Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks [*]

Venkatesh Rajendran
venkat@soe.ucsc.edu

Katia Obraczka
katia@soe.ucsc.edu

J.J. Garcia-Luna-Aceves
jj@soe.ucsc.edu

Department of Computer Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064

ABSTRACT

The traffic-adaptive medium access protocol (TRAMA) is introduced for energy-efficient collision-free channel access in wireless sensor networks. TRAMA reduces energy consumption by ensuring that unicast, multicast, and broadcast transmissions have no collisions, and by allowing nodes to switch to a low-power, idle state whenever they are not transmitting or receiving. TRAMA assumes that time is slotted and uses a distributed election scheme based on information about the traffic at each node to determine which node can transmit at a particular time slot. TRAMA avoids the assignment of time slots to nodes with no traffic to send, and also allows nodes to determine when they can become idle and not listen to the channel using traffic information. TRAMA is shown to be fair and correct, in that no idle node is an intended receiver and no receiver suffers collisions. The performance of TRAMA is evaluated through extensive simulations using both synthetic- as well as sensor-network scenarios. The results indicate that TRAMA outperforms contention-based protocols (e.g., CSMA, 802.11 and S-MAC) as well as scheduling-based protocols (e.g., NAMA) with significant energy savings.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication; C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithm, Design, Performance

[*]This work was supported in part by the U.S. Air Force/OSR under grant No. F49620-00-1-0330, by the NSF-NGI under grant No. ANI-0196418, and by the Jack Baskin Chair of Computer Engineering.

Keywords

Sensor networks, energy-efficient scheduling, traffic-adaptive medium access, ad hoc networks

1. INTRODUCTION

Sensor networks typically refer to large ensembles of interconnected nodes that have processing and communication capabilities, and one or more sensing devices (e.g., thermistors, magnetometers, light detectors). The deployment of such networks is usually done in an ad-hoc manner (e.g., dropping sensors from an aircraft on the field) which implies that sensor-network nodes need to self-organize into a multi-hop wireless network. Many of the large-scale sensor networks of the future will consist of battery-powered sensor nodes whose battery may be difficult to recharge, or that the nodes themselves may be so cheap that recharging them may not be cost effective.

As the hardware for sensor nodes has become affordable, sensor networks have emerged as an ideal solution to a number of applications in both civilian and military scenarios, including monitoring and surveillance of large, remote or inaccessible areas over extended periods of time. However, a major challenge facing the development and eventual deployment of large-scale sensor networks is the scheduling of transmissions among nodes in a way that (a) is self adaptive to changes in traffic, node state, or connectivity; and (b) prolongs the battery life of each node.

Background

There is an extensive body of work on MAC (medium access control) protocols for multihop wireless networks, dating back to the DARPA packet radio program (e.g., [16, 17, 18, 7, 8, 11]). These MAC protocols can be categorized as contention- and schedule-based.

The best-known example of contention-based protocols is the distributed coordination function (DCF) of the IEEE 802.11b standard [13], which uses the carrier sense multiple access (CSMA) technique combined with a four-way handshake that attempts to avoid collisions of data packets. In terms of energy consumption, a key limitation of traditional contention-based schemes is that nodes consume energy needlessly when they are idle (i.e., not transmitting or receiving) as well as when collisions occur. Until recently, very little work has been reported on contention-based schemes that focus on energy efficiency.

PAMAS [20] is one of the earliest contention-based proposals to address power efficiency in channel access. PAMAS saves energy by attempting to avoid over-hearing among neighboring nodes. To achieve this, PAMAS uses out-of channel signaling. Woo and Culler [23] address variations of CSMA tailored for sensor networks, and propose an adaptive rate control mechanism to achieve fair bandwidth allocation among sensor network nodes. In the power save (PS) mode in IEEE 802.11 DCF, nodes sleep periodically. Tseng *et al.* [22] investigated three sleep modalities in 802.11 DCF in multi-hop networks. The sensor-MAC protocol [24], or S-MAC, exhibits similar functionality to that of PAMAS and the protocol by Tseng *et al.*. Like the other approaches, S-MAC avoids overhearing and nodes periodically sleep. However, unlike PAMAS, S-MAC uses in-line signaling, and unlike modalities of the PC mode in 802.11 DCF, neighboring nodes can synchronize their sleep schedules.

The probability of collisions of control or data packets in any contention-based scheme increases with the offered load, which degrades channel utilization and further reduces battery life. This motivates the need for establishing transmission schedules statically or dynamically to allow nodes to receive data packets without collisions. The transmission schedule established in a wireless network can be topology independent or topology dependent [6, 15, 19, 5]. The scheduled-access MAC protocol described by Sohrabi and Pottie [21] uses a combination of TDMA and FDMA or CDMA for accessing the channel. The main drawback of this scheme is that, like most fixed scheduling mechanisms, time slots are wasted if a node does not have any data to send to the intended receiver.

The Node Activation Multiple Access (NAMA) [5] uses a distributed election algorithm to achieve collision-free transmissions. For each time slot, NAMA selects only one transmitter per two-hop neighborhood and hence all nodes in the one-hop neighborhood of the transmitter are able to receive data collision-free. However, NAMA does not address energy conservation. In fact, very few proposals, if any, implement energy-aware medium access scheduling.

Focus

Section 2 introduces the TRaffic-Adaptive Medium Access (TRAMA) protocol, which provides energy-efficient conflict-free channel access in wireless sensor networks. Channel access in TRAMA is energy efficient while maintaining good throughput, acceptable latencies, and fairness. Energy efficiency is attained by (1) transmission schedules that avoid collisions of data packets at the receivers, and (2) having nodes switch to low power radio mode when there is no data packets intended for those nodes. Adequate throughput and fairness is achieved by means of a transmitter-election algorithm that is inherently fair and promotes channel reuse as a function of the competing traffic around any given source or receiver. TRAMA derives collision-free transmission schedules based on the identifiers of nodes one and two hops away, the current time slot, and traffic information that specifies which node intends to transmit to which other node. Hence, the "sleep schedule" of a node is a direct function of the traffic going through the node and its neighbors, and is synchronized automatically when nodes exchange information about their identifiers and their traffic.

TRAMA is similar to the Node Activation Multiple Access (NAMA) protocol [5], in that the identifiers of the nodes within a two-hop neighborhood are used to give conflict-free access to the channel to a given node during a particular time slot. However, NAMA does not address energy efficiency, and nodes that are not transmitting switch to receiver mode. In contrast, TRAMA addresses energy efficiency by having nodes going into sleep mode if they are not selected to transmit and are not the intended receivers of traffic during a particular time slot. Furthermore TRAMA uses traffic information to influence the schedules, which makes TRAMA far more adaptive to the sensor network application at hand. For instance, an event-tracking application will likely generate data only when an event is detected. On the other hand, monitoring applications may generate data continuously. In either case, TRAMA can adapt its schedules accordingly, delivering adequate performance and energy efficiency.

In contrast to prior MAC protocols proposed for sensor networks, TRAMA provides support for unicast, broadcast *and* multicast traffic (i.e., transmitting to only a set of one-hop neighbors). TRAMA differs from S-MAC (which also provides explicit energy conservation mechanisms) in two fundamental ways: (1) TRAMA is inherently collision-free as its medium access control mechanism is schedule-based as opposed to S-MAC's which is contention-based; and (2) TRAMA's uses an adaptive, dynamic approach based on current traffic patterns to switch nodes to low power mode, while S-MAC's scheme is static based on a pre-defined duty cycle.

In Section 3, we discuss why TRAMA is fair and provides transmission schedules in which no collisions, idle listening, and idle senders occur.

