Funneling-MAC: A Localized, Sink-Oriented MAC For Boosting Fidelity in Sensor Networks

Gahng-Seop Ahn[†], Emiliano Miluzzo[‡], Andrew T. Campbell[‡] Se Gi Hong[†], Francesca Cuomo[††]

[†]EE Dept., Columbia University New York, NY, USA

[‡]CS Dept., Dartmouth College Hanover, NH, USA

[††]University "La Sapienza" Rome, Italy

Abstract

Sensor networks exhibit a unique funneling effect which is a product of the distinctive many-to-one, hop-by-hop traffic pattern found in sensor networks, and results in a significant increase in transit traffic intensity, collision, congestion, packet loss, and energy drain as events move closer toward the sink. While network (e.g., congestion control) and application techniques (e.g., aggregation) can help counter this problem they cannot fully alleviate it. We take a different but complementary approach to solving this problem than found in the literature and present the design, implementation, and evaluation of a localized, sink-oriented, *funneling-MAC* capable of mitigating the funneling effect and boosting application fidelity in sensor networks. The funneling-MAC is based on a CSMA/CA being implemented network-wide, with a localized TDMA algorithm overlaid in the funneling region (i.e., within a small number of hops from the sink). In this sense, the funneling-MAC represents a hybrid MAC approach but does not have the scalability problems associated with the network-wide deployment of TDMA. The funneling-MAC is 'sink-oriented' because the burden of managing the TDMA scheduling of sensor events in the funneling region falls on the sink node, and not on resource limited sensor nodes; and it is 'localized' because TDMA only operates locally in the funneling region close to the sink and not across the complete sensor field. We show through experimental results from a 45 mica-2 testbed that the funneling-MAC mitigates the funneling effect, improves throughput, loss, and energy efficiency, and importantly, significantly outperforms other representative protocols such as B-MAC, and more recent hybrid TDMA/CSMA MAC protocols such as Z-MAC.

Categories and Subject Descriptors: C.2.2 [Computer Communication Networks]: Network Protocols, Wireless Communications

General Terms: Algorithms, Design, Experimentation.

Keywords: MAC, Wireless Sensor Networks, Funneling Effect.

SenSys'06, November 1-3, 2006, Boulder, Colorado, USA.

1. Introduction

Wireless sensor networks exhibit a unique funneling effect [7] where events generated in the sensor field travel hop-by-hop in a many-to-one traffic pattern toward one or more sink points, as illustrated in Figure 1. This combination of hop-by-hop communications and centralized data collection at a sink creates a choke point on the free flow of events out of the sensor network. For example, the funneling of events leads to increased transit traffic intensity and delay as events move closer toward the sink, resulting in significant packet collision, congestion, and loss; at best this leads to limited application fidelity measured at the sink, and at worst the congestion collapse [15] of the sensor network. Other drawbacks exist. The sensors nearest to the sink, typically within a small number of hops loose a disproportionate larger number of packets (we call this region of the funnel the *intensity region*, as illustrated in Figure 1) and consume significantly more energy than sensors further away from the sink, hence, shortening the operational lifetime of the overall network. Mitigating the funneling effect represents an important challenge to the sensor network community and is the subject of this paper.

Researchers have proposed distributed congestion control algorithms [15], tiered network design [7], and data aggregation techniques [16] [17] to respond to increased load and congestion in sensor networks. But as the literature [15] [7] indicates these techniques alone cannot fully alleviate the problem because it is very difficult to effectively rate control traffic at aggregation points or sources to match the bottleneck conditions observed at the sink nodes. In this paper, we show that the majority of packet loss in a sensor network occurs within the first few or more hops from the sink, even under light traffic conditions. We conjecture that by putting additional control within the first few or more hops from the sink we can significantly improve communication performance and eradicate the funneling effect.

We propose a *localized, sink-oriented funneling-MAC* that explicitly recognizes the existence of funneling effect in its design. While there have been a number of important new MAC protocols proposed for sensor networks, to the best of our knowledge none have addressed the funneling effect. The funneling-MAC represents a hybrid (schedule-based) TDMA and (contention-based) CSMA/CA MAC scheme that operates in the intensity region of the event funnel, as illustrated in Figure 1. Pure CSMA/CA operates network-wide in addition to acting as a component of the funneling-

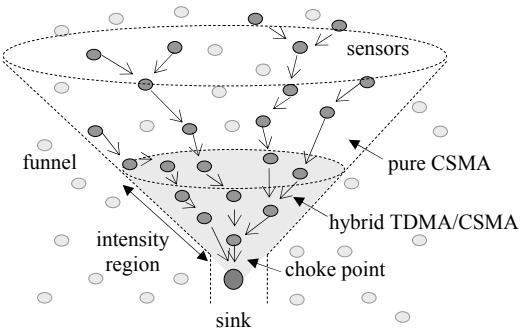


Figure 1. Funneling effect in sensor networks

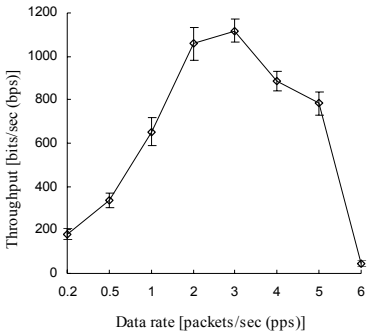


Figure 2. Throughput of CSMA with varying data rates

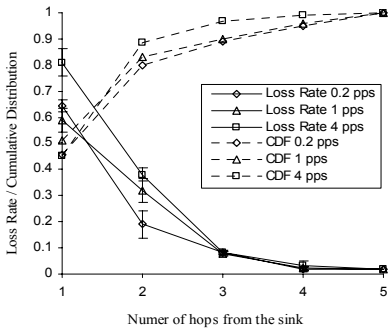


Figure 3. Loss rate and cumulative distribution function of loss over varying distance from the sink for CSMA

MAC that operates in the intensity region. The funneling-MAC mitigates the funneling effect by using local TDMA scheduling in the intensity region only, providing additional scheduling opportunities to nodes closer to the sink, which typically carry considerably more traffic than nodes further away from the sink. 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, and not on resource limited sensor nodes. The funneling-MAC is localized in operation because TDMA only operates in the intensity region close to the sink and not across the complete sensor field. The burden of computing and maintaining the depth of the intensity region also falls on the sink. We assume that the sink is likely to have more computational capability and energy reserves than simple sensors; however, the funneling-MAC does not rely on this to operate efficiently. By using TDMA in this localized manner, and putting more management onus on the sink not the sensors, we offer a scalable solution for the deployment of TDMA scheduling in sensor networks, one that is capable of boosting application fidelity as measured at the sink, but does not have the scalability problems associated with the network-wide deployment of TDMA, which, we believe, is untenable today as a network-wide deployment strategy for large-scale sensor networks.

The structure of the paper is as follows. In Section 2 we show the impact of the funneling effect using results from an experimental sensor network. The effectiveness of existing MACs to counter the funneling effect is discussed in Section 3. Following this, we present the detailed design of the funneling-MAC algorithms in Section 4 that include: on-demand beaconing, which both provides light-weight clock synchronization for TDMA scheduling in the intensity region, and regulates effectively boundary of that region; sink-oriented scheduling, which computes and distributes new schedules when needed in an efficient low cost manner; and dynamic depth-tuning, which dynamically adjusts the depth of TDMA operating in the intensity region with the goal of maximizing the throughput of the sink choke point while minimizing the packet loss in the funnel. The Appendix in our technical report [23] provides important analytical foundations that justify the choice of dynamically controlling the depth of the intensity region in response to measured traffic conditions at the sink node. We take an experimental systems approach to the validation of the funneling-MAC's performance. Section 5 presents results from a number of experiments using a 45 mica-2 mote network. We consider a number of different node densities, and traffic characteristics to study the performance of the funneling-MAC in comparison to other representative protocols such as the TinyOS [11] default protocol B-MAC [3], and more recently proposed, and comparative protocol Z-MAC [10], which is also based on a hybrid TDMA/CSMA approach. We show by simply exerting control over the first few or more hops from the sink that the funneling-MAC significantly outperforms B-MAC and Z-MAC, which we show are not capable of dealing with the funneling effect.

2. Funneling Problem

We begin by first quantifying the impact of the funneling effect in a sensor network using the TinyOS CSMA-based B-MAC protocol, the MintRoute routing protocol, and the Surge application in a 45 mica-2 testbed. The network is deployed as a 5x9 rectangular grid of equally spaced motes in a large open room, making sure there are no interference and near-field issues [12] during the experiments. The mote at the bottom left corner operates as the sink in the grid, as illustrated in Figure 4. Node spacing and transmission power are set such that one-hop neighbors achieve > 80% delivery, while two-hop neighbors achieve < 20% delivery. In this way, a fairly strict and dense multi-hop radio environment is constructed for experimentation.

