percentage of sleep time is almost constant regardless of transmission radii as nodes are only active during their scheduled time slots.

5.2 Network Performance

In principle, higher communication range provides higher connectivity. Krishnamachari et al. suggest that there are “zero-one” phase transitions for network connectivity [32]. Based on our simulation results, Figure 11 shows that the probability of network connectivity in WSNs

No. of nodes	30 m	40 m	50 m	60 m	70 m
50	95.6374%	95.6426%	95.7764%	95.4091%	95.3925%
75	97.0771%	97.1528%	97.1024%	97.2049%	96.9501%
100	97.6579%	97.8072%	97.7942%	97.6785%	97.5284%

Table 4: Average percentage of sleep time per data gathering cycle versus default transmission radius.

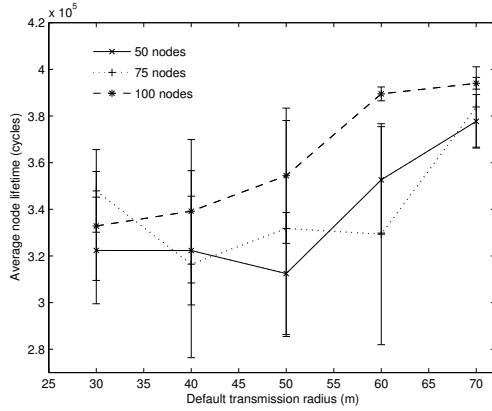


Figure 10: Average node lifetime versus default transmission radius.

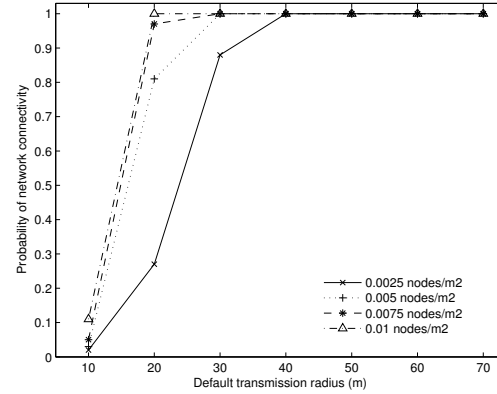


Figure 11: Average network connectivity versus default transmission radius.

asymptotically transits sharply from zero to one at a critical threshold value depending on the network's density. It is then adequate to formulate the density per square unit to be as follows:

$$\text{Density square unit threshold} = \frac{\text{No. of nodes}}{\text{Network dimensions}} \quad (3)$$

This simple formula is useful to predetermine settings for finding the required number of nodes and network dimension of a WSN before deployment. For example with Equation 3, we can find the number of nodes required to achieve good network connectivity if a network dimension and a critical density threshold value are known, and vice versa.

In QMAC, the end-to-end latency of a single datum transmission is measured by the amount of time required for a datum to go through a source node's parent and ancestors to the base station. Thus, data must go through n (where n is the number of routers) transmit and $n-1$ receive time slots. Figure 12 shows the average end-to-end latency of data in the networks in terms of the number of hops a datum must travel to reach the base station from a source node. As the transmission radius increases, the average end-to-end latency decreases. For low transmission radii, the distance among nodes are shorter. This means that more routers with shorter links on the path to the base station, which leads to large number of hops required to send data

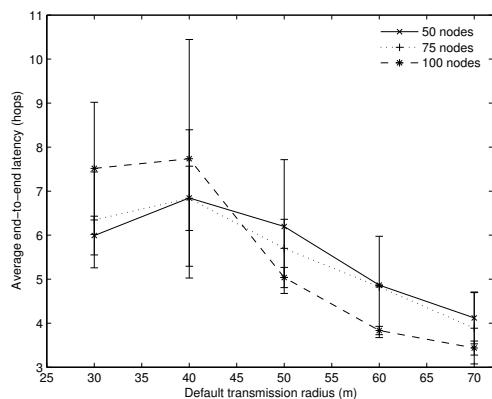


Figure 12: Average end-to-end latency versus default transmission radius.