We evaluate the performance of TRAMA through extensive simulations using the Qualnet network simulator [3]. We compare the performance of TRAMA against three contention-based MAC protocols, namely CSMA, 802.11's DCF, and S-MAC, and also against NAMA, which is a good example of collision-free channel access based on dynamic schedules. Section 4 describes our simulation setup, and Section 5 presents simulation results. Our simulation results show that TRAMA exhibits superior end-to-end throughput (around 40% over S-MAC and CSMA and around 20% for 802.11) for both synthetic traffic models and traffic models that are sensor-network specific, because it avoids collisions due to hidden terminals. Our results also show that TRAMA achieves significant energy savings (since nodes can sleep for up to 87% of the time) and higher throughput. Section 6 presents concluding remarks.

2. TRAMA

2.1 Protocol Overview

TRAMA employs a traffic adaptive distributed election scheme that selects receivers based on schedules announced by transmitters. Nodes using TRAMA exchange their two-hop neighborhood information and the transmission schedules specifying which nodes are the intended receivers of their traffic in chronological order, and then select the nodes that should transmit and receive during each time slot. Accordingly, TRAMA consists of three components: the Neighbor Protocol (NP) and the Schedule Exchange Protocol (SEP), which allow nodes to exchange two-hop neighbor information and their schedules; and the Adaptive Election Algorithm (AEA), which uses neighborhood and schedule

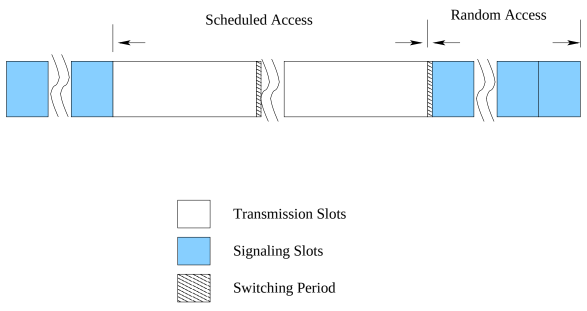


Figure 1: Time slot organization

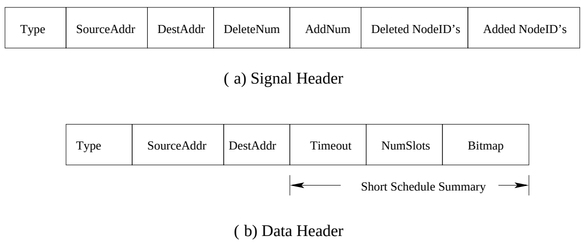


Figure 2: Signaling and data packet header format

information to select the transmitters and receivers for the current time slot, leaving all other nodes in liberty to switch to low-power mode.

TRAMA assumes a single, time-slotted channel for both data and signaling transmissions. Figure 1 shows the overall time-slot organization of the protocol. Time is organized as sections of random- and scheduled-access periods. We refer to random-access slots as *signaling slots* and scheduled-access slots as *transmission slots*. Because the data rates of a sensor network are relatively low, the duration of time slots is much larger than typical clock drifts. For example, for a 115.2 $Kbps$ radio, we use a transmission slot of approximately $46ms$ to transmit 512-byte application layer data units. Hence, clock drifts in the order of ms can be tolerated, and yet typically clock drifts are in the order of microseconds or even less. This allows very simple timestamp mechanisms (e.g., [10]) to be used for node synchronization. When much smaller clock drifts must be assumed and more expensive nodes can be used, nodes can be time synchronized using techniques such as GPS [9]. Accordingly, in the remainder of our description of TRAMA, we simply assume that adequate synchronization is attained.

NP propagates one-hop neighbor information among neighboring nodes during the random access period using the *signaling slots*, to obtain consistent two-hop topology information across all nodes. As the name suggests, during the random access period, nodes perform contention-based channel acquisition and thus signaling packets are prone to collisions.

Transmission slots are used for collision-free data exchange and also for schedule propagation. Nodes use SEP to exchange traffic-based information, or schedules, with neighbors. Essentially, schedules contain current information on traffic coming from a node, i.e., the set of receivers for the traffic originating at the node. A node has to announce its schedule using SEP before starting actual transmissions. SEP maintains consistent schedule information across neighbors and updates the schedules periodically.

AEA selects transmitters and receivers to achieve collision-free transmission using the information obtained from NP and SEP. This is the case, because electing both the transmitter and the receiver(s) for a particular time slot is a necessity to achieve energy efficiency in a collision-free transmission schedule. Random transmitter selection leads to collisions, and electing the transmitters and not the receivers for a given time slot leads to energy waste, because all the neighbors around a selected transmitter have to listen in the slot, even if they are not to receive any data. Furthermore, selecting a transmitter without regard to its traffic leads to low channel utilization, because the selected transmitter may not have any data to send to the selected receiver. Hence, AEA uses traffic information (i.e., which sender has traffic for which receivers) to improve channel utilization.

The length of a transmission slot is fixed based on the channel bandwidth and data size. Signaling packets are usually smaller than data packets and thus transmission slots are typically set as a multiple of signaling slots to allow for easy synchronization. In our implementation, transmission slots are seven times longer than signaling slots.

2.2 Access Modes and the Neighbor Protocol

In sensor networks, nodes may fail (e.g., power drained) or new nodes may be added (e.g., additional sensors deployed). To accommodate topology dynamics, TRAMA alternates between random- and scheduled access.

TRAMA starts in random access mode where each node transmits by selecting a slot randomly. Nodes can only join the network during random access periods. The duty cycle of random- versus scheduled access depends on the type of network. In more dynamic networks, random access periods should occur more often. In more static scenarios, the interval between random access periods could be larger, because topology changes need to be accommodated only occasionally. In the case of sensor networks, there is very little or no mobility, depending on the type of application. Hence, the main function of random access periods is to permit node additions and deletions. Time synchronization could be done during this period. During random access periods, all nodes must be in either transmit or receive state, so they can send out their neighborhood updates and receive updates from neighbors. Hence, the duration of the random access period plays a significant role in energy consumption.

During random access periods, signaling packets may be lost due to collisions, which can lead to inconsistent neighborhood information across nodes. To guarantee consistent neighborhood information with some degree of confidence, the length of the random access period and the number of retransmissions of signaling packets are set accordingly. In [4], it is shown that, for a network with an average of N two-hop neighbors, the number of signaling packet retransmissions should be 7 and the retransmission interval $1.44 * N$ to guarantee packet delivery of 99%. Thus, the length of the random access period will then be $7 * 1.44 * N$.

NP gathers neighborhood information by exchanging small signaling packets during the random access period. Figure 2(a) shows the format of the header of a signaling packet. Signaling packets carry incremental neighborhood updates and if there are no updates, signaling packets are sent as "keep-alive" beacons. Each node sends incremental updates about its one-hop neighborhood as a set of added and deleted neighbors. These signaling packets are also used to maintain connectivity between the neighbors. A node times out a neighbor if it does not hear from that neighbor for a certain period of time. The updates are retransmitted such that we ensure 0.99 probability of success. Because a node knows the one-hop neighbors of its one-hop neighbors, eventually consistent two-hop neighborhood information makes its way across the network.

2.3 Schedule Exchange Protocol

SEP establishes and maintains traffic-based schedule information required by the transmitter (i.e., slot re-use) and receiver (i.e., sleep state switching) selection. A node's schedule captures a window of traffic to be transmitted by the node. This information is periodically broadcast to the node's one-hop neighbors during scheduled access.

