Pervasive and Mobile Computing 5 (2009) 750–765

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks☆

Wen-Zhan Song[a,*], Renjie Huang[a], Behrooz Shirazi[a], Richard LaHusen[b]

[a] Sensorweb Research Laboratory, Washington State University, Vancouver, WA 98686, United States

[b] Cascades Volcano Observatory, U.S. Geological Survey, United States

ARTICLE INFO

Article history:
Received 26 February 2009
Received in revised form 7 June 2009
Accepted 2 July 2009
Available online 14 July 2009

Keywords:
TreeMAC
TDMA
Sensor network
Realtime
High-data-rate
Fairness

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 each frame into *slots*. A parent node determines the 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. First, given any node, at any time slot, there is at most one active sender in its neighborhood (including itself). Second, the packet scheduling with TreeMAC is bufferless, which therefore minimizes the probability of network congestion. Third, the data throughput to the gateway is at least 1/3 of the optimum assuming reliable links. Our experiments on a 24-node testbed show that TreeMAC protocol significantly improves network throughput, fairness, and energy efficiency compared to TinyOS's default CSMA MAC protocol and a recent TDMA MAC protocol Funneling-MAC. Partial results of this paper were published in Song, Huang, Shirazi and Lahusen [W.-Z. Song, R. Huang, B. Shirazi, and R. Lahusen, TreeMAC: Localized TDMA MAC protocol for high-throughput and fairness in sensor networks, in: The 7th Annual IEEE International Conference on Pervasive Computing and Communications, PerCom, March 2009]. Our new contributions include analyses of the performance of TreeMAC from various aspects. We also present more implementation detail and evaluate TreeMAC from other aspects.

1. Introduction

In typical sensor network applications, sensor nodes monitor fields and generate events which traverse hop-by-hop towards 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 consumption. It also affects 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. 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

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

* Corresponding author.
E-mail addresses: songwz@wsu.edu (W.-Z. Song), renjie_huang@wsu.edu (R. Huang), shirazi@wsu.edu (B. Shirazi), rlahusen@usgs.gov (R. LaHusen).

1574-1192/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2009.07.004

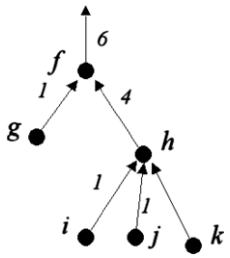


Fig. 1. The fairness of MAC protocol in data gathering scenarios. In the figure, the numbers denote the bandwidth demand of corresponding links.

the nodes further away. For example, in Fig. 1, assume all nodes in one collision domain, every node delivers data towards 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 [1]). To ensure fairness, node f and h should get 6 times and 4 times the opportunities of nodes g, i, j, k. In other words, nodes in a network should get channel access opportunity proportional to their demands [2]. 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 networks where *equal* means *fair*. *This is the key motivation of TreeMAC design*.

Earlier sensor network MAC protocols [3,4] are designed for low duty-cycle and low data-rate applications with the primary goal of energy conservation. More recent applications involve high-data-rate signals, such as monitoring industrial processes [5], geophysical environments [6,7] and civil structures like buildings and bridges [5,8,9]. For those high-data-rate applications, there is a key challenge on how to collect these high-fidelity signals subject to the limited radio bandwidth available to sensor nodes. In addition to the limited physical bit rate of the radios used on 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 network 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 neighborhoods 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.

Motivated by the need of localized TDMA MAC protocols for high throughput and fairness, we propose TreeMAC, an innovative MAC protocol. With TreeMAC, channel access is well regulated for throughput maximization and energy conservation, and every node gets a number of time slots proportional to its bandwidth demand. TreeMAC divides each cycle into frames and each frame into three slots. By making use of the parent–children relationship established by any data collection routing protocol, the frame–slot assignment is determined and exchanged between parent and children only, hence it is truly localized. The parent determines the 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; the frame assignment eliminates horizontal two-hop interference, and the 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 theoretically bounded by 1/3 of the optimum (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 testbed show that TreeMAC protocol significantly improves the network throughput and energy efficiency compared to the TinyOS's default CSMA MAC protocol and a recent TDMA MAC protocol Funneling-MAC [1]. Partial results of this paper were published in [14]. Our new contributions include Section 4, which analyzes the performance of TreeMAC from various aspects. We also present more implementation detail and evaluate TreeMAC from new aspects.

The rest of the paper is organized as follows. In Section 2, we review existing MAC protocols for sensor networks. In Section 3, we introduce the principles and design of TreeMAC protocol and analyze its theoretical properties. Section 4 theoretically analyzes the performance of TreeMAC from various aspects. In Section 5, we present the system design and implementation details by considering various limitations and dynamics in a realistic network environment. We then evaluate the system performance on a real sensor network testbed in Section 6. Finally, Section 7 concludes the paper and discusses future work.