We randomly select 16 of the 44 sensing nodes to generate event rates ranging from 0.2-5 *packets/sec (pps)* where the packet size is 36 bytes. The goal is to gradually drive the sensor network from low to moderate load and then into a congested and saturated state, while studying the choke point throughput measured at the sink and the loss in the network. Typically, events travel over multiple hops, 2-5 hops in the case of the experiment. Figure 2 shows the resulting fidelity (i.e., throughput curve), as measured at the sink as we increase the event rate of all 16 sources. Note that we exclude the preamble and CRC sizes, and count the packet size as 36 bytes when calculating the throughput fidelity. We can clearly see that the throughput measured at

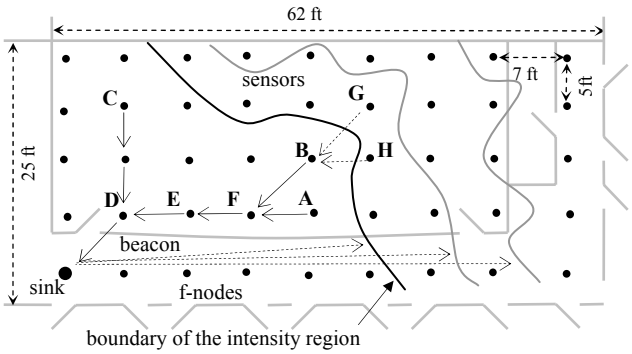


Figure 4. Dartmouth College sensor testbed

the sink rises to a peak of approximately 1100 bps before the network falls into a congested and saturated state. Further increase in source rate only drives the network into further overload and eventual collapse with increasing load. We observe from Figure 2 that source rates of 0.2 pps, 1 pps, and 4 pps can be considered to be light, medium (near optimal load), and overload traffic scenarios, respectively. We use these rates to further study the impact of the funneling effect on loss distributions across the network. We consider the overall loss rate in the network to be the number of packets lost in the network divided by the number of packets transmitted in the network. The overall loss rates measured for increasing load are approximately 67%, 72%, and 95% loss rate for 0.2 pps, 1 pps, and 4 pps, respectively. What is surprising about these results in that at low load there is still significant loss (67%), which rises to the point where 95% of events transmitted in the network are lost at high load. This also translates to significant energy waste. Such loss is unacceptable for many applications and would quickly deplete the sensors energy reserves. Note that in the case of light and medium traffic scenarios, packet loss is mainly due to collision and hidden terminal problem, whereas in the high and overloaded traffic scenarios loss is due to buffer overflow in addition to collision and hidden terminal problem.

Next, we consider the distribution of the loss across the hops in the network. The solid lines in Figure 3 show the loss rate at the i-th hop (i.e., the number of packets transmitted and lost by i-th hop divided by the number of packets transmitted by i-th hop). The result clearly quantifies the funneling effect for this experiment and shows its debilitating impact on network performance. These results represent the average of five runs of the same experiment and the 95% confidence intervals. What is interesting about these results is that Figure 3 clearly shows that there is increasing loss at nodes closer to the sink, which is a product of the many-to-one, hop-by-hop traffic pattern of the funneling effect. For example, for all traffic rates the vast majority of packet loss occurs in the first two hops from the sink and drops of quickly for hops further away from the sink. These are fingerprints of the funneling effect. Note, that even for a light traffic load of 0.2 pps this trend is still dominant with significant loss registered in the first few hops. These per-hop loss rates for the low rate traffic explain why at such a low rate we still can record an overall loss rate for the network of 67%, as discussed above. The dotted lines in Figure 3 show a cumulative distribution function (CDF) of the per-hop losses. We can observe from the plot that between approximately 80-90% of the losses across the three low, medium, high rates happened within the first two hops from the sink. We can conclude that funneling effect is mostly invariant to source rate.

These results indicate that by adding addition controls (e.g., scheduling) in the network over the first few hops could offer significant gains across all traffic rates considered in the experiment (viz. light, medium, heavy). We can also conclude that even at low rates the CSMA-based B-MAC cannot mitigate the funneling effect. These are important insights. Therefore, we conjecture that new MAC approaches other than B-MAC are needed to fully address the funneling problem.

3. Related Work

In what follows, we discuss a number of sensor network MAC protocols and traffic control mechanisms found in the literature and comment on how they would fair in mitigating the funneling effect discussed in the previous section.

S-MAC [1], T-MAC [2], B-MAC [3] and the MAC discussed by Woo and Culler in [19] represent well-known contention-based (CSMA) MAC protocols for sensor networks. In [19] the authors discuss an early contribution to sensor network MACs that uses adaptive rate control mechanisms on top of CSMA to achieve energy efficiency and fairness. This MAC [19] represents a network-aware scheme like the funneling-MAC in the sense that it considers route-through traffic when using rate control. S-MAC avoids idle listening by putting sensor nodes to sleep periodically. S-MAC requires time synchronization but the time-scale is much larger than TDMA. T-MAC provides almost the same functionality as S-MAC except that it is capable of further reducing the idle listening by transmitting all messages in the buffer of each node at the beginning of the active period, allowing it to sleep instantly once the buffer is flushed. B-MAC provides well-defined interfaces to low power listening (LPL), clear channel assessment (CCA) and acknowledgements. LPL improves the energy efficiency and throughput with the cost of transmitting a long preamble by sources. We show that B-MAC is not capable of mitigating the funneling effect because of the large build up of losses in nodes closer to the sink, as discussed in the previous section. We conjecture that Woo's MAC [19], S-MAC and T-MAC based on similar contention-based approaches as B-MAC would likely be as non-responsive and show the same poor trends as B-MAC in dealing with the funneling effect.

There are several schedule-based (TDMA) MAC algorithms proposed in the sensor network literature that do better at mitigating the funneling effect. The energy-aware TDMA-based MAC [4] achieves collision free access and energy efficiency by assigning each node their own time slots (listening slot and transmitting slot), allowing nodes to

sleep when it is not their slot time. This approach [4] may be impractical because the sink requires complete topology information to compute the TDMA schedule and every node requires precise time synchronization. Furthermore, from [4] every node would need to communicate directly with the sink (using high power). These issues indicate that the actual implementation of such a scheme in a large sensor network would have scalability problems.

Another TDMA protocol called TRAMA [5] performs an adaptive election algorithm to overcome this drawback of wasting time slots. TRAMA is a scalable distributed algorithm where each node schedules time slots among its two hop neighbors using a neighbor protocol and schedule exchange protocol as discussed in [5]. One drawback of implementing TRAMA in a mote network (no current implementation exists for TinyOS, as far as we are aware) is that the overall signaling overhead of these fairly complicated protocols may present scalability problems, particularly if implemented in a large-scale testbed. There are a number of other TDMA-based algorithms found in the literature [6] [8] [9] (but not implemented in mote networks) that suffer from similar problems when targeted toward large-scale sensor deployment because of the need for global network-wide schedule computation and distribution, and time synchronization.

The most suitable protocol for potentially mitigating the funneling effect that is available in source code for mica-2 motes is the Z-MAC protocol. Z-MAC [10] is a hybrid protocol that acts like a contention-based protocol under low traffic conditions and a schedule-based protocol under high traffic conditions by using the schedule computed by DRAND (Distributed RAND) as a hint. DRAND is a fairly complex coloring algorithm to explain here in detail, sufficient too say that it 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. Z-MAC reduces the hidden terminal problem by not allowing two nodes in two-hop distance to transmit at the same time. In order to improve utilization, Z-MAC allows 'non-owners' of a slot to contend for the slot if it is not being used by its 'owner'. Z-MAC requires global time-synchronization in the initial phase, and then it performs local synchronization by sending periodic sync packets between nodes. Z-MAC requires that DRAND is run at startup to set up the TDMA schedule, which may be a heavy burden for light-weight sensor devices. The message complexity of DRAND is $O(\delta)$, where δ is the local neighborhood size of each node while the message complexity of the funneling-MAC (detailed in the next section) is O(1). Because of the overhead of running DRAND, the Z-MAC authors do not recommend that it be run periodically. We choose to compare the funneling-MAC to Z-MAC in the experimental evaluation section (Section 5). We note in those experiments that Z-MAC is susceptible to "schedule drift" (i.e., when the schedule allocated by DRAND to nodes drifts out of sync because of various time varying radio impairments). We discuss these issues and show that, while Z-MAC offers scheduling support, it is not designed to schedule more traffic at nodes closer to the sink in its current form, and therefore, cannot mitigate the effects of funneling events to a sink choke point. Because of the potential for schedule drift, Z-MAC's performance ends up degrading to being only marginal better than B-MAC under a number of experimental scenarios, as we discuss in Section 5.

Flexible Power Scheduling (FPS) [20] also represents a hybrid approach that provides coarse grain scheduling that computes radio on/off times, and fine grain MAC control for channel access. The coarse grain scheduling of FPS represents a distributed approach where each node schedules its own children. The funneling-MAC and Z-MAC have some similarities to FPS. However, FPS is limited when dealing with the funneling effect because it does not prevent nodes with different parents from using the same slot. FPS simply relies on CSMA to provide collision avoidance in this case.

