# An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Sensor Networks

Gang Lu, Bhaskar Krishnamachari, Cauligi S. Raghavendra
Department of Electrical Engineering, University of Southern California
Los Angeles, CA 90089
{ganglu, bkrishna, raghu}@usc.edu

***Abstract*— In many sensor network applications the major traffic pattern consists of data collected from several source nodes to a sink through a unidirectional tree. In this paper, we propose DMAC, an energy efficient and low latency MAC that is designed and optimized for such data gathering trees in wireless sensor networks.**

**We first show that previously proposed MAC protocols for sensor networks that utilize activation/sleep duty cycles suffer from a *data forwarding interruption problem*, whereby nodes on a multihop path to the sink are not all notified of data delivery in progress, resulting in significant sleep delay. DMAC is designed to solve the interruption problem by giving the active/sleep schedule of a node an offset that depends upon its depth on the tree. This scheme allows continuous packet forwarding because all nodes on the multihop path can be notified of the data delivery in progress. DMAC also adjusts node duty cycles adaptively according to the traffic load in the network by varying the number of active slots scheduled in an interval. We further propose a *data prediction* mechanism and the use of *more to send* (MTS) packets in order to alleviate problems pertaining to channel contention and collisions. Our simulation results show that by exploiting the application-specific structure of data gathering trees in sensor networks, DMAC provides significant energy savings and latency reduction while ensuring high data reliability.**

## I. Introduction

A wireless sensor network is a distributed system comprised of large numbers of small battery-powered devices that sense and collect information about the environment. WSN can be used in a wide range of applications, such as target tracking, habitat sensing and fire detection. Typically in WSN, local nodes coordinate on local data processing and deliver messages to a common sink. The important design features for medium access control protocols in a WSN are:

- Energy: It is often not feasible to replace or recharge batteries for those nodes. Energy efficiency is a critical issue in order to prolong network lifetime. Measurements have shown that communication consumes much more energy than computation. An energy efficient MAC is thus needed to reduce energy cost of sensor nodes.
- Latency: Latency requirement depends on the applications. In an environment surveillance application, when an event is detected, sensor nodes should be able to report the local processing result to sink in real time so that appropriate action can be taken promptly.
- Throughput: Throughput requirement varies with different applications too. Some applications need to sample the environment with fine temporal resolution and the more data received at the sink, the better. In other applications, such as fire detection, it may suffice for a single report to arrive at the sink.
- Fairness: Another important concern in WSN is fairness at the MAC layer. This concern is addressed in [1] through the use of adaptive techniques to balance route-through and originating traffic. However, we shall consider fairness issues to be beyond the scope for this paper, although the techniques proposed in [1] may be adaptable to our work.

Among these important requirements for MACs, energy efficiency is typically the primary goal in WSN. Previous works (in particular [2], [4], [5], [7], [8], [15], [13]) have identified idle listening as a major source of energy wastage. Measurements show that idle listening consumes nearly the same power as receiving. Since in sensor network applications, traffic load is very light most of the time, it is often desirable to turn off the radio when a node does not participate in any data delivery. The scheme proposed in [5] puts idle nodes in power saving mode and switches nodes to full active mode when a communication event happens. However, even

when there is traffic, idle listening still may consume most of the energy. For example, consider a sensor node that reports its sensing reading via one packet each second. Suppose the packet length is 100 bytes, its transmission takes only $8ms$ for a radio of 100Kbps data rate, while the other $992ms$ is wasted in idle listening. S-MAC [2] reduces idle listening energy cost by reducing the duty cycle of a sensor node in which a node follows a periodic active/sleep schedule. During sleep periods, nodes turn off radio to conserve energy. During active periods, nodes turn on radio to Tx/Rx messages.

Although a low duty cycle MAC is energy efficient, it has three side-effects. First, it increases the packet delivery latency. At a source node, a sampling reading may occur during the sleep period and has to be queued until the active period. An intermediate node may have to wait until the receiver wakes up before it can forward a packet received from its previous hop. This is called *sleep latency* in SMAC [2], and it increases proportionally with hop length by a slope of schedule length (active period plus sleep period). Secondly, a fixed duty cycle does not adapt to the varying traffic rate in sensor network. A fixed duty cycle for the highest traffic load results in significant energy wastage when traffic is low while a duty cycle for low traffic load results in low message data delivery and long queuing delay. Therefore it is desirable to adapt the duty cycle under variant traffic load. Thirdly, a fixed synchronous duty cycle may increase the possibility of collision. If neighboring nodes turn to active state at the same time, all may contend for the channel, making a collision very likely.

There are several works on reducing sleep delay and adjusting duty cycle to the traffic load. Those mechanisms are either implicit (e.g. [2], [4]), in which nodes remain active on overhearing of ongoing transmission or explicit (e.g. [7]), in which there are direct duty cycle adjusting messages. SMAC [2] proposed adaptive listening to reduce the sleep delay. In adaptive listening, a node who overhears its neighbor's transmission wakes up for a short period of time at the end of the transmission, so that if it is the next hop of its neighbor, it can receive the message without waiting for its scheduled active time. In TMAC [4], a node keeps listening and potentially transmitting as long as it is in an active period. An active period ends when no activation event has occurred for a certain time. The activation time events include reception of any data, the sensing of communication on the radio, the end-of-transmission of a node's own data packet or acknowledgement, etc. FRTS is employed to solve the

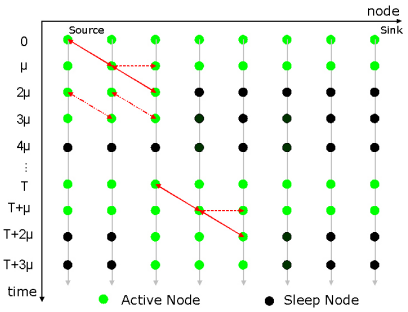


Fig. 1. SMAC with adaptive listening in a chain.

early sleep problem. The authors in [7] proposed a slot-based power management mechanism. If the number of buffered packets for an intended receiver exceeds a threshold $L$, the sender signals the receiver to remain on for the next slot. A node requested to stay awake sends an acknowledgement to the sender, indicating its willingness to remain awake in the next slot. The sender can then send a packet to the receiver in the following slot. The request is renewed on a slot-by-slot basis.

However, in previously proposed mechanisms (whether explicit or implicit), not all nodes beyond one hop away from the receiver can overhear the data communication, and therefore packet forwarding will stop after a few hops. As we shall describe in section II, this *data forwarding interruption problem* causes sleep latency for packet delivery.

