

Z-MAC: a Hybrid MAC for Wireless Sensor Networks

Injong Rhee, Ajit Warriar, Mahesh Aia, Jeongki Min and Mihail Sichitiu

Abstract—

This paper presents the design, implementation and performance evaluation of a hybrid MAC protocol, called Z-MAC, for wireless sensor networks that combines the strengths of TDMA and CSMA while offsetting their weaknesses. Like CSMA, Z-MAC achieves high channel utilization and low-latency under low contention and like TDMA, achieves high channel utilization under high contention and reduces collision among two-hop neighbors at a low cost. A distinctive feature of Z-MAC is that its performance is robust to synchronization errors, slot assignment failures and time-varying channel conditions; in the worst case, its performance always falls back to that of CSMA. Z-MAC is implemented in TinyOS.

Index Terms—MAC, CSMA, TDMA, wireless sensor networks.

I. INTRODUCTION

A radio channel cannot be accessed simultaneously by two or more nodes that are in a radio interference range – neighboring nodes may cause “conflict” or signal interference at some nodes if transmitting at the same time on the same channel. In wireless sensor networks, controlling access to the channel, generally known as *medium access control* (MAC), plays a key role in determining channel utilization, network delays and more important, power consumption, also influencing congestion and fairness in channel usage.

Sensor networks serve many diverse applications from low data rate event driven monitoring applications to high data rate real-time industrial applications. Balakrishnan [1] reports that some high data rate applications can reach sensing rates of 10^2 to 10^5 Hz and consume from a few bytes per seconds up to 10 or 100 Mbps aggregate bandwidth; these applications require over 5 times improvement on the channel utilization of existing sensor networking technologies. Notwithstanding high channel utilization, traditional sensor network requirements such as power efficiency, scalability, robustness and small footprints must also not be compromised.

CSMA (carrier sense multiple access) is popular in wireless networks due to its simplicity, flexibility and robustness. It does not require much infrastructure support: no clock synchronization and global topology information are required, and dynamic node joining and leaving are handled gracefully without extra operations. These advantages, however, come at the cost of trial and error – a trial may cost access *collision* where more than two “conflicting” nodes transmit at the same time

causing signal fidelity degradation at destinations. Collision can happen in any two-hop neighborhood of a node. While collision among one-hop neighbors can be greatly reduced by carrier sensing before transmission, carrier sensing does not work beyond one hop. This problem, called the *hidden terminal* problem, causes a serious throughput degradation especially in high data rate sensor applications. Although RTS/CTS can alleviate the hidden terminal problem, it incurs high overhead (40% to 75% of the channel capacity in sensor networks [2], [3]) because data packets are typically very small in sensor networks.

TDMA (time-division multiple access), on the other hand, can solve the hidden terminal problem without extra message overhead because it can schedule transmission times of neighboring nodes to occur at different times. However, TDMA has many other disadvantages as documented in [4]. First, finding an efficient time schedule in a scalable fashion is not trivial. It often requires a centralized node to find a collision-free schedule. Furthermore, developing an efficient schedule with a high degree of concurrency or channel reuse is very hard (the optimal solution is NP-hard [5]). Second, TDMA needs clock synchronization. Although clock synchronization is an essential feature of many sensor applications, tight synchronization incurs high energy overhead because it requires frequent message exchanges. Third, sensor networks may undergo frequent topology changes because of time-varying channel conditions, physical environmental changes, battery outage and node failures. Handling dynamic topology changes is expensive, possibly requiring a global change. Fourth, it is difficult to ascertain the interference relation among neighboring nodes because radio interference ranges are different from communication ranges and some interfering nodes may not be in a direct communication range (this phenomenon is known as *interference irregularity* [6]). Therefore, any channel assignment that uses the communication ranges, in place of interference ranges, for building the conflict relations does not necessarily yield an interference free schedule. Furthermore as interference ranges and also channel conditions are highly time-varying, it is unlikely that one fixed schedule is sufficient to prevent collision all the time. Fifth, during low contention TDMA gives much lower channel utilization and higher delays than CSMA because in TDMA, a node can transmit only during its scheduled time slots whereas in CSMA, nodes can transmit at any time as long as there is no contention.

These difficulties with TDMA suggest that a stand-alone TDMA scheme is not practical. Even if we have an efficient TDMA schedule, the other factors such as interference irregularity, time-varying channel conditions and clock syn-

Injong Rhee, Ajit Warriar, Mahesh Aia and Jeongki Min are with the Computer Science Department, and Mihail Sichitiu is with the Dept. of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC-27695, USA. (email: {rhee,acwarrie,mgaia,jkmin,mlsichit,}@ncsu.edu).

chronization errors would diminish the benefits of TDMA. Nevertheless, we posit that the information provided by an efficient TDMA schedule, in particular, the independent sets of nodes that can transmit concurrently can be used in curtailing occurrences of collision especially under high contention. This position greatly motivates our work.

In this paper, we present a new hybrid MAC scheme, called *Z-MAC* (Zebra MAC), for sensor networks that combines the strengths of TDMA and CSMA while offsetting their weaknesses. The main feature of *Z-MAC* is its adaptability to the level of contention in the network – under low contention, it behaves like CSMA, and under high contention, like TDMA. It is also robust to dynamic topology changes and time synchronization failures commonly occurring in sensor networks.

Z-MAC uses CSMA as the baseline MAC scheme, but uses a TDMA schedule as a “hint” to enhance contention resolution. In *Z-MAC*, a time slot assignment is performed at the time of deployment - higher overhead is incurred at the beginning. Its design philosophy is that the high initial overhead is amortized over a long period of network operation, eventually compensated by improved throughput and energy efficiency. We use *DRAND* [7], an efficient scalable channel scheduling algorithm. *DRAND* is a distributed implementation of *RAND* [5], a centralized channel reuse scheduling algorithm. After the slot assignment, each node reuses its assigned slot periodically in every predetermined period, called *frame*. We call a node assigned to a time slot an *owner* of that slot and the others the *non-owners* of that slot. There can be more than one owner per slot because *DRAND* allows any two nodes beyond their two-hop neighborhoods to own the same slot.

Unlike TDMA, a node may transmit during any time slot in *Z-MAC*. Before a node transmits during a slot (not necessarily at the beginning of the slot), it always performs carrier-sensing and transmits a packet when the channel is clear. However, an owner of that slot always has higher priority over its non-owners in accessing the channel. The priority is implemented by adjusting the initial contention window size in such a way that the owners are always given earlier chances to transmit than non-owners. The goal is that during the slots where owners have data to transmit, *Z-MAC* reduces the chance of collision since owners are given earlier chances to transmit and their slots are scheduled a priori to avoid collision, but when a slot is not in use by its owners, non-owners can steal the slot. This priority scheme has an effect of implicitly switching between CSMA and TDMA depending on the level of contention. An important feature of this priority scheme is that the probability of owners accessing the channel can be adjusted independently from that of non-owners. We show that this feature contributes to increasing the robustness of the protocol to synchronization and slot assignment failures while enhancing its scalability to contention.

By mixing CSMA and TDMA, *Z-MAC* becomes more robust to timing failures, time-varying channel conditions, slot assignment failures and topology changes than a stand-alone TDMA; in the worst case, it always falls back to CSMA. Since *Z-MAC* needs only local synchronization among senders in two-hop neighborhoods, we devise a simple local synchronization scheme where each sending node adjusts its

synchronization frequency based on its current data rate and resource budget.

In what follows, we describe the design, implementation and performance of *Z-MAC* in detail.

II. RELATED WORK

S-MAC [4] and *T-MAC* [8] are a hybrid of CSMA and TDMA in that they also maintain the synchronized time slots, but unlike TDMA their slots can be much bigger than normal TDMA slots and synchronization failures do not necessarily lead into communication failure because they employ RTS/CTS. Nodes maintain periodic duty cycle to listen for channel activities and transmit data. As these protocols use RTS/CTS, the overhead of the protocols is quite high because most data packets in sensor networks are small. *T-MAC* [8] improves the energy efficiency of *S-MAC* by forcing all the transmitting nodes to start transmission at the beginning of each active period.

B-MAC [3] is the default MAC for Mica2. *B-MAC* allows an application to implement its own MAC through a well-defined interface. They also adopt LPL (low power listening) [9] and engineer the clear channel sensing (CCA) technique to improve channel utilization. *CSMA/p** [10] uses the optimal probability distribution in determining the channel access probability for CSMA when the number of senders N is known. When N is unknown, it provides sub-optimal performance. *Sift* [11] adapts *CSMA/p** [10] for a network where N is unknown. The result is high success probability for channel access and reduced collision probability, thus achieving good throughput under both low and high contention. However the optimal probability distribution works only when senders always have data to transmit and they are synchronized for the channel access, and thus, when data arrivals to a node are highly random and senders cannot sense each other for data transmission (as in two-hops), its performance degenerates to the case of CSMA with the uniform access probability distribution. *Sift* relies on RTS/CTS to handle hidden terminals.

TDMA has long been dismissed as an impractical solution for wireless ad hoc networks for its lack of scalability and adaptability to changing environments. However, it provides a good energy efficiency and collision-freedom. Recently, several proposals [12], [13] are made for TDMA in sensor networks. However, these protocols still fail to address the fundamental difficulties that stand-alone TDMA schemes face.

