TreeMAC: Localized TDMA MAC Protocol for Real-time High-data-rate Sensor Networks

Wen-Zhan Song† Renjie Huang† Behrooz Shirazi† Richard LaHusen‡
† Sensorweb Research Laboratory, Washington State University, Vancouver, WA 98686
‡ Cascades Volcano Observatory, U.S. Geological Survey
Email: † {songwz, renjie_huang, shirazi}@wsu.edu ‡rlahusen@usgs.gov

***Abstract*—Earlier sensor network MAC protocols focus on energy conservation in low-duty cycle applications, while some recent applications involve real-time high-data-rate signals. This motivates us to design an innovative localized TDMA MAC protocol to achieve high throughput and low congestion in data collection sensor networks, besides energy conservation. TreeMAC divides a time cycle into *frames* and frame into *slots*. Parent determines children's *frame* assignment based on their relative bandwidth demand, and each node calculates its own *slot* assignment based on its hop-count to the sink. This innovative 2-dimensional frame-slot assignment algorithm has the following nice theory properties. Firstly, given any node, at any time slot, there is at most one active sender in its neighborhood (including itself). Secondly, the packet scheduling with TreeMAC is bufferless, which therefore minimizes the probability of network congestion. Thirdly, the data throughput to gateway is at least $1/3$ of the optimum assuming reliable links. Our experiments on a 24 node test bed demonstrate that TreeMAC protocol significantly improves network throughput and energy efficiency, by comparing to the TinyOS's default CSMA MAC protocol and a recent TDMA MAC protocol Funneling-MAC [8].** [1]

I. Introduction

In typical sensor network applications, sensor nodes monitor fields and generate events which traverse hop-by-hop toward one or more sink nodes. The major traffic pattern is therefore many-to-one forming a *tree*. It serves many diverse applications from low data rate event-driven monitoring applications to high data rate real-time industrial applications.

Controlling access to the channel, generally known as MAC protocol, plays a key role in determining channel capacity utilization, network delays and energy consumptions. It also influences congestion and fairness in channel usage. The fundamental task of any MAC protocol is to regulate the access of a number of nodes to a shared medium in such a way that certain application-dependent performance requirements are satisfied. The traditional MAC protocols (including ALOHA, CSMA and their variants) assume random any-to-any communication and tend to give nodes *equal* channel access. However, *equal* channel access is *not fair* in the data collection scenario: the nodes closer to the sink need to forward more data than the nodes further away. For example, in figure 1, assume all nodes in one collision domain, every node delivers data up to a sink through the subtree root f. If node k gets equal medium sharing opportunity as node f, it is obviously *unfair* and will cause congestion at node f (e.g., funneling effect [8]). To ensure fairness, node f and h should get 6 times and 4 times opportunities of nodes g, i, j, k. In other words, nodes in a network should get channel access opportunity proportional to their demands [9]. Many-to-one routing, which is commonly used in most sensor networks, needs a fundamental revolution of MAC protocol design, as it is different from random any-to-any communication network where *equal* means *fair*. *This is the key motivation of TreeMAC design.*

Earlier sensor network MAC protocols [6], [7] are designed for low duty-cycle and low data-rate applications with the goal of energy conservation. More recent applications involve high-data-rate signals, such as monitoring industrial processes [3], geophysical environments [1], [2] and civil structures such as buildings or bridges [3]–[5]. In those high-data-rate applications, there is a key challenge on how to collect those high-fidelity data subject to the limited radio bandwidth available to sensor nodes. In addition to the limited physical bit rate of the radios used in those low power platforms, radio links often experience packet loss due to congestion, interference, and multi-path effects. These problems are exacerbated over multi-hop routing paths. TDMA-based MAC protocol is therefore a natural choice to increase the throughput and reduce congestion while conserving energy, with the aid of time synchronization protocols [10]–[12]. Many existing TDMA MAC protocols are based on graph coloring to avoid scheduling conflicts in 2-hop and the message overhead is relatively high. In addition, their goals are to maximize the slot (e.g., spectrum) spatial reuse for all nodes in the network. However, *slot reuse maximization based on graph coloring does not necessarily mean throughput maximization to the sink*. In other words, it is true that every node can send out more data, but the sink does not necessarily get more data.

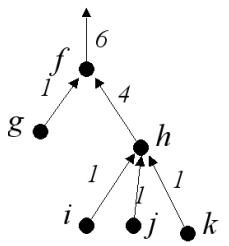


Fig. 1. The fairness of MAC protocol in data gathering scenario. In the figure, the numbers denote the corresponding link's bandwidth demand.

Motivated by the needs of localized TDMA MAC protocol for high-throughput and fairness, in this paper, we propose an innovative TreeMAC protocol. With TreeMAC, channel access is well regulated for throughput maximization and energy conservation, and every node gets time slots proportional to

[1]This work is supported by NASA ESTO AIST program and USGS Volcano Hazard program under the research grant NNX06AE42G.

its bandwidth demands. TreeMAC divides time into frames and each frame consists of three slots. By making use of the parent-children relationship formed by any data collection routing protocol, the frame-slot assignment is determined and exchanged between parent and children only, hence it is truly localized. Parent determines children's *frame* assignment based on their relative bandwidth demands; and each node calculates *slot* assignment based on its hop-count to the sink. This frame-slot assignment algorithm is an innovative two dimensional approach: frame assignment eliminates horizontal two-hop interference, and slot assignment eliminates vertical two-hop interference. We proved that, given any node, at any time slot, there is at most one active sender in its 1-hop neighborhood (including itself). We call it conflict-free packet sending/receiving and snooping, which is much stronger than conflict-free packet scheduling (e.g., conflict-free sending/receiving) in the literature. Snooping has been used by many wireless protocols (e.g., ExOR [13]) to reduce communication cost and control overhead. With TreeMAC, the packet scheduling is bufferless, and the network throughput is bounded by $1/3$ of the optimum in theory (assuming reliable links). To our best knowledge, TreeMAC is the first localized TDMA MAC protocol to achieve these nice properties. Our experiments on a 24 node test bed demonstrate that TreeMAC protocol significantly improves the network throughput and energy efficiency, by comparing to the TinyOS's default CSMA MAC protocol and a recent TDMA MAC protocol Funneling-MAC [8].

