An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless 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 not all nodes on a multihop path to the sink are notified of data delivery in progress, resulting in significant sleep delay. DMAC is designed to solve the interruption problem and allow continuous packet forwarding by giving the sleep schedule of a node an offset that depends upon its depth on the tree. DMAC also adjusts the duty cycles adaptively according to the traffic load in the network. 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.

1 Introduction

Wireless sensor networks (WSN) are expected to be used in a wide range of applications, such as target tracking, habitat sensing and fire detection. Typically in WSNs, nodes coordinate locally to perform 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 sensor nodes. Energy efficiency is a critical issue in order to prolong network lifetime. In particular MAC protocols must minimize the radio energy costs in sensor nodes.

Latency: Latency requirements depend on the application. In surveillance applications, an event detected needs to be reported to a 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. In such applications, the more data the sink receives the better. In other applications, such as fire detection, it may suffice for a single report to arrive at the sink.

Fairness: In many applications, particularly when bandwidth is scarce, it is important to ensure that the sink receives information from all sources in a fair manner. While we do not explicitly consider fairness issues in this paper, adaptive techniques such as those 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], [13]) have identified idle listening as a major source of energy wastage. As traffic load in many sensor network applications 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. Consider a sensor node with 1 report per second at 100 bytes per packet — data transmission takes only 8ms for a 100Kbps radio, 992 ms are wasted in idle listening between reports. S-MAC [2] provides a tunable periodic active/sleep cycle for sensor nodes. 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 still has three shortcomings. First, it increases the packet delivery latency. An intermediate node may have to wait

Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

until the receiver wakes up before it can forward a packet. This is called *sleep latency* in SMAC [2]. The sleep latency increases proportionally with respect to number of hops, with the constant of proportionality being the duration of a single cycle (active period plus sleep period). Secondly, a fixed duty cycle does not adapt to the traffic variation 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 delivery and long queuing delay. 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.

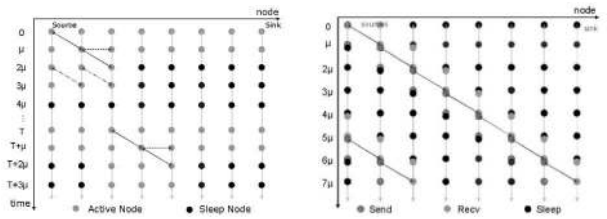
There are several works on reducing sleep delay and adjusting duty cycle to the traffic load. Those mechanisms are either implicit (e.g. [2], [4]), in which nodes remain active when they overhear ongoing transmissions in the neighborhood; or they are explicit (e.g. [7]), in which there are direct duty cycle adjustment messages. In the adaptive listening scheme proposed in SMAC [2], 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, etc. The authors in [7] proposed a slot-based power management mechanism. If the number of buffered packets for an intended receiver exceeds a threshold, the sender signals the receiver to remain on for the next slot. The receiver sends back an acknowledgement, indicating its willingness to remain awake in the next slot. The sender can then send packets in the following slot.

In all these previously proposed mechanisms, nodes on the path to the sink that are more than one or two hops away from the receiver cannot be notified of the ongoing traffic, and therefore packet forwarding will stop after a few hops. As we shall describe in section 2, this *data forwarding interruption problem* causes significant sleep latency for packet delivery.

The protocol that we propose in this paper, 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.

2 Data Forwarding Interruption Problem

The data forwarding interruption problem (DFI) exists in implicit adaptive duty-cycle techniques because the overhearing range is limited by the radio sensitivity. 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.



(a) SMAC (b) DMAC

Figure 1. (a)DFI causes sleep delay. (b)DMAC reduce sleep delay.

Assume the active period in each cycle 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 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 cycle duration. Therefore, compared with the case of no adaptive listening, the delay is only reduced by half. Meanwhile, neighboring 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], since a node remains active if it senses any communication on the air, 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 unnecessarily. Meanwhile nodes out of the interference range on the multi-hop path still go to sleep after their basic active period, causing the data forwarding interruption problem. The FRTS proposed in TMAC can increase the number of

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

packets delivered in one frame and as a side effect, can help forward a packet only one hop further. The same problem happens in the scheme proposed in [7], in which the request for a next active slot can only be received by the next hop. The nodes beyond that will still go to sleep after their basic active period.

Figure 1(a) 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 only reduced by half. If both node 0 and node 1 need to transmit packets, collision may happen.