Schedule generation works as follows. Each node computes a SCHEDULE_INTERVAL based on the rate at which packets are produced by the higher layer application. The SCHEDULE_INTERVAL of a node represents the number of slots for which the node can announce the schedule to its neighbors according to the current state of its MAC-layer queue. The node then pre-computes the number of slots in the interval $[t, t + SCHEDULE\_INTERVAL]$ for which it has the highest priority among its two-hop neighbors (contenders), which we call "winning slots". Because these are the slots for which the node will be selected as the transmitter, the node announces the intended receivers for these slots. Alternatively, if a node does not have enough packets to transmit, it announces that it gives up the corresponding slot(s). Other nodes that have data to transmit can make use of these "vacant" slots. A node's last winning slot in this interval is reserved for broadcasting the node's schedule for the next interval. For example, suppose that node u's SCHEDULE_INTERVAL is 100 slots. During time slot 1000, u computes its winning slots between $[1000, 1100]$. Let us assume that these slots are 1009, 1030, 1033, 1064, 1075, and 1098. Node u uses slot 1098, its last winning slot in this interval, to announce its next schedule by looking ahead from $[1098, 1198]$, and so on. The time corresponding to the last winning slot is fixed as the lifetime for the schedule.

Nodes announce their schedule via *schedule packets*. Because nodes have two-hop topology information obtained through NP, there is no need to send receiver addresses in the schedule packet. Instead, nodes convey intended receiver information using a bitmap whose length is equal to the number of one-hop neighbors. Each bit in the bitmap corresponds to one particular receiver ordered by their identities. The total number of receivers supported by this scheme depends on the size of the data slot and the number of slots for which receivers are announced. [1] For example, a node with four one-hop neighbors with identities 14, 7, 5 and 4 will have a bitmap of size four with first MSB corresponding to node 14, second MSB to node 7. An advantage of using bitmaps is the ease with which broadcast and multicast communication can be supported. To broadcast a packet, all bitmap bits are set to 1, indicating that all one-hop neighbors are intended receivers of the packet. If the packet needs to be multicast to just 14 and 4, then only these bits are set in the bitmap. A node forms the bitmap for the winning slots based on the current traffic information for its queue. If the node's queue size is smaller than the number of bitmaps contained in the schedule, some of the winning slots will go unused. For these "vacant" slots, the node announces a zero bitmap. Slots with zero bitmaps could potentially be used by some other node in the two-hop neighborhood. The slot after which all the winning slots go unused is called *ChangeOver* slot. All unused slots happen contiguously toward the end before the last winning slot, which is reserved for announcing the next schedule. This maximizes the length of sleep periods.

Figure 3 shows the schedule packet format. *SourceAddr* is the address of the node announcing the schedule, *timeout* is the number of slots for which the schedule is valid (starting from the current slot), *width* is the length of the neighbor bitmap (i.e the number of one-hop neighbors), and *numSlots* is the total number of winning slots (i.e., the number of bitmaps contained in the packet). The last winning slot is always reserved for announcing the next schedule.

Additionally, a summary of a node's schedule is sent with every data packet. Schedule summaries help minimize the effects of packet loss in schedule dissemination. As shown in Figure 2(b), the summary includes the schedule's *timeout*, *numSlots*, and a bitmap corresponding to the winning slots in the current interval. The size of the bitmap is *numSlots* and is used to indicate whether the node is transmitting or giving up the corresponding slot. Note that, in order not to incur excessive overhead,[2] schedule summaries do not carry intended receiver information. They are meant to maintain synchronization[3] among one-hop neighbor schedules even in the face of losses. For example by inspecting the values of *numSlots* and the *bitmap* in the schedule summary, the receiving node can update or re-synchronize its stored schedule information. Each schedule has an associated timeout and nodes are not allowed to change the schedule until this timeout expires. This is required to ensure consistency across one-hop neighborhood schedules.

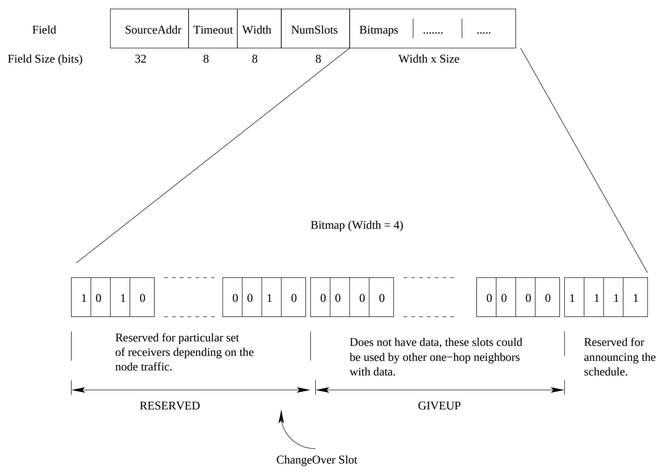
Nodes maintain schedule information for all their one-hop neighbors. The schedule information is consulted whenever a node has the highest two-hop priority to decide if the node will actually transmit (i.e., it has data to send and thus will use the slot) or will give up the slot to another node in the neighborhood. Based on this decision, the schedule information for the node is updated either using the short summary from the data packet (if the node is receiving), or assuming transmissions (if the node is sleeping since it is not the intended receiver of transmitter). In the latter case, its schedule is in an unsynchronized state until the node verifies or updates it based on the schedule summary piggybacked in a future data packet from that transmitter.

All nodes listen during the *ChangeOver* slot of the transmitter to synchronize their schedule. For instance, if a node u keeps assuming transmissions for a particular neighbor tx at different timeslots and the neighbor does not transmit any packets due to a contender that is hidden from u, then the schedule at node u for node tx will be unsynchronized. If node u does not listen during the *ChangeOver* slot,

[1]Assuming the schedule is announced for 16 slots, the scheme can support 256 or 512 neighbors for a 512 or 1024 byte transmission slot size respectively.

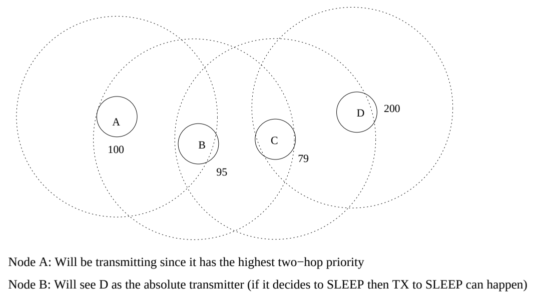
[2]The overhead in the current implementation due to schedule summary is 6 bytes per data packet.

[3]As demonstrated in Section 3, TRAMA's correctness is not affected by unsynchronized schedules.



Field
SourceAddr
Timeout
Width
NumSlots
Bitmaps
....
.....
Field Size (bits)
32
8
8
8
Width x Size
Bitmap (Width = 4)
1 0 1 0
0 0 1 0
0 0 0 0
0 0 0 0
1 1 1 1
Reserved for particular set of receivers depending on the node traffic.
Does not have data, these slots could be used by other one−hop neighbors with data.
Reserved for announcing the schedule.
RESERVED
GIVEUP
ChangeOver Slot

Figure 3: Schedule packet format



A
100
B
95
C
79
D
200
Node A: Will be transmitting since it has the highest two−hop priority
Node B: Will see D as the absolute transmitter (if it decides to SLEEP then TX to SLEEP can happen)

Figure 4: Inconsistency problem

which is the last slot in the current schedule interval that will be used by node tx for transmission, it may assume that the node tx is transmitting the data corresponding to the *ChangeOver* slot and update the corresponding schedule. From now on until the schedule announced by node tx expires, node u considers that node tx gives up winning slots for re-use. This can lead to collisions if node u tries to reuse the winning slot of node tx that is actually used for transmission. Hence, schedules among neighbors could be unsynchronized only until the *ChangeOver* slot and a node has to listen during the *ChangeOver* slot of the transmitter.