Seamlessly switching between TDMA and CSMA according to the level of contention was previously explored by Ephremides and Mowafi [14] for a wireless LAN (or 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 (“a”) and that of non-owners (“b”), PTDMA adapts the behavior of MAC between TDMA and CSMA depending on contention. These probabilities are adjusted by a function $a + (M - 1)b = 1$ where M is the number of senders. While PTDMA and *Z-MAC* share a common goal, PTDMA, being designed primarily for a one-hop wireless LAN environment, does not deal with many difficulties that TDMA faces in ad hoc sensor networks such

as time synchronization errors, interference irregularity and topology changes. These failures can drastically reduce the performance of PTDMA. PTDMA also assumes buffered senders where all nodes experience the same statistical arrival. In a network where only a subset of nodes is active data sources (which is a common scenario in sensor networks), PTDMA exhibits very low channel utilization and does not behave like CSMA. This is because “b” cannot be arbitrarily set to a high value without reducing “a” (reducing “a” also causes MAC not to behave like TDMA) due to the dependency between “a” and “b”. The effect of probability “a” is also not clear; it seems that the authors want to adjust “a” for different contention levels but the paper does not mention how this can be achieved (Z-MAC does not need to dynamically adjust its parameters to achieve the desired effect).

III. DESIGN OF Z-MAC

Z-MAC has a setup phase in which it runs the following operations in sequence: *neighbor discovery*, *slot assignment*, *local frame exchange* and *global time synchronization*. These operations run only once during the setup phase and do not run until a significant change in the network topology (such as physical relocation of sensors) occurs. The idea is that the initial upfront costs for running these operations are amortized by improved throughput and energy efficiency during data transmission. In this section, we first describe how we implement these setup phase operations and then discuss how they are integrated with transmission control in Z-MAC.

A. Neighbor Discovery and Slot Assignment

As a node starts up, it first runs a simple neighbor discovery protocol where it periodically broadcasts a ping to its one-hop neighbors to gather its one-hop neighbor list. A ping message contains the current list of its one-hop neighbors. In our implementation, each node sends one ping message at a random time in each second for 30 seconds. Through this process, each node gathers the information received from the pings from its one-hop neighbors which essentially constitutes its two-hop neighbor information.

The two-hop neighbor list is used as input to a time slot assignment algorithm. The current implementation of Z-MAC uses DRAND [7], a distributed implementation of RAND [5], to assign time slots to every node in the network. DRAND ensures a broadcast schedule where no two nodes within a two-hop communication neighborhood are assigned to the same slot. This assignment guarantees that no transmission by a node to any of its one-hop neighbors interferes with any transmission by its two-hop neighbors. Note that a broadcast schedule can handle any routing changes among its one-hop neighbors.

The performance of DRAND is scalable because it does not depend on the network size, but on the local neighborhood size of each node. The protocol produces a very efficient time schedule where the slot number assigned to a node does not exceed the size of its local two-hop neighborhood (δ) – in most cases, much less than that. The running time and message complexity of DRAND is also bounded by $O(\delta)$.

Thus, its energy cost is linearly proportional to the size of the local neighborhood. When only a small number of new nodes are joined late, DRAND can also perform localized time slot assignment without modifying the time slots already assigned to the existing nodes. The detailed performance analysis of DRAND can be found in [7].

B. Local Framing

Once a node picks a time slot, each node needs to decide on the period in which it can use the time slot for transmission. This period is called the *time frame* of the node. The conventional wisdom is that all nodes must keep the same time frame while all nodes synchronize to have their time slot 0 at the same time. But this requires to propagate the maximum slot number (MSN) to the entire network and is also not adaptive to local time slot changes. When new nodes are added to the network, DRAND can run local slot assignment while maintaining the existing assignment. If this assignment causes the MSN to be changed, that change must be propagated again to the entire network. This could incur high cost for adapting to a small change in the network topology. (Note that network topology changes by unstable radio channel conditions are handled by the inherent operation of Z-MAC so it does not incur new assignment, but new node joining or node redeployment can cause slot changes.)

We present a new scheme where each node maintains its own local time frame that fits its local neighborhood size, but avoids any conflict with its contending neighbors. The main idea is as follows.

Time frame rule (TF rule). Let a node i be assigned to a slot s_i according to DRAND and the MSN within its two-hop neighborhood be F_i . We set i 's time frame to be $L_i = 2^a$ where a positive integer “a” is chosen to satisfy condition $2^{a-1} \leq F_i < 2^a - 1$. That is, i uses the s_i -th slot in every L_i time frame (i ' slots are $l \cdot L_i + s_i$, for all $l = 1, 2, 3, \dots$). The RHS of the inequality constrains the set of feasible values for a to avoid conflict while the LHS of the inequality forces us to pick the minimum of these feasible values. We now prove why the RHS of the inequality avoids conflict among contending neighbors.

Theorem 3.1: If every node i uses only slots $l \cdot 2^a + s_i$, for all $l = 1, 2, 3, \dots$, then no node j in the two-hop neighborhood of i uses any slot that i uses.

Proof: We can prove the theorem by contradiction. Let j be a node that is in the two-hop neighborhood of i , but it happens to use one of the slots that is used by i . It does that at the m -th time frame of j . By the TF rule, j is assigned to s_j by DRAND and j 's time frame is 2^b for some b . Then without loss of generality, we assume that $2^a \leq 2^b$ and $s_i < s_j$. Note that by DRAND, $s_i \neq s_j$, and i and j are assigned to only one slot within F_i and F_j respectively. Then because of the way that a and b are chosen by the TF rule, it is true that i uses only one slot within a 2^a time frame, so does j within a 2^b time frame. Then for all $l = 1, 2, 3, \dots$, $l \cdot 2^a = l \cdot 2^{b-a} 2^a$. This means that whenever j starts its own frame, i starts its time frame, and whenever j ends its own frame, i ends its time frame (i.e., no time frame of j starts or ends in the middle

of i 's). Because i and j are in conflict, s_j must be less than or equal to F_i . Because the beginning of j 's time frame is always aligned with that of i 's, and j 's slots occur always at slots $l \cdot 2^b + s_j$. Then by way of the contradiction, in order for i and j to use the same slot at the m -th time frame of j , s_i and s_j must be the same since j uses only one slot in the frame and that is s_j , which is a contradiction. ■

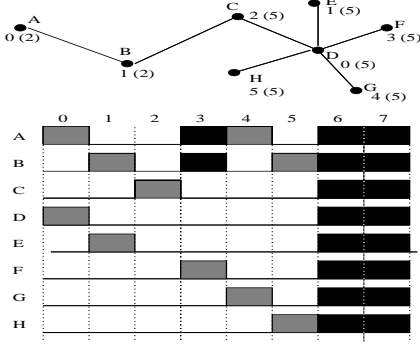


Fig. 1. An example of the TF rule. The top figure shows a network topology and the numbers indicate the slot numbers assigned by DRAND and the numbers in parenthesis are F_i . The bottom figure shows the slot schedule of all nodes (shaded slots are ones where each node transmits, and dark slots are the “empty” slots that are not used by any one-hop or two-hop neighbors).

The TF rule allows nodes to pick their own time frame sizes based on their local two-hop information. This rule makes DRAND adaptive to dynamic time frame changes (caused by local topology changes) without incurring any global changes. Figure 1 shows an example of a TDMA schedule obtained by the TF rule. If the global time frame is used, then 6 will be the time frame size. Then nodes A and B can use their slots only once every 6 slots although their frame sizes are 2 each. But if the TF rule is used, we can allow them to use frame size 4. This increases the concurrency in the channel usage and reduces the message delays for node A and B . However, we find that slots 6 and 7 are not assigned to any node in the neighborhood. This is a trade-off; when the network is uniformly dense, the global time frame would create a smaller number of empty slots. But if the network contains many sparse areas with only a few dense areas, then the local framing would be more preferable. In Z-MAC, since empty slots are available for CSMA (more details on Z-MAC’s transmission control are given next section), they are not necessarily wasted.

Synchronizing on slot 0. The local framing rule implicitly assumes that every node starts its time slot 0 at the same time. This can be achieved without any communication, if clocks are synchronized, by fixing a predetermined absolute time to synchronize slot 0. For instance, we can set the beginning of the real time (i.e., when the synchronized clock value is zero) to be the beginning of slot 0. New nodes can easily synchronize their slots if they synchronize their clocks to the global clock. To allow this synchronization, Z-MAC performs global clock synchronization such as TPSN [15], only once at the beginning. After the initial synchronization, each node runs a low-cost local synchronization protocol discussed in Section III-F.

C. Transmission Control of Z-MAC

At the end of the DRAND phase, every node forwards its frame size and slot number to its two-hop neighborhood. Thus, a node knows about the slot and frame information of its one-hop and two-hop neighbors at the beginning of the Z-MAC phase. At this point, every node synchronizes to slot 0 and then they are finally ready to run the transmission control of Z-MAC.

In Z-MAC, a node can be in one of two modes: *low contention level (LCL)* or *high contention level (HCL)*. A node is in HCL only when it receives an *explicit contention notification (ECN)* message from a two-hop neighbor within the last t_{ECN} period. Otherwise, the node is in LCL. A node sends an ECN when it experiences high contention. The details on ECN are in next section.