2. Related works

The earlier sensor network MAC protocols mainly address energy conservation in low-duty cycle applications. S-MAC [3] expects sensor networks to be deployed in an ad hoc fashion, with individual nodes remaining largely inactive for long time periods, 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 [15], instead of having fixed sleep-wakeups, sensor nodes determine sleep-wakeup schedules adaptively. The schedules are determined based on a node's own traffic and that of its neighbors. B-MAC [4] employs an adaptive preamble sampling scheme to reduce duty cycle and minimize idle listening. It supports on-the-fly reconfiguration and provides well-defined interfaces to optimize throughput, latency, or power conservation. The protocol in [16] 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 [1], 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 [17] 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 and 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 [18] 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, 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 [19] uses CSMA as the baseline MAC scheme, and uses a TDMA schedule as a hint to enhance contention resolution. Z-MAC uses DRAND (Distributed RAND) [20] 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 running it periodically, hence it is not really traffic adaptive. Those MAC protocols are not optimized for data gathering sensor network applications. In high-rate data collection networks, the nodes nearest to the sink (the intensity region) often experience heavy congestion since all data is forwarded toward the sink through those nodes, which have limited buffer size. This is called the funneling effect. Funneling-MAC [1] is a hybrid TDMA and CSMA protocol proposed to address this problem. 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 the intensity region. This assumption may be applicable in some applications, but it is not always true. In some environmental applications, it is not uncommon that some stations of intensity region are behind obstacles, or the sink's radio range is not long enough to reach all nodes in the 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 networks. It does not assume that any node is different from any other node, and is fully distributed and localized. Our experiments on a sensor network testbed show that it can achieve higher network throughput and fairness than Funneling-MAC.

3. 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, as illustrated in Fig. 2. Hence, we can make use of the parent-child relationship to assign slots locally. The basic idea is 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 networks. With that assumption, some nodes may send out more data, but the sink does not necessarily get more data, since those data may worsen the network congestion and be dropped in the congested funneling region. Appropriate slot reuse maximization shall consider proportional bandwidth demands.

TreeMAC protocol divides time into cycles, each cycle into N frames and each frame into 3 slots $\{0, 1, 2\}$, as illustrated in Fig. 3. 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 bandwidth demand with tradeoffs. In this paper we use the total data rate of u's subtree plus that of itself.
5. C_u: the children set of node u.

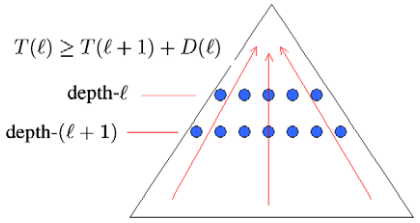


Fig. 2. The pattern of bandwidth demand. $T(\ell)$ denotes the demand of the depth-ℓ nodes. $D(\ell)$ indicates the bandwidth demand for the depth-ℓ nodes' own data.

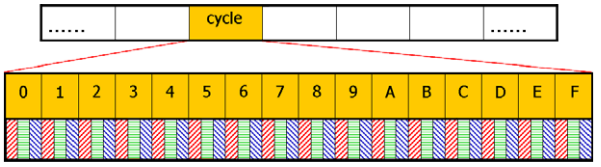


Fig. 3. Relationship of cycle, frame and slot: $cycle = N \cdot frames = 3N \cdot slots$. In this illustration, $N = 16$.

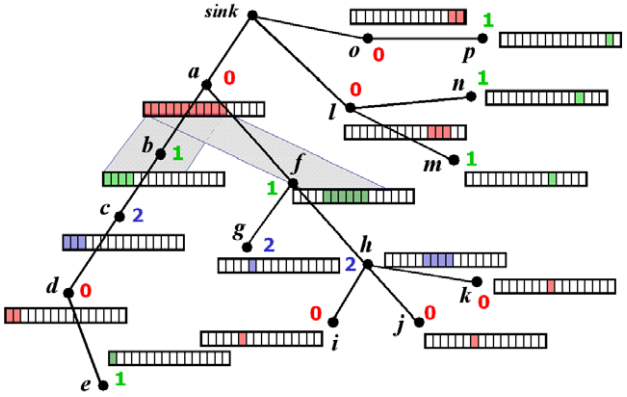


Fig. 4. Frame and slot assignment illustration in an example tree. For each node, the bar illustrates its frame assignment, and the number denotes its transmittable slot in its assigned frame.

6. $c_u^k \in C_u$: the kth child of node u.
7. p_u: the parent of node u.

Each node u calculates its *transmittable slot* number S_u according its own depth ℓ_u in the tree: $S_u = (\ell_u - 1) \bmod 3$. For example, in Fig. 4, 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 seen 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 subsets of its assigned frames to different children, then all depth $\ell+1$ nodes will get non-overlapping frames. For instance, in Fig. 4, node a got 11 frame assignments from its parent, then it may assign [0, 3] frame range to node b and [4, 9] frame range to node f, then messages from node b and f will never conflict with each other. This recursive frame assignment eliminates horizontal 2-hop interferences, as seen in Theorem 1.

Formally speaking, each node runs the frame–slot assignment procedure in Algorithm 1.

Algorithm 1 TreeMAC frame–slot assignment

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

1. each node u calculates its *transmittable time slot* $S_u = (\ell_u - 1) \bmod 3 \in \{0, 1, 2\}$.
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) $size(F_{c_u^i}) = size(F_u) \cdot \beta_{c_u^i} / (\sum_{c_u^k \in C_u} \beta_{c_u^i} + \beta_u)$.

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 numbers. 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 cannot 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 messages 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 knows 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 scheduled for sending or receiving, a node can switch to sleeping mode to conserve energy and prolong network lifetime. For some up-layer 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 remains collisions-free even at a snooper, hence these 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 the same transmission power, and a many-to-one routing protocol will generate a 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 the interferer is. It is reasonable to count these 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 that 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 enabled by the radio broadcast nature, has been used by many protocols [21–23,13,24] to reduce communication cost. The differences of those concepts are illustrated in Fig. 5.