After describing the data forwarding interruption problem, we will describe the proposed DMAC mechanism in section III. DMAC employs a *staggered active/sleep schedule* to solve this problem and enable continuous data forwarding on the multihop path. In DMAC, *data prediction* is used to enable active slot request when multiple children of a node have packets to send in a same sending slot, while *More to Send packet* is used when nodes on the same level of the data gathering tree with different parents compete for channel access. Once we describe the DMAC protocol, we shall evaluate its performance via simulations in section IV.

## II. Data Forwarding Interruption Problem

The data forwarding interruption problem exists in implicit adaptive duty-cycle techniques because the overhearing range is limited by radio's sensitivity to signals on air. Nodes that are out of the hearing range of both the sender and the receiver are unaware of ongoing data transmissions, and therefore go to sleep until the next cycle/interval. The data forwarding process will then stop at the node whose next hop towards the sink is out of the

overhearing range because it is in sleep mode. Packets will then have to be queued until the next active period which increases latency. Also, for explicit mechanism, the duty cycle adjusting messages can only be forwarded limited hops in an active period. So nodes out of the range go to sleep after their basic duty cycle, leading to interrupted data forwarding.

Assume an active period (i.e. the portion of time in each interval when a node is active, unless there is more data to be sent/received) is only long enough to transmit one packet each hop. In SMAC, only the next hop of the receiver can overhear the data transmission and remains active for a long period. Other nodes on the multihop path do not overhear the data transmission thus go to sleep after the basic active period, resulting in the interruption of packet forwarding to the sink till the next duty cycle. It is shown theoretically in [2] that the delay with adaptive listening still increases linearly with the number of hops with a slope that is half of the interval length. Therefore, compared with the case of no adaptive listening, the delay is only reduced by half. Meanwhile, nodes other than the next-hop in the neighborhood of the sender and the receiver also overhear a data transmission and thus may remain active unnecessarily. Similarly, in TMAC [4], a node remains active if it senses any communication on the air. Typically, a radio's interference range is larger than its transmission range (e.g. in ns-2, the interference range is set to more than twice the transmission range). In TMAC, any neighbor nodes in the interference range of either the sender or the receiver will remain active. Many of the nodes do not participate in the data delivery but remain active for an unnecessarily long period which wastes energy. Meanwhile only nodes in the interference range hear the communication, while other nodes out of the interference range on the multi-hop path still go to sleep after their basic active period. Thus packets still suffer from the data forwarding interruption problem. The FRTS proposed in TMAC can increase the number of packets delivered in one frame and as a side effect, can help forward a packet one hop further. The same problem happens to [7], in which the request for a next active slot can be only received by the next hop. The nodes beyond that will still go to sleep after their basic active period.

Figure 1 illustrates this data forwarding interruption problem using SMAC with adaptive listening as an example. There is a chain of nodes with a single source on the far left and the sink on the far right. We assume an active period is only long enough to transmit one packet one hop. By adaptive listening, the next hop of the receiver overhears the receiver's ACK or CTS packet, then remains active an additional slot. But other nodes still go to sleep after their active periods. If the source has multiple packets to send, those packets can only be forwarded two hops away every interval $T$. Latency is also only reduced by half. Collision is also depicted in the figure. Suppose in slot between $2\mu$ and $3\mu$, both node 0 and node 1 need to transmit packets, a collision could happen. Things will be even worse if between $0$ and $\mu$, all nodes have packets to send.

The hearing/interference range also causes a tradeoff between the latency and energy. If the hearing range is long, latency is reduced since more nodes on the path can overhear the communication and remain active. Meanwhile, more nodes not on the path also overhear the communication and waste energy in idle listening on the increased active periods. We need a MAC that can tell all nodes on the path to stay active and/or increase their duty cycles and all other nearby nodes to sleep in order to enable continuous data forwarding without incurring energy waste of unrelated nodes.

## III. DMAC Protocol Design

### A. Staggered Wakeup Schedule

One can identify three main communication patterns in sensor network applications. The first involves local data exchange and aggregation purely among nearby nodes (these can be handled by clustering or simple medium access mechanisms). The second involves the dispatch of control packets and interest packets from the sink to sensor nodes. Such sink-originated traffic is small in number and may not be latency sensitive. We can reserve a separate active slot periodically with a larger interval length for such control packets. The third and most significant traffic pattern in WSN is data gathering from sensor nodes to sink. For a sensor network application with multiple sources and one sink, the data delivery paths from sources to sink are in a tree structure, an *data gathering tree* [14], [17]. Routes may change during data delivery, but we assume that sensor nodes are fixed without mobility and that a route to the sink is fairly durable, so that a data gathering tree remains stable for a reasonable length of time. Flows in the data gathering tree are unidirectional from sensor nodes to sink. There is only one destination, the sink. All nodes except the sink will forward any packets they receive to the next hop (except local processing packets which are handled in cluster). Our key insight in designing a MAC for such a tree is that it is feasible

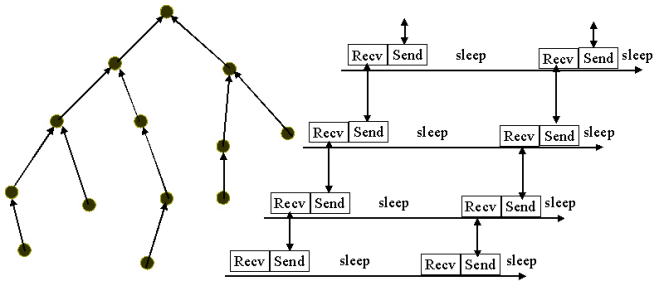


Fig. 2. DMAC in a data gathering tree.

to stagger the wake-up scheme so that packets flow continuously from sensor nodes to the sink. *DMAC is proposed to deliver data along the data gathering tree*, aiming at both energy efficiency and low latency.

In DMAC, we stagger the activity schedule of nodes on the multihop path to wake up sequentially like a chain reaction. Figure 2 shows a data gathering tree and the staggered wake-up scheme. An interval is divided into receiving, sending and sleep periods. In receiving state, a node is expected to receive a packet and send an ACK packet back to the sender. In the sending state, a node will try to send a packet to its next hop and receive an ACK packet. In sleep state, nodes will turn off radio to save energy. The receiving and sending periods have the same length of $\mu$ which is enough for one packet transmission and reception. Depending on its depth $d$ in the data gathering tree, a node skews its wake-up scheme $d\mu$ ahead from the schedule of the sink. In this structure, data delivery can only be done in one direction towards the root. Intermediate nodes have a sending slot immediately after the receiving slot.