The rest of the paper is organized as follows. In Section II, we review the existing MAC protocol for sensor networks. In Section III, we introduce the principles and design of TreeMAC protocol and analyze its theory properties. In Section IV, we present our system design and implementation details by considering various limitations and dynamics in a practical network environment. We then evaluate the system performance in Section V. Finally, we conclude the paper in Section VI.

II. Related Works

The earlier sensor network MAC protocols mainly address energy conservation in low-duty cycle applications. S-MAC [6] expects sensor networks to be deployed in an ad hoc fashion, with individual nodes remaining largely inactive for long periods of time, but then becoming suddenly active when some events are detected. Energy conservation and self-configuration are the primary goals of S-MAC, while per-node fairness and latency are less important. S-MAC reduces idle listening energy waste by putting sensor nodes to sleep periodically. Neighboring nodes form virtual clusters to auto-synchronize on sleep schedules. In T-MAC [14], instead of having fixed sleep-wakeups, sensor nodes determine sleep-wakeup schedules adaptively. The schedules are decided based on a node's own traffic and that of its neighbors. B-MAC [7] employs an adaptive preamble sampling scheme to reduce duty cycle and minimize idle listening. It supports on-the-fly re-configuration and provides well-defined interfaces to optimize throughput, latency, or power conservation. The protocol in [15] is a network-aware scheme in the sense that it considers route-through traffic when using rate control, and uses adaptive rate control mechanisms on top of CSMA to achieve energy efficiency and fairness. As pointed out in [8], they did not consider the funneling effect in high data rate data collection networks. After all, they are designed for energy conservation, not for throughput maximization.

Naturally, to improve network throughput and support real-time data delivery, schedule-based TDMA protocols have been studied in the literature. TRAMA [16] 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. It performs an adaptive election algorithm to overcome the drawback of wasting time slots. Seamlessly adapting the MAC behavior between TDMA and CSMA according to the level of contention was explored by [17] for a wireless one-hop environment using a scheme called Probabilistic TDMA (PTDMA). As in TDMA, real time is slotted and by adjusting the access probability of owners and that of non-owners, PTDMA adapts the behavior of MAC between TDMA and CSMA depending on contention. However, PTDMA, being designed primarily for a one-hop wireless LAN environment, does not deal with many difficulties that TDMA faces in multi-hop networks. Inspired from that, Z-MAC [18] uses CSMA as the baseline MAC scheme, but uses a TDMA schedule as a hint to enhance contention resolution. Z-MAC uses DRAND (Distributed RAND) [19] to compute time slot assignments. However, DRAND is a complex coloring algorithm that allocates time slots to every node ensuring that no two nodes among a two-hop neighborhood are assigned to the same time slot by broadcasting the TDMA schedule of each node to its two hop neighbors. Because of the overhead of running DRAND, Z-MAC does not recommend to run DRAND periodically hence it is not really traffic adaptive.

In high-rate data collection networks, the nodes closer to the sink are heavily congested, since all data is forwarded toward the sink through those nodes, which have limited buffer size. This is called funneling effect. Funneling-MAC [8] realizes this problem and proposes a hybrid TDMA and CSMA protocol. It uses TDMA scheduling in the intensity region to mitigate the funneling effect, while keeping CSMA in the rest of the network to provide flexibility. The funneling-MAC is sink-oriented because the burden of managing TDMA scheduling of sensor events in the intensity region falls on the sink node. It is essentially a centralized TDMA scheduling protocol. However, the funneling-MAC only operates in the intensity region close to the sink and not across the complete sensor field. It assumes that the sink node has relatively longer transmission range and can reach all nodes in intensity region. This assumption may be applicable in some applications, but not always true. In some environment applications, it is not uncommon that some stations of intensity region are behind barriers, or the sink's radio range is not long enough to reach all nodes in intensity region. In this paper, we

present a localized TDMA MAC protocol to achieve high throughput and low congestion, by making use of the unique characteristics of data collection network. It does not assume any node is different from others, and is fully distributed and localized. Our experiments on sensor network test bed also demonstrates its better performance than Funneling-MAC.

III. TreeMAC Protocol Principles and Design

In a many-to-one routing tree structure, to minimize congestion and maximize throughput, a key observation is that the time slots of depth-ℓ nodes shall be no less than the sum of the time slots of depth-$(\ell+1)$ nodes and the slot demand for the depth-ℓ nodes themselves. Hence, we can make use of the parent-child relationship to assign slots locally. The basic idea is that: if a node u gets X slots assigned from its parent, then it can assign at most $X-x$ slots to its children, where x is its own slot demand. This is the key observation inspiring the TreeMAC protocol design. Notice that, conventional TDMA MAC protocols assume random traffic and maximize slot reuse based on graph coloring. This assumption is not necessarily true in data collection network. With that assumption, some nodes may send out more data, but the sink does not necessarily get more data, since those data may be dropped in the congested funneling region. Appropriate slot reuse maximization shall consider proportional bandwidth demands.

TreeMAC protocol divides time into cycles, and divides each cycle into N frames and each frame into 3 slots {0,1,2}, as illustrated in Figure 2. For each node u, its assigned frame-slot pair (F_u, S_u) determines when u can send, receive or sleep, as will be described in algorithm 2.

The following notations will be used in the rest of the paper.

1) F_u: the assigned frames set of each cycle to node u. For sink node, F_{sink} is the set of all N frames.
2) $S_u \in \{0,1,2\}$: the assigned transmittable time slot of each frame to node u.
3) ℓ_u: the depth of node u in the routing tree, e.g., hop count to the sink.
4) β_u: the slot demands of node u. Note that there are many ways to reflect bandwith demand with tradeoffs. In this paper we use the total data rate of u's subtree plus that of itself.
5) γ_u: the number of pending packets generated by node u itself in the link layer packet queue. Certainly, $\gamma_u \le \beta_u$.
6) C_u: the children set of node u.
7) $c_u^k \in C_u$: the k-th child of node u.
8) p_u: the parent of node u.

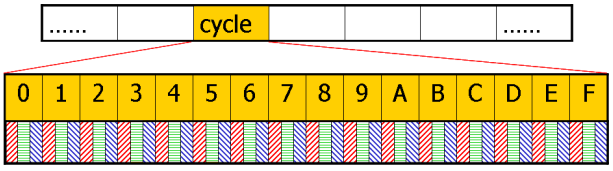


Fig. 2. Relationship of cycle, frame and slot: $cycle = N \cdot frames = 3N \cdot slots$. In this illustration, $N = 16$.