In LCL, any node can compete to transmit in any slot, but in HCL, only the owners of the current slot and their one-hop neighbors are allowed to compete for the channel access. In both modes, the owners have higher priority over non-owners. If a slot does not contain an owner or its owner does not have data to send, non-owners can steal the slot. This feature achieves high channel utilization even under low contention as a node can transmit as soon as the channel is available. Z-MAC implements LCL and HCL using the backoff, CCA and LPL interfaces of B-MAC.

The transmission rule. As a node i acquires data to transmit, it checks whether it is the owner of the current slot. If it is the owner of the slot, it takes a random backoff within a fixed time period T_o . When the backoff timer expires, it runs CCA and if the channel is clear, it transmits the data. If the channel is not clear, then it waits until the channel is not busy and repeats the above process. If node i is a non-owner of the current slot and it is in LCL, or if it is in HCL and the current slot is not owned by its two-hop neighbors, then it waits for T_o and then performs a random backoff within a contention window $[T_o, T_{no}]$. When the backoff timer expires, it runs CCA and if the channel is clear, then it starts transmission. If the channel is not clear, then it waits until the channel is clear, and repeats the above process. If node i is a non-owner of the current slot and is in HCL (this means that a two-hop neighbor of i has sent an ECN in the last t_{ECN}), postpones its transmission (it may sleep) until it finds a time slot that either (1) is not owned by a two-hop neighbor or (2) is its owner. After waking up, it repeats the above process.

According to the above transmission rules, in the LCL mode, a node can compete in any slot, albeit with different priorities. In HCL mode, it can compete in the current slot only if it is the owner of the slot or a one-hop neighbor to the owner of that slot. Note that it is possible that a transmission started in the previous slot crosses over to an HCL slot causing collision with the owner of the slot. One way to prevent this is to restrict a transmission not to cross over an HCL slot. We opt not to support this because this restriction makes the system design more complicated especially in a network where tight time synchronization is difficult to achieve. Besides packets do not come at a regular interval and may not be of the same

size. Another reason for allowing slot “crossing” is due to the following tradeoff in channel utilization. If such a crossing is not allowed, then even when there is some remaining time in a slot, that time may be unused if a packet transmission by the owner cannot be finished within that time slot. On the other hand, if we allow the crossing, then it is possible that a packet transmission by the next owner (which could act as a hidden terminal to the current owner) could cause a collision, thus wasting the time for transmitting the packet. Now the tradeoff is whether we pro-actively prevent such a collision by not transmitting during that remaining time in the slot and thus wasting that time or we make the transmission during that time but possibly risking channel wastage due to a packet collision at the next slot. Both cases waste some amount of slot time but in the first case, we always waste that time, but in the second case we waste the time only when a collision happens. Our initial test result is consistent with our intuition in that the second case results in more channel utilization.

Specific values of T_o and T_{no} have performance impact. The choice of T_o determines the robustness of Z-MAC in the face of time synchronization errors or slot assignment failures which cause some slots to have more than one owner. If the synchronization error is no more than one TDMA slot size, then there can be at most two to three conflicting owners at any time. We can analytically obtain the optimal size of T_o to handle contention among two to three owners. Based on this, we set T_o to 8 contention window slots (also a power of 2 for efficient implementation). We set T_{no} to 32 slots (which is also the initial contention window size in B-MAC).

Slot sizes also have a performance implication. If the slot size is too small, clock synchronization errors will have higher performance impact because it will allow more nodes to overlap over slot boundaries. For slot size x ms, as long as the synchronization error is less than $x/2$ ms, a slot will have no more than two conflicting owners. Another way to look at the problem is that since the effect of clock synchronization errors will likely occur around the boundaries of slots, as the slot size increases, the performance impact of such errors asymptotically reduces (because within a unit time, the number of boundaries gets smaller). On the other hand, increasing the slot size tends to increase the transmission delay because it increases the frame size. If a node misses its time slot, it takes one frame size before it becomes an owner again. Therefore, the choice of the slot size should be a function of the accuracy of clock synchronization and also the desired network delay in the network. In our system, the slot size is a system parameter tunable depending on the application.

The transmission rule of Z-MAC is different from that of PTDMA. Unlike PTDMA, the owner and non-owner access probabilities of Z-MAC (“a” and “b” in Eq. 1 in [14]) are independently adjusted by T_o and T_{no} since non-owners cannot compete during T_o . This enhances the ability to increase the robustness of the protocol without affecting the general behavior of the protocol. For instance, increasing T_o does not change the priority between owners and non-owners, thus preserving the performance swing between TDMA and CSMA depending on contention. In PTDMA, this is not possible due to dependency between “a” and “b”.

D. Explicit Contention Notification

ECN messages notify two-hop neighbors not to act as hidden terminals to the owner of each slot when contention is high. Each node makes a local decision to send an ECN message based on its local estimate of the contention level. There are two ways to estimate two-hop contention. One is to receive acknowledgment from the one-hop receiver and measure the packet loss rate. Since two-hop contention causes collision, it is highly related to the loss rate. However this technique requires the receiver to send feedback and incurs extra overhead. Unless the acknowledgment feature is enabled by the application, this overhead can unduly reduce the channel utilization. The other technique is to measure the noise level of the channel. When high contention occurs, it tends to increase the noise level. This technique does not require any extra overhead as the noise level can be measured passively at the time of data transmission. In order to measure the noise level passively without actively sampling the channel, we measure the average number of *noise backoffs* that a sender takes before transmitting a packet. A noise backoff is the backoff taken by a transmitter when it senses the channel using CCA before packet transmission (it transmits only when the channel is clear). When the noise level is higher than the CCA threshold, the node takes backoff. In order to see the correlation between the noise backoff and two-hop contention, we took a Mica2 experiment where two clusters of nodes transmit to a common receiver called *sink*. The nodes in different clusters are in a two-hop distance to each other and the nodes in the same cluster are one-hop away from each other. In one cluster, we fix one sender, called *measurement node*, and in the other cluster, we vary the number of senders. We measure the correlation between two-hop contention at the sink and the noise level at the measurement node as we vary the number of senders in one cluster and their transmission rate. The two-hop contention is measured by the number of times per second that the sink leaves the idle state into the receiving state but fails to receive the data because of corrupted data or high noise in sampled data (including loss of sync, CRC fail, and preamble fail).

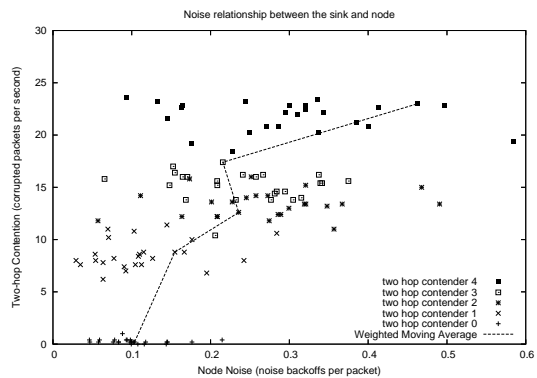


Fig. 2. The correlation between noise level and two-hop contention.

Under low transmission rates, even if we increase the number of senders, the average noise level and two-hop contention are very low, below 0.1 per packet and 5 per second respectively. However, we increase the transmission rate to the full rate (all senders always have data to send), the noise

level increases beyond 0.2. Figure 2 shows the correlation between the average noise level and two-hop contention under the full data rate. The *simple correlation coefficient* [16], the ratio of co-variance of the two metrics over the product of the variances of individual metrics, is 0.68 (1 and -1 indicate the maximum positive and negative correlations and 0 indicates no correlation), indicating high correlation. The exponentially moving average value (with weight 0.5) of the noise level when the two-hop contention is higher than 20 per second increases beyond 0.3 backoffs/packet. We repeated the experiment many times and confirmed that average 0.3 noise backoffs per packet consistently indicates high two-hop contention. However, this is only a conservative metric because even one-hop contention can cause high noise backoffs as well, but it is clear that low noise indicates low contention.

As a transmitting node detects high contention, the node sends a unicast message, *one-hop ECN*, to a destination to which the node is experiencing contention. If multiple destinations experience contention, it sends one broadcast with information about the multiple destinations. Typically, in sensor networks, since each node has one parent to transmit data to, a node has one destination. When a node j receives a one-hop ECN message triggered by its one-hop neighbor i , it first checks whether j is the destination of the ECN message. If so, it then broadcasts the ECN to its one-hop neighbors (these ECN messages are called *two-hop ECN*). If j is not the destination, it simply discards the one-hop ECN. When a node receives a two-hop ECN, then it sets its HCL flag.

The HCL flag is only a *soft state*, meaning that unless another two-hop ECN message is received within the last t_{ECN} period, the flag is reset. Thus, if node i continually experiences contention, it needs to transmit the ECN message periodically. This refresh period t_{ECN} is set by the system.

Typically, when a node detects contention, it is likely that its neighboring senders will do so at the same time. Therefore, we will have many duplicate ECN messages forwarded to routing nodes. To prevent ECN implosion, we use overhearing to suppress ECN. When a node i detects high contention, it takes random backoffs before the transmission of a one-hop ECN message. In the mean time, if it receives a one-hop ECN intended for another node that has the same destination as i 's ECN, then node i suppresses its ECN and cancels the transmission of the ECN. After t_{ECN} , if it still experiences high contention, it schedules another ECN by taking a random backoff and repeats the above process. The same suppression rule applies to routing nodes. If a routing node receives a one-hop ECN and it has forwarded an ECN within t_{ECN} period, it does not forward a two-hop ECN.