A staggered wake-up schedule has four advantages. First, since nodes on the path wake up sequentially to forward a packet to next hop, sleep delay is eliminated if there is no packet loss due to channel error or collision. Second, a request for longer active period can be propagated all the way down to the sink, so that all nodes on the multihop path can increase their duty cycle promptly to avoid data stuck in intermediate nodes. Third, since the active periods are now separated, contention is reduced. Fourth, only nodes on the multihop path need to increase their duty cycle, while the other nodes can still operate on the basic low duty cycle to save energy.

In a multi-hop wireless network, it is well known that contention-based MACs suffer from the hidden node problem. In MACAW [16], virtual and physical carrier sense and RTS/CTS exchange are utilized to reduce hidden node problem. For large packet sizes, these small control packets are efficient in saving the possible high cost of a packet loss. However, for sensor networks where packet size is usually small, the overhead of RTS/CTS could be very high compare to the actual data transmission cost. Therefore we do not advocate the use of RTS/CTS in DMAC. DMAC, however, employs link layer ARQ through ACK control packet and data retransmission, and the hidden node problem is mitigated to some extent through the manner in which active slots are scheduled so that nodes on the same path do not cause hidden node collisions. Although ACK packets consume energy and bandwidth, we believe these are essential for the link reliability to recover lost packet due to harsh quality wireless channel and contention (though there is always the possibility of using implicit ACKs [1] in case of highly reliable links). If a sending node does not receive an ACK packet from receiving node, it will queue the packet until next sending slot. After 3 retransmissions, the packet will be dropped.

In DMAC, nodes with the same depth will have same offset, and thus a synchronous schedule. During the sending period, nodes will compete for the channel. To reduce collision during this period, every node backs off for a backoff period ($BP$) plus a random time within a contention window at the beginning of a sending slot. Since the length of a sending slot is only enough for one packet transmission, there is no need for exponential contention window increase, and therefore we employ a fixed contention window. When a node receives a packet, it waits for a short period ($SP$) then transmits the $ack$ packet back to the sender. $BP$ and $SP$ are two inter-frame spaces with $BP > SP$ in order to assure the collision free reception of the $ack$ packet [1].

Based on the above choices, the sending and receiving slot length $\mu$ is set to:

$$\mu = BP + CW + DATA + SP + ACK$$

where $CW$ is the fixed contention window size, $DATA$ is the packet transmission time(we assume all packets are in the same length) and $ACK$ is the ACK packet transmission time.

Synchronization is needed in DMAC. However, local synchronization is enough since a node only need to be aware of its neighbors' schedule. There exist techniques such the reference broadcast synchronization scheme (RBS)[6] that can achieve time synchronization precision of $3.68 \pm 2.57 \mu sec$ after 4 hops. Given that typical slot lengths are on the order of $10ms$ in length, we will

[1]They are similar to the $difs$ and $sifs$ in IEEE 802.11 protocol.

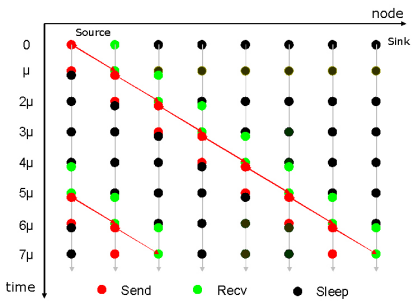


Fig. 3. DMAC in a chain.

assume that synchronization is available in the following discussions.

We should mention that ongoing work to improve SMAC [12] also explores the possibility of using offsets/phase differences in scheduling to reduce latency. It does a simple analysis for two cases. In case 1 where the phase difference is in the same direction of the data flow, delay is reduced. In case 2 where phase difference is in the opposite direction, delay is increased. It then proposes a scheme to design global offset synchronization to minimize delay.

### *B. Data Delivery and Duty Cycle Adaptation in Multihop chain*

Figure 3 shows DMAC operation in a multihop chain. Every node periodically turns to receiving, sending and sleep states. It is shown that when there is no collision, a packet will be forwarded sequentially along the path to the sink, without sleep latency.

However when a node has multiple packets to send at a sending slot, it needs to increase its own duty cycle and requests other nodes on the multihop path to increase their duty cycles too. We employed a slot-by-slot renewal mechanism. We piggyback a *more data* flag in the MAC header to indicate the request for an additional active periods. The overhead for this is very small. Before a node in its sending state transmits a packet , it will set the packet's *more data* flag if either its buffer is not empty or it received a packet from previous hop with *more data* flag set. The receiver check the *more data* flag of the packet it received, and if the flag is set, it also sets the *more data* flag of its ACK packet to the sender. With the slot-by-slot mechanism and the policy to set *more data* flag when buffer is not empty, DMAC can react quickly to traffic rate variation to be both energy efficient and maintain low data delivery latency.

A node will decide to hold an additional active period if:

1) It sends a packet with the *more data* flag set and receives an ACK packet with the *more data* flag set.
2) It receives a packet with *more data* flag set.

In DMAC, even if a node decides to hold an additional active period, it does not remain active for the next slot but schedules a $3\mu$ sleep then goes to the receiving state as shown in Figure 3. The reason for a $3\mu$ sleep is that it knows the following nodes on the multihop path will forward the path in the next 3 slots. In [3], it is shown that the maximum utilization of a chain of ad hoc nodes is $\frac{1}{4}$ if the radio's interference range is twice the transmission range. So the maximum sending rate for a node is one packet per 4 slots. However, to accommodate the possibility of short range between two neighbor nodes, a node will only send one packet every $5\mu$ in DMAC in order to avoid collision as much as possible. Of course, this may reduce the maximum network capacity by about 20%, but if the traffic load is more than 80% of the maximum channel capacity duty-cycled mechanisms would not function efficiently in any case, making this a moot point.

A good result of the staggered wake up schedule is that the *more data* flag can be propagated to all the nodes on the multi-hop path. In Figure 3, suppose the source sets the *more data* flag of the first packet, since this packet can be forwarded to the sink without interruption, all nodes will receive the first packet with *more data* flag set thus will hold an additional active period $3\mu$ later after their sending slot. So at time $5\mu$, the second packet from the source can still be delivered to the sink with very short delay.