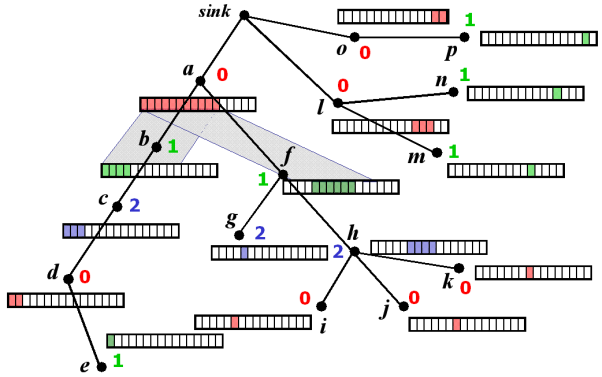


Fig. 3. Frame and slot assignment illustration in an example tree. For each node, the bar illustrates its frame assignment, and the number denotes its transmittable slot in its assigned frame.

Each node u calculates its *transmittable slot* number S_u according its own depth ℓ_u in the tree: $S_u = (\ell_u - 1) \mod 3$. For example, in Figure 3, the node b has depth $\ell_b = 2$, hence $S_b = 1$. It means that node b's transmittable slot number is 1 in its each assigned frame, that is to say, node b may transmit data in slot 1, but definitely not in slot 0 or slot 2. Whether a node can send or not in a time slot also depends on its frame assignment F_u (see algorithm 2). Notice that, assigning *transmittable slot* based on tree depth eliminates vertical 2-hop interferences, as will see in theorem 1.

For frame assignment, as mentioned earlier, the number of transmitting slots in depth-ℓ should be no less than the sum of the number of transmitting slots in depth-$(\ell+1)$ and the slot demands of depth-ℓ's own data. Hence, each depth-ℓ node could assign different subset of its own frames to different children, then all depth $\ell+1$ nodes will get non-overlapping frames. For instance, in Figure 3, node a got 11 frame assignment from its parent, then it may assign $[0,3]$ frame range to node b and $[4,9]$ frame range to node f, then message from node b and f will never conflict with each other. This recursive frame assignment eliminates horizontal 2-hop interferences, as will see in theorem 1.

Formally speaking, in each cycle, each node runs the following frame-slot assignment procedures in Algorithm 1.

Algorithm 1 TreeMAC frame-slot assignment

Each cycle is divided to N frames. In each cycle,

1) each node u calculates its *transmittable time slot* $S_u = (\ell_u - 1) \mod 3 \in \{0,1,2\}$.
2) each node u assigns its children frames such that: (1) $F_{c_u^i} \subseteq F_u$; (2) $F_{c_u^i} \bigcap F_{c_u^j} = \phi$ (if $i \neq j$) (3) $size(F_{c_u^i}) = size(F_u) \cdot \beta_{c_u^i} / (\sum_{c_u^k \in C_u} \beta_{c_u^i} + \gamma_u)$.

Algorithm 1 is simple yet powerful. TreeMAC's frame-slot assignment is a two dimensional approach: frame assignment eliminates horizontal two-hop interference, and slot assignment eliminates vertical two-hop interference. The graph coloring approaches used by conventional TDMA require at least 2-hop message exchanges to (try to) avoid hidden terminal

problems, while TreeMAC ensures conflict-free scheduling by making use of the unique characteristics of many-to-one routing tree. Each node's transmittable time slot number only depends on its hop distance to the sink, which can be easily obtained from its own routing table. The frame assignment from parent to children can be piggybacked in data packets of previous cycle. Hence, the frame-slot assignment has near-zero control overhead. Each node's message transmission will not collide with the transmission from its parent, children, grandparent and grandchildren, since they get different transmittable slot number. In other words, the vertical two-hop interference is mitigated. Also, each node's message transmission will not collide with its sibling nodes (and their parents/children), since they get non-overlapping frames per cycle therefore can not transmit concurrently. In fact, we later will show that (in theorem 1), given any node, there is at most one active sender in its neighborhood.

After getting its frame-slot assignment, each node performs the packet scheduling as described in Algorithm 2.

Algorithm 2 TreeMAC packet scheduling

Let F be the current frame per cycle and S be current slot per frame, each node u takes the following actions:
switch(F, S)

1) $F \in F_u$:
   a) **case** $S = S_u$**:** node u may send a packet immediately. If there is no packet to send, it switches to sleeping mode to conserve energy.
   b) **case** $S = S_{p_u}$**:** node u switches to receiving mode to receive beacon or command and control information from parent.
   c) **case** $S = S_{c_u}$**:** node u switches to receiving mode to receive data message from children.
2) $F \notin F_u$:
   a) **case** $S = S_u$**:** node u may switch to sleeping mode to conserve energy, or to receiving mode to snoop the neighborhood.
   b) **case** $S = S_{p_u}$**:** node u switches to receiving mode to receive command and control information from parent, or snoop the neighborhood. If node u know its parent's frame set F_{p_u} and knows $F \notin F_{p_u}$, it can switch to sleeping mode as well.
   c) **case** $S = S_{c_u}$**:** node u may switch to sleeping mode to conserve energy, or to receiving mode to snoop the neighborhood.

As described in Algorithm 2, if a slot is not for sending or receiving, a node can switch to sleeping mode to conserve energy and prolong network lifetime. For some uplayer protocols (e.g., ExOR [13]) making use of snooping to improve communication efficiency, nodes may choose to stay at *receiving* mode. We will later show that TreeMAC guarantees collisions-free even at a snooper, hence those snooping-depended protocols will work effectively with TreeMAC.

In the following, we analyze the theory properties of TreeMAC protocol by adopting some simplified network models. We assume every node has same transmission power and many-to-one routing protocol will generate Shortest Path Tree (SPT), e.g., each node finds the lowest cost path to the root. We adopt the widely used protocol interferences model. In this model, a transmission by a node v_i is successfully received by a node v_j iff v_j is not in the transmission range of the source of any other simultaneous transmission. The protocol interference model does not necessarily provide a comprehensive view of reality due to the aggregate effect of interference in wireless networks. However, it does provide some good estimations of interference and most importantly it enables a theoretical performance analysis of a number of protocols designed in the literature. We understand the interference is the sum of all emissions by interferers (e.g., unintended senders). However, if a interferer's radio signal reaches a node v weakly, which can not be decoded by node v, then v can not tell who is the interferer anyway. It is reasonable to count those weak radio signals as background noises, which shall already be counted in the link quality measurement of routing protocols.