In [7] the authors propose to add multi-radio virtual sinks to sensor networks as a means of dealing with loss at the physical sink. Virtual sinks address the funneling effect by adding more 'capacity' in an on-demand manner to the network using network layer routing to redirect traffic off the primary mote radio network (reducing the funneling effect on the physical sink) and onto an overlay network. While virtual sinks are effective they require specialized multi-radio nodes and an overlay network to siphon packets off the primary network. In addition, virtual sinks themselves can experience a mini-funneling effect [7].

4. Funneling-MAC Design

We now discuss the detail design of the funneling-MAC algorithms, and issues related to timing and framing.

4.1 On-Demand Beaconing

The funneling-MAC localized TDMA is triggered by a beacon broadcast by the sink. All sensor nodes perform CSMA by default unless they receive a beacon and are then deemed *f-nodes*. The sink regulates the boundary of the intensity area (see Figure 4) by controlling the transmission power of the beacon. The dynamic depth-tuning algorithm discussed in Section 4.5 determines this transmission power. The sink then transmits the beacon message at the computed transmission power. The nodes that received the beacon consider themselves to be in the intensity region and f-nodes. These nodes can perform TDMA while the nodes that do not receive the beacon (e.g., those nodes outside the intensity region) perform CSMA.

F-nodes need to synchronize their clock to perform TDMA but the funneling-MAC does not rely on any synchronization protocol. If a network synchronization protocol is present then the funneling-MAC can use that and further minimize its active beacon signaling. However, in our implementation of the funneling-MAC we do not assume this and integrate a light-weight clock synchronization scheme embedded in the beacon messaging. Therefore, f-nodes rely

on the beacon sent to activate TDMA and regulate the boundary of the intensity region for clock synchronization. As soon as a node receives a beacon, it becomes an f-node and synchronizes with other f-nodes by initializing its clock. The propagation delay of a beacon is on the scale of microseconds in wireless sensor networks while the accuracy of synchronization required for the funneling-MAC is on the scale of milliseconds, so beacon-based synchronization can keep the synchronization tight enough to perform TDMA scheduling. Because the beacon is broadcast across the complete intensity region then all f-nodes receive the beacon at the same time and are tightly synchronized. This is a similar approach to reference-broadcast synchronization [21] but much simpler.

The beacon packet contains a small number of control fields including the *beacon interval*, *superframe duration*, and the *TDMA duration*. The superframe duration and TDMA duration are explained in Section 4.3 on framing. The beacon is sent periodically every beacon interval specified in the beacon packet. Experimentally we set the beacon interval so it is responsive to possible changes in routing, traffic rates, and clock drift of f-nodes. The beacon interval is determined by taking into account the accuracy of the local clock of the motes and required accuracy of the synchronization, as discussed in Section 5.1.

The beacon is sent only when it is necessary and in an on-demand basis. The beacon is not sent when the network is idle or receiving very low traffic. Note that every f-node keeps a timer that expires if the f-node does not receive a beacon for a period longer than the beacon interval. When the timer expires, the node performs pure CSMA. As soon as the sink receives a sufficient amount of data packets as determined by a change in the weighted moving average of the traffic (measured at the sink) from all paths then it begins to transmit a beacon periodically, based on the computed beacon interval. Conversely, if the sink does not receive sufficient traffic to allocate slots in the network in one or more beacon interval times, then it stops sending beacons until the sink registers such a positive change. F-nodes use the beacon interval to synchronize with future beacon transmissions from the sink. A mote based beacon interval timer allows motes to defer from transmitting when a beacon is due which could potentially interfere with the beacon if left unregulated.

When the sink starts beaconing at start-up or just after an idle period, it starts with the minimum transmission power (i.e., the same transmission power as ordinary sensor nodes). This is because the depth-tuning algorithm (as described in Section 4.5) uses an incremental increase/decrease rule when calculating the beacon/schedule transmission power. Gradually the sink will increase the transmission power as the measured traffic increases and the throughput/loss objectives are met (as addressed in Section 4.5) using the dynamic depth-tuning algorithm. Conversely, if the sink was to send the beacon not at the minimum power as discussed but rather high transmission power from start-up or after an idle period, then the beacon would likely interfere with contention based incoming CSMA data packets. This is because motes in a start-up state or just after an idle period are not aware when a beacon will be transmitted. This problem is resolved by the funneling-MAC because the starting point for the dynamic depth-tuning algorithm is always the same as the common default power used by motes (which is considered to be the power floor for the depth-tuning algorithm). Hence, the impact of interference is minimized. Since the objective of the tuning algorithm is to increase the depth of the intensity region and therefore the transmission power there is a case that nodes not reachable by the existing power level will be interfered with when the tuning algorithm increments the beacon transmission power. The funneling-MAC resolves this potential interference issue by introducing a 'meta-schedule advertisement', which is discussed in Section 4.4.

Our design goal is to limit the cost of supporting periodic beacons by making them on-demand. One other parameter we consider is to extend the beacon interval to trade off signaling overhead, the reception power used by motes in the existing intensity region, and reduce the energy demands on the sink. We introduce the notion of 'lazy beaconing', which pushes out the optimal beacon interval that is used to maintain tightness of clock synchronization and slot scheduling at f-nodes. By pushing out the beacon interval in this manner there can be some performance penalties if left unbounded. In Section 5.1, we discuss the optimal beacon interval used to maintain tight synchronization and slot scheduling, and optimal throughput, and contrast this to lazy beaconing which allows us to triple the optimal beacon interval for only a small reduction in the performance of the network, as measured by sink fidelity.

4.2 Sink-Oriented Scheduling

The sink monitors the traffic that arrives at the sink on a per-aggregated-path basis, calculates the TDMA schedule based on the monitored traffic (initially based on only new CSMA events and thereafter including existing TDMA traffic) for all paths, and distributes the schedule by broadcasting a schedule packet at the same transmission power used by beaconing. We define an *aggregated path* as a path which results from the merge of two or more paths at or before entering the intensity region. The funneling-MAC treats an aggregated path as a single path entry. For example in Figure 4, the funneling-MAC keeps information associated with paths G-B-F-E-D and H-B-F-E-D as a single aggregated path entry B-F-E-D. The funneling-MAC scales well because the number of aggregated paths entering the intensity region is bounded by the number of nodes in the intensity region. We use the term path to indicate aggregated path in the remainder of the paper for convenience. In what follows, we provide a detailed discussion of sink-oriented scheduling. See [23] for the pseudo code of the algorithm that is not presented in this paper because of space limitations.

In order to compute the schedule the sink needs to determine the identity of the *path-head* f-nodes and the

weighted average of the traffic on the path in order to correctly schedule the path. The concept of a path represents the direction taken by a train of events from a path-head (e.g., mote A in Figure 4) on a hop-by-hop basis along a route (e.g., determined by the TinyOS MintRoute routing protocol in our experiments) to the sink (e.g., path A-F-E-D-Sink). The sink measures the weighted moving average of each path and allocates slots according to an allocation rule, which we discuss below. In order to enable the sink to acquire this information the funneling-MAC reserves 3 bytes in the packet header called the path information field. The path information field is only updated by the f-nodes along a certain path in the intensity region. The sink gathers this information from incoming packets on a per-path basis for all paths in the intensity region. The path information field contains the *path head id* (2 bytes) and the *number of hops* (1 byte). The path-head lies near the intensity region boundary where the path head id equals the *node id* of the path-head, and the number of hops field reflects the number of hops the packet traverses on the path between the path-head and the sink. For example in Figure 4 if a packet generated from outside of the intensity region is received by node A, node A forwards the event packet toward the sink following the path A-F-E-D-Sink. In this simple example, the path head id is A, and the value of number of hops is 4. Importantly, node A identifies itself as the path-head when it receives a data event packet with a value of the path information field set to zero. In addition, source nodes inside the intensity region identify themselves as a path-head when they generate a new packet. A path-head puts its *id* in the path head id field and a value 1 in the number of hops field. All f-nodes along the path increment the value of the number of hops field by 1 when they forward the event data packet. Consequently, each packet that arrives at the sink carries the path head id of the path it traversed as well as the number of hops.

The sink monitors incoming data packet and keeps track of incoming traffic rate for each path along with the path head id and number of hops. The sink keeps the traffic rate on a per path basis in the path table. The sample period is one superframe (as defined in Section 4.3) and the sink measures the number of incoming packets in one superframe per path. Then, the sink calculates the weighted moving average of the measured traffic rate per path.

The sink computes the schedule by allocating time slots per-path rather than on per-node basis. This is because the sink only has the information about the paths and not about the nodes in the paths. This makes the scheme scalable and not coupled to any tree generated by a particular routing scheme; that is, the schedule computation operates on a simple path abstraction of path-end and hop count and not topological routing information. Therefore, the funneling-MAC is agnostic to the routing scheme or routing tree formations. The sink stores per-path state information in a path-table, which is indexed using the path-head id, per-path measurement statistics are also maintained in this table. Each entry contains a path head id, number of hops, and incoming rate. The incoming rate represents the number of packets each path should carry during one superframe. Note that the sink ages each entry every beacon interval and if the table overflows the sink replaces the oldest entry with a new entry.