However, there is a possibility of inconsistency on the new active period request. We may have a situation where the receiving node is awake, while the sending node is off. This could happen when the receiving node received a packet with *more data* flag, but the ACK packet sent by the receiver is not received by the sender. In this case, the receiving node will waste an active period in idle listening. However, the slot-by-slow renewal mechanism will make sure that a node will only waste one additional active period, though packets will have a sleep delay. The situation where the sending node is awake but the receiving node is off is not possible since the sending node will hold an additional active period only if it successfully received an ACK packet with *more data* which guaranteed the receiver is awake. DMAC avoids this situation because transmission is more energy costly than receiving and a packet retransmission chance will be wasted.

Measurements have showed that the cost for switching radio between active and sleep is not free. However, the overhead of this switching is likely to be small [11] compared to energy savings in a $3\mu$ sleep period of around $30ms$.

### C. Data Prediction

In last section, we assume a single source needs a higher duty cycle than the basic lower duty cycle. In a data gathering tree, however, there is a chance that each source's rate is small enough for the basic duty cycle, but the aggregated rate at an intermediate node exceeds the capacity of basic duty cycle. For example, suppose a node C has 2 children A and B. Both children has only one packet to send every interval. At the sending slot of an interval, only one child can win the channel and send a packet to the node. Assume A wins the channel and sends a packet to C. Since A's buffer is empty, the *more data* flag is not set in A's packet. C then goes to sleep after its sending slot without a new active period.. B's packet would then have to be queued until next interval. This results in sleep delay for packets from B.

We propose a scheme called *data prediction* to solve this problem. If a node in receiving state receives a packet, it predicts that its children still have packets waiting for transmission. It then sleeps only $3\mu$ after its sending slot and switches back to receiving state. All following nodes on the path also receive this packet, and schedule an additional receiving slot. In this additional data prediction receiving slot, if no packet is received, the node will go to sleep directly without a sending slot. If a packet is received during this receiving slot, the node will wake up again $3\mu$ later after this sending slot.

For a node in sending state, if during its backoff period, it overhears the ACK packet from its parent in the data gathering tree, it knows that this sending slot is already taken by its brother but its parent will hold an additional receiving slot $3\mu$ later, so it will also wake up $3\mu$ later after its sending slot. In this additional sending slot, the node then can transmit a packet to its parent. Figure 4 shows an example of the *data prediction* scheme.

Of course, this generalizes beyond the case of a node having two children. If a node has more children, in the additional receiving slot, the remaining children would compete for the channel again. This process would repeat until eventually, all children will be able to transmit their packet to the parent one by one with shortest delay. However if a collision happens, all children nodes have to wait until next interval. But since those nodes have the same parent, they are at most two hops away. Hence they can detect each other's transmission, and the chance of a collision due to hidden node problem is small.

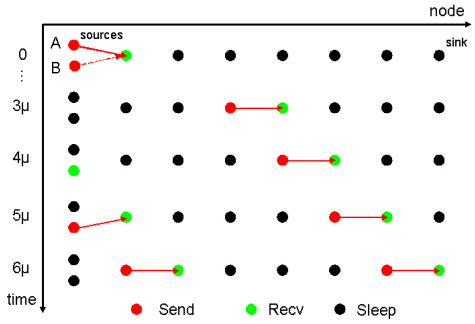


Fig. 4. Data prediction scheme reduces sleep delay.

There is an overhead brought by the *data prediction* scheme. After the reception of the last packets from its children, a node will remain idle for a receiving slot which waste energy in idle listening. Compared to the huge latency reduction by the *data prediction*, we believe this additional overhead would be worthwhile.

### D. MTS

Although a node will sleep $3\mu$ before an additional active period to avoid collision, there is still a chance of interference between nodes on different branches of the tree. Consider the example in Figure 5; two nodes A and B are in interference range of each other with different parents in the data gathering tree. In the sending slot of one interval, A wins the channel and transmits a packet to its parent. Neither B nor its parent C holds additional active slots in this interval. Thus B can only send its packet in the sending slot of next interval, resulting a sleep latency of $T$. Since C does not receive any packet in its receiving slot and B does not overhear ACK packet from C in its sending slot, *data prediction* scheme will not work.

We propose the use of an explicit control packet, that we refer to as *More to Send* (MTS), to adjust duty cycle under the interference. The MTS packet is very short with only destination's local ID and a flag. A MTS packet with flag set to 1 is called a request MTS. A MTS packet with flag set to 0 is called a clear MTS.

A node sends a request MTS to its parent if either of the two conditions is true:

1) It can not send a packet because channel is busy. After the node's back-off timer fires, it finds there is not enough time for it to send a packet and it does not overhear its parent's ACK packet. It then

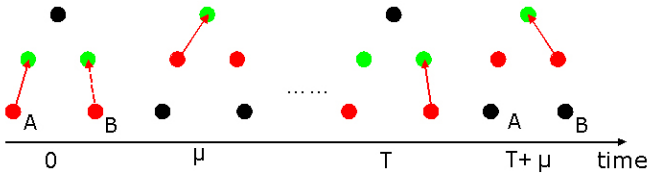


Fig. 5. Interference between two sending nodes causes sleep delay.

assume it lost the channel because of interference from other nodes.

2) It received a request MTS from its children. This is aimed to propagate the request MTS to all nodes on the path.

A request MTS is sent only once before a clear MTS packet is sent.

A node sends clear MTS to its parent if the following three conditions are true:

1) Its buffer is empty.
2) All request MTSs received from children are cleared.
3) It sends a request MTS to its parent before and has not sent a clear MTS.

A node which sent or received a request MTS will keep waking up periodically every $3\mu$. It switches back to the basic duty cycle only after it sent a clear MTS to its parent or all previous received request MTS from its children were cleared.

Same as the slot-by-slot renewal scheme and data prediction scheme, the higher duty cycle request by MTS packets are forwarded through the staggered schedule to all nodes on the multihop path. However, to reduce the overhead of MTS packets, instead of sending MTS packets to renew active period slot by slot, only two MTS packets are sent for a MTS request/clear period.

Inconsistent schedules are possible due to the loss of MTS packets. A soft timer is maintained to ignore current request MTS if no data is received or transmitted after a certain number of receiving slots, in order to avoid unnecessary active slots because of loss of clear MTS packets.

Slot length has to be increased to enable the transmission of MTS packets after a data transmission. Since the MTS packet is very short, the increase in slot length is small. Energy consumption also increases because the overhead of MTS packets and the longer slot. In the simulation section, we show that the use of MTS can significantly reduce latency in a sensor network at only small cost of energy consumption.