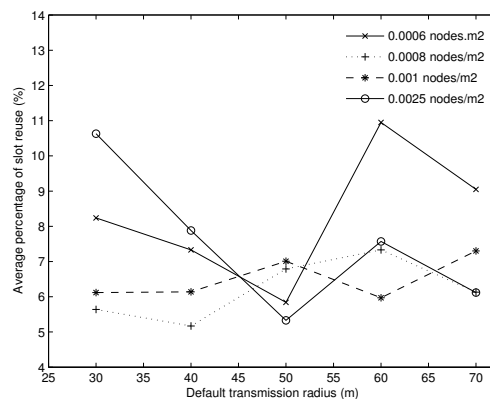


Figure 13: Average percentage of slot reuse versus default transmission radius.

and so results in an increased average end-to-end latency. Furthermore, there are more hops (average end-to-end latency) required to send data to the base station when the total number of nodes increases for the same low transmission radius. In contrast, for high transmission radii, the distance between nodes are longer, the end-to-end latency is decreased because fewer hops required for data to reach the base station. However, the average end-to-end latency in the low network density is higher than that of the high network density. This is because the network is sparse and so nodes that are further away from the base station need more routers to relay their data.

The QMAC TDMA scheme is better than CSMA scheme as nodes do not need to contend for the medium to send data, so that the time it takes for the data to reach from a source node to the base station is reduced and predictable. Contention-based protocols such as SMAC and TMAC periodically put nodes into sleeping mode after overhearing other nodes' communication. This approach introduces high latency because routers on the path to the base station that are more than one or two hops away from a source node are not notified of the ongoing traffic, and therefore the source node must wait for the next active period before it can transfer data [12, 33]. Furthermore, the QMAC TDMA scheme guarantees fair access as no node can monopolize the medium and nodes have their own slot to transmit data.

Sensor nodes have highly constrained computing power and memory. Multi-hop routing requires routers to buffer multiple packets prior to forwarding them to the next hop. This method can result in a low throughput as the number of packets dropped is high due to nodes' memory buffer over-flow. QMAC addresses this issue so that nodes only buffer one packet at a time to

be sent to the next hop. This ensures regular data delivery and hence provides a predictable throughput. Furthermore, the QMAC time slot assignment scheme ensures that nodes and their first-level and second-level neighbors cannot use the same transmit slot to ensure collision-free traffic. In order to validate this argument, we observed the collision at both senders and receivers in each slot in the network. Simulation results show that all packets sent by source nodes in networks reached the base station because there were no collisions. It is evident from our arguments and simulation results that QMAC protocol maintains a high throughput due to reliable packet delivery for different network configurations. According to Gao et al. [34], hops that are too long lead to excessive path loss. Therefore, in the real world implementation, the transmission radius should be made as small as possible to reduce the probability of interference and the probability of error in order to maximize the average throughput per node.

In QMAC, all nodes always claim the lowest available slot and so the slot reuse can be maximized. This method will also minimize the number of slots required in a network. Hence, QMAC spatial slot reuse is good. Figure 13 shows that QMAC provides slot reuse of at least 5% of the total transmission slots used in the network across different network scenarios and network densities. Furthermore, higher network densities (higher number of nodes and larger network area) with low transmission radii improve the percentage of slot reuse.

5.3 Fault Tolerance

We tested QMAC robustness by focusing on the most problematic scenarios, such as adding or removing a substantial number of nodes to or from the network. The default transmission radius was set to 30 m so that the networks were organized into short links. We simulated a short fault tolerant listening slot (FTS) such that nodes in a network could only take one new child in each data gathering cycle. We observed that as the number of new nodes introduced to the network increases, the average number of FTS required for these nodes to select a parent increases. Based on simulation results, it is apparent that QMAC is robust to node additions as shown in Figure 14. However, QMAC local repair performance starts to deteriorate when the number of nodes added is too large. This is because new nodes have to wait for a few more FTS before they manage to select a parent. We also observed that the increase in the network density reduces the latency of new nodes in selecting a parent. This is as expected because in high network densities, new nodes have a high probability of getting a parent within their default transmission radius. For low network densities, the networks are sparse so that new nodes have a high probability of increasing their transmission radius to find a parent, possibly

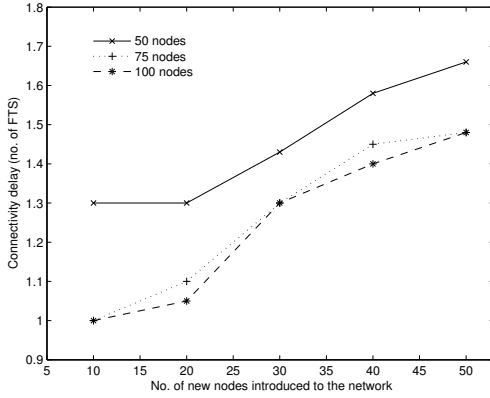


Figure 14: Average network connectivity delay when new nodes introduced to the network through short FTS.