ECN is similar to RTS/CTS in CSMA/CA. But the difference is that HCL uses topology information (i.e., slot information) to avoid two hop collision. The cost of ECN is also far less than RTS/CTS since it is triggered only when contention is high. Using ECN suppression, only a small number of ECN messages need to be forwarded. Since the HCL state may last for a much longer term than a single packet transmission, its cost is amortized over many packet transmissions. ECN can also be viewed similar to the *suppression* message in CODA[17]. However the difference is that ECN suppresses

two-hop neighbors only for the time slot of the ECN originator whereas a suppression message suppresses all receivers except the one designated in the message.

E. Receiving Schedule of Z-MAC

DRAND defines only the transmission schedule of nodes. In Z-MAC, a node can transmit in any slot. On the other hand, Z-MAC does not define a receiving schedule for nodes. Instead, it relies on the LPL mode of B-MAC for receiving packets. Therefore, the energy consumption of Z-MAC for idle listening especially under low duty cycles is comparable to that of B-MAC.

The check period is also a factor in determining the slot size because a slot must be big enough to transmit one packet. Thus, the slot size must be larger than the sum of the check period, T_o , T_{no} , the CCA period and one packet propagation time. There is a trade-off between the slot size and the network delay, especially under high contention. Under low contention the slot size does not affect the delay since a node can transmit at any time. But under high contention, a node is in HCL and transmits only during a few designated slots. Therefore, a large slot size can incur a large delay. We leave the choice of the slot size to the application designers who has to evaluate the trade-offs and find the slot size that fits their needs.

F. Local Time Synchronization

Z-MAC requires clock synchronization under high contention to implement HCL. However, note that synchronization is required only among neighboring senders and also when they are under high contention. This offers us an excellent opportunity to optimize the overhead of clock synchronization because synchronization is required only locally among neighboring senders, and the frequency of synchronization can be adjusted according to the transmission rates of senders so that senders with higher data rates transmit more frequent synchronization messages. In this scheme, receivers passively synchronize their clocks to the senders' clocks and do not have to send any synchronization messages.

To implement the local clock synchronization among senders, Z-MAC adopts a technique from RTP/RTCP (real-time transport protocol) [18]. In this protocol, the control message transmission rate is limited to a small fraction of session bandwidth and each session member adjusts its sending rate of control messages according to the allocated session bandwidth. In Z-MAC, each data sender limits its bandwidth consumed by synchronization messages to a predetermined fraction of its data sending rate, B_{synch} (e.g., one synchronization packet per every 100 data packets). In fact, each sender can independently determine this fraction by some function of its energy and bandwidth budget. We currently set B_{synch} to 1% of the sending rate.

In our local synchronization protocol, each *data sender* transmits a synchronization message containing its current clock value periodically. When a node receives a synchronization message, it updates its clock value by taking a weighted moving average of its current value and the newly received value. Because only senders transmit synchronization

messages, it is possible that some nodes located in a low traffic area might have clock values drift far away from the other synchronized nodes. When those nodes start transmission, their clock values are unsynchronized (note that the maximum clock drift rate of Mica2 is around $40\mu s$ [15] per second). Thus, their clock values must not be trusted. To avoid honoring clock values from unsynchronized senders, we adjust the averaging weight by applying a *trust factor* (β_t) that reflects the frequency of synchronizations of the message senders. β_t is computed by the frequency of transmitted and received synchronization messages as below.

Let r_{drift} be the maximum clock drift rate of each sensor and ϵ_{clock} be the maximum acceptable clock error. Then $I_{\text{synch}} = \epsilon_{\text{clock}}/r_{\text{drift}}$ determines the minimum synchronization interval required to achieve the maximum clock error or less. Let S be the average rate at which a node receives or sends synchronization messages, and α_{synch} be the maximum weight that applies to the new clock value received. Then the β_t of the node can be computed by $\beta_t = \min\{\alpha, S \times I_{\text{synch}} \times \alpha_{\text{synch}}\}$. The weighted moving average value C_{avg} of a clock can be computed by taking a weighted moving average of a newly received clock value C_{new} and C_{avg} : $C_{\text{avg}} = (1 - \beta_t)C_{\text{avg}} + \beta_t \cdot C_{\text{new}}$.

In Mica2, to maintain 1 ms clock accuracy with $40\mu s$ per second maximum drift rate and one synchronization packet per every hundred packets (packet size 49 bytes), a node needs to maintain its sending and/or receiving data rate to 1.5 Kbps or higher. At that rate, the trust factor of the node becomes α_{synch} , consuming only 1% of the sending rate, 150 bps (or 1/3 packets per second with 49 byte packets), for synchronization. If the data rate (sum of sending and receiving rates) goes below this, then the trust factor of that node gets discounted. But this does not pose any threat to throughput because it is likely that the node does not experience much contention below that data rate and CSMA works effectively under low contention.

In the above scheme, the nodes that send and receive synchronization messages more often tend to have a higher trust factor and their values will be reflected more heavily in updating clock values. Typically, these nodes on routing paths tend to have higher trust factors because they tend to send more packets than the others. Similarly, source nodes that infrequently send data have lower trust factors. When a source starts sending data again after a long hibernation, its clock could be drifted far apart from other more synchronized clocks. But as it increases its rate and its data being routed to the sink, its clock value will come closer to the clock values of other routing nodes. In our experiment, we find that even if an island of 30 nodes is not synchronized, it resynchronizes with the rest of the network within 10 synchronization messages.

IV. ANALYSIS OF CHANNEL UTILIZATION

In this section, we formulate the closed form expression of channel utilization for various existing MAC schemes for wireless sensor networks, namely B-MAC, Sift, PTDMA and Z-MAC in a one-hop environment where all nodes can sense the transmission of the other nodes. We do not analyze S-MAC and T-MAC as [3] shows that these protocols perform much worse than B-MAC. These expressions are validated in the next section by the simulation and experiment.

A. Model and Definitions

N nodes are in the system and they are all in a radio range of each other, i.e., a transmission by a node can be sensed by all other nodes. B out of N nodes are sources and the remaining nodes do not send any packets. Sources always have packets to send, i.e., applications are continually transmitting. We assume that all packets are of the same size. As we increase B , we vary the level of contention in the system. N poses as the maximum potential number of contenders in a neighborhood. We assume that T_s is the time taken to sense the radio. We call T_s a *contention slot*. Assuming negligible propagation delay, collision always occur at the beginning of a packet transmission. Note that under no delay, collision cannot occur in other times because it can be sensed by all other transmitting nodes. Thus, the time wasted because of collision is the same as the transmission time of a packet which is denoted by T_p . Out of T_p , let T_d the time spent to transmit the payload of a packet excluding the time taken to send its header. In our analysis, we measure the effective channel utilization expended to transmit data payload, considering the header transmission to be overhead. We assume a *slotted-time* model where real time is divided in the unit of a slot time and all transmissions occur at the boundary of a slot.

B. B-MAC

We approximate the performance of B-MAC by slotted CSMA with one backoff window size. In this model, a node takes a random backoff time before a transmission. When the backoff timer expires, it senses the channel. If the channel is not busy, it starts transmission. If not, it waits until the transmission is over and then take another backoff and repeat the above process. The backoff time value is randomly set in the unit of slots within the backoff contention window. This scheme only approximates B-MAC because B-MAC is implemented as a non-slotted CSMA protocol and also uses two different backoff windows: the initial backoff before the transmission of a packet and the backoff after sensing the channel are taken from two different window sizes. Despite these differences, we shall see that our analysis is fairly close to its simulation result.

Nodes pick a random backoff uniformly over $[1, CW]$. Hence, the average window size observed by a node would be $W_{\text{avg}} = (CW + 1)/2$. Now, consider a contention time slot i . Since all nodes pause their backoff timer as soon as they detect that the channel is busy, from the viewpoint of the nodes, the duration of an entire packet transmission (whether successful or collided) is counted as a single contention slot. Seen this way, contention slot i can be in one of three states: Collision(s) occurred during i , successful data transmission occurred during i , or i was idle. Let the probability that i is in each of these states be P_c , P_d and P_i respectively. Hence,

$$P_c + P_i + P_d = 1 \quad (1)$$

Given B contending nodes, we can calculate utilization ($S(B, CW)$) achieved as follows:

$$S(B, CW) = \frac{T_d P_d}{T_p P_c + T_p P_d + T_s P_i} \quad (2)$$

The probability that a contention slot is idle is the probability that none of the nodes selected that slot. Given B contending nodes, each with an average backoff window size of W_{avg} , the probability of a node selecting a slot is $1/W_{avg}$. Hence,

$$P_i = (1 - 1/W_{avg})^B \quad (3)$$

Along the same lines, the probability of a contention slot being used for successful data transmission is the probability of a node selecting a contention slot, and all others choosing different slots. Hence,

$$P_d = B \times (1/W_{avg}) \times (1 - 1/W_{avg})^{B-1} \quad (4)$$

This system of equations can be solved for P_c , P_i , and P_d to get the utilization S .