TABLE I
RADIO PARAMETERS

| | |
|---|---|
| Radio bandwidth | 100Kbps |
| Radio Transmission Range | 250 m |
| Radio Interference Range | 550 m |
| Packet Length | 100 bytes |
| Transmit Power | 0.66W |
| Receive Power | 0.395W |
| Idle Power | 0.35W |

## IV. PERFORMANCE EVALUATION

We implemented our prototype in the ns-2 network simulator with the CMU wireless extension. For comparison, we also implement a simple version of SMAC with adaptive listening, but without its synchronization and message passing scheme. We will also compare with a full active CSMA/CA MAC without periodical sleep schedule. This will serve as the baseline of latency, energy and throughput performance.

We choose 3 metrics to evaluate the performance of DMAC: **Energy Cost** is the total energy cost to deliver a certain number of packets from sources to sink. This metric shows the energy efficiency of the MAC protocols. **Latency** is the end to end delay of a packet. **Throughput or Delivery ratio** is the ratio of the number of packets arrived at the sink to the number of packet sent by sources.

The radio characteristics are shown in Table I. The energy costs of the Tx:Rx:Idle radio modes is about 1.67:1:0.88 [2]. The sleeping power consumption is set to 0 [3]. A MTS packet is 3 bytes long.

According to the parameters of the radio and packet length, the receiving and sending slot $\mu$ is set to $10ms$ for DMAC and $11ms$ for DMAC/MTS. The active period is set to $20ms$ for SMAC with adaptive listening. All schemes have the basic duty cycle of 10%. This means a sleep period of $180ms$ for DMAC and SMAC, $198ms$ for DMAC/MTS.

All simulations are run independently under 5 different seeds. All sources generate packets at constant averaged rate with 50% randomization in inter-packet interval.

[2]The power consumption numbers are chosen according to the default values in ns-2. Although not based on real radio in sensor node, the ratio of the Tx, Rx and Idle power is typical value and is sufficient to show the energy efficiency performance.

[3]The sleep power of real radio is not 0. However, in our simulations where each run lasts less than 100 second, the sleep power consumption is negligible.

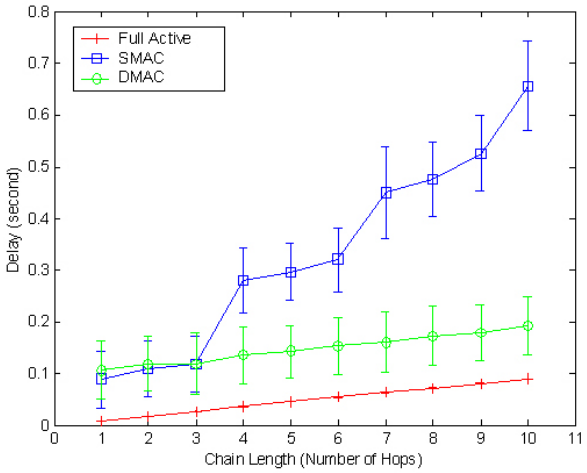


Fig. 6. Mean packet latency on each hop under low traffic load.

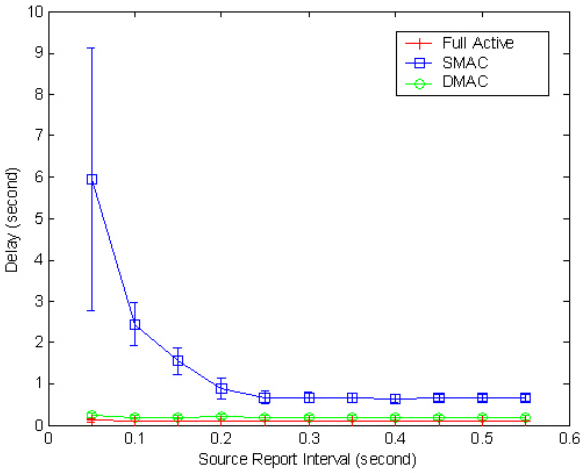


Fig. 8. Mean packet latency for 10 hops chain under different source report interval.

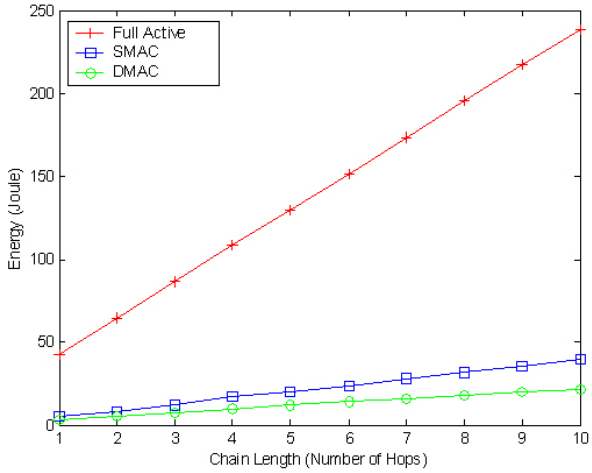


Fig. 7. Total energy consumption on each hop under low traffic load.

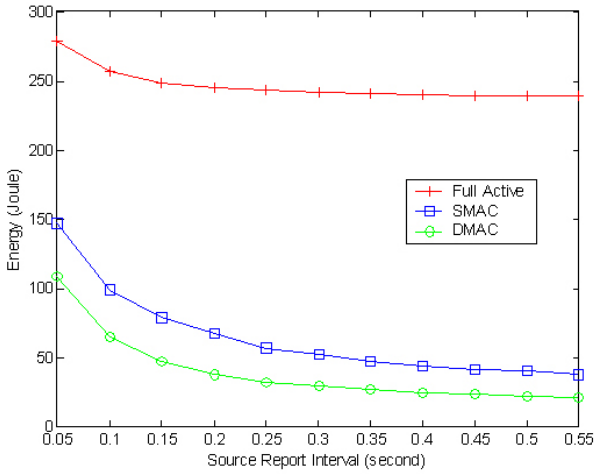


Fig. 9. Energy consumption for 10 hops chain under different source report interval.

### A. *Multihop chain*

To reveal the fundamental performance of DMAC, we first performed a test on a simple multihop chain topology with 11 nodes. The distance between adjacent nodes is 200 meters. First in order to show the capability of reducing the sleep delay in DMAC, we measure the end-to-end latency of packets under very light traffic rate of source report interval $0.5s$. In this light traffic load, there is no queuing delay but only a sleep delay that is caused by periodic sleep.