Slot Allocation Rule: The sink allocates slots to each path using the information in the path table. For example, assume that the traffic rate of a path is k and the number of hops of the path is h. The sink should allocate every node in the path with $\lfloor k \rfloor$ slots so the sink allocates $\lfloor k \rfloor \times h$ slots to the path. If the traffic rate of a path is less than 1, the sink does not follow the above rule, instead, the sink allocates $1 \times h$ slots to the path. The traffic rate can be less than 1 in the case where periodic traffic with data generation rates of less than 1 packet in one superframe or in the case where event-driven traffic happens. As shown in Section 2, the funneling effect is active under light traffic load conditions as well at increased loads so there is a need to schedule paths that have a traffic rate less than 1. If the traffic rate of a path is low, the sink should allocate the minimum number of slots to such a path. The minimum number of slots that the sink can allocate to a node is 1 slot. Therefore, the sink should allocate every node in the path 1 slot so the sink allocates $1 \times h$ slots to the path. This rule turns out to be good because the testbed evaluation result in Section 5.5 show that the funneling-MAC improves the throughput in light traffic scenario compared to pure CSMA.

Simple Spatial Reuse: To enhance the throughput inside the funnel area, the sink considers spatial reuse. It is very difficult to design an optimal spatial reuse scheme without having the complete physical topology information of the network. However, the sink can compute sub-optimal spatial reuse using only the per-path number of hops state information. The funneling-MAC takes this simple sub-optimal approach and reuses the same slot if two nodes are more than 2 hops away from each other. In this case, f-nodes are unlikely to interfere because one of the nodes may back off due to the fact that in the funneling-MAC carrier sensing is used even for the scheduled access. For example in Figure 4, the f-nodes A or B can share the same slot with f-node D because they are 3 hops away. In this case, sink based schedule computation allows f-node B to start transmission three slots after f-node A's slot (i.e., at the slot which belongs to f-node D). As a result, the computed schedule is as follows: 3 slots are allocated to the path A-F-E-D, and 4 slots to path B-F-E-D.

Header	**A ; 3**	**B ; 4**	**C ; 3**

Figure 5. Schedule packet structure

Once the sink computes the schedule, it broadcasts a schedule packet for all paths in its path-table immediately after the next beacon. The sink transmits the schedule packet using the same power level that the sink uses for the beacon so all f-nodes in the intensity region are likely to hear the schedule. Because new schedules are not typically sent each beacon interval the sink sets a schedule expected bit in the

beacon header. The payload of the schedule packet contains the path head ids of the scheduled paths and the number of slots allocated to each path, respectively. This resulting per-path schedule is stored in a tuple [path head id (2 bytes), number of slot (1 byte)] in the packet payload. For example in the simple schedule packet shown in Figure 5 all f-nodes are informed that there are 3 active paths scheduled in the intensity region and that the 3 paths are allocated, 3, 4, and 3 slots, respectively. F-nodes receive the schedule packet and figure out which slots are assigned to them. Each f-node keeps a table where it stores the path head node ID of each path going through it and the number of hops to the path head when they forward data packets. Using this table, the f-node can compute which slots are allocated to itself. For example, the entries of {path-head id, number of hops} maintained by the node E are {A, 2} and {B, 2} so the node E understands that it can transmit two slots after A's slot and two slots after B's slots.

4.3 Timing and Framing Issues

Once f-nodes receive a schedule packet, they synchronize their communication to the funneling-MAC framing structure, as illustrated in Figure 6. F-nodes transmit their scheduled packets at their allocated slots times in the TDMA frame. To enhance the robustness and flexibility of the funneling-MAC, a CSMA frame (random access period) is reserved between two consecutive TDMA frame (scheduled access period) schedules, and carrier sensing is performed even for scheduled transmissions. The combination of a TDMA and CSMA frame forms what we call a *superframe*. Several superframes are repeated between two beacons, as illustrated in Figure 6, where a schedule packet typically follows a beacon.

The aim of the CSMA frame is to allow for the transmission of event data packets that are generated by sensors but have not been allocated slots to be scheduled yet. Other scenarios arise: management, routing, and event data from new nodes that suddenly require transport. One other scenario that is commonly experienced in our testbed is new event data appears on a path due to route changes that occur due to radio vagaries. The sink detects these events using its traffic measurement algorithm. Another reason we always offer some CSMA access in the intensity region is to support the transmissions of asynchronous management and control packets such as routing, hello messages, and packet retransmissions for event data packets that are not successfully transmitted during the TDMA frame. Note that the retransmission policy is only an optional part of the funneling-MAC that can be activated should link reliability be required.

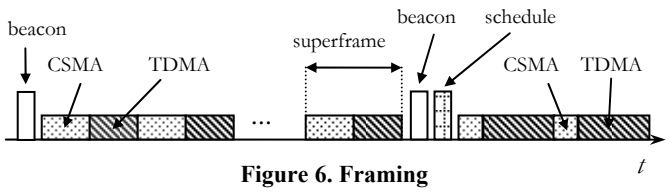


Figure 6. Framing

The beacon delivered to f-nodes includes all the necessary frame timing information for the f-nodes to correctly schedule their traffic or contend for the CSMA access in a superframe. Note that from Figure 6 the superframe duration is fixed while TDMA duration changes dynamically. The superframe duration has no significant impact on the performance because the sink adapts the schedule to the superframe duration. The sink measures the incoming traffic every superframe and computes the schedule based on the results of sampling process, as described in Section 4.2. The TDMA duration changes when the sampled traffic rate at the sink changes. If the traffic load increases sufficiently, the sink allocates more slots in a superframe so that the TDMA duration grows and more events get scheduled in the intensity region. The portion of a superframe that is not used by TDMA is allocated to the CSMA frame. In our implementation, we limit the maximum ratio of TDMA/CSMA in a superframe to 80% so that at least there is a minimum allocation of CSMA to support control packets and unscheduled data packets, as discussed.

The funneling-MAC improves robustness by performing carrier sensing even for scheduled transmissions to avoid possible collisions in transmission anomalies such as in the presence of nodes inside the intensity region that do not receive beacons nor meta-schedule advertisements, as discussed in Section 4.4. Finally, in terms of framing we note that the funneling-MAC uses the low power listening (LPL) algorithm and preamble technique proposed in B-MAC [3] to reduce energy consumption for sensor networks with low duty cycle. However, unlike B-MAC f-nodes do not need to transmit a long preamble in the LPL mode because their communications are synchronized by the superframe. This frees f-nodes to use the standard short radio preamble. During TDMA access f-nodes wake-up at the beginning of their scheduled listening slot and in the case of CSMA frame f-nodes wake-up periodically based on the wake up periods suggested in [3]. During CSMA access, f-nodes can transmit with the standard preamble because all f-nodes can wake-up and listen at the same time. The nodes outside the intensity region use the long preamble used in the LPL mode before transmitting a data.

4.4 Meta-Schedule Advertisement

A number of MAC interference issues arise with the funneling-MAC due to its hybrid MAC nature and its broadcasting of sink signaling (i.e., beaconing, schedules) at potentially high power over the complete intensity region. In order not to interfere with any on-going sensor communications in the network (e.g., CSMA forwarding between sensors toward the sink) by such a high power sink transmission, nodes must be capable of learning the superframe timing details from beacon messages. Another interference issue arises where nodes inside the intensity region may not receive beacons (e.g., due to fading, asymmetric links, etc.) and therefore can become potential "interferers" by not having the timing and framing information carried in the beacon. One final scenario can

occur where nodes outside of the boundary of the intensity region may not be aware of the funneling-MAC frame timing because they do not receive beacons, and as a result, also represent potential interferers. To deal with these interference scenarios (i.e., between scheduled and random access transmissions) the funneling-MAC embeds a low cost meta-schedule advertisement in the first event data packet transmitted by f-nodes, after a new schedule is received.

All f-nodes that received the beacon and schedule embed the meta-schedule in the first event data packet transmitted toward the sink every beacon interval. The mini-schedule contains the following information: superframe duration, TDMA duration, time left of the current TDMA frame, and number of superframe repetitions before the beacon interval expires. The meta-schedule is only 4 bytes in length.

Nodes that are either inside the intensity region and miss a beacon or outside the intensity region but near the boundary can overhear the transmission of meta-schedule carried in a data event. Reception of a meta-schedule allows these nodes to transmit in the CSMA portion of the current superframe mitigating the likelihood of interfering. Now, let's consider a case when an intermediate node of a path inside the intensity region misses a beacon. For example, node F in Figure 4 misses a beacon while the path A-F-E-D is scheduled. The path-head f-node A sends a data packet with meta-schedule and node F receives the data packet with meta-schedule. This way, node F can determine that the data packet is scheduled at the current time slot so node F transmits the data packet immediately. Node F uses CSMA frame for its other data packets. Now, let us assume the path A-F-E-D is not yet scheduled and the path-head f-node A transmits a data packet with its path information field using CSMA frame. Node F receives the data packet with path information field and node F updates the number of hops field and forwards the data packet so the sink can still schedule the path A-F-E-D. Therefore, the meta-schedule advertisement allows seamless interoperation between TDMA inside the intensity region, and CSMA operating outside of that region. The use of meta-schedules in this manner resolves potential erroneous behavior.