Theorem 1: With TreeMAC protocol, for any node, at any time, there is at most *one* active sender in its neighborhood (including itself). We call it conflict-free packet sending/receiving and snooping.

Proof: In the literature, conflict-free packet scheduling usually means the packet sending/receiving is conflict-free. In other words, the concurrent communication pairs do not conflict with each other, e.g., sending and receiving are conflict free. Here TreeMAC supports a much *stronger* conflict-free packet scheduling: message snooping is also conflict-free. That is to say, even for a snooper, at any time, there is at most *one* active sender in its neighborhood. Snooping, uniquely enable by the radio broadcast nature, has been used by many protocols [13], [20]–[22], [26] to reduce communication cost. The difference of those concepts are illustrated in Figure 4.

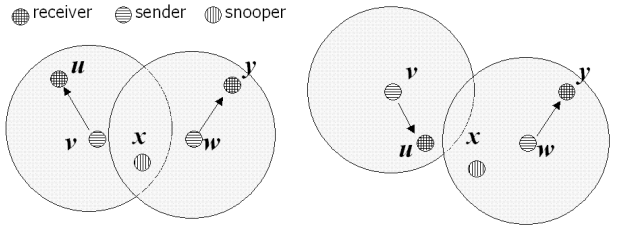


Fig. 4. The definition of conflict-free. **(Left) conflict-free sending/receiving**. Concurrent transmissions $v \rightarrow u$ and $w \rightarrow y$ are conflict-free, but collides at a snooper x. **(Right)conflict-free sending/receiving and snooping**. Concurrent transmissions $v \rightarrow u$ and $w \rightarrow y$ are conflict-free, and have no collisions at snooper x.

To prove *conflict-free sending/receiving and snooping*, it is equivalent to show that: *given any node, at any time, there is at most* one *active sender in its neighborhood (including itself)*. We prove this result by inducing contradictions. Hypothesize that, two sender v and w sends at same time and their messages collides at a receiver/snooper x. We will show this can not happen with TreeMAC protocol.

The relationship of two senders v and w can be classified into the following three cases:

1) The depth difference between v and w is 0. According to algorithm 1, they have same transmittable slot (e.g., $S_v = S_w$), but have non-overlapped frames, since $F_{c_u^i} \bigcap F_{c_u^j} = \phi$ (if $i \neq j$) and so do their ascendants. Hence v and w can not send concurrently. Contradiction is induced.

2) The depth difference between v and w is 1 or 2. According to algorithm 1, they must have different slot, since $S_u = (\ell_u - 1) \mod 3 \in \{0, 1, 2\}$. Hence v and w can not send at same time. Contradiction is induced.

3) The depth difference between v and w is more than 2. First of all, if $(\ell_w - \ell_v) \mod 3$ equals to 1 or 2, similar as above, they must have different slot and can never send at same time. Then the only left case is $(\ell_w - \ell_v) \mod 3 = 0$, indeed, they could transmit simultaneously, e.g., at same slot in same frame. W.L.O.G, we may assume $\ell_w - \ell_v = 3$ (the other cases can be proved similarly). Also, notice that $abs(\ell_x - \ell_v) \leq 1$ since they communicate directly in a tree. Base on the hypothesis that node w can reach x, then x must be able to reach w too. Then if w chooses x as parent, then the depth of w should be reduced. Hence the given SPT is not be a shortest path tree. Contradiction is induced.

Consequently, the hypothesis is wrong. That is to say, given any node, there is at most one active sender in its neighborhood. ■

To our best knowledge, this is the first MAC protocol to support conflict-free snooping, besides conflict-free sending/receiving.

Theorem 2: TreeMAC protocol is a fair and bufferless packet scheduling protocol.

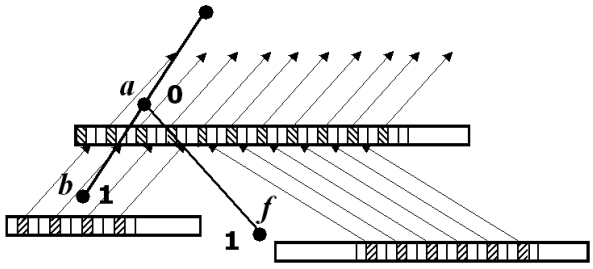


Fig. 5. Illustration of bufferless scheduling in TreeMAC. Node a is the parent of node b and node f. Once a receives a data in slot 1 from either b or f, it will forward it up in slot 0 of next frame.

Proof: In Algorithm 1, each node u assigns $F_{c_u^k}$ frame set to its child c_u^k, such that $F_{c_u^i} \bigcap F_{c_u^j} = \phi$ (if $i \neq j$) and $size(F_{c_u^i}) = size(F_u) \cdot \beta_{c_u^i} / (\sum_{c_u^k \in C_u} \beta_{c_u^i} + \gamma_u)$. That is to say, children get a subset number of slot of a parent. The ratio $size(F_{c_u^i}) / size(F_{c_u^j})$ is the ratio of their bandwidth demands. In other words, it is fair.

To show bufferless, recall that parent and children has adjacent transmittable slots per frame, as illustrated in figure 5. Once the parent receives a packet from a child, it could forward it up in next slot (or next frame). Hence, parent does not need to buffer packets for more than one frame time. In other words, it is a bufferless packet scheduling, which minimizing the probability of network congestion. In addition, this also means that the sink node can receive a packet from its children in every frame. ■

Theorem 3: The throughput of TreeMAC protocol is at least $1/3$ of the optimum.

Proof: It is easy to see that the optimum solution is that, at most, the sink can receive a packet in every slot. With TreeMAC protocol, as illustrated in Theorem 2, the sink can receive a packet in every frame or every 3 time slots. Hence the throughput is bounded by $1/3$ optimum. ■

In certain network configurations (if applicable), as illustrated in Figure 6), TreeMAC can even maximize the throughput to the optimum:

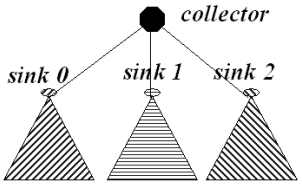


Fig. 6. Possible network configuration to maximize throughput: (1) three sinks are used to bridge sensor networks and a collector. (2) three different sink trees use three different channels; (3) given a node u in the sub-tree of sink i, the transmittable slot $t_u = (\ell_u + i) \mod 3$.