It is possible for a node to get some extra slots for transmissions in addition to the original winning slots computed while announcing the schedule. To prevent inconsistencies and collisions when transmitting the schedule packet, a node should always send the schedule packet only on the previously announced timeout. Because all the slots after the *Changeover* slot are assumed as give-up slots by the neighbors, the schedules might be unsynchronized. Hence, transmitting a schedule before the timeout can potentially cause collisions with neighbors. In the following section, we describe how these schedule information is used to adaptively decide the node state.

2.4 Adaptive Election Algorithm

In the original NCR algorithm [5], a node is selected to transmit if it has the highest priority among its *contending set*. Node u's *contending set* is the set of all nodes that are in u's two-hop neighborhood. Node u's priority at time slot t is defined as the pseudo-random hash of the concatenation of node u's identity and t, or

$$prio(u, t) = hash(u \oplus t) \tag{1}$$

Assuming that node identities are unique and nodes are synchronized, all nodes compute the same priority value at any given time slot. However, if the selected node does not have any data to send, then the slot is wasted. Furthermore, nodes are free to transmit to any one-hop neighbor, because there is no sleep state in NCR.

For energy efficiency, TRAMA switches nodes to sleep state whenever possible, and attempts to re-use slots that are not used by the selected transmitter for bandwidth efficiency. A selected node may give up its transmission slot if it does not have any packets to send; this slot could then be used by another node. Nodes exchange current traffic information with their neighbors to make effective use of low-power, idle radio mode and accomplish slot re-use.

At any given time slot t during the scheduled access period, the state of a given node u is determined based on u's two-hop neighborhood information and the schedules announced by u's one-hop neighbors. Possible states are: transmit (TX), receive (RX), and sleep (SL).

At any given slot t, a node u is in the TX state if: (1) u has the highest *priority*, i.e., $prio(u, t)$ among its *contending set* and (2) u has data to send.

A node is in the RX state when it is the intended receiver of the current transmitter. Otherwise, the node can be switched off to the SL state, because it is not participating in any data exchange. This means that, if a node is not the selected transmitter, it will decide whether it needs to be in RX state by consulting the schedule sent out by the selected transmitter. If the transmitter does not have traffic destined for that node in the current slot, the node can then sleep.

Each node executes AEA to decide its current state (TX, RX, or SL) based on current node priorities (within its two-hop neighborhood) and also on the announced schedules from one-hop neighbors. The algorithm's pseudo-code is provided in Figure 5. Table 1 lists some basic terminology and notation used in the description of AEA.

The state of a node depends on the *Absolute Winner* and the announced schedules from its one-hop neighbors. From node u's point of view, the *Absolute Winner* at any given time slot t can be: (1) node u itself, (2) node v that lies in the two-hop neighborhood of node u in which case the *Alternate Winner* $atx(u)$ needs to be accounted for if hidden from node v, or (3) a node w that lies in node u's one-hop neighborhood.

Whenever a node becomes an *Absolute Winner* for a particular timeslot and has announced a non-zero bitmap for this slot, it knows that no other node in its two-hop neighborhood will be transmitting in this slot. Thus, the node can transmit collision-free to its intended receiver(s). When a node is not an *Absolute Winner*, it is not certain who the actual transmitter for a particular slot is. For example, consider the topology shown in Figure 4. Let D be the node

Table 1: Notations and terminologies

$\mathbf{N2}(u)$	Set of neighbors of node u which are two-hops away.
$\mathbf{N1}(u)$	Set of neighbors of node u which are one-hop away.
$\mathbf{CS}(u)$	u's *Contending Set* is the set of nodes in u's two-hop neighborhood such that $\{u \cup \mathbf{N1}(u) \cup \mathbf{N2}(u)\}$.
$tx(u)$	*Absolute Winner* is the node with the highest priority in $\mathbf{CS}(u)$.
$atx(u)$	*Alternate Winner* is the node which has the highest priority among u's one-hop neighbors, i.e., over the set $\{u \cup \mathbf{N1}(u)\}$.
$\mathbf{PTX}(u)$	*Possible Transmitter Set* is the set of all nodes in $\{u \cup \mathbf{N1}(u) - atx(u)\}$ that satisfy the condition given in Equation 2.
$\mathbf{NEED}(u)$	*Need Contender Set* is the set of nodes in $\{\mathbf{PTX}(u) \cup u\}$ that are in need of additional transmission slots.
$ntx(u)$	*Need Transmitter* is the node with the highest priority among the set of nodes $\mathbf{NEED}(u)$ containing valid synchronized schedule.

with highest priority in node B's two-hop neighborhood in a given time slot and let A be the highest 2-hop priority node in node A's two-hop neighborhood. Both A and D could transmit in the time slot because they are *Absolute Winners*; the *Absolute Winner* to node B is node D. Therefore, if B looks at its schedule information for D and finds out that it is not D's intended receiver for the current slot, it will decide to switch to SL mode. However, if it happens to be A's intended receiver, it will end up missing A's transmission. Hence, before switching to SL mode, a node must also account for the *Alternate Winner.* This potential inconsistency occurs only if the *Alternate Winner* is hidden from the *Absolute Winner* i.e., they are three hops away.

To avoid wasting slots when the *Absolute Winner* has no data to send, TRAMA keeps track of nodes that could use extra slots to send their data. It first computes the set of nodes that can possibly transmit at the current time slot. They are kept in the *Possible Transmitter Set*, which contains all nodes in the one-hop neighborhood that can possibly transmit without any collision. A node can transmit without collisions only if it has the highest priority in the two-hop neighborhood. Hence, a node checks for possible transmitters in the one-hop neighborhood using the available information. Because a node cannot know the entire two-hop neighborhood of its one-hop neighbors, it can only check if this neighbor has the highest priority among the nodes that are known to be the neighbor's two-hop neighbor. In other words, for a one-hop neighbor of node u, say node y, the following condition should be satisfied to be in the $\mathbf{PTX}(u)$:

$$prio(y) > prio(x) \forall x, x \in \mathbf{N1}(\mathbf{N1}(y)) \text{ and } x \notin \mathbf{N1}(y)) \quad (2)$$

The *Need Contender Set* is the subset of the *Possible Transmitter Set* and contains only those nodes that have data to send. Nodes for which node u does not have (valid) schedules are also included in this set as node u does not know whether these nodes have data to send.

The *Absolute Winner* is the assumed transmitter for a node, unless the *Alternate Winner* is hidden from *Absolute Winner* and it belongs to the *Possible Transmitter Set.* In the latter case, the *Alternate Winner* is the assumed transmitter. Whenever the assumed transmitter gives up, the *Need Contender Set* is checked and the node with the highest priority within this set is selected as the *Need Transmitter* $ntx(u)$. Nodes that are not in the schedule listed by the assumed transmitter can switch to SL mode to save energy. This is especially beneficial in scenarios in which only a few nodes generate data at a time and data are destined to small subset of receivers.

3. TRAMA CORRECTNESS

TRAMA is correct, if it avoids collisions and transmissions to a sleeping node - both can cause packet losses. A node can, however, assume that some of its neighbors is transmitting when the transmission does not actually happen. Though this will lead to increased energy consumption because nodes may be in receive mode unnecessarily, it does not affect the correctness of the algorithm.