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 senders v and w send at same time and their messages collide at a receiver/snooper x. We will show this cannot 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 cannot send concurrently. Contradiction is induced.

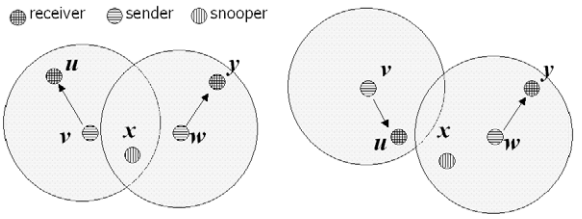


Fig. 5. The definition of conflict-free. **(Left) conflict-free sending/receiving**. Concurrent transmissions $v \to u$ and $w \to y$ are conflict-free, but collide at a snooper x. **(Right) conflict-free sending/receiving and snooping**. Concurrent transmissions $v \to u$ and $w \to y$ are conflict-free, and have no collisions at snooper x.

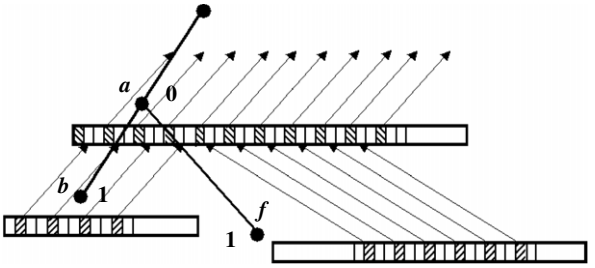


Fig. 6. Illustration of bufferless scheduling in TreeMAC. Node a is the parent of node b and node f. Once a receives a data packet in slot 1 from either b or f, it will forward it up in slot 0 of the next frame.

(2) The depth difference between v and w is 1 or 2. According to algorithm 1, they must have different slots, since $S_u = (\ell_u - 1) \bmod 3 \in \{0, 1, 2\}$. Hence v and w cannot send at the same time. Contradiction is induced.
(3) The depth difference between v and w is more than 2. First of all, if $(\ell_w - \ell_v) \bmod 3$ equals 1 or 2, similar as above, they must have different slots and can never send at the same time. Then the only case left is $(\ell_w - \ell_v) \bmod 3 = 0$. Indeed, they could transmit simultaneously, e.g., at the same slot in the 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 would 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 as well as conflict-free sending/receiving.

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

Proof. In Algorithm 1, each node u assigns the $F_{c_u^k}$ 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 $size(F_{c_u^i}) = size(F_u) \cdot \beta_{c_u^i} / (\sum_{c_u^k \in C_u} \beta_{c_u^i} + \beta_u)$. That is to say, children get a subset number of slots of their 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 that it is bufferless, recall that parent and children have adjacent transmittable slots per frame, as illustrated in Fig. 6. Once the parent receives a packet from a child, it could forward it up in the next slot (or the next frame). Hence, a parent does not need to buffer packets for more than one frame time. For example, as illustrated in Fig. 6, a packet sent by node b in slot 1 is forwarded by node a in slot 0 of the next frame. In other words, the packet scheduling is bufferless and it minimizes the probability of network congestion. The bufferless scheduling is especially important in these resource-constrained wireless sensor nodes since less memory is required to buffer the forwarding data. 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 Fig. 7, TreeMAC can even maximize the throughput to the optimum:

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 $t_u = (\ell_u + i) \bmod 3$ (including the sink with $\ell_{sink} = 0$), instead of $t_u = (\ell_u - 1) \bmod 3$ in Algorithm 1.

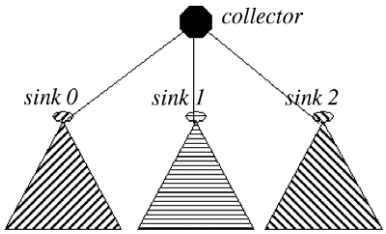


Fig. 7. Possible network configuration to maximize throughput: (1) three sinks are used to bridge sensor networks and a collector. (2) three different sink trees use three different channels; (3) given a node u in the sub-tree of sink i, the transmittable slot $t_u = (\ell_u + i) \bmod 3$.

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 uses 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 [25]. Using three different channels on three subnetworks increases system robustness and throughput. In case of broken links or link quality degradation in one subnetwork, the affected child node shall join the other subnetworks through the appropriate channel surfing [26] technique. The basic idea is that, if a node cannot 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 connect to the new parent.

To our best knowledge, TreeMAC is the *first* localized TDMA MAC protocol to achieve all these nice theoretical 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 the optimum. Analysis establishes the theory foundation of TreeMAC, and our experiments later show that it indeed improves network throughput and reduces network congestion significantly in a real sensor network testbed.

4. Performance analysis

We have proved that theoretically TreeMAC can achieve a network throughput bounded by 1/3 optimum and its bandwidth assignment is fair. Besides above properties, we further analyze the performance of TreeMAC from other aspects.

Delivery latency. Delivery latency is the time a packet takes to traverse from the source node to the gateway. Many multimedia applications such as video surveillance typically have stringent requirements on the packet delivery timeliness. In TreeMAC, once the parent receives a packet from a child, it will forward it in the next slot or the next frame. The next frame may be in a new cycle. Assume the TDMA cycle is T, then the average hop-delay is $O(T/3)$. In a topology tree of size n, the average depth of each node is $O(\log n)$. The time taken by a packet to traverse to the gateway is $O(\log n) \times O(T/3) = T/3 \times O(\log n)$. We can see that, when the network size is determined, the delivery latency is proportional to the TDMA cycle period T.