C. Sift

Sift [11] is a slotted fixed window CSMA protocol. Backoff values are randomly chosen based on the following probability distribution $P(r)$ where r is the contention slot number in range $1, \dots, CW$, and $P(r)$ is the probability that contention slot r is chosen as a backoff value:

$$P(r) = \frac{(1 - \alpha)\alpha^{CW}}{1 - \alpha^{CW}} \alpha^{-r} \quad (5)$$

Once the slot is chosen, the node transmits at that slot. If a node finds the channel busy, it waits till the channel is idle, and tries again as before by choosing a new slot. This behavior is different from B-MAC where the backoff timers are paused when the channel is found to be busy, and later resumed again from the last values before the pause. This makes our analysis for Sift a little different from that for B-MAC.

Let the probability of a successful transmission in a slot r be $P_s(r)$:

$$P_s(r) = B \times P(r) \times (1 - \sum_{i=1}^r P(i))^{B-1} \quad (6)$$

which is the probability that any one node chooses slot r , all other $B - 1$ nodes do not select any slot from 1 to r , and since any of the B nodes could be the winner, we multiply this probability by B to get the desired probability.

Let the probability of collision in slot r be $P_c(r)$:

$$P_c(r) = \sum_{x=2}^B (C_x^B \times P(r)^x \times (1 - \sum_{i=1}^r P(i))^{B-x}) \quad (7)$$

Here, we are counting the probabilities of x collisions occurring in slot r . So literally the probability is that any x nodes select the same slot r , and the remaining $B - x$ do not select slots from 1 to r . Since any x out of the B nodes can be the nodes involved in the collision, we need to multiply this probability by C_x^B . Finally, we need to sum up probabilities due to 2, 3, 4... B collisions, all occurring in slot r .

Let the total probability of success and collision for one trial be P_s and P_c respectively. Hence:

$$P_c = \sum_{r=1}^{CW} P_c(r), \quad P_s = \sum_{r=1}^{CW} P_s(r), \quad \text{and } P_s + P_c = 1$$

Let the expected lengths of successful and collided transmissions be E_s and E_c respectively. Hence:

$$E_s = \sum_{r=1}^{CW} P_s(r) \times ((r - 1) \times T_s + T_p) \quad (8)$$

$$E_c = \sum_{r=1}^{CW} P_c(r) \times ((r - 1) \times T_s + T_p) \quad (9)$$

We can hence calculate the utilization $S(B, CW)$ as:

$$S(B, CW) = \frac{P_s T_d}{E_s + E_c} \quad (10)$$

D. PTDMA

Consider a TDMA time slot i whose owner is O_i . Note that TDMA time slots are different from contention slots and in PTDMA, there is no contention slots and all transmissions are done once at the beginning of each TDMA slot. It does not perform any carrier sensing either. If O_i is a source, then it transmits in slot i with probability a while the remaining $B - 1$ sources transmit in slot i with probability b . If more than two nodes transmit during the same slot, the TDMA slot is wasted with collision.

Let P_{s_a} and $P_{s_{in}}$ be the probabilities that successful packet transmission occurs in a slot of an active source and a non-source respectively:

$$P_{s_a} = a(1 - b)^{B-1} + b(1 - a)(1 - b)^{B-2}(B - 1) \quad (11)$$

$$P_{s_{in}} = b(1 - b)^{B-1}B \quad (12)$$

P_{s_a} is calculated as the probability that the owner of the slot wins, all other $B - 1$ sources lose, *OR*, the owner of the slot loses, one of the non-owners of the slot wins, and all other $B - 2$ sources lose (we multiply this by $B - 1$, since there can be $B - 1$ such winners). $P_{s_{in}}$ is obtained as the probability that one source wins with probability b , and all other $B - 1$ sources lose – we again multiply by B since there can be B such winners. Given these probabilities, the utilization $S(N, a, b)$ is:

$$S(N, B, a, b) = \frac{P_{s_a}B + P_{s_{in}}(N - B)}{N} \times T_d/T_p \quad (13)$$

The factor T_d/T_p accounts for the fraction of bandwidth lost due to the header.

E. Z-MAC

Owners of a slot pick a random backoff uniformly over $[1, T_o]$, while non-owners do so within $[1, T_{no}]$. Hence the average window size of owners and non-owners would be $W_o = (T_o + 1)/2$ and $W_{no} = T_o + (T_{no} + 1)/2$ respectively. Assuming strict time synchronization between nodes, the owner grabs the channel every time because of its smaller backoff window. Hence, in B slots out of N slots, the corresponding owners will always succeed. Z-MAC behaves like a slotted, memory-less CSMA scheme with just one contender. We can apply the analysis in Section IV-C to this case, with one small

modification – the probability of choosing a slot r is governed by a uniform distribution, hence:

$$P_s(r) = 1/W_o \quad (14)$$

Let utilization obtained by owners be denoted by $S_o(1, W_o)$. In the remaining $N - B$ slots, all B sources contend with a fixed window of size W_{no} . Z-MAC behaves like a slotted, fixed window, memory-less CSMA scheme with B contenders. Applying the analysis in Section IV-C again, with:

$$P_s(r) = 1/W_{no} \quad (15)$$

We denote the utilization obtained in the $N - B$ slots as $S_{no}(B, W_{no})$. Given S_o and S_{no} , the average utilization, $S(N, B, T_o, T_{no})$ is a weighted average on the B and $N - B$ slots respectively:

$$S(N, B, T_o, T_{no}) = \frac{S_o(1, W_o)B + S_{no}(B, W_{no})(1 - B)}{N} \quad (16)$$

F. Discussion

Our goal for this analysis is to ascertain the best performance achievable by a given protocol under idealized channel conditions when factors such as channel losses and noise are factored out. We shall show in Section V-B, Figures 6 and 7 that the analysis closely follows the simulation results and that Z-MAC performs well compared to other protocols. With this validation of our design, we proceed to implement Z-MAC in the real network environment and present the corresponding results in Section V.

V. EXPERIMENTAL EVALUATION

In this section, we validate the analytical results in the previous section experimentally and also test the performance of the MAC protocols in more diverse but realistic environments.

A. Experimental Method

To evaluate the performance of Z-MAC, we implemented Z-MAC in both ns-2 and Mica2/TinyOS. We use ns-2 simulation to compare the performance with existing protocols whose TinyOS implementation does not exist at the time of preparing this work. We compare the performance of Z-MAC with that of PTDMA (ns-2), Sift (ns-2) and B-MAC (ns-2 and TinyOS). We do not run S-MAC and T-MAC as [3] shows that these protocols perform much worse than B-MAC. Although our performance evaluation does not cover all the available sensor MAC protocols, we believe that the evaluated protocols constitute a good representation of existing protocols.

Unless specified otherwise, we use the default settings of B-MAC as described in [3]. Since Z-MAC is implemented on top of B-MAC, we use the same packet format as B-MAC (shown in Table 4 in [3]). The default initial and congestion backoff window sizes of B-MAC are 32 and 16 slots respectively (each slot is 400 μ s). Except the throughput tests where we vary the backoff window sizes to see the impact of window sizes on

TinyOS and ns-2 experiment parameter	Default
Owner contention window size (T_o)	8 slots
Non-owner contention window size (T_{no})	32 slots
Contention window per-slot duration	400 μ s
ECN refresh period (t_{ECN})	10 seconds
Averaging weight for time synchronization (α_{synch})	0.25
Maximum clock drift rate (r_{drift})	40 μ s
Maximum clock error (ϵ_{clock})	1 ms
Synchronization bandwidth (B_{synch})	1 %
Z-MAC TDMA slot size	50 ms
Communication range (ns-2)	200 ft
Interference range (ns-2)	300 ft
Communication bandwidth (TinyOS, ns-2)	19.2 Kbps

TABLE I
THE DEFAULT SETTINGS OF Z-MAC PARAMETERS

channel utilization, we keep the default window sizes. The default values of Z-MAC parameters are shown in Table I.

We use three benchmark setups in our experiment: *one-hop*, *two-hop* and *multi-hop* benchmarks.

One-hop benchmark. This benchmark is reproduced from [3] - n nodes placed equidistant from a receiver in a circle transmit as quickly as possible with full transmission power. Before each run, we ensured that all nodes are in a one-hop distance to each other so that there are no hidden terminals. This benchmark is used to measure the achievable throughput of different MAC protocols for different levels of contention within a one-hop neighborhood. All nodes are placed at least 2 feet apart and the distance to the receiver was approximately 2 meters. The setup is placed in an open conference room without any obstruction. ns-2 one-hop simulation follows the same setup.

Two-hop benchmark. We create this benchmark to test the performance of different protocols when hidden terminals are present. We organize nodes into two clusters where 7 and 8 sending nodes are located in each cluster respectively. The two clusters are placed approximately 5 meters apart in a house with drywall. A receiver node (or routing node) is placed in the middle of the two clusters. Nodes within the same cluster are placed about 2 feet apart. In this environment, we cannot get a sharp boundary of interference but we ensure that all senders find the receiver as a one-hop neighbor and all nodes are reachable by two hop communications. We also reduce the transmission power of senders to 1 dBm (1.3 mW) to control the number of hidden terminals. Since the number of hidden terminals varies with the transmission power, we get more hidden terminals with a low transmission power. On the other hand, ns-2 simulation of the two-hop benchmark can define a clear separation of the two clusters so that they become always two-hop to each other.