The hearing/interference range is not a useful tunable parameter because it results in an undesirable energy-latency tradeoff. If the hearing range is large, latency is reduced since more nodes on the path can overhear the communication and remain active. However, if the hearing range is large, more nodes that are not on the path also overhear the communication and waste energy in idle listening. We need a MAC that can tell all the nodes on the path to stay active and/or increase their duty cycles and all other nearby nodes to sleep.

3 DMAC Protocol Design

3.1 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, a *data gathering tree* [12]. 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 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 μ 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.

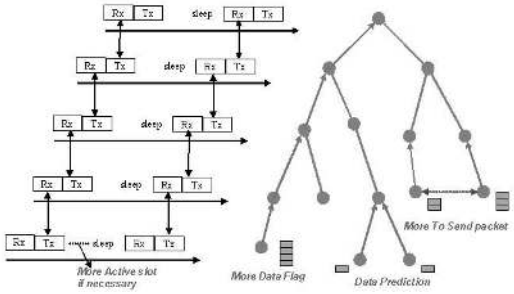


Figure 2. DMAC in a data gathering tree.

A staggered wake-up schedule has four advantages. First, since nodes on the path wake up sequentially to forward a packet to next hop, so sleep delay is reduced. 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. 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 DMAC, RTS/CTS control packets are not employed because as they would add unnecessary overhead given the relatively small packet size in most sensor applications. However, link layer ARQ through ACK packet and data retransmission are necessary to recover lost packet due to harsh quality wireless channel and contention. A sending node will queue the packet until next sending slot in case

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

no ACK packet received. After a fixed typically small number of retransmissions (e.g. 3), the packet will be dropped.

To reduce collision during the Tx period of nodes on the same tree level, every node backs off for a period (BP) plus a random time within a contention window before packet transmission. We employ a fixed contention window since the length of a sending slot is only enough for one packet transmission. 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 μ 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.

Local synchronization is needed in DMAC since a node needs to be aware of its neighbors' schedule. There exist techniques such as the reference broadcast synchronization scheme (RBS)[6] that can achieve time synchronization precision less than $10\mu sec$ even for multiple hops. Given that typical slot lengths are on the order of $10ms$ in length, we will assume that sufficiently fine-grained synchronization is available in the following discussions.

We should mention that ongoing work to improve SMAC [11] 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.

3.2 Data Delivery and Duty Cycle Adaptation in Multihop chain

Figure 1(b) 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 period with little overhead. 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 checks 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 this slot-by-slot mechanism, DMAC can react quickly to traffic rate variations to be both energy efficient and to maintain low data delivery latency.

A node will decide to hold an additional active period if either it sends a packet with the *more data* flag set and receives back an ACK packet with the *more data* flag set, or if 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 3μ sleep then goes to the receiving state as shown in Figure 1. The reason is that it knows the following nodes on the multihop path will forward the packet in the next 3 slots. It is shown in [3] 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. To accommodate the possibility of shorter range between two neighbor nodes, in DMAC a node will only send one packet every 5μ 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.

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.

Measurements have shown that the cost for switching radio between active and sleep is not free. However, the overhead of this switching is likely to be small [10] compared to energy savings in a 3μ sleep period of around $30ms$.

3.3 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

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

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

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 have only one packet to send in every interval. At the sending slot of an interval, only one child can win the channel and send a packet to node C. Assume A wins the channel and sends a packet to C. Since A's buffer is now 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 anticipates that its children still have packets waiting for transmission. It then sleeps only 3μ 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 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 3μ later after the current 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 3μ later, so it will also wake up 3μ later after its sending slot. In this additional sending slot, the node then can transmit a packet to its parent.

There is an overhead entailed 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 wastes energy in idle listening. Compared to the potentially great latency reduction by the *data prediction*, we believe this additional overhead would be worthwhile.

3.4 More-To-Send Packet

Although a node will sleep 3μ before an additional active period to avoid collision, there is still a chance of interference between nodes on different branches of the tree. Assume 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.

Table 1. 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

A node sends a request MTS to its parent if either of these two conditions is true. First 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 assume it lost the channel because of interference from other nodes. Second 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: Its buffer is empty, all request MTSs received from children are cleared and it sends a request MTS to its parent before and has not sent a clear MTS.

A node which sends or received a request MTS will keep waking up periodically every 3μ. 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.