Responsiveness to topology dynamics. Topology dynamics are common in multi-hop wireless networks due to time-varying channel conditions, physical environmental changes, inaccurate link estimation, and node mobility. Handling the topology change requires a update of the transmission schedule. In TreeMAC, a parent node can immediately detect whether a node leaves/joins its children set by monitoring the continuous upstream data. So the convergence speed of the schedule update mostly depends on the time delay of the dissemination of scheduling messages. If a node u switches its parent from u_{p_0} to u_{p_1} and the common ancestor of u_{p_0} and u_{p_1} is u_p, the bandwidth demand of the nodes on the path $u_{p_0} \to \cdots \to u_p$ and $u_{p_1} \to \cdots \to u_p$ will be affected as the amount of data they need to forward has changed and it requires an update in the bandwidth allocation. In the best case, the update takes places in merely one-hop range. In the worse case, u_p is the sink. The time taken by the scheduling message to traverse hop by hop to get to intended nodes is $T/3 \times O(\log n)$.

Control overhead. The control message overhead of TreeMAC is primarily the periodic scheduling messages. In the best case, the topology is a balanced tree, thus half of the nodes are leaves, who do not need to send out scheduling messages since they have no children. In this case, the control message overhead is $n/(2T_S)$. Here T_S is the schedule beaconing period. In the worse case, the topology is a line and only one node is a leaf. So the message cost is $(n-1)/T_S$.

5. TreeMAC implementation

In this section, we introduce the implementation of TreeMAC in the TinyOS-1.x [27] operating system, running on the iMote2 sensor network platform. The Imote2 sensor mote is an advanced wireless sensor node platform. It is equipped with 256 KB SRAM, 32 MB SDRAM, and an 802.15.4 compliant radio chip CC2420 which provides an effective data rate of 250 kbps. Its PXA271 processor can be configured to work from 13 MHz to 416 MHz. We configured its PXA271 processor to operate in a low voltage (0.85 V) and low frequency (13 MHz) mode in normal operations.

The implementation adopts a cross-layer design to have a joint optimization, as illustrated in Fig. 8. It learns its parent and children set from the network layer. The link reliability known from transmission statistics in the link layer helps each node to allocate frames proportionally for its children based on demands. We activate the CCA (Clear Cannel Assessment)

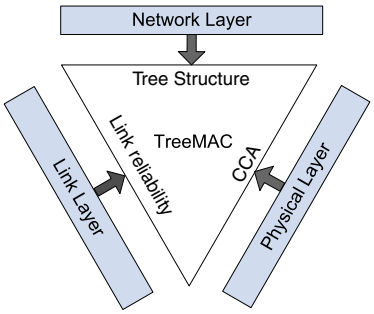


Fig. 8. The cross layer design of TreeMAC.

function of the CC2420 radio to enhance the protocol robustness and ease the need of fine-grained time synchronization. During the transmittable slots, a node starts transmission only when the CCA indicates the channel is idle. The CC2420 radio will backoff for a random period bounded by a time window when it detects the channel as busy.

5.1. System initialization

When the network starts, every node runs CSMA to discover its 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 for channel access 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 each parent will announce the schedule periodically.

When nodes are in TreeMAC mode, congestion backoff is enabled to resist interference from external environments. A white paper published by a European manufacturing group (associated with the development of a competing standard, Z-Wave) claims that wireless technologies such as ZigBee, which operate in the 2.4 GHz RF band, are subject to significant interference—enough to make them unusable [28]. It claims that this is due to the presence of other wireless technologies like Wireless LAN in the same RF band. For example, the typical TX power output of Wi–Fi transceiver is up to 20 dBm, while the maximum signal strength of most 802.15.4 radio chips is no more than 0 dBm [29]. It is obvious that a Wi–Fi signal can easily overwhelm the 802.15.4 radio signal. This is also confirmed in our lab experiments. We set a pair of sensor nodes S (sender) and R (receiver) to maximum power level 0 dBm. It shows that when either initial backoff (the delay before a packet transmission is attempted on the channel) or congestion backoff (the backoff time period when the channel is sensed busy) is turned off, the radio link between S and R suffers high packet loss up to 20%, while in the case they are turned on, the packet loss is near 0. Moreover, CSMA uses a large backoff window size to avoid high channel intention and successive collision. On the contrary, the most significant interference for TreeMAC is external, rather than inter-node, therefore a shorter backoff window is preferred to provide better throughput.

5.2. Time synchronization

In the literature, there are several known solutions on time synchronization [10,12,11]. 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 μs 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 the correct time are processed to get an accurate global time. In our experiments, the timing errors normally are no more than 1 ms (the time service to measure the error is of 1 ms resolution). That meets the requirements of TreeMAC since we enable CCA carrier sensing to ease the need of fine-grained time synchronization.