Multi-hop benchmark. We consider two multi-hop topologies – a 10-hop chain topology and a full-fledged wireless sensor network testbed comprising of 42 Mica2 nodes.

The 10-hop chain experiment is reproduced from [3] to measure the latency of different protocols where 11 nodes are lined up side by side to create a line topology. The source and sink are placed at the two different ends of the topology. The source sends 20 messages with a payload of 100 bytes without any fragmentation. The intermediate nodes forward the messages to the sink.

For a realistic multi-hop scenario, we construct a network of 42 Mica2 nodes, each placed in faculty offices and classrooms of our computer science building. Figure 3 shows the testbed and wireless communication links among nodes. In this testbed, the maximum two-hop neighborhood size of all nodes is 27 and the maximum local frame size is 32 (many nodes have smaller local frame sizes). To remove any effect of routing differences, we use fixed routing paths for all tests. The paths are taken from one run of Mint [19], the default routing protocol of TinyOS. Figure 4 shows the routing paths we used for 30 nodes in the testbed with node node 36 as the sink. Thicker lines indicate links with more traffic.

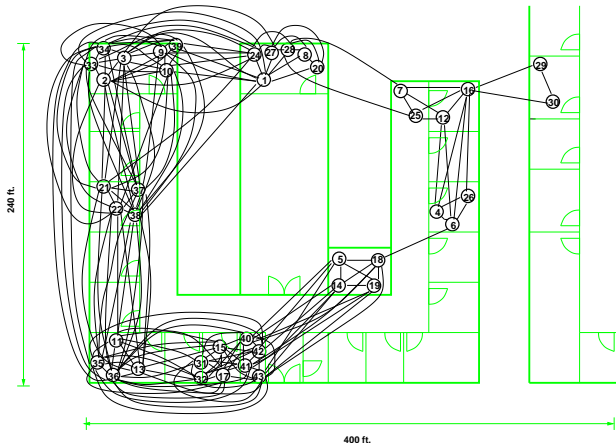


Fig. 3. NCSU testbed with 42 Mica2 nodes.

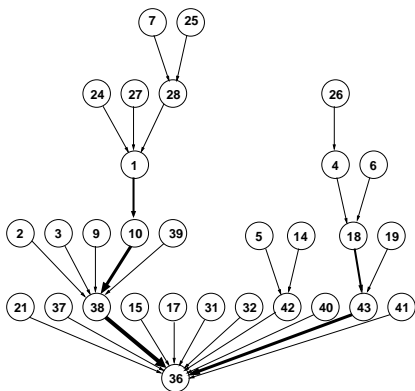


Fig. 4. Routing paths used in the multi-hop Mica2 benchmark.

B. Throughput

In this experiment, we measure and compare the effective channel utilization of each MAC protocol. We measure only data throughput as done in [3] where the data portion of each packet consists of 36 bytes (29 bytes for the data payload, 5 bytes for the header, and 2 bytes for CRC).

One-hop Benchmark. In this test, all senders are transmitting at their full transmission power and the receiver has its radio on always (i.e., no duty cycle). The effective maximum data throughput on Mica2 is 15.6 Kbps (excluding preamble and sync bytes). Figure 5 shows the data throughput of B-MAC and Z-MAC from Mica2 one-hop benchmark. Unfortunately,

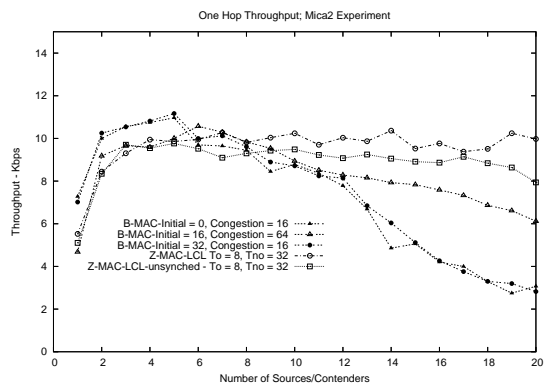


Fig. 5. The data throughput comparison in the one-hop Mica2 benchmark.

we are not able to reproduce the same performance of B-MAC as shown in [3]. Our result is significantly less than what they report ([3] reports approximately the maximum throughput of 13 Kbps with one sender). We conjecture this discrepancy could be due to a higher noise floor level in our experiment environment. B-MAC with congestion window size 64 performs much better than that with congestion backoff window size 16. This happens because the larger congestion window size reduces the contention among senders.

For the Z-MAC tests, we fix the frame size to 20 for all experiments and vary the number of senders. HCL is disabled because the performance of HCL and LCL is the same when all nodes are in a one-hop distance to each other. Before running Z-MAC, we run DRAND and TPSN to get slot assignments and to synchronize the clocks of the senders. The data throughput is obtained after these protocols finish. The data throughput of Z-MAC with one sender is about 40% less than that of B-MAC with window sizes (0,16) and (32,16). This happens because Z-MAC uses a larger congestion backoff window size. With one source, it sends as non-owners at most times except for its own slot. Therefore, it incurs the cost of waiting for T_o for the non-owner slots. The throughput of Z-MAC is almost independent of the number of senders. When the number of senders is small, most senders are sending as non-owners. Thus, they can utilize the unused slots that belong to the other nodes. As the number of senders increases, so does the number of senders transmitting during their own slots. Thus, when contention is high, it can maintain good throughput since it works more like TDMA. The throughput with 20 senders is much higher than that of B-MAC (in fact, higher than that shown in [3]).

We also run Z-MAC with no clock synchronization. At the beginning of the run, we randomize the clock values of all nodes. We turn off the local clock synchronization protocol as well. This allows some slots to be overlapped with each other so that several nodes consider themselves as owners at the same time. Thus, this scenario essentially emulates slot assignment failures as well. We observe that although the Z-MAC performance drops in the presence of time synchronization errors, it does no worse than CSMA, and under high contention gives comparable performance to Z-MAC with synchronized clocks. This is because T_o is

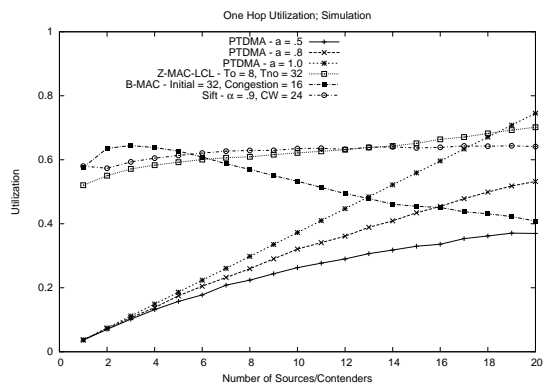


Fig. 6. The data throughput comparison in the one-hop ns-2 benchmark.

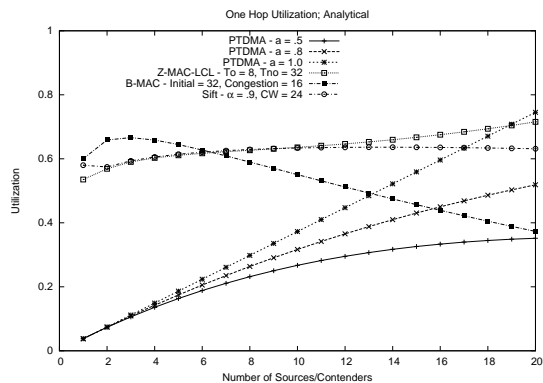


Fig. 7. The analytical data throughput comparison in the one-hop benchmark

sufficiently large to handle multiple owners within a slot. This shows that when the information about interference relation and synchrony are inaccurate, the performance of Z-MAC gracefully degrades to that of CSMA. When this information is accurate, Z-MAC performs really well under high contention.

Figure 6 shows ns-2 simulation results for one-hop involving PTDMA, Sift, B-MAC and Z-MAC - which agree closely with the analytical utilization shown in Figure 7. For PTDMA and Z-MAC, we set the number of stations M to be 21. For PTDMA, we report utilization for access probability (“a”) values of 0.5, 0.8 and 1. The slot size for PTDMA is set to 20 ms which is enough to send one packet. For Sift, the probability distribution parameter α is set to 0.9 and the contention window size is set to 24 slots. PTDMA, for any value of “a”, shows very low utilization under a small number of sources. As the number of senders increases to M , it shows its maximum channel utilization. But under lower values of “a”, its performance becomes close to that of CSMA (which is B-MAC in this figure). Only when the number of senders is equal to M , it becomes closer to TDMA as “a” increases (these data points can be verified from the results in [14] as well). The B-MAC simulation result closely follows that in [3]. The good performance of Sift is because all nodes are one-hop and data are always available for transmission so the senders are synchronized with each other to the boundary of packet transmissions.

Two-hop Benchmark. In the two-hop benchmark, we measure

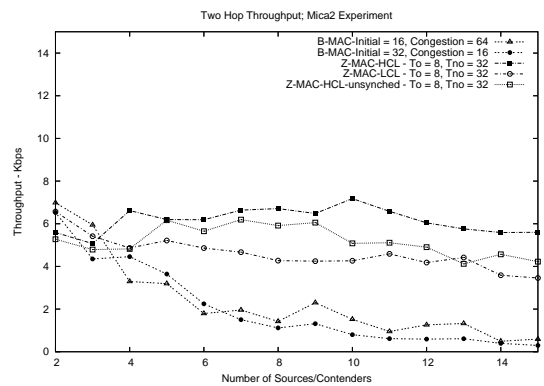


Fig. 8. The data throughput comparison in the two-hop Mica2 benchmark.