Figure 6 shows the averaged packet latency with different hop length. In both DMAC and full active CSMA/CA, the latency increases linearly with the number of hops with almost the same slop. The additional latency of DMAC is at the source when a sensor reading occurs during the sleep period and has to wait until the node wakes up. The SMAC protocol with adaptive listening, however, has higher latency. In particular, the latency sees a "jump" every 3 hops. SMAC can forward a packet 2 hops in $20ms$ active period. With adaptive listening, a packet can be forwarded three hops instead of two hops without adaptive listening. However the packet has to queued for a scheduled interval for the fourth hop. This is shown clearly in the figure.

Figure 7 shows the energy cost with different hop length. In all MAC protocol, the energy cost increase linearly with the number of hops. However, the energy

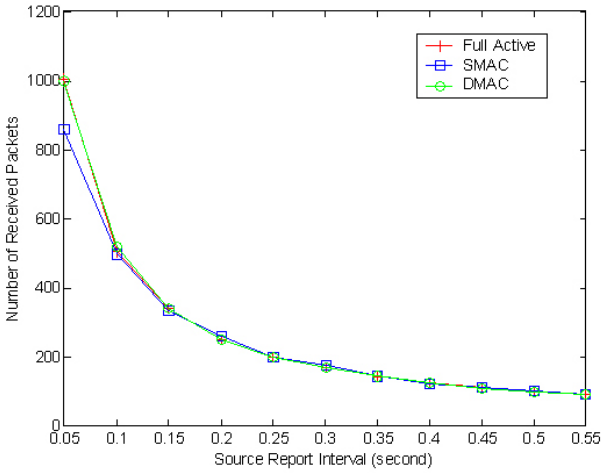


Fig. 10. Throughput for 10 hops chain under different source report interval.

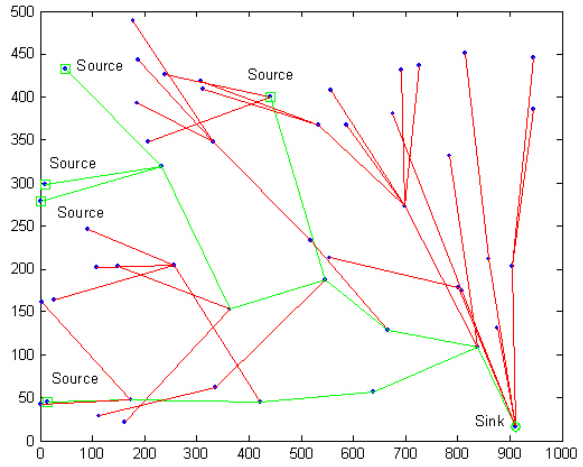


Fig. 11. A random data gathering tree.

cost of the full active CSMA/CA increases much faster than other two MAC protocols. DMAC consumes less energy cost than SMAC. This is due to the additional active period in SMAC for nodes that are not the next hop of a data packet (but are within overhearing range).

We then test the traffic adaptation of these MAC protocols, by varying the sensor report interval on the source node from 0.05s to 0.55s. The hop length is fixed at 10 hops.

Figure 8 shows the averaged packet latency for different source report intervals. Clearly, full active CSMA/CA has the lowest latency. DMAC has a slightly higher latency due to the initial latency at the source. SMAC, however, has much higher latency, especially when traffic load is heavy (e.g. at small source report interval). The reason is that since packets can be forwarded only three hops every interval, packets suffer from both sleep delay and queuing delay. When traffic load is very high, collisions would significantly increase packet latency as a retransmission can only be done after one total schedule interval. When source report is less than 0.05s, the traffic load will be more than 80% of the maximum channel capacity. Only full active CSMA/CA can handle such a high traffic load.

Figure 9 shows the total energy cost for different source report intervals. Energy cost decreases as traffic load decreases. For full active CSMA/CA, however, the decrease is small since without radio off, the idle listening still consume significant energy. SMAC has a higher energy cost than DMAC due to that nodes other than next hop of a data packet remain active unnecessarily.

Figure 10 shows the throughput achieved for different MACs. All MAC have quite good data delivery ratio near 1 under the simple multihop chain topology.

### *B. Random Data gathering Tree*

In this topology, 50 nodes are distributed randomly in a $1000m \times 500m$ areas shown in Figure 11. The sink node is at the right bottom corner. A data gathering tree is constructed by each node choosing from its neighbor the node closest to the sink as its next hop. In order to show the different packet latency, a source should be at least 3 hops away from the sink. Five nodes at the margin are chosen as sources to testify the mechanism of data prediction and MTS. All sources generate reports at the same rate.

Packet latency under different source report intervals is shown in Figures 12. Full active CSMA/CA has small delay for all traffic load. However, other three MACs' latency increases significantly when the traffic load is larger than a certain threshold. DMAC/MTS can handle the highest traffic load with small delay among the three MACs with periodical sleep. Compared to the multihop chain under the same heavy traffic load, the latency in a data gathering tree is much higher. This is due to the interference between nodes in the same depth of the tree. The interference could result in data loss, schedule inconsistency and MTS packet loss which increase the sleep latency.

Figure 13, 14 shows the energy and throughput performance. We collect the energy costs of all the 50 nodes in the network because potentially a MAC could cause

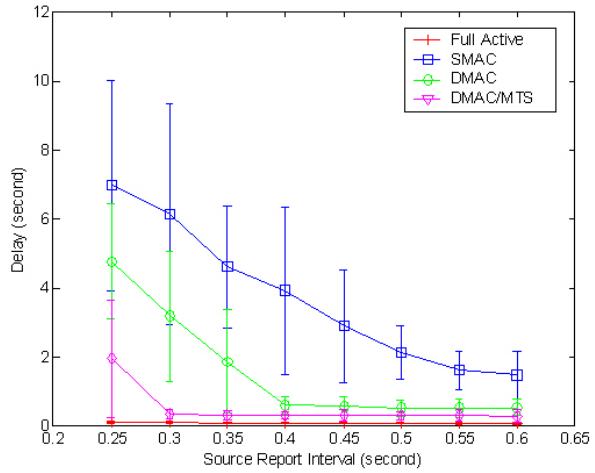


Fig. 12. Mean packet latency for a data gathering tree under different traffic load.