Arguing for collision freedom is simple. The only two ways in which a node u can be a transmitter is through line 4 in the pseudo-code, where the node is the *Absolute Winner* and line 34 in the pseudo-code, where the node is the *Need Transmitter.* In both cases, there cannot be a node two-hops away from node u and transmitting. This follows from Equation 2 and by the definition of *Absolute Winner.* Hence, there can be no collisions due to transmissions from two-hop neighbors. Assuming that schedules are synchronized (which allows a node to know exactly whether the elected one-hop neighbor uses the slot or gives it up for re-use) and by virtue of the election mechanism, no other node that is one-hop away from node u can transmit. Hence, there can be no collisions due to a neighbor transmitting at the same time and the protocol maintains collision freedom at all the times.

To show that TRAMA never looses a packet due to an invalid state assignment (i.e., a node transmitting to a sleeping node), it is enough to show that, whenever a node u goes to sleep assuming that some node v is transmitting in its one-hop neighborhood (given that node u is not the intended receiver of node v), then no other node except node v can transmit in the one-hop neighborhood. A node always considers the transmitter to be either the *Absolute Winner*, the *Alternate Winner*, or the *Need Transmitter.* Because a node can receive only from a node that is one-hop away, the *Absolute Winner* should be a one-hop neighbor. Hence, if a node assumes that a neighbor is transmitting, it is either the *Alternate Winner* or the *Need Transmitter.*

Consider the case in which node u decides to sleep during a time slot t assuming that node v is the transmitter. Hence, node v has the highest priority among the two-hop neighbors of node v known to node u and among the one-hop neighbors of node u that have data to send. Let node

```
Compute $tx(u)$, $atx(u)$ and $ntx(u)$
**if** ($u = tx(u)$) **then**
  **if** ($u.isScheduleAnnouncedForTx = TRUE$) **then**
    **let** $u.state = TX$
    **let** $u.receiver = u.reported.rxId$
    Transmit the packet and update the announced schedule
  **else if** ($u.giveup = TRUE$) **then**
    **call** HandleNeedTransmissions
  **endif**
**else if** ($tx(u) \in N1(u)$) **then**
  **if** ($tx(u).announcedScheduleIsValid = TRUE\ AND\ tx(u).announcedGiveup = TRUE$) **then**
    **call** HandleNeedTransmissions
  **else if** ($tx(u).announcedScheduleIsValid = FALSE\ OR\ tx(u).announcedReceiver = u$) **then**
    **let** $u.mode = RX$
  **else**
    **let** $u.mode = SL$
    Update schedule for $tx(u)$
  **endif**
**else**
  **if** ($atx(u)$ hidden from $tx(u)$ $AND$ $atx(u) \in \mathbf{PTX}(u)$) **then**
    **if** ($atx(u).announcedScheduleIsValid = TRUE\ AND\ atx(u).announcedGiveup = TRUE$) **then**
        **call** HandleNeedTransmissions
    **else if** ($atx(u).announcedScheduleIsValid = FALSE\ OR\ atx(u).announcedReceiver = u$) **then**
        **let** $u.mode = RX$
    **else**
        **let** $u.mode = SL$
        Update schedule for $atx(u)$
    **endif**
  **else**
    **call** HandleNeedTransmissions
**endif**
**procedure** HandleNeedTransmissions
**if** ($ntx(u) = u$) **then**
  **let** $u.state = TX$
  **let** $u.receiver = u.reported.rxId$
    Transmit the packet and update the announced schedule
**else if** ($ntx(u).announcedScheduleIsValid = FALSE \ || \ ntx(u).announcedReceiver = u$) **then**
  **let** $u.mode = RX$
**else**
  **let** $u.mode = SL$
  Update the schedule for $ntx(u)$
**endif**
```

Figure 5: Pseudo-code description of AEA

w be the actual transmitter for the time slot t in the one-hop neighborhood of node u. This means that node w has the highest priority among the two-hop neighbors of node w and it has the highest priority among the one-hop neighbors of node w that have data to send. Node w can either be a one-hop neighbor to node v or a two-hop neighbor to node v. If node w is a one-hop neighbor of node v, then node w should have higher priority than node v. Because, node v and node w are neighbors of node u and the schedules are synchronized, it contradicts the fact that node v has the highest priority among the one-hop neighbors of node u that have data to send. Hence, node w cannot be the actual transmitter. When node w is a two-hop neighbor to node v, it should have higher priority than node v, which contradicts the fact that node v has the highest priority among the two-hop neighbors of node v known to node u. Hence, there cannot be a transmitter, node w in the one-hop neighborhood of node u.

Schedules can get unsynchronized for different reasons, and TRAMA is also correct in the face of unsynchronized schedules. For example when a node assumes that node v is transmitting and decides to sleep, node v may not transmit. This will make the schedules unsynchronized and the *ChangeOver* slot for node v is reached earlier. The requirement that all the nodes should listen during the *ChangeOver* slot of the neighbors, prevents a node going to sleep or transmit state assuming that the neighbor is giving up and ensures correctness. Whenever a node assumes that a neighbor is transmitting a data to it, the schedules are updated only after receiving the data from the neighbor using the schedule summary. Hence, packet losses due to transmission errors can cause the schedules to be unsynchronized and forces a node to listen whenever the unsynchronized neighbor is elected for transmission. This continues until the node receives a data packet from the unsynchronized neighbor, and also prevents invalid state assignment. Hence, TRAMA is correct even when the schedules are not synchronized.

4. EXPERIMENTAL SETUP

Through simulations, we evaluate TRAMA and compare its performance against both contention- and scheduling-based protocols. While we consider Carrier-Sense Multiple Access (CSMA) [16], IEEE802.11 [13] and S-MAC [24] as example contention-based protocols, we use Node Activation Multiple Access (NAMA) [5] as example scheduling-based protocol.

We used Qualnet [3] as our simulation platform and we present the results for a variety of scenarios. The underlying physical layer model used for all the experiments was based on the TR1000, a typical radio used in sensor networks. The TR1000 [2], the radio used by the UC Berkeley Motes [1], are short range, low data-rate (a maximum of 115.2KBPS) radios with built-in support for low-power sleep state. The average power consumption in transmit, receive and sleep modes is $24.75mW$, $13.5mW$ and $15\mu W$, respectively. The maximum transition time for switching is $20\mu S$. The modulation type used in the physical layer is ASK and the receiver threshold is $-75dBm$. Fifty nodes are uniformly distributed over a 500m x 500m area in all the experiments. The transmission range of each node is 100m and the topology is such that the nodes have 6 one-hop neighbors on average. The average size of the two-hop neighborhood for this network is 17 nodes. Two different types of traffic load are considered in our study. We used a scenario in which node traffic is statistically generated based on a exponentially distributed inter-arrival time. We chose this to stress-test protocol performance for different arrival rates. We also test TRAMA's performance when driven by data gathering applications, which are considered typical of sensor networks. Below, we describe these traffic scenarios as well as other simulation parameters in detail.

4.1 Protocol Parameters

In both scenarios we fixed up the SCHEDULE_INTERVAL to be 100 transmission slots for TRAMA. The maximum size of a signaling packet is fixed at 128 bytes which gives to a slot period of $6.82ms$ with guard time to take care of switching. Transmission slots are seven times longer than the signaling slots supporting a maximum data fragment size 896 bytes. The random access period is fixed to 72 transmission slots and is repeated once every 10000 transmission slots.

S-MAC is a contention-based channel access protocol and it uses periodic sleep intervals to conserve energy. Sleep schedules are established using $SYNC$ packets which are exchanged once every SYNC_INTERVAL. The duty cycle determines the length of the sleep interval.