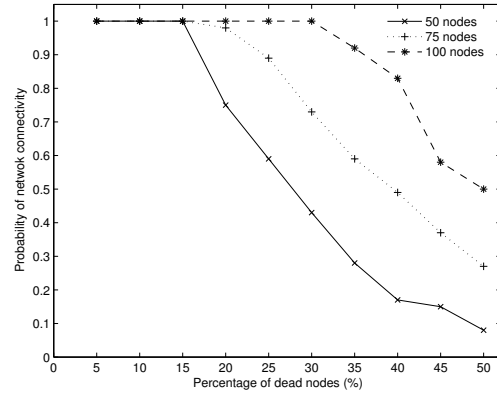


Figure 15: Average network connectivity re-establishment after a certain percentage of node failure.

causing nodes to overhear each other’s transmission. When that happens, nodes that lose in a contention wait for the next FTS and hence increases the latency for selecting a parent. The latency for selecting a parent can be improved by increasing the FTS period and so nodes in a network can take more new children during the FTS period in each data gathering cycle, i.e., more new nodes can join the network in each FTS. The human manager can customize this feature according to the application’s dynamics and needs.

The re-establishment of network connectivity after node failures is depicted in Figure 15. It shows that the network get partitioned as the number of node failures increases. The network connectivity represents the total number of nodes that are actually connected to the tree over the total number of nodes that are expected to be connected to the tree after a number of nodes in the network become faulty, i.e., dead nodes. Figure 15 also shows that QMAC network is adaptive to node failures as the remaining nodes still formed a connected network. However, depending on the network density, QMAC local repair breaks down at certain points. Based on simulation results of random faulty nodes, QMAC local repair starts to break down when the number of faulty nodes is equal or more than 25% of the total nodes in the networks. We also simulated the worst case scenarios in which roots of a sub-tree are gradually switched off over time. We found that the network connectivity reestablishment after QMAC local repair improves as the network density increases. For example, in the case of a network of 100 nodes, six roots of sub-trees that have more than three children are switched off and 28 nodes became orphan nodes. Since the network is dense, after QMAC local repair, only one orphan node is

unable to join the network. Although that orphan node has neighbors but it cannot select one of them as its new parent because they do not have an equal or a lower tree level than itself.

The energy expenditure for rebuilding nodes' schedule is free since QMAC piggybacks the *forward_slot_request* and *inform_slots* data onto existing data transmission and multi-function packets. We measured the average network energy expenditure to re-establish a network connectivity by computing the following energy costs:

- the cost for orphan nodes to send a distress signal to find a parent,
- the cost for prospective parents to send a reply to the orphan nodes,
- the cost for the orphan nodes to select a parent by sending a confirmation signal, and
- the cost for the selected parent to send a schedule to the orphan nodes.

Based on the previous fault tolerance scenario in which six roots of sub-trees are removed from the network of 100 nodes, the average energy expenditure for all nodes that involved in the local repair is 0.0018 J that is only around 0.000012% of the node's initial energy. This is mainly due to the QMAC parent selection scheme in which neighbors that have a higher tree level than an orphan node's tree level will not send a reply signal. Thus, only prospective parents are involved in the local repair.

We envision that the re-establishment of network connectivity can be improved by allowing a reverse path link in which a node can select a descendant or a node with a higher tree level than itself to be its parent if it cannot reach any neighbor. To ensure a full network connectivity, one of the node's descendants is required to select a parent that is attached to the network tree. This can be done by allowing descendants to communicate with each other during the FTS (peer-to-peer communication) and to elect one of them to find a new parent. The easiest election scheme is based on lowest NID. This method will be explored in the future. Furthermore, the location of faulty nodes and the network density are also critical for the network connectivity re-establishment. Nodes that are close to the base station exhaust their batteries more rapidly than other nodes due to relaying other nodes' data. They tend to dominate the network connectivity and possibly causing network partition when they are dead. Therefore, another avenue of research is to find a scheme to distribute excessive burden on these nodes.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we argue that to achieve good QoS in WSNs, end-to-end guarantees on data delivery (such as latency and throughput), fair access to the network for all sensor nodes, and robust self healing network should be guaranteed, whilst also respecting the severe operating constraints of WSNs (energy and memory). To the best of our knowledge, QMAC is the first MAC protocol for WSNs that proposes a scheduling scheme that provides a balance among QoS delivery, energy efficiency, and memory efficiency.