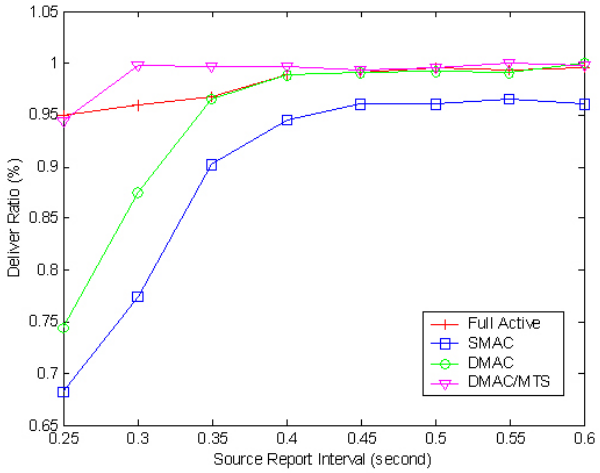


Fig. 14. Data delivery ratio for a data gathering tree under different traffic load.

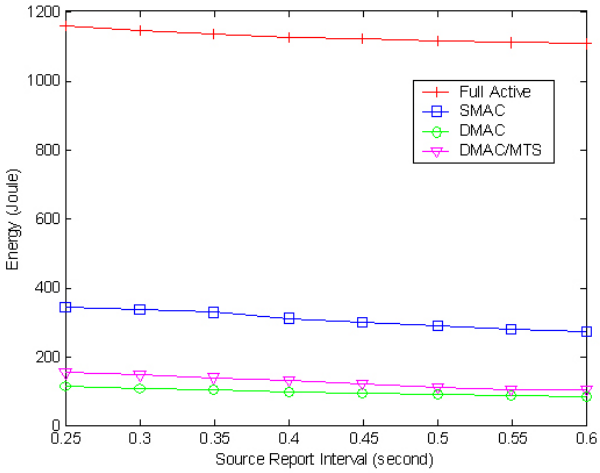


Fig. 13. Energy consumption for a data gathering tree under different traffic load.

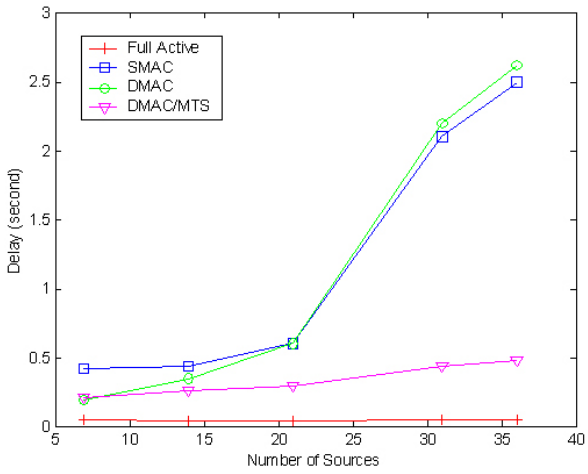


Fig. 15. Mean packet latency for data gathering different source number.

unrelated nodes to maintain a higher duty cycle. It is shown in the figure that DMAC and DMAC/MTS are the two most energy-efficient MAC protocols. DMAC/MTS, however, consumes higher energy than DMAC because of the overhead of MTS packets and more active period requested by MTS packets. In terms of end-to-end throughput, DMAC/MTS has a good delivery ratio while SMAC and DMAC's delivery ratio decreases when traffic load is heavy.

We further evaluate the scalability of DMAC under a dense network, in which 100 nodes are randomly placed in a $100m \times 500m$ area. A data gathering tree is constructed rooted at the sink on the right bottom corner. All sources generate traffic at one message per 3 seconds. We vary the number of sources which are chosen randomly from the margin nodes in the network.

Figure 15 shows the averaged delay under different number of sources. As source number increases, interference increases which results in increased latency for SMAC and DMAC without MTS. DMAC/MTS, however, can still maintain quite low latency. This low latency is achieved at very small overhead in energy compared to DMAC without MTS, which is shown in figure 16. DMAC/MTS also has the second delivery ratio next to full active CSMA. This clearly shows the effectiveness of DMAC/MTS.

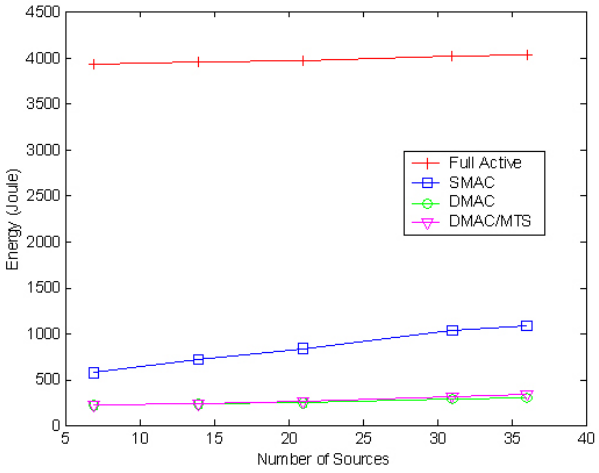


Fig. 16. Energy consumption for data gathering different source number.

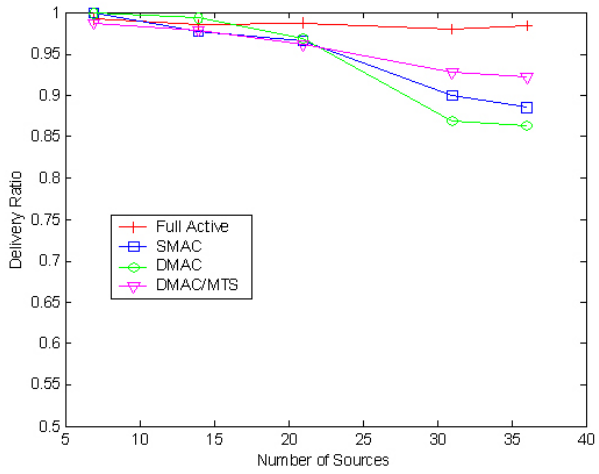


Fig. 17. Data delivery ratio for data gathering with different source number.