FTSP synchronizes all nodes' time to a global clock, but it does not guarantee time slots are aligned since different nodes may start their timer at different time points. To solve that problem, we designed a RTC (Real Time Clock) module, which maintains a millisecond resolution timer, with the support of iMote2's microsecond resolution clock. The timer service provided by RTC is different from the default timer service in TinyOS [27]; it synchronizes its clock value t to the UTC time and fires based on the clock value instead of the interval. In other words, the timer fires when the clock value t satisfies $t\%T = 0$, where T is the sampling interval. For example, if $T = 10$ ms and the timer starts at 20 : 00 : 00 : 422, then the next fire point is 20 : 00 : 00 : 430, not 20 : 00 : 00 : 432. This enables strict synchronized timing event triggering: the slot

timer of all nodes in the network will trigger at the same time point. However, time synchronization with FTSP may still have milliseconds of errors, and cause unaligned time slots. The problem is exacerbated in a multi-hop network. To prevent such mismatch errors from propagating to new cycles, TreeMAC uses two timers of millisecond resolution for transmission scheduling: slot timer and cycle timer. When the cycle timer fires, it stops and restarts the slot timer, so that the slot timer gets re-synchronized in each cycle.

5.3. 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 to the gateway for each node.

TreeMAC updates its frame–slot assignment periodically according to an 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 has left. In our implementation, we designed a neighbor management module linked to the routing module, which maintains the 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 a 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 $\delta = 6$ s 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.

5.4. 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, a 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 toward their parents. To ensure fairness, we adjust the bandwidth demand level to R/p, where p is the link reliability of a node to its parent. To notify its parent of its bandwidth demand, a sensor node piggybacks this information in upstream packets. Originally the demand information is piggybacked in routing beacon messages. The advantage is that the signaling overhead is rather low. However, because beacon packets do not require acknowledgement, they are likely to be lost. In addition, the beacon interval can be as large as several cycles. As a result that may make TreeMAC not responsive enough to topology update (topology update will result in change of bandwidth demand). On the contrary, piggyback throughput upstream packets can provide much better reliability and consistency.

As described in the algorithm 1, every node uses a 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 that every node has an opportunity to transmit data, each node is at least assigned one frame if its bandwidth demand is not 0. If the total number of required frames is more than 24, TreeMAC can increase the cycle size adaptively to meet the demand. Although TreeMAC algorithm supports fraction frame allocation, that is multiple nodes share the same frame in different cycles in a round robin way when the network size is large, the current implementation only supports integer frame allocation. We also allow multiple packets transmission in one time lot, if possible. The basic principle is that each node should fully utilized its own slots. How to efficiently encode schedules for children is also important to reduce the control message overhead. Normally encoding a frame's position in a cycle of size N can take up to the order of $\log N$ bits. In our design,a bitmap is utilized to improve the schedule encoding efficiency. Each bit in the bitmap denotes a corresponding frame in each cycle. In a schedule message (see Fig. 9), a bitmap that encodes a node's schedule follows its address. A bit set as '1' in the bitmap denotes an assigned frame.

5.4.1. Maximize channel utilization

TreeMAC is primarily designed to support multi-hop TDMA, but there are nodes of depth less than 3. For those nodes, it is a waste to use only one slot of each assigned frame, since their data only require one-hop forwarding or even fewer. We improve the channel utilization by using Algorithm 3 to determine the transmittable slot of each assigned frame. For instance,

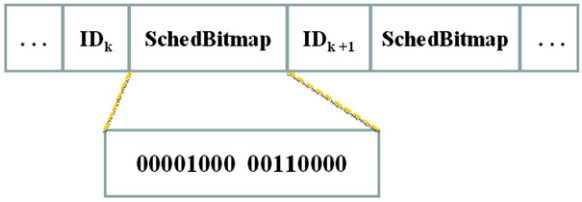
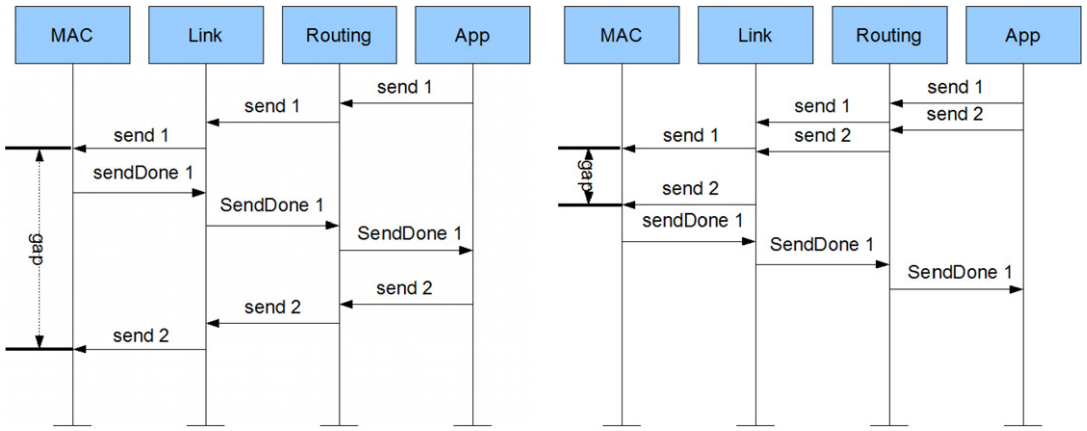


Fig. 9. Schedule encoding using bitmap: In the illustrated example, the bitmap for child k is 0×0830, which child k is assigned frames: 5, 6, 12.



(a) Time sequence of simplex sending.

(b) Time sequence for pipelined sending.

Fig. 10. TinyOS communication stack optimization.

nodes with a depth of 1 do not need intermediate nodes to forward data for them, so they can use all 3 slots of each assigned frame. That enables TreeMAC to seamlessly support one-hop TDMA without the sacrifice of channel utilization.