QMAC offers a flexible slot structure in which nodes in the network simply build, modify, or extend their schedules based on the local information available to them (RSL, TSL, CSL). QMAC also utilizes TDMA scheduling for efficient data gathering in wireless sensor networks. Nodes only transmit and receive packets at their own time slot(s) and sleep until their slots turn up again. This fixed scheduling approach offers a high energy saving by reducing idle listening, avoiding collisions, and avoiding overhearing. QMAC also minimizes energy consumption by adaptively adjusting node transmission power for a given topology. Furthermore, a TDMA scheme is better than a CSMA scheme as nodes do not need to contend for the medium to send data, so that the time it takes for the data to reach from a source node to the base station is reduced, i.e., eliminates queue delay (resulting in a lower latency).

In assigning time slots, QMAC uses the depth-first-search technique, where data sent by a source node is forwarded by routers to the base station in an interleaving manner so that buffer overflow in routers can be avoided. Furthermore, QMAC allows a slot to be used by many nodes as long as they are not within interference range of each other. Based on simulation results, QMAC time slot assignment approach is proven to be scalable with respect to the network density as it can guarantee a predictable throughput and predictable latency for different network topologies and densities. Furthermore, the schedule is fixed and maintained throughout the lifetime of a node to ensure fair access to the network for all sensor nodes.

QMAC implements local synchronization to minimize clock drifts among nodes: a parent synchronizes its children during a designated multi-function period. Furthermore, nodes in a WSN can continue to function accurately in the event of failure of individual nodes by performing a local repair in the fault tolerant listening period and the multi-function slot in each data gathering cycle. Thus, QMAC is robust to network dynamics such as erroneous network link, dying node, and topology changes. Both the time synchronization and fault tolerant scheme take advantage of collision-free traffic. They are also achieved almost cost-free by utilizing one

communication only during the multi-function slot.

Through comprehensive analysis and simulation results we substantiate our argument that QMAC ensures good QoS by offering predictable latency, predictable throughput, fair access, and robust self healing network, while achieving energy efficiency and memory efficiency, for different network configurations. We also analyzed the relationship between transmission radii and QoS. Although high transmission radii give a lower energy consumption and lower latency than low transmission radii, in the real world implementation low transmission radii are desirable for effective communication. This is because a high signal-to-noise (SNR) ratio is obtained at the short range and so ensures a low probability of error, i.e., a tree constructed by QMAC will contain more hops over stronger signal. Thus, low transmission radii reduces energy consumption by increasing the probability of reliable data delivery and so maximizes the average throughput per node.

Currently, QMAC assumes a regular data reporting by all nodes in the network. In future work we plan to extend QMAC to allow it to be adaptive to changing data values and dynamic traffic loads. This work is developed to meet the need of critical applications such as disaster monitoring of bushfires and flooding, which require a protocol to be delay intolerant and mission critical. In this type of application, the sensor nodes are expected to detect “interesting” events and accordingly take an appropriate action as quickly and reliably as possible. For example, in the situation where the nodes’ data value (e.g., wind, soil moisture, temperature) in a certain part of the network changes rapidly, it is desirable to allow this part of the network to send data to the base station more often, while nodes in other parts of the networks whose data value do not change much can give up their schedules. We plan to extend QMAC to accommodate this priority-based decision making through systematic resource transfers in which the resource (slots) from one part of the network can be used by another part of the network. The problem of systematic resource transfers is interesting by itself and is an important property for most WSN applications. To the best of our knowledge, this approach has never been investigated before.

7 Acknowledgements

The authors would like to thank David Glance for many constructive suggestions on a previous draft of this report.

References

- [1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, “Wireless Sensor Networks for Habitat Monitoring,” in *Proc. ACM WSNA Conf.*, Sep. 2002.
- [2] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, “Instrumenting the World with Wireless Sensor Networks,” in *Proc. IEEE ICASSP Conf.*, May 2001.
- [3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless Sensor Networks: A Survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.
- [4] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, “A Wireless Sensor Network for Structural Monitoring,” in *Proc. ACM SenSys Conf.*, Nov. 2004.
- [5] R. Iyer and L. Kleinrock, “QoS Control for Sensor Networks,” in *Proc. IEEE ICC Conf.*, May 2003.
- [6] Dazhi Chen and Pramod K. Varshney, “QoS Support in Wireless Sensor Networks: A Survey,” in *Proc. ICWN Conf.*, June 2004.
- [7] L. H. A. Correia, D. F. Macedo, A. L. dos Santos, and J. M. Nogueira, “Issues on QoS Schemes in Wireless Sensor Networks,” Tech. Rep. RT.DCC.004/2005, DCC/UFMG, Apr. 2005.
- [8] G. Lu, B. Krishnamachari, and C. S. Raghavendra, “An Adaptive Energy-Efficient and Low-latency MAC for Data Gathering in Wireless Sensor Networks,” in *Proc. IEEE IPDPS Conf.*, Apr. 2004.
- [9] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer, “A Reactive Soil Moisture Sensor Network: Design and Field Evaluation,” *International Journal of Distributed Sensor Networks*, vol. 1, no. 2, pp. 149–162, 2005.
- [10] M. Bhardwaj, T. Garnett, and A. P. Chandrakasan, “Upper Bounds on the Lifetime of Sensor Networks,” in *Proc. IEEE ICC Conf.*, June 2001.
- [11] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori, “Performance Measurements of Motes Sensor Networks,” in *Proc. ACM MSWiM Conf.*, Oct. 2004.