We set SYNC_INTERVAL as $10sec$ and we varied the duty cycle (10% and 50%). All the nodes are time synchronized and hence we favored S-MAC by allowing the listen and sleep periods synchronized across the entire network. We also observed that S-MAC needed larger time to set up the listen/sleep schedules with the neighbor. This is because S-MAC does not have a proper neighbor discovery protocol, it has to rely on the $SYNC$ packets for doing this. $SYNC$ packets are transmitted only once and are transmitted unreliably. Hence, a large warmup time of $20sec$ is allowed for the neighbor information to settle down. Because the queuing delay for the scheduling-type MAC's is higher, we allowed some more time for delivering the queued packets before ending the simulation. The simulation is run for $400sec$ and the results are averaged over multiple runs.

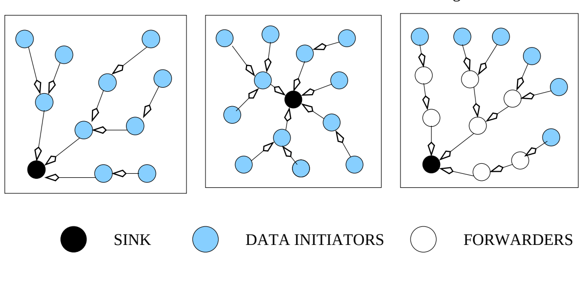
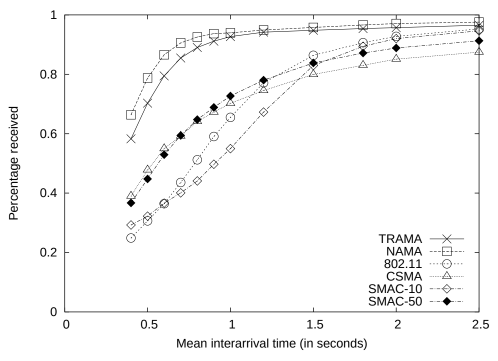
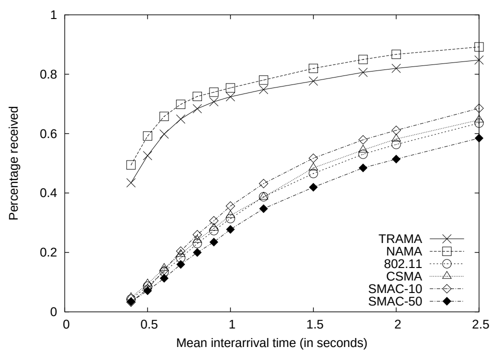


Figure 6: Data gathering application



(a) Unicast Traffic



(b) Broadcast Traffic

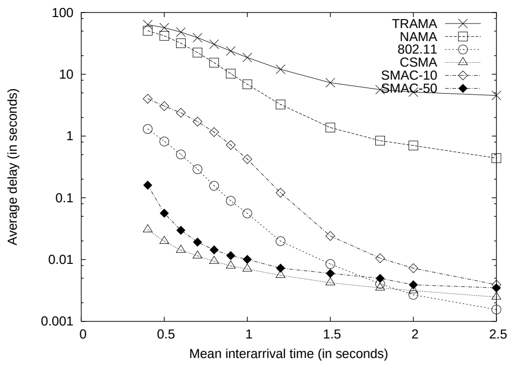
Figure 7: Average packet delivery ratio for synthetic traffic

4.2 Synthetic Data Generation

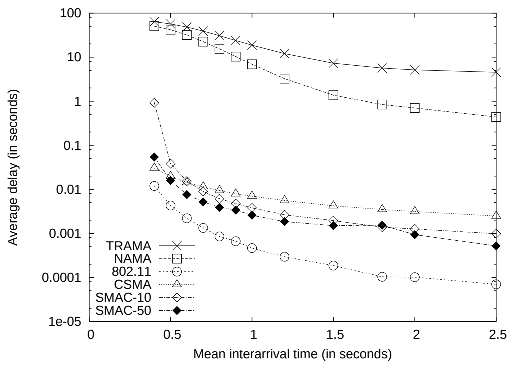
The objective of this experiment is to measure the performance of TRAMA when all the nodes in the network generate traffic based on some statistical distribution. We used exponential inter-arrival for generating data and varied the rate from 0.5 to 2.5 seconds. A neighbor is randomly selected as a next-hop every time a node transmits a packet. We tested both unicast and broadcast data generation separately. The performance metrics are:

- **Average Packet Delivery Ratio:** It is the ratio of number of packets received to the number of packets sent averaged over all the nodes. For broadcast traffic a packet is counted to be received only if it is received by all the one-hop neighbors.

- **Percentage Sleep Time:** It is the ratio of the number of sleeping slots to the total number of slots averaged over the entire network.

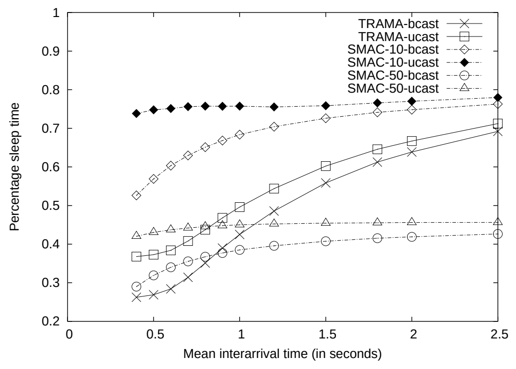


(a) Unicast Traffic

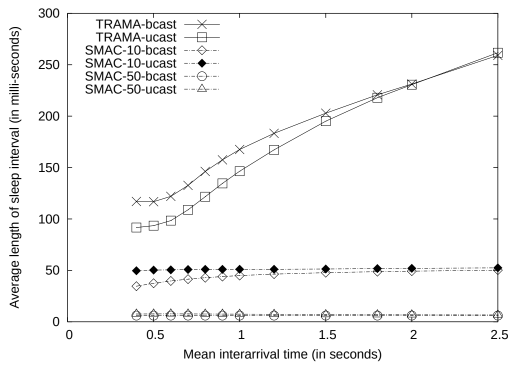


(b) Broadcast Traffic

Figure 8: Average queuing delay for synthetic traffic



(a) Percentage Energy Savings



(b) Average Sleep Interval

Figure 9: Energy savings and average sleep interval for synthetic traffic

- **Average queuing Delay:** Average delay for the packet to be delivered to the receiver
- **Average Sleep Interval**: This is the average length of sleeping interval. This measures the number of radio mode switching involved. Frequent switching can waste energy due to the transient power consumption involved in switching.[4]

4.3 Data-Gathering Application

We assume a sink is collecting data from all the sensors for these experiments. The sink sends out a broadcast query requesting data from all the sensors. The sensors respond back with the data, which are generated periodically to the sink. We implemented a simple reverse-path routing to forward the data from the sensors to the sink. Figure 6 shows the three different scenarios considered for this study. Data-collection node or sink is placed in the corner for the first case and in the middle for the second case.

All the sensors respond with periodically generated data in both cases. Because data aggregation [14] or grouping data to minimize traffic, are advantageous, we also emulated data aggregation in a third case. Here, only the nodes at the edge generate traffic and we assume that the nodes do data aggregation and appends its reading to the parent node. To measure performance in these experiments we use the metrics defined for the synthetic case. The average packet delivery ratio is measured as the ratio of total number of data received by the sink to the total number of data sent by all the sensors, unlike the per-hop delivery ratio used for synthetic traffic generation.

[4]Measurements for 802.11 based radios are available in [12].