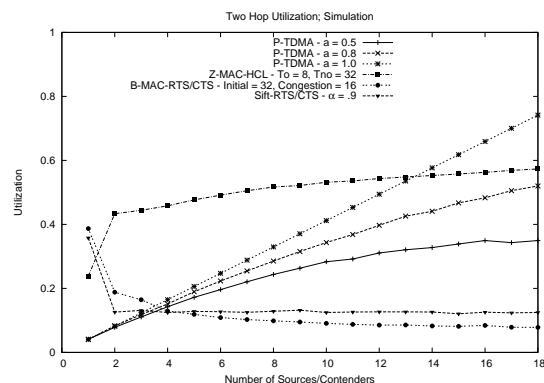


Fig. 9. The data throughput comparison in the two-hop ns-2 benchmark.

the data throughput when hidden terminals are present. We vary the number of senders while fixing the number of neighbors. As in the one-hop benchmark, all senders always have data to send. Each additional sender is chosen from the alternating clusters. For Z-MAC tests, we set the frame size to 16. In this test, we run Z-MAC with HCL disabled (marked as Z-MAC-LCL) and with HCL enabled (marked as Z-MAC-HCL). Both cases run along with the local clock synchronization protocol in which each sender sends one synchronization packet in every 100 packets transmitted. The data throughput reported by Z-MAC includes the overhead of the clock synchronization and ECN.

Figure 8 shows the results of the two-hop Mica2 benchmark. Since the transmission power is low (1.3 mW), the maximum achievable throughput also gets lower. As the number of hidden terminals increases along with more senders, the throughput of LCL drops more than that of HCL. On the other hand, HCL performs relatively well maintaining around 6 Kbps even under high contention. According to [3], the protocols with RTS/CTS (S-MAC and B-MAC with RTS/CTS) achieve around 2 Kbps when more than 15 nodes (even when no hidden terminals are present). This confirms that the overhead of ECN is much lower than that of RTS/CTS. B-MAC shows high sensitivity to hidden terminals as its throughput drops to 1 Kbps under high contention. Z-MAC-HCL when run under unsynchronized clocks, shows a drop in performance but is still better than B-MAC.

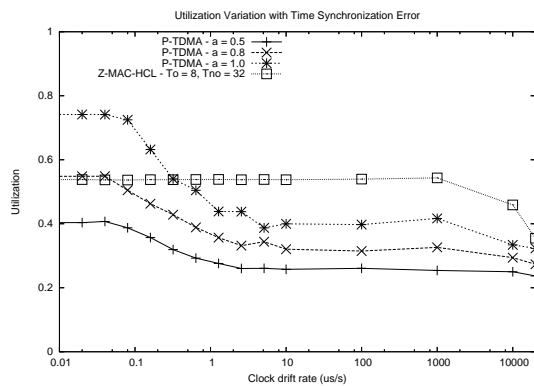


Fig. 10. The data throughput in the two-hop ns-2 benchmark as we add clock drifts. The figure reports throughput when the number of senders is 18 as we vary the drift rate (shown in a logarithmic scale).

Figure 9 shows the results of the two-hop ns-2 benchmark tests. We make sure that the two node clusters do not sense each other, thus maximizing the number of hidden terminals. To be fair, we add RTS/CTS and data acknowledgment for the CSMA techniques (B-MAC and Sift). The size of RTS and CTS including the TinyOS default preamble and synchronization bytes is set to 15 bytes and the acknowledgment size is 5 bytes. the throughput of B-MAC and Sift immediately dropped to zero without RTS/CTS because of hidden terminals so we did not plot them. The utilization of B-MAC and Sift with RTS/CTS reaches around 10 to 12% as RTS/CTS/ACK incur high overhead. These results are similar to the result of B-MAC with RTS/CTS in [3]. The performance of PTDMA does not change much from the one-hop benchmark result since time is completely synchronized and they use DRAND time slots (note that in PTDMA, senders always send at the beginning of a slot without sensing the channel). Z-MAC-HCL shows a good sustained performance independent of the number of senders. Its performance degrades slightly from that in the one-hop ns-2 benchmark because nodes can compete only during their own slots and the slots of their one-hop neighbors and also because of the overhead of ECN messages.

To see more effect of time synchronization errors to the performance of PTDMA and Z-MAC, we add some clock drift in our two-hop ns-2 benchmark. A randomized clock drift value is added at every 1 s and the average throughput of PTDMA and Z-MAC measured over 600 seconds is plotted in Figure 10 as we increase the drift rate. We fix the number of senders to 18. For Z-MAC, we turn off the local clock synchronization feature. The results are that PTDMA shows high sensitivity to even a small clock drift rate ($1 \mu\text{s/s}$), losing its throughput down to 38% of the channel capacity. As we increase the drift rate, the channel utilization of PTDMA quickly drops to 25%. This is because PTDMA relies on a high value of “a” to get the effect of TDMA under high load and with a high “a” value, any overlapping with neighboring slots increases the chance of collision among the owners of neighboring slots. In contrast, Z-MAC shows good robustness to the synchronization errors as it sustains its superior performance until the drift error becomes larger than 1 ms/s even without its local clock synchronization.

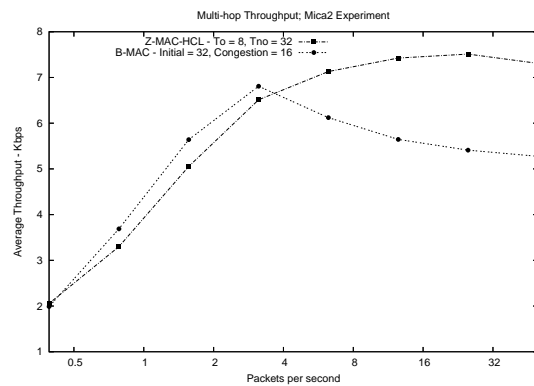


Fig. 11. The data throughput in the multi-hop Mica2 benchmark as we vary the transmission rate of each sender. In this experiment, all nodes except the sink (node 36) are sending.

Multi-hop Benchmark. Figure 11 shows data throughput of Z-MAC-HCL and B-MAC under the multi-hop Mica2 benchmark of 42 nodes. Under transmission rate less than 3.12 packets/sec, both protocols deliver all the packets and achieve about the same throughput. B-MAC shows slightly better throughput than Z-MAC. This is because the backoff congestion window value of Z-MAC for non-owners ($T_o + T_{no} = 8 + 32 = 40$) is larger than B-MAC (16). The backoff value make difference because contention is low and most transmissions in Z-MAC are done as non-owners. As the transmission rate increases beyond 3 packets/sec, we observe that Z-MAC achieves about 20 to 30% higher throughput than B-MAC. Under the full data rate (50 packets/sec), Z-MAC achieves about 7.2 Kbps while B-MAC achieves about 5.2 Kbps. These figures are slightly higher than the values from the two-hop benchmark under low contention. This is because as the network is so densely populated, nodes can sense each other very well so one-hop contention dominates two-hop contention.

C. Fairness

We measure the fairness index [20] of delivered packets of all the senders. As the number of packets delivered to the sink is more uniformly distributed among all the senders, the index approaches one. We compute fairness index from the average number of packets delivered per sender within 10 second intervals.

Figure 12 shows the fairness index from the multi-hop Mica2 experiment. Under low transmission rates, both Z-MAC-HCL and B-MAC show high fairness. However as the transmission rate increases, their fairness indices drop (more precipitously for B-MAC). This is because in the testbed some links are so unreliable that under high load, we see high packet losses from the links. Z-MAC still shows about 40% higher fairness index than B-MAC, under the full rate.

D. Latency

We replicate the same latency experiment in [3] using the multi-hop benchmark. [3] uses the sending rate of one packet in every 10 second to measure the latency. We perform the

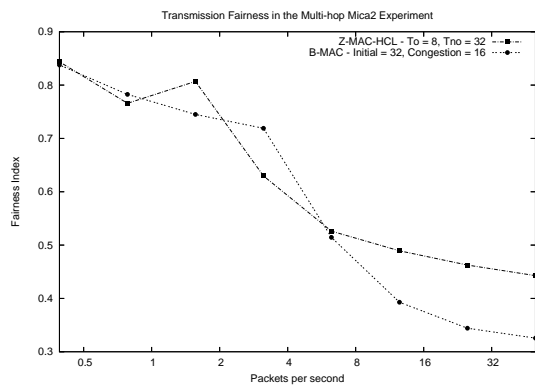


Fig. 12. Fairness index from the multi-hop Mica2 benchmark

Operation	Average (J)	StdDev
Neighbor discovery	0.73	0.0018
DRAND	4.88	3.105
Local Frame Exchange	1.33	1.39
TPSN	0.28	0.036

TABLE II
AVERAGE ENERGY CONSUMPTION (IN JOULE) DURING THE SETUP OPERATIONS IN THE MULTI-HOP MICA2 TESTBED.

same experiment with Z-MAC with HCL enabled (but at this source rate, ECN is never sent; the result is the same as Z-MAC LCL). We run TPSN at the beginning to synchronize the clocks of all the nodes in the line topology and take the latency measurement of each packet using the timestamps taken at the source and sink. Both B-MAC and Z-MAC are tested under LPL with 100ms check interval, and with full duty cycle.