- [12] W. Y., J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. IEEE INFOCOM Conf.*, June 2002.
- [13] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks," in *Proc. ACM Sensys Conf.*, Mar. 2003.
- [14] L.F.W van Hoesel and P.J.M. Havinga, "A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks," in *Proc. INSS Conf.*, June 2004.
- [15] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for Multi-Hop Wireless Sensor Networks," in *Proc. SpringerVerlag ALGOSENSORS Conf.*, July 2004.
- [16] S. S. Kulkarni and M. U. Arumugam, "TDMA Service for Sensor Networks," in *Proc. IEEE ICDCSW Conf.*, Mar. 2004.
- [17] M. J. Miller and N. H. Vaidya, "A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio," *IEEE Transactions on Mobile Computing*, vol. 4, no. 3, pp. 228–242, 2005.
- [18] Y. Yuan, Z. Yang, Z. He, and J. He, "An Integrated Energy Aware Wireless Transmission System for QoS Provisioning in Wireless Sensor Network," *Elsevier Computer Communications (Article in Press)*, May 2005.
- [19] M. L. Sichitiu, "Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks," in *Proc. IEEE INFOCOM Conf.*, Mar. 2004.
- [20] K. Langendoen and G. Halkes, *To appear in The Embedded Systems Handbook*, chapter Energy-efficient Medium Access Control, 2005.
- [21] J. Elson and D. Estrin, "Time Synchronization for Wireless Sensor Networks," in *Proc. IEEE IPDPS Conf.*, Apr. 2001.
- [22] M.L. Sichitiu and C. Veerarittiphan, "Simple, Accurate Time Synchronization for Wireless Sensor Networks," in *Proc. IEEE WCNC Conf.*, Mar. 2003.

- [23] A. Boukerche, I. Chatzigiannakis, and S. Nikolettseas, "Power-Efficient Data Propagation Protocols for Wireless Sensor Networks," *To appear in the SCS Journal of Simulation: Transactions of the Society for Modeling and Simulation International*, 2005.
- [24] T. Armstrong, "Wake-Up Based Power Management in Multi-hop Wireless Networks," <http://www.eecg.toronto.edu/~trevor/Wakeup/survey.pdf>, 2005.
- [25] A. Gonzalez, I. Marshall, L. Sacks, I. Henning, and T. Khan, "A Self-Synchronised Scheme for Automated Communication in Wireless Sensor Networks," in *Proc. IEEE ISSNIP Conf.*, Dec. 2004.
- [26] G. Gupta and M. Younis, "Fault-Tolerant Clustering of Wireless Sensor Networks," in *Proc. IEEE WCNC Conf.*, Mar. 2003.
- [27] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, "Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks," in *Proc. IEEE Sensors*, June 2002.
- [28] "MICA2 Mote Datasheet," http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf, 2005.
- [29] G. Xing, C. Lu, Y. Zhang, Q. Huang, and R. Pless, "Minimum Power Configuration in Wireless Sensor Networks," in *Proc. ACM MobiHoc Conf.*, May 2005.
- [30] E. W. Weisstein, "Lagrange Interpolating Polynomial," <http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html>, 2005.
- [31] I. Raicu, L. Schwiebert, S. Fowler, and S. K.S. Gupta, "Local Load Balancing for Globally Efficient Routing in Wireless Sensor Networks," *To appear in the International Journal of Distributed Sensor Network*, 2005.
- [32] B. Krishnamachari, S. B. Wicker, R. Bejar, and M. Pearlman, *Communications, Information and Network Security*, chapter Critical Density Thresholds in Distributed Wireless Networks, 2002.
- [33] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. ACM SenSys Conf.*, Mar. 2003.
- [34] Q Gao, K J Blow, D J Holding, I Marshall, and X H Peng, "Radio Range Adjustment for Energy Efficient Wireless Sensor Networks," *To appear in the Ad-Hoc Networks*, 2005.