1) Configure three sinks to bridge sensor networks and a collector (e.g., a PC or laptop with a radio attached).
2) The i-th sink-tree uses i-th frequency channel for communications.
3) Each node u in the i-th sink-tree calculates its transmittable slot $t_u = (\ell_u + i) \mod 3$ (including the sink with $\ell_{sink} = 0$), instead of $t_u = (\ell_u - 1) \mod 3$ in algorithm 1.

With this configuration, the three sinks may deliver data to the collector in a round-robin fashion: in each frame of each cycle, the i-th sink use i-th channel in i-th slot to forward data to the collector. With this configuration, the collector can get data in every time slot, instead of every three time slots, the network throughput is therefore maximized to optimum. This network configuration is feasible even preferred in some field deployments [23]. Using three different channels on three subnetworks increases system robustness and throughput. In case of link broken or link quality degradation in one subnetwork, the affected child node shall join the other subnetworks, through appropriate channel surfing [24] technique. The basic idea is that, if node can not find a parent in one channel, it tunes to the other two channels. If it finds a better alternative parent node in channel i, it may switch to the channel i and connects to the new parent.

To our best knowledge, TreeMAC is the *first* localized TDMA MAC protocol to achieve all theses nice theory properties: (1) sending/receiving and snooping are all conflict-free; (2) packet scheduling is fair and bufferless; (3) network throughput is bounded by a constant ratio of optimum. These analysis establishes the theory foundation of TreeMAC, and our experiments later show that it indeed improves network throughput and reduces network congestion significantly in real sensor network testbed.

IV. TreeMAC Protocol Implementation

In this section, we introduce the implementation details of TreeMAC protocol in TinyOS sensor network test bed.

A. System Initialization and Time Synchronization

When the network starts, every node runs CSMA to discover neighbors, run data collection routing protocols and establish time synchronization. Once the network is time synchronized, the sink node initializes the TreeMAC mode by assigning schedules to its children. For any node who receives schedules from its parent, it will switch to TreeMAC mode and start to assign the frames to its own children as well. During the transition period from CSMA to TreeMAC mode, some nodes in CSMA mode may potentially compete channel with those nodes in TreeMAC mode. To avoid this, nodes in CSMA mode are set with longer initial Backoff window and congestion Backoff window, hence have less opportunity to grasp the channel. After all, those nodes in CSMA mode will eventually switch to TreeMAC mode, as parent will announce the schedule periodically.

In the literature, there are several known solutions on time synchronization [10]–[12]. In our testbed experiments, we adopt FTSP [11] (Flooding Time Synchronization Protocol), a multi-hop time synchronization protocol, which achieves high precision performance by utilizing MAC-layer time stamping and comprehensive error compensation including clock skew estimation. In the original FTSP work [11], timing errors of less than 67 microseconds were reported for an 11-hop network of Mica2 nodes. The protocol worked well in laboratory experiments. However, the original FTSP does not check the validity of synchronization messages, so a node reading an incorrect value can corrupt the calculation of the global time. In our implementation, the corrupted packet will be filtered to guarantee that only FTSP packets with correct time are processed to get an accurate global time. However, time synchronization with FTSP may still have milliseconds errors, and cause time slot unaligned. To avoid such mismatching error to propagate to new cycle. TreeMAC uses two timers of millisecond resolution for transmission scheduling: slot timer and cycle timer. When the cycle timer fires, it stops and restart the slot timer, so that the slot timer gets re-synchronized after each cycle.

B. Obtaining Tree Structure Information

TreeMAC protocol does *not* need separate tree formation and maintenance, instead, it obtains the parent and children information from the routing tables of data collection routing protocol. In our implementation, we utilized an improved version of MultiHopLQI in TinyOS-1.x as the routing protocol. MultiHopLQI is a distance vector routing protocol that proactively discovers a path of lowest cost the gateway for each node.

TreeMAC updates its frame-slot assignment periodically according to updated routing table and traffic. In data collection routing, the parent information is easy to get, but children information is not obvious as we can only passively learn whether a child is alive or left. In our implementation, we design a neighbor management module linked to the routing module, which maintains children set information based on upstream packets. Every time a sensor node receives a packet from a child, the liveliness of that child is set to a maximum value δ. When the liveliness times out, that is δ reaches 0, the child is removed from the children set. There is a tradeoff between the responsiveness to topology change and the protocol robustness. With a smaller δ, topology change can be detected with a shorter latency. However, sometimes a sensor node may fail to receive packets successfully from a child for certain period due to network congestion. It is observed from our test bed that sometimes some part of the network is "frozen" for up to a few seconds because of high contention. That is to say, no packets from the nodes in that part of the network arrive at the sink during that period. We choose $\delta = 6$ seconds as an appropriate value, based on our experimental test.

If a node changes parent during a cycle, we do not adjust frame and slot assignment immediately; instead, we wait until next cycle. We agree there could be some temporal mismatch, but it does not necessarily break the foundation: the number of time slots in depth-ℓ are no less than the sum of the number of time slots in depth-$(\ell + 1)$ and the number of slots for the depth-ℓ nodes' own data. Hence this temporal side effect is limited. In other words, TreeMAC is not fragile to tree topology updates; instead, it just makes use of the routing tree structure and its performance is not necessarily affected by topology changes.

C. Frame-Slot Schedule Assignment and Adjustment

After parent-child relationships are known, every node assigns frames to its children periodically. When a network stabilizes, the frame-slot schedule assignment run less frequently.

Before assigning frames, parent node needs to collect the bandwidth demand of its children. At the first glance, a node's bandwidth demand will be the total data rate R of its subtree (including itself). However, nodes may have different link reliability ratio to their parents. To ensure fairness, we adjust the bandwidth demand level to R/p, here p is the link reliability of a node to its parent. To notify parent of its bandwidth demand, a sensor node piggybacks this information in a routing beacon message. Thus only two extra bytes are used. This is one of the measures taken to reduce signaling overhead in the design.

As described in the algorithm 1, every node uses round robin method to assign frames to its children based on their proportional bandwidth demand. In our implementation, by default the number of frames per cycle is 24 and each slot is 20 ms. To guarantee every node has opportunity to transmit data, each node is at least assigned one frame. If the total number of required frames is more than 24, TreeMAC increases the cycle size adaptively to meet the demand. We allow multiple packets transmission in one lot. Bitmap is widely used in the design to encode schedules compactly and thus reduce

control message overhead. Each bit in a bitmap denotes the corresponding frame is assigned or not.