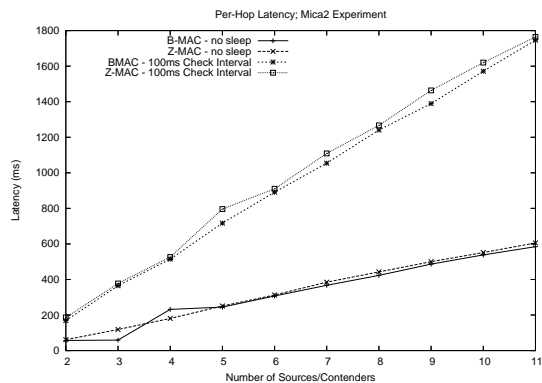


Fig. 13. The end-to-end latency on the 10-hop chain topology

Our result is very similar to that in [3]. Both protocols show very similar latency in all tests. This indicates that the protocol overhead of Z-MAC is quite comparable to B-MAC's.

E. Energy Efficiency

Table V-E shows the itemized energy cost of the Z-MAC setup phase operations in the multi-hop benchmark. We run the setup phase for 30 times and report the average values and standard deviations. Total 7.22 J/node on average is consumed for the setup phase which constitutes about 0.03% of the total energy available per node with 2500 mAh and 3 V battery (the

same battery used in Table 3 [3]). Although DRAND and the other operations are not optimized for energy saving, this is still a substantial amount of energy consumption compared to the per-transmission energy cost. However, the idea is that this upfront energy cost is later compensated by increased energy efficiency during the regular transmission of Z-MAC. In this section, we summarize our energy efficiency result from the Mica2-based benchmarks.

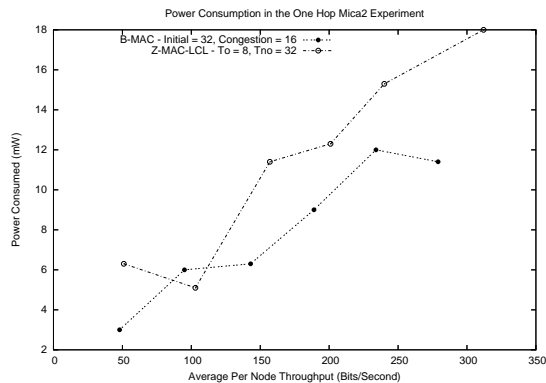


Fig. 14. The power efficiency in low-data rate applications with low duty cycle.

Z-MAC uses the CCA and LPL features of B-MAC. Thus, its energy efficiency is no better than B-MAC's under low-data applications. We run the same energy efficiency test described in Section 6.2 of [3] using the one-hop Mica2 benchmark, and plot the results in Figure 14. As we vary the transmission rate, we compute the optimal check interval for the traffic pattern. The power consumption of Z-MAC is slightly worse than that of B-MAC. This is because in Z-MAC, (1) nodes tend to wake up longer for transmission since their backoff window sizes are larger and (2) clock synchronization messages are periodically sent. In this test, data rates being low, all nodes are in LCL and no overhead for ECN is incurred.

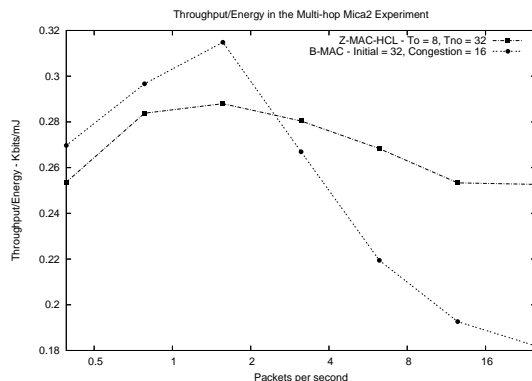


Fig. 15. The power efficiency in the multi-hop Mica2 benchmark.

We measure the energy efficiency of Z-MAC and B-MAC in the multi-hop Mica2 benchmark. For each sending rates, we vary the duty cycle from 20% to 60% and measure the energy efficiency in terms of throughput over power. Figure 15 presents the best ratio of throughput over power for a given sending rate among all duty cycle runs. As we observe in

the multi-hop throughput test, under low data rates, B-MAC has slightly higher throughput. Also we observe in the energy efficiency test, that B-MAC also has slightly less power consumption (up to 10%) under low transmission rates. This is again because as B-MAC has a smaller contention window size than Z-MAC, its idle time is less under low transmission rates. But as the transmission rate increases beyond 3 packets per second, Z-MAC's energy efficiency improves and beats that of B-MAC by about 40% under the full rate. This higher energy efficiency under high transmission rates is attributable to the efficiency in the contention resolution of Z-MAC-HCL.

VI. CONCLUSION

This paper presents Z-MAC, a new MAC protocol for sensor networks that can dynamically adjust the behavior of MAC between CSMA and TDMA depending on the level of contention in the network. The protocol uses the knowledge of topology and loosely synchronized clocks as hints to improve MAC performance under high contention. Under low contention, and when these hints are not reliable, the protocol behaves like CSMA. Z-MAC is useful for applications where expected data rates and two-hop contention are medium to high.

ACKNOWLEDGMENT

The work reported in this paper is financially supported in part by NSF-NOSS 0435157.

REFERENCES

- [1] H. Balakrishnan, "Opportunities in high-rate wireless sensor networking," October 2004, NSF NOSS Principal Investigator and Informational Meetings.
- [2] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *ACM MobiCom '01*. New York, NY, USA: ACM Press, 2001, pp. 221–235.
- [3] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *ACM SenSys '04*. New York, NY, USA: ACM Press, 2004, pp. 95–107.
- [4] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 493–506, 2004.
- [5] S. Ramanathan, "A unified framework and algorithms for (T/F/C)DMA channel assignment in wireless networks," in *IEEE INFOCOM 1997*, 1997, pp. 900–907.
- [6] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," in *ACM MobiSys '04*. New York, NY, USA: ACM Press, 2004, pp. 125–138.
- [7] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND:: Distributed randomized TDMA scheduling for wireless ad-hoc networks," in *ACM MobiHoc '06*. New York, NY, USA: ACM Press, 2006, pp. 190–201.
- [8] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *ACM SenSys '03*. New York, NY, USA: ACM Press, 2003, pp. 171–180.
- [9] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization," U. C. Berkeley, Tech. Rep., 2001.
- [10] Y. Tay, K. Jamieson, and H. Balakrishnan, "Collision-Minimizing CSMA and its Applications to Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications*, August 2004.
- [11] K. Jamieson, H. Balakrishnan, and Y. C. Tay, "Sift: A mac protocol for event-driven wireless sensor networks," in *EWSN*, 2006, pp. 260–275.
- [12] J. Li and G. Y. Lazarou, "A bit-map-assisted energy-efficient mac scheme for wireless sensor networks," in *IPSN '04*. New York, NY, USA: ACM Press, 2004, pp. 55–60.
- [13] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," *Wirel. Netw.*, vol. 12, no. 1, pp. 63–78, 2006.
- [14] A. Ephremides and O. A. Mowafi, "Analysis of a hybrid access scheme for buffered users—probabilistic time division," in *IEEE Transactions on Software Engineering*, Vol. SE-8, No. 1. IEEE, Jan. 1982, pp. 52–61.
- [15] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *ACM SenSys '03*. New York, NY, USA: ACM Press, 2003, pp. 138–149.
- [16] A. L. Edwards, "The correlation coefficient," in *An Introduction to Linear Regression and Correlation*. W. H. Freeman, 1976, pp. 33–46.
- [17] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: congestion detection and avoidance in sensor networks," in *ACM SenSys '03*. New York, NY, USA: ACM Press, 2003, pp. 266–279.
- [18] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RFC 1889: RTP: A transport protocol for real-time applications," Jan. 1996.
- [19] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *ACM SenSys '03*. New York, NY, USA: ACM Press, 2003, pp. 14–27.
- [20] R. Jain, D.-M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Digital Equipment Corporation, Tech. Rep., 1984.

Injong Rhee (SM'89) received his Ph.D. from the University of North Carolina at Chapel Hill. He is an associate professor of Computer Science at North Carolina State University. His areas of research interests include computer networks, congestion control, wireless ad hoc networks and sensor networks. He works mainly on network protocol designs optimizing the transport performance of networks.



Ajit Warrier received his B.E. from Nirma Institute of Technology, Ahmedabad, India and his M.S. degree from North Carolina State University, where he is currently working toward his Ph.D. degree. His research interests are in multi-hop wireless networks.

Mahesh Aia received the M.S. degree from North Carolina State University and now works for TapRoot Systems. His research interests are in mobile wireless networks and multimedia networks.

Jeongki Min received his B.S. from Hanyang University, South Korea and his M.S. degree from North Carolina State University, where he is currently working towards his Ph.D. degree. His research interests are in multi-hop wireless networks.

Mihail L. Sichertiu (M'98) received the B.S. and M.S. degrees from the Polytechnic University of Bucharest, Bucharest, Romania, in 1995 and 1996, respectively, and the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, in 2001. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh. His research is focused in the area of wireless networking.