Algorithm 3 Maximize Channel Utilization

Each node u of network depth ℓ_u takes the following actions for each assigned frame:
switch(ℓ_u)

1. **case** $\ell_u \leq 1$: $S_u \leftarrow 0$, 1, and 2;
2. **case** ℓ_u is 2: $S_u \leftarrow 0$ and 1;
3. **default:** $S_u \leftarrow \ell_u \bmod 3$;

5.5. Pipelined sending and receiving between layers

TinyOS-1.x is an event-driven system and does not support multi-threading. In the recommended practice of TinyOS, an up-layer component cannot 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. Fig. 10 (a) illustrates this issue. Similar situations happen during receiving. We enable pipelined sending/receive approach through buffering messages in sending/receiving queues at each layer, as illustrated in Fig. 10 (b). Pipelined sending/receiving helps to improve the performance of a TDMA MAC protocol. Because when using a TDMA protocol, once a node is assigned a time slot, the node should use the slot efficiently, especially when high throughput is demanded. However, in the 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, precious channel time is wasted. In the design of TreeMAC, we overcome the problems by adding queues at each communication component. In this way, when the MAC layer finishes sending out one packet, it will immediately fetch the next packet in the queue, instead of waiting for the upper layer to send down a new packet.

In addition, the queue size in upper layer shall be no bigger than that in lower layer. Otherwise when the queue in lower layer cannot accommodate more messages, new incoming messages from upper layer will fail to be transmitted. Moreover, we observe that for intermediate nodes, locally originated packets have a higher privilege to be stored and sent, while forwarding packets are unfavored. We mitigate that problem by restricting the queue size in application layer to be no more than $\frac{1}{3}$ of that in the network layer.

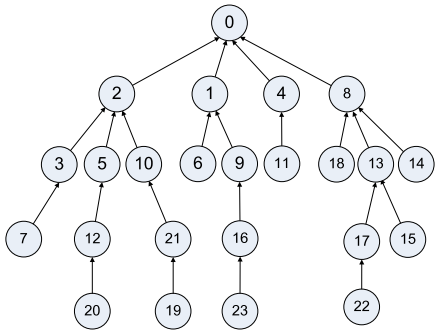


Fig. 11. Snapshot of network topology on the testbed.

Table 1
TreeMAC experiment parameters.

Parameter	Value
Default data transmission power	−25 dBm
Slot size	20 ms
Frame size	3
Cycle size	24
Bandwidth demand update interval	10 s
Schedule update interval	8 s
FTSP synchronization message interval	5 s

Table 2
Funneling-MAC experiment parameters.

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

6. Sensor testbed evaluation

6.1. Experiment method

We conducted experiments on a sensor network testbed with 24 Imote2 motes. The sensor nodes are placed in a 4×6 grid without any obstruction. Fig. 11 shows a snapshot of the network topology on the testbed. The data payload size is 74 bytes. We conducted experiments with varying network traffic and network topologies to evaluate the capacity of TreeMAC compared to Funneling-MAC and CSMA (the default MAC protocol in TinyOS-1.x for CC2420 radio). Originally, Funneling-MAC is only implemented on a mica2 platform that uses CC1000 radio, so we ported it to a 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]. The maximum TX power output is 0 dBm for both radio chips. We did not compare TreeMAC to Z-MAC [19] and B-MAC [4], because Funneling-MAC [1] has shown better performance than them. Tables 1 and 2 show the value of key parameters for TreeMAC and Funneling-MAC respectively in evaluation experiments. We use USB cable to supply power for sensor nodes to prevent their performance from being effected by battery power level. 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.

6.2. 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.

6.2.1. Throughput v.s. data rate

First, we evaluated the network throughput with a varying data rate and fixed network size 24. We conducted the experiment by varying the data rate of each node at 5 levels: 4, 6, 8, 10 and 12 pps (packet per second). The results in

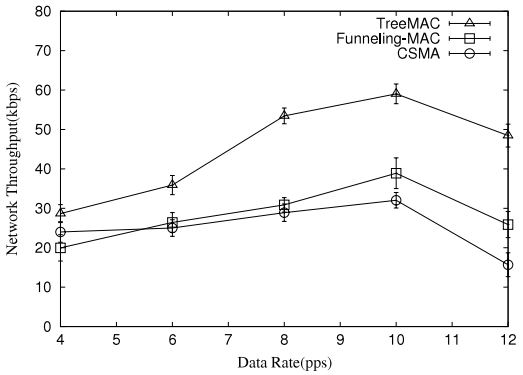


Fig. 12. Network throughput v.s. data rate.

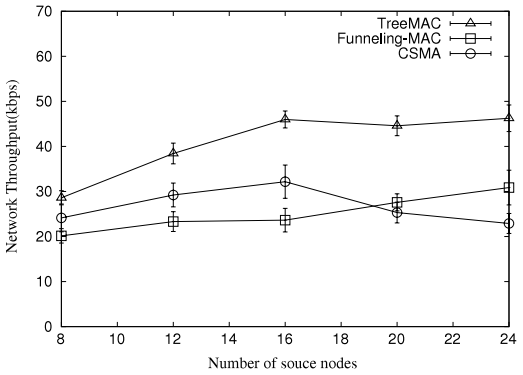


Fig. 13. Network throughput v.s. data rate.

Fig. 12 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 are 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, Fig. 12 shows that TreeMAC always achieves the best throughput compared to Funneling-MAC and CSMA.