5. SIMULATION RESULTS

5.1 Synthetic Traffic

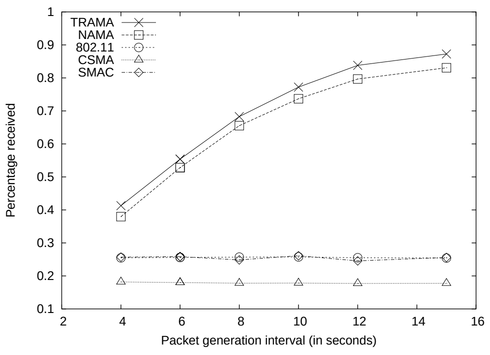
The packet delivery ratio, average queuing delay, percentage of sleep time, and average length of sleep intervals for synthetic traffic scenarios are shown in Figure 7, Figure 8, Figure 9(a) and Figure 9(b), respectively. We present results for S-MAC using two different duty cycles, namely 50% and 10%. Our results indicate that, in general, schedule-based MACs based on NCR achieve better delivery than IEEE802.11, CSMA and S-MAC. The main reason for the improvement in delivery is the collision freedom guaranteed at all times during data transmission. The effect is more noticeable when all nodes generate broadcast traffic. In CSMA, S-MAC and IEEE802.11, broadcasting is unreliable and susceptible to hidden-terminal collisions. This reduces broadcast delivery significantly when we increase the load as our results indicate. For IEEE802.11 and S-MAC, the RTS/CTS/DATA/ACK exchange for unicast traffic improves delivery when compared to broadcast traffic because it reduces hidden-terminal collisions by doing collision avoidance.

Schedule-based MACs, on the other hand, incur higher average queuing delays. We should point out that when measuring average delay, we account for the delay of packets successfully delivered. However, TRAMA and NAMA deliver more packets than contention-based MACs and this will reduce the retransmissions at the higher layers. Hence, the end-to-end delay perceived by the application will be comparable to that of contention-based protocols. The average queuing delay for TRAMA is higher than that of NAMA due to the overhead involved in propagating scheduling information. Once every *SCHEDULE_INTERVAL*, a transmission slot is used for announcing schedules. This decreases the effective channel access probability for data transmission. This scenario is not a favorable scenario for traffic-adaptive elections, because the traffic is homogeneous across the network and all nodes periodically generate traffic. The throughput of TRAMA is comparable with that of NAMA and is significantly better than contention-based protocols for both unicast and broadcast traffic scenarios. The performance of the only other energy-efficient protocol, S-MAC is comparable with IEEE802.11 in terms of throughput. However, the delay is slightly higher than that of IEEE802.11 or CSMA due to the sleep periods, and it increases as the duty cycle of the listen periods is decreased. For a duty cycle of 50%, the delay of S-MAC is smaller than that of IEEE802.11 for the unicast-data generation scenario. This is because S-MAC frequently switches between sleep and listen modes(average sleep interval plotted in Figure 9(b) reflects this) for that duty cycle. This is equivalent to a node being awake most of the time, and the delay is less because S-MAC does not have any contention resolution algorithm.

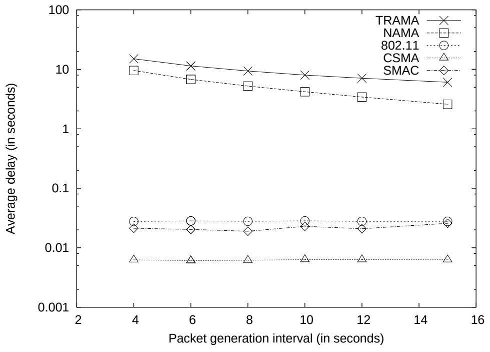
The energy savings of TRAMA depend mainly on the traffic pattern, while the energy savings of S-MAC are dependent on the duty cycle. The total energy savings depend on both percentage sleep time and average length of sleep interval. Percentage sleep time metric does not account for the performance loss that is possible due to frequent radio mode switching. The average length of sleep interval quantifies the amount of radio-mode switching involved. A higher value of average sleep length is preferred because this implies less radio-mode switching and hence more savings. The results indicate that the percentage of sleep time is less for broadcast traffic when compared to unicast traffic, which is intuitive. The percentage sleep time of S-MAC increases as the duty cycle decreases. The price paid for decreasing the duty cycle is an increase in latency. The throughput decreases more steeply for a lower duty cycle as traffic increases. For the broadcast packets, a 50% duty cycle achieves less throughput than 10% duty cycle. Broadcast is done by plain carrier sensing and is more prone to hidden terminal collisions. The timing structure for 512 bytes data that favors low duty cycle as it reduces the channel contention and hence the collisions are reduced. Note that a broadcast packet is counted as delivered ONLY if it is delivered to all the neighbors.

Compared with TRAMA, S-MAC with 10% duty cycle exhibits higher percentage of sleeping time. But the average length of sleep intervals is much lower for S-MAC when compared to TRAMA. This reduces the overall energy savings due to the overhead involved in mode switching. This is the case even though S-MAC is being favored by assuming that synchronized listen/sleep schedules are established across the nodes due to the simulation setup. The performance of TRAMA is not affected by nodes joining at discrete intervals because it does not require any synchronization of listen/sleep schedules.

In the subsequent experiments, we only consider S-MAC with 10% duty cycle, which has better performance than with a 50% duty cycle.



(a) Average delivery ratio

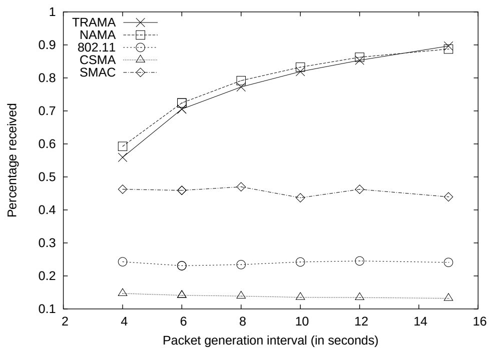


(b) Average queuing delay

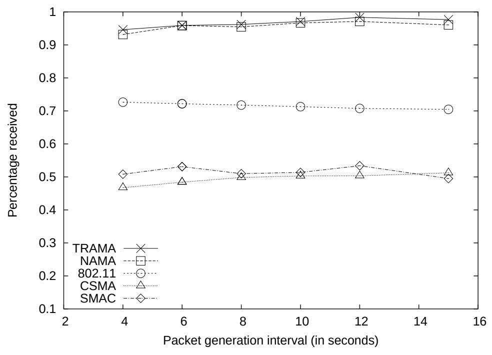
Figure 10: Corner Sink

5.2 Sensor Network Application

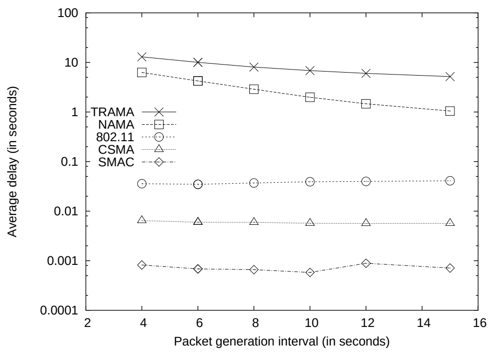
We tested the protocols using a sensor network data gathering application. One of the nodes in the network is designated as the sink and the sink starts sending a broadcast query. All nodes receiving a non-duplicate query add the sender of the query as the next hop for data forwarding, establishing a reverse-shortest path tree with the sink node as the root. Figures 10(a), 11(a), and 12(a) show the average packet delivery ratio for the corner sink, center sink, and edge sink scenarios respectively. Schedule-based MAC protocols outperform the contention-based MAC protocols in all the cases. The delivery is highest for the scenario in which the edge nodes are generating traffic. This is be-