### C. Discussion

To understand the trade off between energy, throughput and latency, Figure 18 shows the number of packets that can be sent per unit resource measured in terms of $Energy \times Latency$ for the scenario in Figure 11, as a function of the traffic load. From the figure, we see that because SMAC achieves energy efficient at the sacrifice of latency, it sends the least number of packets per $Joule - second$. This suggests that for applications that can tolerate message latency, SMAC is a reasonable solution. But for applications that require real-time data delivery, SMAC is not feasible due to the data forwarding interruption problem. DMAC and DMAC/MTS, however, can achieve both energy efficiency and low message latency. DMAC/MTS can operate with even smaller base duty cycle to save more energy when traffic is light and can still adapt to traffic bursts with high throughput, low latency and small energy consumption. However, this figure also shows that when traffic load exceeds a certain threshold, a full active MAC is most suitable when taking both energy and delay into account.

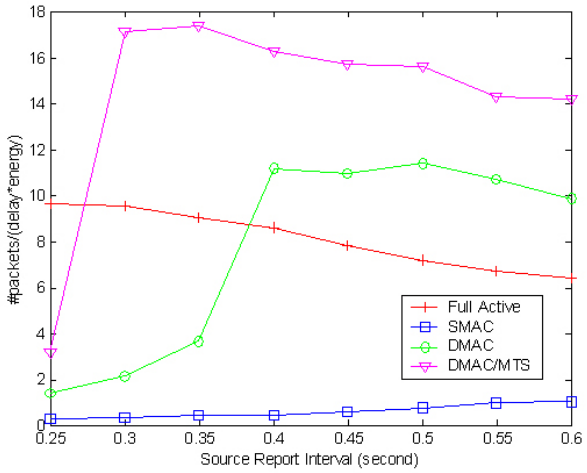


Fig. 18. Trade off among energy, latency and throughput for a data gathering tree under different traffic load.

Since DMAC can adjust duty cycle to traffic load with small latency, we can set the basic duty cycle even smaller. But a lower duty cycle could have longer initial sleep delay at the source node when a sensing reading occurs during the source's radio is off. So there is a limitation on lowest basic duty cycle DMAC can operate on. However, with the same application latency bound requirement, DMAC can operate on a lower basic duty cycle than SMAC or TMAC to be more energy efficient.

Finally, we should note that this comparison between DMAC and SMAC is only applicable under the specific data gathering tree scenario for unidirectional communication flow from multiple sources to a single sink. SMAC is in fact a general-purpose energy-efficient MAC that can handle simultaneous data transmissions and flows between arbitrary source and destination. For applications that require data exchange between arbitrary sensor nodes, DMAC cannot be used while SMAC will be a good choice.

## V. Conclusion and Future Work

This paper has proposed DMAC, an energy efficient and low latency MAC protocol for tree-based data gathering in wireless sensor networks. The major traffic in

wireless sensor networks are from sensor nodes to a sink which construct a data gathering tree. DMAC utilizes this data gathering tree structure specific to sensor network applications to achieve both energy efficiency and low packet delivery latency. DMAC staggers the active/sleep schedule of the nodes in the data gathering tree according to its depth in the tree to allow continuous packet forwarding flow in which all nodes on the multihop path can be notified of the data delivery in progress and duty cycle adjustment command.

Data prediction is employed to solve the problem when each single source has low traffic rate but the aggregated rate at an intermediate node is larger than the basic duty cycle can handle. The interference between nodes with different parents could cause one traffic flow be interrupted because the nodes on the multihop path is not notified of the data transmission requirement. The use of an MTS packet is proposed to command nodes on the multihop path to remain active when a node fails to send a packet to its parent due to interference.

Our simulation results have shown that DMAC achieves both energy savings and low latency when used with data gathering trees in wireless sensor networks. In our future work, we aim to implement this MAC on a Mote-based sensor network platform and evaluate its performance through real experiments.

## VI. Acknowledgments

We would like to thank Marco Zuniga from USC, Dr. Wei Ye from USC-ISI, and Prof. Koen Langendoen and Gertjan Halkes from TUDelft for their helpful feedback and for providing useful references.

## References

[1] A. Woo, D. Culler, “A Transmission Control Scheme for Media Access in Sensor Networks”, in *Mobicom*, July 2001.

[2] W. Ye, J. Heidemann, and D. Estrin, “Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks”, in *IEEE/ACM Transaction on Networking*, To Appear.

[3] J. Li, C. Blake, D. Couto, H. Lee and R. Morris, “Capacity of Ad Hoc Wireless Networks”, in *ACM Mobicom* July 2001

[4] Tijs van Dam, Koen Langendoen, “An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks”, in *ACM Sensys* Nov. 2003

[5] Rong Zheng, Robin Kravets, “On-demand Power Management for Ad Hoc Networks”, in *IEEE Infocom* 2003

[6] Jeremy Elson, Lewis Girod and Deborah Estrin, “Find-Grained Network Time Synchronization using Reference Broadcasts”, in *ACM SIGOPS* 2002

[7] Rong Zheng, Jennifer C. Hou and Lui Sha, “Asynchronous Wakeup For Ad Hoc Networks”, in *ACM MobiHoc* 2003

[8] Eun-Sun Jung, Nitin H. Vaidya, “An Energy Efficient MAC Protocol for Wireless LANs”, in *IEEE Infocom* 2002

[9] Brad Karp, H. T. Kung, “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks”, in *ACM MobiCom* 2000

[10] Chalermek, Ramesh Govindan, Deborah Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks”, in *MobiCom* 2002

[11] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, “Energy-aware wireless microsensor networks”, in *IEEE Signal Processing Magazine* 2002

[12] Yuan Li, Wei Ye, John Heidemann “Schedule and Latency Control in S-MAC”, Poster, in *UCLA CENS research review* 2003

[13] A. El-Hoiydi, J. D. Decotignie, C. Enz and E. Le Roux, “WiseMAC: an Ultra Low Power MAC Protocol for the WiseNET Wireless Sensor Network”, Poster, in *ACM Sensys* 2003

[14] B. Krishnamachari, D. Estrin and S. Wicker, “The impact of data aggregation in wireless sensor networks”, in *International Workshop on Distributed Event-based Systems*, 2002

[15] C. S. Raghavendra and S. Singh, “PAMAS-power aware multi-access protocol with signaling for ad hoc networks”, in *Computer Communication Revies*, 1998

[16] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, ”MACAW: A Media Access Protocol for Wireless LAN’s” in *ACM SIGCOMM*, 1994

[17] Huang, Y., Huang, J., Hester, L., Allen, A., Andric, O., Chen, P., O’Dea, B., “OPNET Simulation Of A Multi-hop Self-organizing Wireless Sensor Network”, in *Proceedings of OPNETWORK2002 conference*, Washington D.C., August 2002.