4.5 Dynamic Depth-Tuning

The dynamic depth-tuning algorithm enables the funneling-MAC to maximize the throughput and minimize the packet loss at the sink point. The sink regulates the boundary of the intensity area where TDMA is performed by controlling the transmission power of the broadcast beacon. The sink can dynamically change the transmission power of the beacon and therefore the area in which TDMA is active by determining the optimal depth d of the intensity area in the funnel. We analytically analyzed how the funneling-MAC should determine the optimal depth of the intensity region. Due to space limitations, we do not include the analysis here (see the Appendix in [23] for the details on the analysis). The Appendix in [23] provides a number of valuable insights that motivate the operations of dynamic depth-tuning algorithm. In [23] it is shown that the optimal value of d to maximize throughput and minimize packet loss can be determined at the sink. This result drives the design of the dynamic depth-tuning algorithm. Based on the analysis in the Appendix [23], we propose the following dynamic depth-tuning algorithm. Suppose that A is the total number of slots scheduled, A_{max} is the number of the maximum available slots in one superframe, and that d_{max} is the upper bound of the depth d; then the sink chooses $d=1$ when the network is saturated, that is, where $A>A_{max}$ even with $d=1$, and if the network is not saturated, then the sink gradually increases d while $A<A_{max}$ and stop increasing d when $A>A_{max}$ or $d>d_{max}$. Since the depth is controlled by the transmission power of beacon signal at the sink, there is an upper bound d_{max} that matches the maximum transmission power available at the sink. We verified in [23] that when $A=A_{max}$, the depth is at the optimal point where the network achieves both the maximum throughput and minimum loss. This analytical result justifies our approach of adjusting the power to reach that optimality.

The actual operation of dynamic depth tuning algorithm is as follows. When the sink starts up, it chooses the transmission power as ordinary sensor nodes operating in the network – this is where all the motes and sink use a common power. The sink monitors the channel and computes the schedule with size A as discussed in Section 4.2. At this point, two different cases may occur: either $A \leq A_{max}$ or $A>A_{max}$. If $A>A_{max}$, then the sink does not increase the transmission power for the next beacon transmission. If $A<A_{max}$, then the sink increments the transmission power of the next beacon by one power level and monitors the performance of channel. The sink keeps incrementing the transmission power in this manner until $A>A_{max}$ or the transmission power reaches its device-limited maximum. If $A>A_{max}$, then the sink decrements the transmission power of the next transmitted beacon by one level. If the transmission power reaches the maximum and $A<A_{max}$, then the sink keeps the transmission power at the maximum. The sink performs this dynamic depth-tuning algorithm on a continued basis, regulating the beacon transmission power accordingly. The pseudo code for dynamic depth-tuning algorithm is presented in [23].

5. Sensor Testbed Evaluation

We take an experimental approach to the evaluation of the funneling-MAC and present a number of experiments that give insights into the performance tradeoffs of the protocol under a wide variety of systems conditions, e.g., different traffic conditions, different mote topologies and densities (from simple benchmarks to more realistic dense grid), and compare the performance of the funneling-MAC to the baseline TinyOS B-MAC protocol and the Z-MAC [18].

5.1 Experimental Set-up

We implement the funneling-MAC on mica-2 motes using the default TinyOS [11] MintRoute routing protocol and Surge applications to drive different source rates. The bit

rate of the radio interface for mica-2 motes is 19.2 kbps. Our experimental testbed comprises of a 45 mote dense grid deployed in a large laboratory room and is configured, as shown in Figure 4 unless specified otherwise. Node spacing and transmission power of the sensors are set such that one-hop neighbors achieve > 80% delivery, while two-hop neighbors achieve < 20% delivery. In this way, a fairly strict and dense multi-hop radio environment is constructed for experimentation. We use the default TinyOS packet size, which is 36 bytes.

We implement the funneling-MAC on B-MAC, which provides the baseline CSMA system. Note, that we do not use fixed routes as in [10] because we are interested in how well the protocols under comparison, B-MAC, Z-MAC, and the funneling-MAC performs in a realistic networking scenario where time-varying radio conditions can impact coverage, link quality, and routing paths. For B-MAC and Z-MAC, we use the default settings described in [3] [10], respectively. The parameter settings of the funneling-MAC are presented in Table 1. The settings that are not specified in Table 1 are the settings used in [3] as the funneling-MAC is built on top of B-MAC. For all experiments, we turned off the low power listening and use the same preamble size for B-MAC, Z-MAC, and the funneling-MAC for fair comparison. We adjusted the data transmission power of sensor nodes at -10 dBm in order to build up a strict multi-hop network (up to 5 hops), as discussed in Section 2. The funneling-MAC dynamically tunes the power of beacon and schedule at the sink node from -10 dBm to 5 dBm (i.e., the maximum transmission power of the CC1000 transceiver [13]) in increments or decrements the power of 1 dBm which is the unit power level, as reported in [13].

Table 1. Funneling-MAC experimental parameters

Parameter	Value
Default data transmission power (C_{data})	-10 dBm
Beacon and schedule transmission power ($C_{control}$)	-10 ~ 5 dBm
Step size of power for dynamic depth-tuning (C_{unit})	1 dBm
Beacon interval (t_b)	20 sec
Superframe size (t_f)	1 sec
Slot size (t_s)	30 *m*sec
Moving average factor (α)	0.9

The beacon interval is initially computed based on the mote's clock accuracy and the required accuracy of synchronization for scheduling on the media. We run some experiments with various values for the beacon interval and we experimentally determine a beacon interval of 20 seconds gives the best performance in terms of throughput with the necessary accuracy. We also experiment with lazy-beaconing where we trade performance for a larger beacon interval. We observed that we can push the beacon interval out to 50 seconds with only a marginal drop in throughput performance. However, for beacon intervals greater than 50 seconds we register a sharp reduction in throughput measured at the sink of approximately 30%, showing that the loss of scheduling accuracy and schedule drift is too costly for the further reduction in signaling overhead. For the experiments reported in this section we chose a beacon interval of 20 seconds for increased scheduling accuracy and to remove any likelihood of schedule drift. Table 1 shows the set of experimental parameters for the funneling-MAC testbed that are consistently applied across all experiments.

5.2 Impact of Depth-Tuning

We are interested in evaluating the impact of the depth of the intensity region on the measured throughput of the sensor network testbed for the following reasons. First, in order to verify that by pushing the TDMA area (i.e., the intensity region) beyond the optimal depth will only degrade in measured throughput at the sink. Second, to show that the dynamic depth-tuning algorithm is valid when implemented in a real sensor testbed. To compare dynamic depth-tuning to the simple case of just scheduling the last hop (i.e., one hop from the sink) we fix the dynamic depth-tuning algorithm to one hop only. Note, that the results in Section 2 indicate that most packet loss occurs over the last hop to the sink. Following this logic, we consider a '*baseline algorithm*' as having a fixed depth of one, which only schedules the last hop, and an '*optimized algorithm*' that schedules additional hops using the fully enabled dynamic depth tuning algorithm. In what follows, we show that the optimized algorithm achieves considerably better performance than the simple baseline algorithm does.

In order to observe the impact of depth on performance, we fix the beacon transmission power to the values of -10, -8, -6, -4, 0, and 4 dBm, respectively. The depth of the intensity region is an approximate function of the beacon transmission power used. In essence, we can approximate the depth in terms of the beacon transmission power coverage distance in terms of number of hops from the sink for our grid network. For example, if the sink transmits a beacon using the default transmission power of ordinary sensor nodes, this will approximate coverage of one hop from the sink. Likewise, we can expect that a beacon will have a greater coverage than one hop for higher transmission powers. The metric that we observe with each beacon power setup is the throughput. We define the choke point throughput of the sink as the amount of data in terms of bits received at sink over a 1 second period. In these experiments, all 44 nodes are sources. We run experiments for 3 different source rates low, medium, high: 0.2 pps, 1 pps, 2 pps, respectively.

We plot the results in Figure 7. For each of the source rates we measured the sink throughput for increasing beacon power (which approximates the depth of the intensity region coverage). The result indicates that there is an approximation of the optimal transmission power for beacons (i.e., optimal depth) that maximize the throughput such that if we use a larger transmission power than the optimal power, the throughput measured at the sink degrades. This means that if we increase the TDMA area further the optimal depth by using more power then it degrades the measured throughput.

Figure 7 validates the dynamic depth-tuning algorithm. According to the analytical result in the Appendix in [23], the optimal depth is close to 1 hop (i.e., the beacon

transmission power is the same as the motes data event transmission power) when the network is saturated, while the optimal depth is greater than 1 hop when the network is not saturated. In fact, if we set the source rate to 2 pps, which drives the network toward saturation, the optimal beacon transmission power from our experimental result is -8 dBm, which provides radio coverage close to 1 hop (i.e., the mote's data event transmission power of -10 dBm). We observe in Figure 7 that the optimal depth is greater than 1 hop when the network is not saturated (i.e., 0 dBm for 1 pps, and 4 dBm or greater for 0.2 pps). These experimental observations validate the analytical observation in [23] and thus provide a sound basis for the dynamic depth-tuning algorithm.