D. Pipelined Sending and Receiving between Layers

TinyOS is an event-driven system and does not support multi-threading. In the recommended practice of TinyOS-1.x, an up-layer component can not send another message to a low-layer component until the sendDone event of the previous message is received, which could cause a big gap between two consecutive sends. Similar situation happens during receiving. We enable pipelined sending/receive approach through buffering messages in sending/receiving queues at each component. Pipelined sending/receiving helps to improve performance of TDMA MAC protocol. Because when using a TDMA protocol, once a node is allocated a time slot, the node should use the slot efficiently, especially when high throughput is demanded. However, in simplex approach, when a packet is sent by the MAC layer, the MAC module will signal up a `sendDone` message, usually via posting a task. And only after the original sender processes the sendDone event, will it make another send request to MAC component. In this way, the precious channel time is wasted. In TreeMAC design, we overcome the problems by adding queues at each communication component. In this way, when the MAC finishes sending out one packet, it will immediately fetch the next packet in the queue, instead of waiting for the upper layers to send down a new packet.

V. Sensor Testbed Evaluation

A. Experiment Method

TABLE I
TreeMAC experiment parameters

Parameter	Value
Default data transmission power	-25 dBm
Slot Size	20 msec
Frame Size	3
Cycle Size	24
Bandwidth demand update interval	10 sec
Schedule update interval	8 sec
FTSP synchronization message interval	5 sec

TABLE II
Funneling-MAC experiment parameters

Parameter	Value
Default data transmission power	-25 dBm
Beacon & schedule transmission power($C_{control}$)	$-25 \sim 0$ dBm
Step size of power for dynamic depth-tunning	1 dBm
Beacon interval(t_b)	10 sec
Super frame size(t_f)	1 sec
Slot size(t_s)	30 msec
Moving average factor α	0.9

To evaluate the performance of TreeMAC, we conducted the experiments on a sensor network test bed with 24 Imote2 motes. The Imote2 sensor mote is an advanced wireless sensor node platform. It is built around the low-power PXA271 XScale CPU and also integrates an 802.15.4 compliant radio chip CC2420, which provides an effective data rate of 250 kbps. In our experiment, Imote2 sensor motes are configured to work in a low frequency(13 MHz) mode. Our testbed comprises of 24 imote2 sensor nodes. The sensor nodes are placed in a 4×6 grid without any obstruction. The data payload size is 74 bytes. We conducted experiments with varying network traffic and network topologies to evaluate the capacity of TreeMAC, and compared TreeMAC with Funneling-MAC and CSMA (the default MAC protocol in TinyOS-1.x for CC2420 radio). Originally, Funneling-MAC is only implemented on mica2 platform that uses CC1000 radio, so we ported it to CC2420 radio. The only change made is scaling the power level. The power level range in CC2420 is [1, 31] while that in CC1000 is [1, 255]. We did not compare TreeMAC to Z-MAC [18] and B-MAC [7], because Funneling-MAC [8] has shown better performance than them. Table I and II show the value of key parameters for TreeMAC and Funneling-MAC respectively in evaluation experiments. Under each network condition, the same experiment is repeated 5 times to get an average. In the experiments, network throughput, energy efficiency, network fairness, signaling overhead are compared between TreeMAC, Funneling-MAC and CSMA.

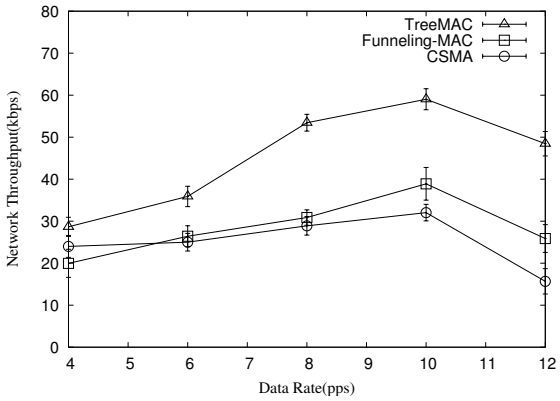
B. Network Throughput

One primary goal of TreeMAC is to achieve higher network throughput. We measure the network throughput by calculating how many data packets are successfully delivered to the sink from all nodes in one second.

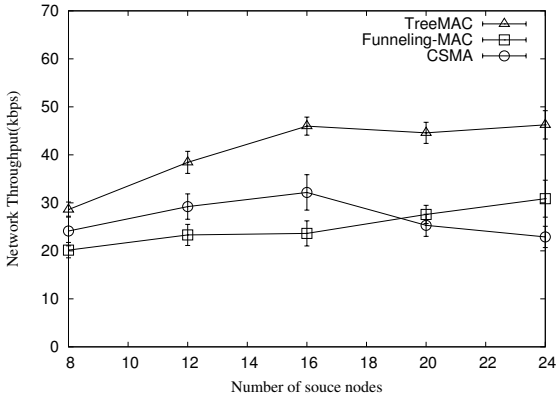
1) Throughput v.s. Data rate: First, we evaluated the network throughput with varying data rate and fixed network size 24. We conducted the experiment by varying data rate of each node at 5 levels: 4, 6, 8, 10 and 12 pps (packet per second). The results in Figure 7 (a) show that, when the injected data rate is 10 pps, all the three MAC protocols achieve their best throughput: the mean throughput for TreeMAC, Funneling-MAC and CSMA are 59.03 kpbs, 38.90 kpbs and 32.04 kpbs respectively. Between the window of [4 pps, 10 pps], the throughput increases as more data is injected. This is because the total data rate does not exceed the network capacity. Within the window of [10, 12], the network is congested and results in a decreasing throughput with a growth of data rate. Moreover, Figure 7 (a) shows that TreeMAC always achieves the best throughput compared to Funneling-MAC and CSMA.

2) Throughput v.s. Network size: We also evaluated network throughput with varying network size and fixed data rate 8 pps. We conducted the experiment by varying network size at 5 levels: 8, 12, 16, 20, 24. Figure 7 (b) shows that the throughput increases as the network size varies from 8 nodes to 24 nodes. Also, TreeMAC outperforms Funneling-MAC $50 - 90\%$ for all experimented network sizes.