Just as in the slot-by-slot renewal scheme and data prediction scheme, the duty cycle adjustment request by MTS packets is 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 [2]

Since the MTS packet is very short, the increase in slot length would be small. Energy consumption also increases because of the overhead of MTS packets and the longer slot. In the simulation section, however, we will show that the use of MTS can significantly reduce latency particularly in a sensor network with multiple sources, with a minimal additional energy cost.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

4 Performance Evaluation

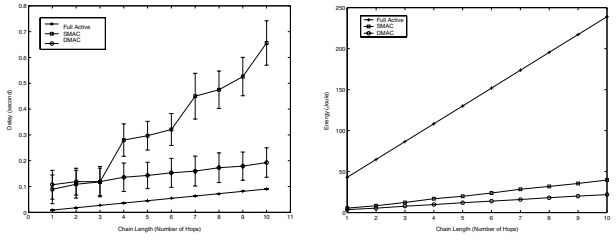
We implemented our prototype in the ns-2 network simulator with the CMU wireless extension. For comparison, we implemented a simple version of SMAC with adaptive listening (but without its synchronization and message passing scheme) and also used a full active CSMA/CA MAC without periodical sleep schedule.

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. **Latency** is the end to end delay of a packet. **Delivery ratio** is the ratio of the successfully delivered packets to the total packets originating from all sources.

The radio characteristics are shown in Table 1. The relative energy costs of the Tx:Rx:Idle radio modes are assumed to be about 1.67:1:0.88 [3]. The sleeping power consumption is set to 0 (i.e. considered negligible). An MTS packet is 3 bytes long. According to the parameters of the radio and packet length, the receiving and sending slot μ 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.

4.1 Multihop chain



(a) latency (b) energy

Figure 3. Packet latency and energy on a chain topology.

To study the performance of DMAC on a more realistic scenario, 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 $0.5s$. There is no queuing delay other than the sleep delay caused by periodic sleep.

Figure 3 shows the simulation results under different hop lengths. The latencies of both DMAC and full active CSMA/CA increase linearly with the number of hops. The SMAC protocol with adaptive listening, however, has higher latency. In particular, the latency sees a "jump" every 3 hops. This is because SMAC can forward a packet 2 hops in $20ms$ active period. With adaptive listening, a packet can be forwarded one more hop then queued for a scheduled interval for the fourth hop. The energy costs in all MACs increase linearly with the number of hops. DMAC consumes less energy cost than SMAC because of the additional active period in SMAC for nodes that are not the next hop of a data packet (but are within overhearing range).

4.2 Random Data gathering Tree

In this topology, 50 nodes are distributed randomly in a $1000m \times 500m$ area. The sink node is at the right bottom corner. A data gathering tree is constructed by each node choosing from its neighbors the node closest to the sink as its next hop. Five nodes at the margin are chosen as sources to test the mechanisms of data prediction and MTS. All sources generate reports at the same rate.

Simulation results are shown in Figures 4. Full active CSMA/CA has the smallest delay for all traffic load, other three MACs' latencies increase significantly when the traffic load is larger than a certain threshold. Among them, DMAC/MTS can handle the highest traffic load with the smallest delay. However, the interference between nodes in the same depth of the tree could result in data loss, schedule inconsistency and MTS packet loss which increase the sleep latency. Also shown in the figure is that DMAC and DMAC/MTS are the two most energy-efficient MAC protocols [4]. DMAC/MTS, however, consumes higher energy than DMAC because of the overhead of MTS packets and more active period requested by MTS packets. DMAC/MTS also achieves a better delivery ratio while SMAC and DMAC's delivery ratio decreases when traffic load is heavy.

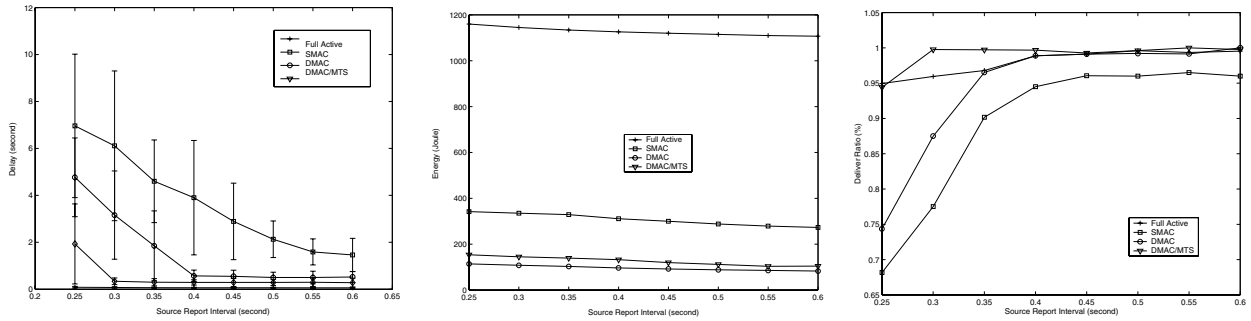
We further evaluate the scalability of DMAC under a dense network, in which 100 nodes are randomly placed in a $100m \times 500m$ area. All sources generate traffic at one message per 3 seconds. We vary the number of sources which are chosen randomly from the margin nodes in the network.