In what follows, we quantify how much gain the baseline and optimized algorithms can achieve over B-MAC. From Figure 13(c), we can observe that the throughput of B-MAC for 0.2 pps, 1 pps, and 2 pps source traffic rates is 272 bps, 1099 bps, and 1631 bps, respectively (we discuss this plot further in Section 5.5). The throughput related to the -10 dBm x-axis value in Figure 7 (i.e., 1583 bps for 2 pps, 1511 bps for 1 pps, and 645 bps for 0.2 pps) represent the performance of the funneling-MAC's baseline algorithm that schedules only the last hop with the depth fixed by -10 dBm beacon power. The throughput shown in Figure 7 at the optimal beacon transmission powers (i.e., 1872 bps at -8 dBm for 2 pps, 1925 bps at 0 dBm for 1 pps, and 1191 bps at 4 dBm or greater for 0.2 pps) represent the performance of the funneling-MAC's optimized algorithm (i.e., when dynamic depth-tuning is fully enabled). The gain over B-MAC for the baseline algorithm with 0.2 pps, 1 pps, and 2 pps is 124%, 37%, and 0%, respectively. The gain over B-MAC for the optimized algorithm with 0.2 pps, 1 pps, and 2 pps is 338%, 75%, and 15%, respectively. For all source traffic rates (viz. 0.2 pps, 1 pps, and 2 pps) the optimized algorithm performs better than the baseline algorithm. More importantly, the baseline algorithm does not achieve any gain over B-MAC when the source rate is 2 pps. This result indicates that the baseline algorithm provides some gain that may be sufficient for simple low complexity deployments (i.e., schedule only the last hop) but the optimized algorithm provides considerably better performance despite that the optimized algorithm comes with some added complexity over the baseline algorithm. As a result, we recommend using dynamic depth-tuning in its fully enabled form as a default.

5.3 Impact of Boundary Node Interference

In what follows, we show that the meta-schedule advertisement is effective at dealing with the interference scenarios discussed in Section 4.4. We study the impact of abruptly changing the depth of the intensity region on boundary node behavior and the measured sink throughput performance. In this experiment meta-schedule advertisements exploit the broadcast nature of the radio medium, where nodes receive the embedded meta-schedule simply by overhearing data event packets with embedded meta-schedules sent by neighboring nodes. The use of meta-schedule allow for the co-existence of TDMA inside the intensity region and pure CSMA outside that region.

We evaluate the behavior of nodes at the boundary of the intensity region for some interfering scenarios. We set up an experiment that studies the impact of boundary variability. In this experiment the sink changes the beacon transmission power for every beacon by selecting the transmission power between two values in turn. We choose the two beacon transmission power values -6 dBm and -8 dBm such that the boundary of the intensity region falls approximately across the center of the grid testbed where there is a higher density of nodes that will be included in TDMA scheduling (at -8 dBm) and then dropped out (at -6 dBm) as they fall outside of the intensity region and operate without the framing and timing information, as shown in Figure 4.

We run the experiment of switching between -6 dBm and -8 dBm for a number of different source data rates. Figure 8 shows the various source rates and the corresponding throughput performance measured at the sink. This is for the case where all the 44 motes are sources. We study two experiments, one called variable power where the transmission is alternating between -6 dBm and -8dBm, and one called fixed power where we fix the beacon transmission power to -7 dBm which represents the average of the variable case. The comparison of the throughput measured on each experiment is shown in Figure 8. We run the experiment five times for each data rate and calculate 95% confidence interval. From the plot we can see that the measured throughput for fixed and variable power cases are

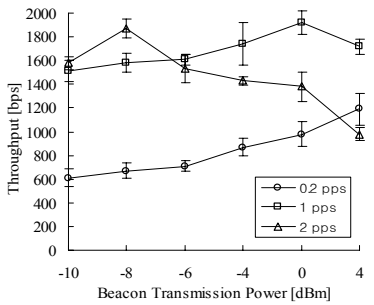


Figure 7. The impact on throughput over varying depth in terms of beacon transmission power

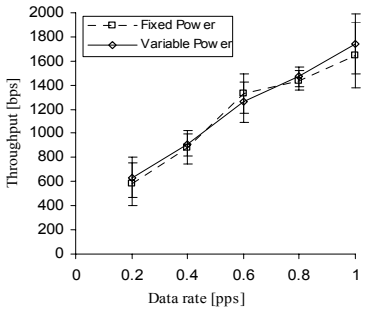


Figure 8. Throughput with fixed/variable beacon transmission power

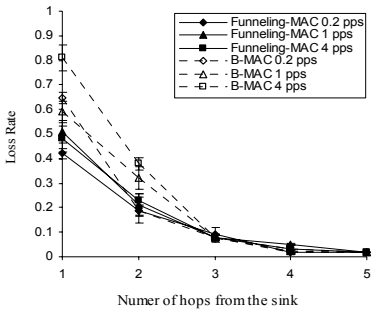


Figure 9. Loss rate over varying distance from the sink for B-MAC and the funneling-MAC

almost the same (i.e., within the confidence interval of each other). This result indicates that boundary variability stressed in this test has little impact on the ability of the funneling-MAC to operate stably. As part of this test we instrument the motes to record if the beacon timeout occurred and the mote had no framing information but overheard meta-schedules. We found that 8% of the boundary motes fall into this category; that is, motes that are consistently inside and outside of the intensity region as the beacon transmission power toggled between -6 dBm and -8 dBm at the beacon interval. This indicates that these 8% of nodes would have become interfering modes if they had not successfully overhead embedded meta-schedule advertisements.

5.4 Loss Rate Distribution

In Section 2, we quantify the impact of the funneling effect on the packet loss rate distribution for B-MAC. In what follows, we now assess the impact of the funneling effect on the funneling-MAC. We use the same setup (i.e., multi-hop testbed using 45 motes) and metric (i.e., loss rate) as in Figure 3. The result is presented in Figure 9. For the comparison, we also include the B-MAC result in the figure. Figure 9 shows the loss rate across an increasing number of hops from the sink. We observe that the funneling effect is mitigated by comparing the steepness of the slopes of the funneling-MAC and B-MAC curves, respectively. The loss rate over the first two hops from the sink is significantly different because the funneling effect is active, before both curves converge on the same performance at three hops from the sink where the funneling effect is no longer present in the experiments. After this point both MACs offer similar CSMA performance for 3, 4, and 5 hops from the sink. However, the loss rate over the first two hops is considerably smaller when using funneling-MAC over B-MAC. For example, at the higher source rate of 4 pps B-MAC's loss rate is 81% at one hop and 40% at two hops from the sink, while the funneling-MAC reduces those loss rates to 48% at one hop and 22% at two hops for the same source rate. The loss rate for the funneling-MAC remains almost the same for varying source traffic rates (viz. 0.2 pps, 1 pps, 4 pps) while the loss rate for B-MAC varies considerably with source rate and across the first two hops from the sink. In the following section, we show how this reduction of loss rate in the first few hops impacts the overall throughput performance of the sensor network.

5.5 Multi-hop Throughput

Due to space considerations we do not include the one-hop and two-hop benchmarks tests that verify the correctness of our testbed setup by reproducing the results achieved with B-MAC [3] and Z-MAC [10] in comparison to the funneling-MAC performance. See [23] for the details of these results. The benchmark results discussed in [23] presents the same pattern reported for B-MAC [3] and Z-MAC [10] but with minor difference in scale. In addition, the results show that the funneling-MAC achieves almost the same throughput as Z-MAC and outperforms B-MAC. In what follows, we compare the throughput of B-MAC, Z-MAC, and the funneling-MAC in our multi-hop testbed consisted of 45 motes.

Figure 10 shows the trace of the throughput of the funneling-MAC, Z-MAC, and B-MAC over time for the experiment where all 44 nodes are sources generating 5 pps. This scenario represents a heavy traffic load. We run the experiments five times with this setup and compute the average throughput with 95% confidence interval. At start of the experiment, Z-MAC and the funneling-MAC perform equally while B-MAC performs worst. It is worth noting that the routing paths from all sources are not completely established until approximately 20 minutes into experiment. Protocols such as MintRoute take a significant amount of time with the default settings to create sufficient routing state [14] before the performance of the network stabilizes at around 20 minutes into the experiment. When a node does not have a route it sends the event data to the broadcast channel, which contributes to congestion and degrades the throughput further. As more source nodes acquire routes and path, the funneling-MAC and B-MAC gain performance in terms of throughput. The funneling-MAC outperforms B-MAC consistently over time.