6.2.2. 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. Fig. 13 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.

6.2.3. 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 ran the experiments for a long time and measured the throughput for each 5-min time window. The comparison between TreeMAC and Funneling-MAC is shown in Fig. 14. We can see 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 path changes) and can obtain stable performance in the long run.

6.3. Energy efficiency

Maximizing energy efficiency is always an important goal to achieve in MAC protocol design. Energy efficiency [30] 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.

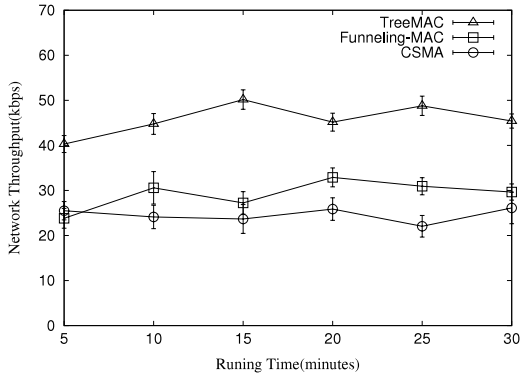


Fig. 14. Network throughput v.s. data rate.

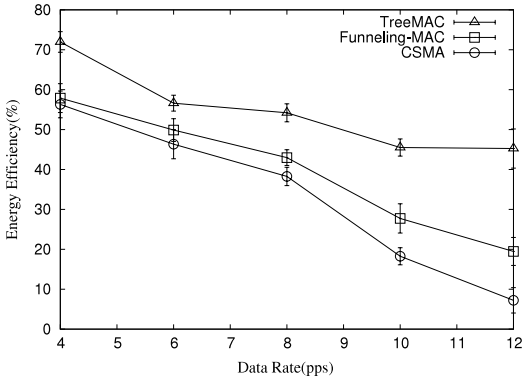


Fig. 15. Energy efficiency.

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). Fig. 15 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 send more packets than their parents can handle, it will result in buffer overflow. This experiment further proves that in data collection networks, 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.

6.4. Energy consumption

In a data collection network, nodes of different depths have varying energy consumption depending on the amount of data they need to transmit. Fig. 16 shows the comparison of average per node energy consumption. When the data rate is low, the energy consumption among CSMA, Funneling-MAC, and TreeMAC is almost at the same level. But as the data rate increases to 12 pps, the energy consumption of CSMA is much higher than Funneling-MAC and TreeMAC due to link layer retransmissions caused by collision; the energy consumption of TreeMAC is also lower than Funneling-MAC by 31.7% at the rate of 12 pps. That is because Funneling-MAC applied CSMA to the nodes outside of the intensity region and collisions are still likely to happen.

6.5. Network fairness

When a sensor network is deployed, users are typically interested in getting real-time data from the 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 [30]: $\phi = \frac{(\sum_{k=1}^{N} r_i)^2}{N \times \sum_{k=1}^{N} {r_i}^2}$. Here, r_i is the average rate of packets delivered from the ith sensor and N is the number of sensors in the network.

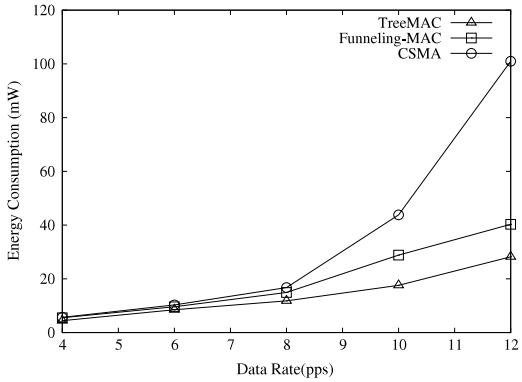


Fig. 16. Average per node energy consumption.

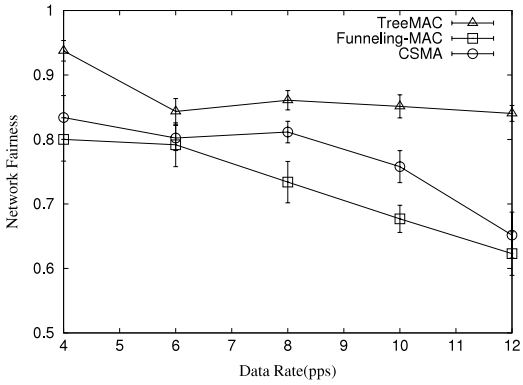


Fig. 17. Network fairness.

If each node sends the same number of packets to the sink, then the fairness index is 1. Fig. 17 shows the comparison of fairness among TreeMAC, Funneling-MAC and CSMA. It shows that as the 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. It is interesting to note that Fig. 17 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.

6.6. Signaling overhead

Both TreeMAC and Funneling-MAC use a TDMA mechanism, and thus have some signaling overhead. CSMA does not need extra control messages, 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. Fig. 18 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.

7. Conclusion and future work

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 networks. We have shown the nice properties of TreeMAC in theory and demonstrated that it achieves much better throughput and energy efficiency than CSMA and Funneling-MAC [1] in a real sensor network test bed.