[2] Loss of clear packets may result in wasted active slots — this can be mitigated by maintaining a soft timer to ignore the current request MTS if no data is received or transmitted after a certain number of receiving slots.

[3] The power consumption numbers are chosen from the default values in ns-2.

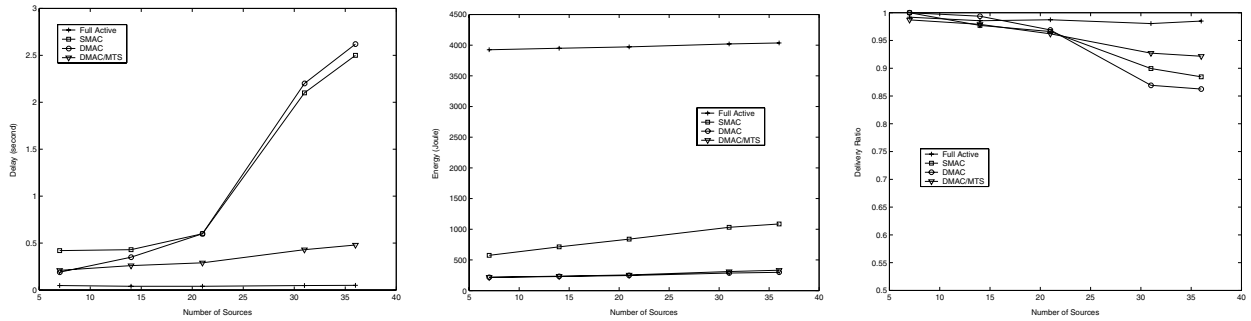
[4] We collect the energy costs of all the 50 nodes in the network because potentially a MAC could cause unrelated nodes to maintain a higher duty cycle.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE



(a) latency (b) energy (c) delivery ratio

Figure 4. Packet latency, energy and delivery ratio under different traffic loads on a tree topology



(a) latency (b) energy (c) delivery ratio

Figure 5. Packet latency, energy and delivery ratio with different numbers of sources on a tree topology.

Figure 5 shows the performance 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 a low latency. This low latency is achieved at very small overhead in energy compared to DMAC without MTS. DMAC/MTS also has the second delivery ratio next to full active CSMA. This clearly shows the effectiveness of DMAC/MTS.

4.3 Discussion

To understand the trade off between energy, throughput and latency, Figure 6 shows the number of packets that can be sent per unit resource measured in terms of $Energy \times Latency$ for the scenario in Figure 4, as a function of the traffic load. From the figure, we see that SMAC achieves energy efficiency at the sacrifice of latency, as it shows the least number of packets per $Joule-second$. This suggests that SMAC may not be well-suited to tree-based applications that require real-time data delivery. DMAC, however, can achieve both energy efficiency and low message latency. DMAC can operate with a 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 [5]. 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.

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.

[5] 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.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

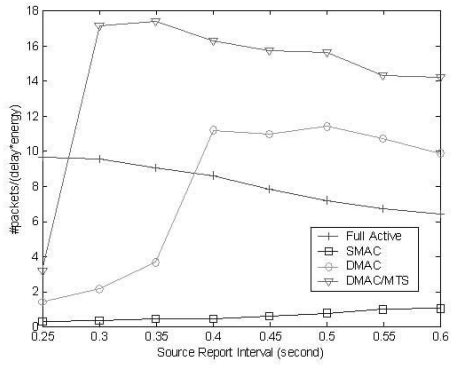


Figure 6. Trade off among energy, latency and throughput.

5 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 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. This allows continuous packet forwarding flow in which all nodes on the multihop path can be notified of the data delivery in progress as well as any duty-cycle adjustments.

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 what the basic duty cycle can handle. The interference between nodes with different parents could cause a traffic flow be interrupted because the nodes on the multihop path may not be aware of the interference. 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.

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

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

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

[12] 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

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

0-7695-2132-0/04/$17.00 (C) 2004 IEEE