Schedule Drift: We can observe from Figure 10 that Z-MAC throughput steadily degrades as time increases. In [10] it is noted that Z-MAC runs DRAND only at the beginning and not periodically. Hence it is possible that the reason Z-

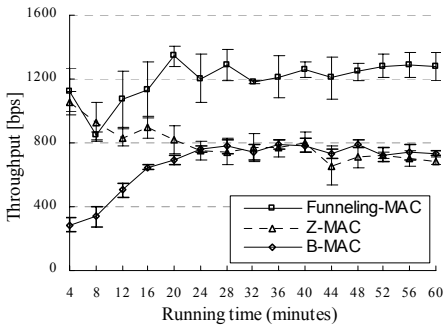


Figure 10. Trace of throughput over running time for the funneling-MAC, B-MAC, and Z-MAC (Dartmouth College Testbed)

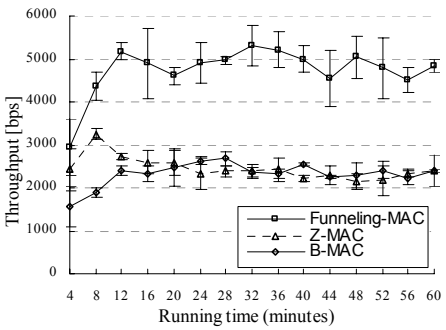


Figure 11. Trace of throughput over running time for the funneling-MAC, B-MAC, and Z-MAC (Columbia University Testbed)

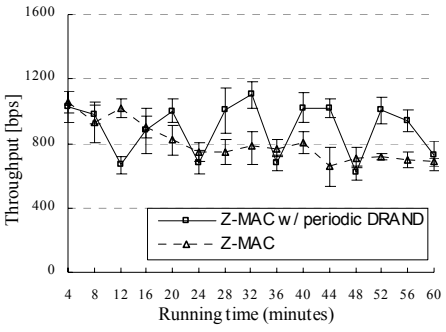


Figure 12. Trace of throughput over running time for Z-MAC with/without periodic DRAND (Dartmouth College Testbed)

MAC degrades is because it is susceptible to schedule drift where the initial schedule computed by DRAND is no longer valid due to time-varying radio conditions and possibly route changes, forcing Z-MAC to fall back to the performance of CSMA, as shown in the plot. To verify that the Z-MAC throughput degradation is not a product of our dense grid sensor testbed setup at Dartmouth College we ran the same experiment on a more sparse sensor testbed at Columbia University [23]. The Columbia testbed consists of only 31 mica-2 motes. The transmission power of each mote is set to -10dbm, and at this power, on average all nodes have at least 7 nodes from which the packet delivery ratio is at least 80%. The result presented in Figure 11 show that the Z-MAC throughput degradation is reproducible on the Columbia University testbed although the scale and timing of the degradation are different. For more details and results from the Columbia University testbed experiments see [23].

To confirm that the throughput degradation is due to schedule drift, we run Z-MAC with DRAND running at the start of the experiment and then every 12 minutes. After each DRAND run, each node reports its neighborhood table to the sink. We analyzed these reports of neighbor table from each node. Each node keeps a table that contains the IDs of its neighbors in its two-hop transmission range. The schedule (computed by DRAND) is the result of the interaction between the nodes within the two-hop transmission range. Hence, the neighbor table is a good indication of the validity of the schedule. Analyzing these reports, we observe that 76.7% of the nodes experience some changes in their neighbor table after each DRAND re-run.

In order to quantify the degree of change in neighborhood, we define a metric for the degree of change in the neighbor table of a node as,

$$X_{ij} = \frac{D_{ij} + D_{ji}}{H_{ij}}, \qquad (1)$$

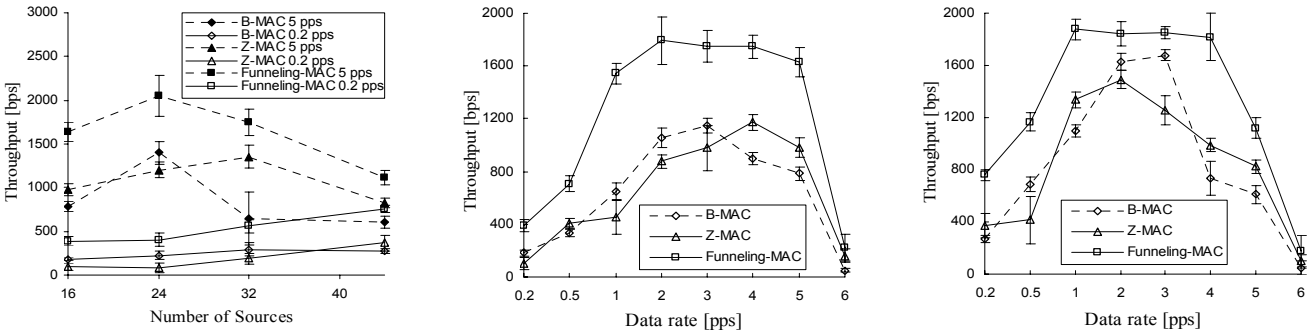
where T_i is the neighbor table of a node after i-th DRAND run, D_{ij} is the number of nodes that are in neighbor table T_i but not in table T_j, and D_{ji} is the number of nodes that are in neighbor table T_j but not in T_i, and H_{ij} is the number of nodes that are in neighbor table T_i OR T_j.

After calculating X_{12}, X_{23}, X_{34}, and X_{45} of each node, we calculated the average of X_{12}, X_{23}, X_{34}, and X_{45} among the nodes. From the experiment, the average of X_{12} (i.e., the degree of changes between the first DRAND run at initial time and the second DRAND) is 45.5%. The average of X_{23} is 25.3%, the average of X_{34} is 30.5%, and the average of X_{45} is 31.0%.

The fact that the majority (76.7%) of the nodes experience some change in their neighbor table and the fact that average degree of change is considerable (ranging from 25.3% to 45.5%) indicates that the schedule computed by DRAND at the start of the experiment does not take into account the changes in neighborhood over the time. The neighborhood change is due to radio characteristic variability and environmental factors.

The throughput comparison between Z-MAC without periodic DRAND and Z-MAC with periodic DRAND is shown in Figure 12. The solid lines in Figure 12 indicate that Z-MAC with periodic DRAND does not suffer from the throughput degradation except for the time DRAND is running. During each DRAND run (at 12, 24, 36, and 48 minute in the experiment), the throughput degrades below 700 bps. At the beginning of the DRAND run, the DRAND neighbor table is initialized (zero entry in the table), so the network operates just like B-MAC during DRAND run. In addition, the signaling overhead of DRAND contributes to the degradation. It is shown in the Section 5.6 that the signaling overhead of DRAND is large. As soon as the DRAND run is complete, the throughput returns back to 1000 bps or greater; the value of which is the same level as the throughput just before the DRAND run begins. This result (that Z-MAC with periodic DRAND does not suffer from throughput degradation) indicates that the schedule computed by DRAND running every 12 minutes is responsive to neighbor changes for our network configuration and conditions. Hence, this result indicates that the cause of throughput degradation is due to the schedule drift where the initial schedule computed by DRAND is no longer valid.

Varying Workload: Figure 13 compares the throughput of the funneling-MAC, Z-MAC, and B-MAC over varying number of sources and data rates. We varied the number of sources each time we run an experiment. When the number of sources is less than 44, the sources are randomly chosen among 44 sensor nodes in the grid. We also varied the packet generation rates of the sources each time we run a test. The



(a) Varying number of sources (b) 16 sources with varying data rate (c) 44 sources with varying data rate

Figure 13. Throughput comparison of the funneling-MAC, Z-MAC, and B-MAC

x-axis values represent the number of packets generated per source node.

From Figure 13 we note that the overall throughput results show that the funneling-MAC consistently achieves greater throughput than B-MAC and Z-MAC under various conditions, such as changing number of sources and changing data rates (from very low rates to very high rates). The throughput curves for Z-MAC and B-MAC shown in Figure 13 follow the same general pattern in [10]. For light traffic, Z-MAC and B-MAC perform the same, while for high loads Z-MAC outperforms B-MAC. Note that for data rates higher than 5 pps, MintRoute protocol cannot setup routing paths for any node, thus, after this rate the network goes into collapse as clearly indicated in Figure 13 (b).

There are two reasons why Z-MAC only shows marginal improvements over B-MAC in the presence of the funneling effect while the funneling-MAC outperforms both B-MAC and Z-MAC by a large margin. One of the reasons is the schedule-drift associated with Z-MAC. The other reason is currently that DRAND cannot take into account the funneling effect and the need to allocate more slots to nodes in the intensity region. This is because DRAND is a pure distributed coloring algorithm that does not take into account the existence of the central entity (i.e., the sink) and the funneling traffic pattern. Hence, DRAND can only allocate the same amount of slots among its two hop neighbors. In contrast, the funneling-MAC is capable of allocating slots based on the funneling traffic pattern so that the funneling-MAC allocates more slots to the nodes closer to the sink providing a big win in performance of the measured throughput at the sink.

We also note that the funneling-MAC performs better than Z-MAC and B-MAC even under light traffic conditions where the funneling effect is evident. So even under light load B-MAC and Z-MAC are not capable of mitigating the negative effects of funneling.