In the design of TreeMAC protocol, time synchronization precision has significant impact on its performance. Meanwhile, as is mentioned in Section 5, FTSP has the problem of propagated synchronization error, as a result of which the

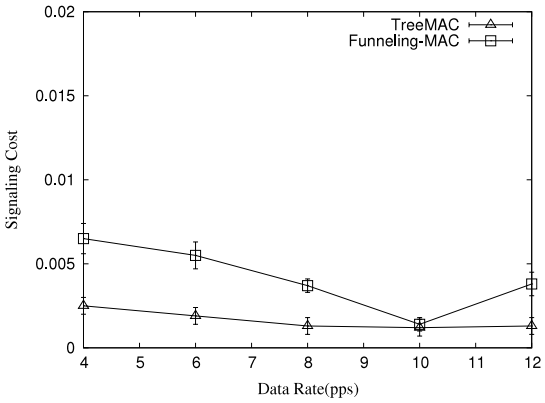


Fig. 18. Signaling overhead.

synchronization error will increase as the hop count to the FTSP root increases. That obviously effects the scalability of TreeMAC. In this paper, the problem is mitigated by using a cycle timer and a slot timer at the same time. Our future work is to make TreeMAC get rid of the need for global time synchronization.

References

[1] G.-S. Ahn, E. Miluzzo, A.T. Campbell, S.G. Hong, 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, Boulder, CO, USA, Nov. 2006.
[2] W.-Z. Song, F. Yuan, 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.
[3] W. Ye, J. Heidemann, 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.
[4] J. Polastre, J. Hill, D. Culler, Versatile low power media access for wireless sensor networks, in: Proc. 2nd ACM Conference on Embedded Networked Sensor Systems, SenSys, Baltimore, MD, USA, Nov. 2004.
[5] R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, L. Krishnamurthy, N. Kushalnagar, L. Nachman, 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, San Diego, CA, USA, Nov. 2005.
[6] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, 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.
[7] 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, D. Pieri, Optimized autonomous space in-situ sensor-web for volcano monitoring, in: 2008 IEEE Aerospace Conference, Big Sky, MT, USA, Mar. 2008.
[8] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, D. Estrin, A wireless sensor network for structural monitoring, in: Proc. 2nd ACM Conference on Embedded Networked Sensor Systems, SenSys, Baltimore, MD, USA, Nov. 2004.
[9] S.N. Pakzad, S. Kim, G.L. Fenves, S.D. Glaser, D.E. Culler, 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.
[10] J. Elson, L. Girod, 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, A. Lédeczi, The flooding time synchronization protocol, in: Proc. 2nd International Conference on Embedded Networked Sensor Systems, SenSys, Baltimore, MD, USA, Nov. 2004.
[12] S. Ganeriwal, R. Kumar, M.B. Srivastava, Timing-sync protocol for sensor networks, in: Proc. 1st International Conference on Embedded networked sensor systems, SenSys, Los Angeles, CA, USA, Nov. 2003.
[13] S. Biswas, 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] W.-Z. Song, R. Huang, B. Shirazi, R. Lahusen, TreeMAC: Localized TDMA MAC protocol for high-throughput and fairness in sensor networks, in: The 7th Annual IEEE International Conference on Pervasive Computing and Communications, PerCom, March 2009.
[15] T. van Dam, K. Langendoen, An adaptive energy-efficient MAC protocol for wireless sensor networks, in: Proc. 1st ACM Conference on Embedded Networked Sensor Systems, SenSys, Los Angeles, CA, USA, Nov. 2003.
[16] A. Woo, 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.
[17] V. Rajendran, K. Obraczka, Garcia, Energy-efficient, collision-free medium access control for wireless sensor networks, in: Proc. 1st ACM Conference on Embedded Networked Sensor Systems, SenSys, Los Angeles, CA, USA, Nov. 2003.
[18] A. Ephremides, O.A. Mowafi, Analysis of a hybrid access scheme for buffered usersprobabilistic time division, IEEE Transactions on Software Engineering 8 (1) (1982) 52–61.
[19] I. Rhee, A. Warrier, M. Aia, J. Min, Z-MAC: A hybrid MAC for wireless sensor networks, in: Proc. 3rd ACM Conference on Embedded Networked Sensor Systems, SenSys, San Diego, CA, USA, Nov. 2005.
[20] I. Rhee, A.C. Warrier, 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.
[21] K.-H. Kim, 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.
[22] F. Stann, J. Heidemann, R. Shroff, M.Z. Murtaza, RBP: Robust broadcast propagation in wireless networks, in: Proc. 4th International Conference on Embedded Networked Sensor Systems, SenSys, Boulder, CO, USA, Nov. 2006.
[23] P. Kyasanur, R.R. Choudhury, 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.
[24] Y. Peng, W. Song, R. Huang, M. Xu, B. Shirazi, Cacades: A reliable dissemination protocol for data collection sensor network, in: 2009 IEEE Aerospace Conference, Big Sky, MT, USA, Mar. 2009.
[25] OASIS: Optimized Autonomous Space In-Situ Sensor Web. http://sensorweb.vancouver.wsu.edu.

[26] W. Xu, W. Trappe, 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.
[27] TinyOS-1.x: http://www.tinyos.net.
[28] BGP, WLAN Interference to IEEE802.15.4 in http://www.z-wavealliance.org, Mar. 2007.
[29] CC2420 Data Sheet. http://focus.ti.com/docs/prod/folders/print/cc2420.html.
[30] B. Hull, K. Jamieson, H. Balakrishnan, Mitigating congestion in wireless sensor networks, in: Proc. 2nd International Conference on Embedded networked sensor systems, SenSys, Baltimore, MD, USA, Nov. 2004.