(a) Average delivery ratio

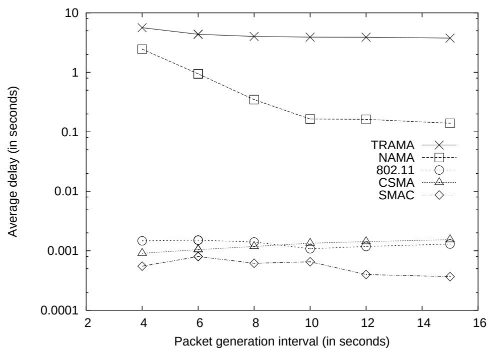


(a) Average delivery ratio



(b) Average queuing delay

Figure 11: Center Sink



(b) Average queuing delay

Figure 12: Edge Sink

cause the overall load in the network is low and well within the capacity of the protocols. Delivery to the center sink is slightly higher than when corner sink is used because the packets need to go through fewer number of hops to reach the sink. TRAMA performs much like NAMA and the decrease in throughput due to scheduling overhead is overcome by TRAMA's adaptive scheduling approach. The average delivery ratio is nearly constant for contention-based protocols as the variation in the offered load is not high.

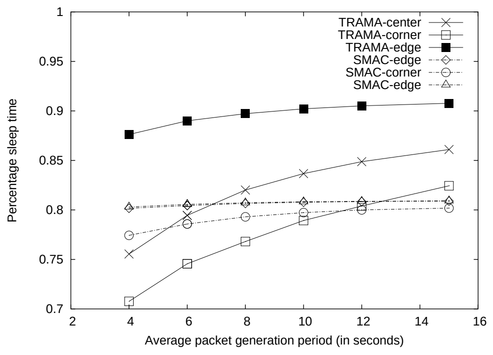
Figures 10(b), 11(b), and 12(b) show the average per-hop delay for all the protocols for the sensor scenarios. Contention-based protocols outperform scheduling-based protocols in terms of delay. This is due to the latency introduced by random scheduling. Finally we show the percentage sleep time achieved by TRAMA and S-MAC in Figure 13(a). The percentage of time nodes can be put to sleep increases with decrease in traffic load. The percentage sleep time is quite high (as high as 85%) for the edge sink scenario which has the lowest load. Again the average length of the sleep interval is also the highest for this case. This clearly shows the benefit of TRAMA's traffic adaptability when compared to S-MAC. The average sleep interval of TRAMA is significantly higher than that of S-MAC. When compared to the edge sink scenario, the percentage sleep time is less for the center and corner sink scenarios due to the increased load. In the corner sink case, data forwarded by the nodes which are closer to the sink is heavier than data forwarded by nodes farther away. This reduces sleep time for these nodes and hence the overall percentage sleep time is lesser than the case where the sink is in the center. This also applies to the average length of sleep period shown in Figure 13(b).

6. CONCLUSIONS

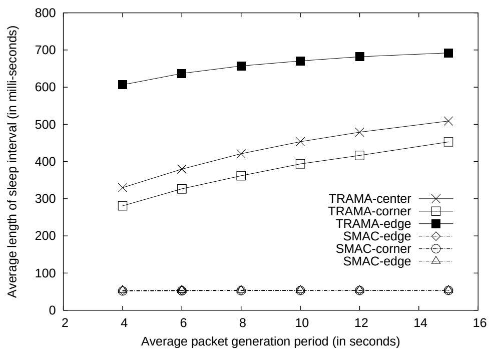
In this paper we presented TRAMA, a new energy-aware channel access protocol for sensor networks. TRAMA uses traffic-based scheduling to avoid wasting slots when nodes do not have data to send and to switch nodes to a low-power standby radio mode when they are not intended receivers of traffic.

Through extensive simulations, we compared TRAMA's performance against a number of contention- and a schedule-based MACs. It is evident from the simulation results that significant energy savings (since nodes can sleep for up to 87% of the time) can be achieved by TRAMA depending on the offered load. TRAMA also achieves higher throughput (around 40% over S-MAC and CSMA and around 20% for 802.11) when compared to contention-based protocols since it avoids collisions due to hidden terminals.

In general, schedule-based MACs exhibit higher delays than contention-based MACs. TRAMA is well suited for applications that are not delay sensitive but require high



(a) Percentage Energy Savings



(b) Average Sleep Interval

Figure 13: Energy savings and average sleep interval for sensor scenarios

delivery guarantees and energy efficiency. A typical example is sensor networks used for periodic data collection and monitoring applications. In summary, TRAMA achieves energy-savings comparable to S-MAC with delivery guarantees comparable to NAMA.

7. REFERENCES

[1] http://www.cs.berkeley.edu/ awoo/smartdust/.

[2] Product specification, http://www.rfm.com/products/data/tr1000.pdf.

[3] Scalable networks, http://www.scalble-solutions.com.

[4] L. Bao and J. Garcia-Luna-Aceves. Hybrid channel access scheduling in ad hoc networks. *Proc. IEEE Tenth International Conference on Network Protocols (ICNP)*, November 2002.

[5] L. Bao and J. J. Garcia-Luna-Aceves. A new approach to channel access scheduling for ad hoc networks. In *The seventh annual international conference on Mobile computing and networking 2001*, pages 210–221, 2001.

[6] I. Chlamtac and A. Farago. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions on Networking*, 2(1):23–29, February 1994.

[7] I. Chlamtac and A. Lerner. Fair algorithms for maximal link activation in multihop radio networks. *IEEE Transactions on Communications*, 35(7):739–746.

[8] I. Cidon and M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Transactions on Computers*, 38(10):1236–1361, October 1989.

[9] E.D.Kaplan. *Understanding GPS: Principles and Applications*. Artech House, 1996.

[10] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *IPDPS 2001*, April 2001.

[11] A. Ephremides and T. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on Communications*, 38(4):456–460, April 1990.

[12] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, 2001.

[13] IEEE. Wireless LAN medium access control (MAC) and physical layer specifications. ANSI/IEEE Standard 802.11, 1999 Edition, 1999.

[14] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks, 2001.

[15] J. Ju and V. Li. An optimal topology-transparent scheduling method in multihop packet radio networks. *IEEE/ACM Transactions on Networking*, 6(3):298–306, June 1998.

[16] L. Kleirock and F. Tobagi. Packet switching in radio channels, part 1: Carrier sense multiple-access models and their throughput-delay characteristics. *IEEE Transactions on Communications*, 23(12):1400–1416.

[17] L. Kleirock and F. Tobagi. Packet switching in radio channels, part 2: Hidden-terminal problem in carrier sense multiple access and the busy-tone solution. *IEEE Transactions on Communications*, 23(12):1417–1433, 1975.

[18] S. Lam. A carrier sense multiple access protocol for local networks. *Computer Networks*, 4:21–32, 1980.

[19] S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5(2):81–94, 1999.

[20] S. Singh and C. Raghavendra. PAMAS: Power aware multi-access protocol with signaling for ad hoc networks, 1999.

[21] K. Sohrabi and G. Pottie. Performance of a novel self-organization protocol for wireless ad hoc sensor networks. *IEEE 50th. Vehicular Technology Conference*, pages 1222–1226, 1999.

[22] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In *Proceedings of the IEEE Infocom*, June 2002.

[23] A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. *ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom) 2001*, 2001.

[24] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. *IEEE Infocom 2002*, pages 1567–1576, June 2002.