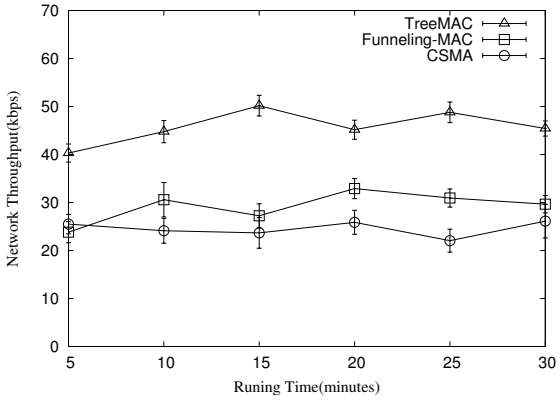
3) Throughput v.s Time: We next investigated the throughput stability of TreeMAC at running time. With a network size of 24 nodes and a data rate of 8 pps, we run the experiments for a long time and measure throughput for each 5-minute time window. The comparison between TreeMAC and Funneling-MAC is shown in Figure 7 (c). We can see



(a) Network throughput v.s. data rate



(b) Network throughput v.s. network size



(c) Network throughput v.s. time

Fig. 7. Network throughput

that at the beginning the throughput is slightly lower. That is because FTSP requires at least 4 time synchronization messages to get a node synchronized to the global time. After that, the throughput of TreeMAC keeps around 47 kbps with a fluctuation of no more than ± 7 kbps. This shows that TreeMAC is robust with respect to dynamic topology (such as routing paths change) and can obtain stable performance in the long run.

C. Energy Efficiency

Maximizing energy efficiency is always an important goal to achieve in MAC protocol design. Energy efficiency [25] is defined as the ratio of the number of delivered distinct packets to the gateway over the total number of packet transmissions in the network. The total number of transmissions includes all link layer retransmissions, as well as transmissions associated with packets that are dropped or corrupted. The energy efficiency η is formally defined as: $\eta = \frac{\sum_{d \in D} hops(d)}{\sum_{p \in P} \sum_{h \in hops(p)} xmits(p,h)}$ It is a value between 0 and 1. D is the set of packets delivered to the sink; P is the set of all injected packets; $hops(p)$ ranges over each hop packet p traverses; $xmits(p,h)$ denotes the number of transmissions a packet p undergoes at hop h.

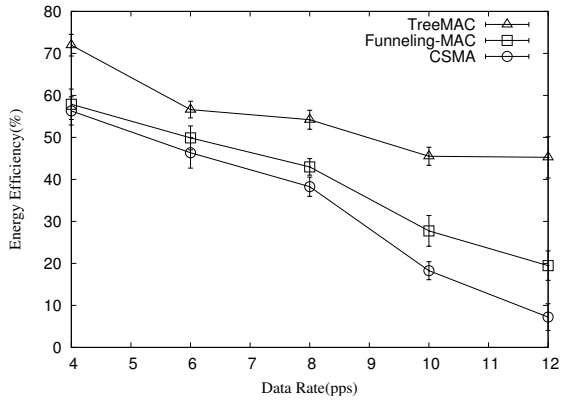


Fig. 8. Energy efficiency

We evaluate the energy efficiency by fixing network size at 24 and varying data rate at 5 levels: 4, 6, 8, 10 and 12 pps (packet per second). Figure 8 shows that both TreeMAC and Funneling-MAC save more energy than CSMA. This is the advantage of TDMA MAC protocol, as the media spectrum is scheduled for access. It is also shown that, TreeMAC is more energy efficient than Funneling-MAC. Especially when the data rate is above 10 pps, the gain is up to twice more energy efficient compared to Funneling-MAC. The high energy efficiency of TreeMAC lies in the fact that TreeMAC assigns slots according to proportional bandwidth demand. TreeMAC would rather drop a packet at the source node, than drop it at the intermediate nodes. In wireless network, buffer overflow occurs when a parent node has to discard a received packet from its children because its buffer queue is full. If children nodes sends more packets than their parents can handle, it will result in a lot buffer overflow. This experiments further proves that: in data collection network, slot reuse maximization based on graph coloring does not necessarily mean throughput maximization to the sink. Appropriate slot reuse in TreeMAC actually improves throughput and reduces energy cost.

D. Network Fairness

When a sensor network is deployed, users are typically interested to get real-time data from whole monitored area.

However, high channel contention in a sensor network can cause channel starvation for some nodes/areas. We expect all nodes in a sensor network can fairly deliver their packets to the sink. We utilize the definition of fairness index ϕ from [25]: $\phi = \frac{(\sum_{k=1}^{N} r_i)^2}{N \times \sum_{k=1}^{N} {r_i}^2}$. Here, r_i is the average rate of packets delivered from the i_{th} sensor and N is the number of sensors in the network. If each node sends the same number of packets

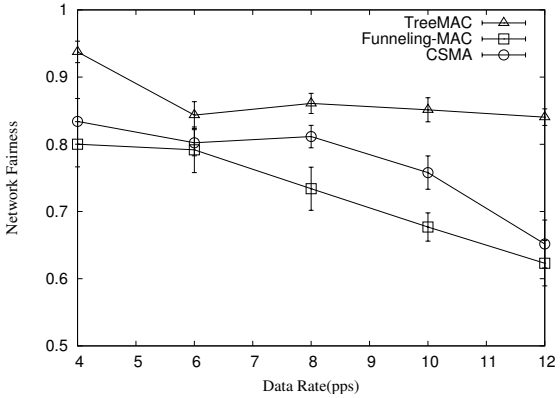


Fig. 9. Network fairness

to the sink, then the fairness index is 1. Figure 9 shows the comparison of fairness among TreeMAC, Funneling-MAC and CSMA. It shows that as data rate varies from 4 pps to 12 pps, TreeMAC always achieves a network fairness above 85%. Also, we can see that, with TreeMAC, varying data rate does not affect the fairness much. That is because TreeMAC assigns slots proportionally to each node's bandwidth demand. Interesting to notice that, Figure 9 also shows that CSMA is better than Funneling-MAC in terms of fairness. The reason may be that, those nodes outside of the funneling region may still overflow the buffer of the nodes in the funneling region, as they are not controlled by the Funneling-MAC protocol.