5.6 Energy Tax and Signaling Overhead Cost

In order to analyze the cost of delivering data packets to the sink, we define the energy tax E_{tax} as,

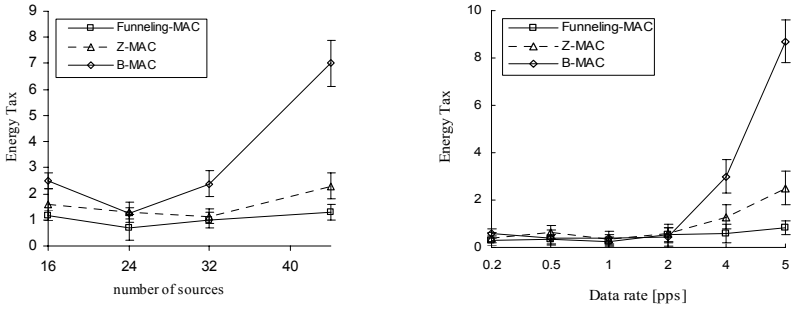
$$E_{tax} = \frac{D_t + C_t}{D_d \cdot n} \quad (2)$$

where D_t is the amount of data packets transmitted (either by source nodes or intermediate nodes) in number of bits, C_t is the total amount of control packets transmitted in number of bits, D_d is the amount of packets delivered to the sink, and n is the number of nodes in the network. We measure the signaling overhead cost E_{sig} as,

$$E_{sig} = \frac{C_t}{D_d \cdot n} \quad (3)$$

The energy tax includes the overhead of control messages if a MAC protocol introduces some control messages. The funneling-MAC introduces signaling for beacon packets, schedule packets, path information field, and meta-schedule, which we considered when computing the energy tax. B-MAC does not introduce any control packets. Z-MAC introduces sync packets for local synchronization. In the Figures 14 and 15 we consider Z-MAC with and without the DRAND overhead. If a data packet has to travel i hops to reach the sink, the energy used in delivering this packet i hops is included in the energy tax as well. If a data packet is lost after traveling j hops, the energy used in delivering this packet j hops is included in the energy tax. The signaling overhead cost only considers the overhead of control messages and not the cost due to packet loss. The testbed settings are the same as the settings in Section 5.5.

The funneling-MAC introduces more signaling overhead compared to B-MAC but the funneling-MAC reduces the energy wasted by reducing the packet losses to the extent that the funneling-MAC has lower or equal energy tax compared to B-MAC despite its signaling overhead, as we can see from Figure 14. B-MAC does not perform well when the data rate is greater than 2 pps resulting in a high energy tax for B-MAC. In contrast, the funneling-MAC exhibits consistently lower energy tax. Z-MAC also reduces the energy wasted by packet losses to the extent that Z-MAC has a lower or equal energy tax compared to B-MAC despite the overhead of sync packets. However, Z-MAC has a greater energy tax than the funneling-MAC when the data rate is greater than 2 pps. This is because the funneling effect impacts Z-MAC's overall energy tax as packet loss increases. In summary, our results indicate that even though the funneling-MAC has more signaling in its basic protocol than the other protocols, it is a more energy efficient than Z-MAC and B-MAC. The signaling overhead cost of the funneling-MAC and Z-MAC are similar if we do not consider the overhead of running DRAND. Also from the schedule drift exhibited in Figures 10, 11 and 12 for Z-MAC it would be



(a) Varying number of sources with 5 pps (b) 44 sources with varying data rate

Figure 14. Energy tax comparison of the funneling-MAC, Z-MAC, and B-MAC

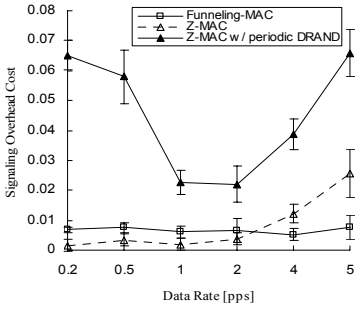


Figure 15. Signaling overhead cost of the funneling-MAC and Z-MAC

necessary to re-run DRAND to boost its performance and resolve the schedule drift. Based on the observations from Figure 12, it would be costly to re-run DRAND every 12 minutes to maintain performance. If we do this then the cost of operating Z-MAC and its DRAND mechanism increases significantly, as indicated in Figure 15. Clearly, it is not tenable to re-run DRAND periodically because of the significant overhead incurred; this is clearly shown in Figure 15.

6. Conclusions

The main contributions of this paper are as follows. We show that by implementing a simple hybrid TDMA/CSMA scheme in the intensity region, under the control of the sink, can significantly improve the throughput and loss performance of sensor networks, even under lightly loaded traffic conditions, and for small intensity region depths of one or two hops. We also show experimentally that multiple MACs can coexist in the sensor network, specifically, we can run a hybrid TDMA/CSMA in the intensity region which seamlessly coexists with pure CSMA outside of that region, in addition, any potential interference caused by dynamically increasing or decreasing the intensity region (i.e., the TDMA/CSMA region) is effectively managed by the funneling-MAC. We show that the funneling-MAC out-performs B-MAC and Z-MAC under a wide variety of network and traffic conditions. The TinyOS source code for the funneling-MAC is available from the web [22].

7. Acknowledgements

This work is supported by the Army Research Office (ARO) under Award W911NF-04-1-0311 on resilient sensor networks. We would like to thank Injong Rhee and Ajit Warrier for providing us with TinyOS code of Z-MAC. We would also like to thank Nirupama Bulusu for shepherding our paper and the anonymous reviewers for their excellent comments and suggestions.

8. References

[1] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks", In Proc. of IEEE INFOCOM 2002, June 2002.

[2] Tijs van Dam, and Koen Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks", In Proc. of 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), November 2003.

[3] J. Polastre, J. Hill, D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks", In Proc. of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), November 2004.

[4] K. A. Arisha, M. A. Youssef, and M. F. Younis, "Energy-Aware TDMA-Based MAC for Sensor Networks", In Proc. of IEEE Workshop on Integrated Management of Power Aware Communications, Computing and NeTworking, IMPACCT 2002, New York, May 2002.

[5] Venkatesh Rajendran, Katia Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks", In Proc. of 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), November 2003.

[6] G. Pei and C. Chien, "Low Power TDMA in large wireless sensor networks", In Proc. of IEEE Communications for Network Centric Operation MILCOM 2001, October 2001.

[7] Chieh-Yih Wan, Shane E. Eisenman, Andrew T. Campbell, and John Crowcroft, "Siphon: Overload Traffic Management using Multi-Radio Virtual Sinks", In Proc. of 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005), November 2005.

[8] Shashidhar Gandham, Milind Dawande, and Ravi Prakash, "Link Scheduling in Sensor Networks: Distributed Edge Coloring Revisited", In Proc. of IEEE INFOCOM 2005, Miami, USA, March 2005.

[9] J. Li and G. Lazarou, "A bit-map-assisted energy-efficient MAC scheme for wireless sensor networks", In Proc. of IEEE Information Processing in Sensor Networks (IPSN '04), Berkeley, USA, April 2004.

[10] Injong Rhee, Ajit Warrier, Mahesh Aia and Jeongki Min, "Z-MAC: a Hybrid MAC for Wireless Sensor Networks", In Proc. of 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005), November 2005.

[11] TinyOS web site: http://www.tinyos.net.

[12] X. Chen, "Dual near field effect in radio frequency simulations", 2002 Summer Computer Simulation Conference, San Diego, USA, July 2002.

[13] Chipcon Corporation, "CC1000 low power FSK transceiver", http://www.chipcon.com/files/CC1000DataSheet.pdf.

[14] A. Woo and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks", In Proc. of 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), November 2003.

[15] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks", In Proc. of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), November 2004.

[16] D. Petrovic, R. C. Shah, K. Ramchandran, J. Rabaey, "Data funnelling: routing with aggregation and compression for wireless sensor networks", In Proc. of IEEE Sensor Network Protocols and Applications (SNPA 2003), May 2003.

[17] N. Shrivastava, C. Buragohain, D. Agrawat, S. Suri, "Medians and Beyond: new aggregation techniques for sensor networks", In Proc. of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, November, 2004.

[18] Z-MAC TinyOS Source Code: http://www.csc.ncsu.edu/faculty/rhee/export/zmac/software/zmac/zmac.htm

[19] A. Woo and D. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks", In Proc. of ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2001), Rome, Italy, July 2001.

[20] B. Hohlt, L. Doherty, and E. Brewer, "Flexible Power Scheduling for Sensor Networks", In Proc. of IEEE Information Processing in Sensor Networks (IPSN '04), Berkeley, April 2004.

[21] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts", In Proc. of 5th Symposiumon Operating Systems Design and Implementation (OSDI 2002), December 2002.

[22] Funneling-MAC project webpage: http://www.cs.dartmouth.edu/~sensorlab/funneling-mac/

[23] Funneling-MAC Technical Report: http://www.cs.dartmouth.edu/~sensorlab/funneling-mac/TAP-TR-2006-08-003.pdf.