E. Signaling Overhead

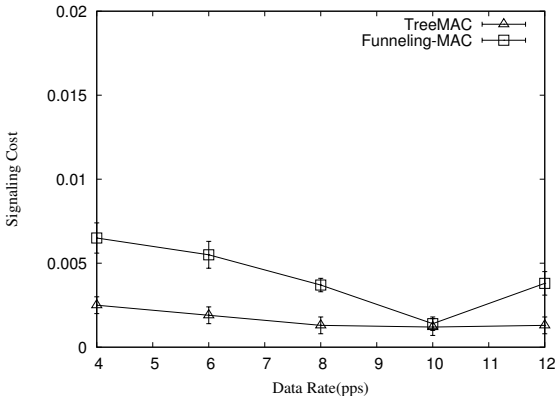


Fig. 10. Signaling overhead

Both TreeMAC and Funneling-MAC use TDMA mechanism, and thus have some signaling overhead. CSMA does not need extra control messagees, so we only compare the signaling overhead between TreeMAC and Funneling-MAC. The signaling overhead is the dissemination of bandwidth demand and frame-slot schedule messages. For Funneling-MAC, the signaling overhead includes beacon packets, schedule packets, path information field, and meta-schedule. The signaling overhead index is defined as the ratio of total control packet bits over total data bits that reach the sink. Figure 10 illustrates that the signaling overhead index of TreeMAC is less than 0.0025 for all tested data rates. TreeMAC achieves lower signaling overhead than Funneling-MAC.

VI. Conclusion

Many-to-one communication pattern is fundamentally different from random any-to-any communication pattern, where equal channel access does not mean fairness and slot reuse maximization based on graph coloring does not mean data throughput maximization to the sink. TreeMAC is an innovative localized TDMA MAC protocol and designed to achieve high throughput and low congestion with low overhead, by utilizing unique characteristics of data collection network. We have showed the nice properties of TreeMAC in theory and demonstrated that it achieves much better throughput and energy efficiency than CSMA and Funneling-MAC [8] in a real sensor network test bed.

References

[1] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proc. 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, WA, USA, Nov. 2006.

[2] W.-Z. Song, B. Shirazi, R. Lahusen, S. Kedar, S. Chien, F. Webb, J. Pallister, D. Dzurisin, S. Moran, M. Lisowski, D. Tran, A. Davis, and D. Pieri, "Optimized autonomous space in-situ sensor-web for volcano monitoring," in *2008 IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2008.

[3] R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, L. Krishnamurthy, N. Kushalnagar, L. Nachman, and M. Yarvis, "Design and deployment of industrial sensor networks: Experiences from the north sea and a semiconductor plant," in *Pro. 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, San Diego, CA, USA, Nov. 2005.

[4] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proc. 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Baltimore, MD, USA, Nov. 2004.

[5] S. N. Pakzad, S. Kim, G. L. Fenves, S. D. Glaser, D. E. Culler, and J. W. Demmel, "Multi-purpose wireless accelerometers for civil infrastructure monitoring," in *5th International Workshop on Structural Health Monitoring (IWSHM)*, Stanford, CA, USA, Sep. 2005.

[6] W. Ye, J. Heidemann, and D. Es, "An energy-efficient mac protocol for wireless sensor networks," in *Proc. 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, New York, NY, USA, Apr. 2002.

[7] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Baltimore, MD, USA, Nov. 2004.

[8] G.-S. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, and F. Cuomo, "Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks," in *Proc. 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, CO, USA, Nov. 2006.

[9] W.-Z. Song, F. Yuan, and R. Lahusen, "Time-optimum packet scheduling for many-to-one routing in wireless sensor networks," in *Proc. 3rd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Vancouver BC, Canada, Oct. 2006.

[10] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, USA, Dec. 2002.

[11] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proc. 2nd international conference on Embedded networked sensor systems (SenSys 2004)*, Baltimore, MD, USA, Nov. 2004.

[12] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st international conference on Embedded networked sensor systems (SenSys 2003)*, Los Angeles, CA, USA, Nov. 2003.

[13] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," in *the annual conference of the Special Interest Group on Data Communication (SIGCOMM)*, Philadelphia, PA, Aug. 2005.

[14] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proc. 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, CA, USA, Nov. 2003.

[15] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks." in *Proc. 7th annual international conference on Mobile computing and networking (MOBICOM)*, Rome,Italy, Jul. 2001.

[16] V. Rajendran, K. Obraczka, and Garcia, "Energy-efficient, collision-free medium access control for wireless sensor networks," in *Proc. 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, CA, USA, Nov. 2003.

[17] A. Ephremides and O. A. Mowafi, "Analysis of a hybrid access scheme for buffered usersprobabilistic time division." in *IEEE Transactions on Software Engineering*, vol. 8, no. 1, pp. 52–61, Jan. 1982.

[18] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a hybrid mac for wireless sensor networks," in *Proc. 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, San Diego, CA, USA, Nov. 2005.

[19] I. Rhee, A. C. Warrier, and L. Xu, "Randomized dining philosophers to TDMA scheduling in wireless sensor networks," in *Technical report, Computer Science Department, North Carolina State University, Raleigh, NC.*, Apr. 2004.

[20] K.-H. Kim and K. G. Shin, "On accurate measurement of link quality in multi-hop wireless mesh networks," in *Proc. 12th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Los Angeles, CA, USA, Sep. 2006.

[21] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza, "RBP: robust broadcast propagation in wireless networks," in *Proc. 4th international conference on Embedded networked sensor systems (SenSys 2006)*, Boulder, CO, USA, Nov. 2006.

[22] P. Kyasanur, R. R. Choudhury, and I. Gupta, "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks," in *Proc. 3rd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Vancouver BC, Canada, Oct. 2006.

[23] "OASIS: Optimized Autonomous Space In-Situ Sensor Web. http://sensorweb.vancouver.wsu.edu."

[24] W. Xu, W. Trappe, and Y. Zhang, "Channel surfing: Defending wireless sensor networks from jamming and interference," in *International Conference on Information Processing in Sensor Networks (IPSN)*, Cambridge, MA, USA, Apr. 2007.

[25] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *Proc. 2nd international conference on Embedded networked sensor systems (SenSys 2004)*, Baltimore, MD, USA, Nov. 2004.

[26] Y. Peng, W. Song, R. Huang, M. Xu, and B. Shirazi, "Cacades: a reliable dissemination protocol for data collection sensor network," in *2009 IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2009.