Z-MAC: a Hybrid MAC for Wireless Sensor Networks

Injong Rhee, Ajit Warrier, Mahesh Aia and Jeongki Min
Dept. of Computer Science, North Carolina State University
Raleigh, NC 27695

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.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Networks]: Network Architecture and Design,Wireless Communications

General Terms: Algorithms, Design, Performance

Keywords: MAC, CSMA, TDMA, wireless sensor networks.

1. 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 *multiple access control* (MAC), plays a key role in determining channel capacity utilization, network delays and more important, power consumption. It also influences 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 [4] 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.

A common MAC paradigm in wireless networks is CSMA (carrier sense multiple access). It is popular because of 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 [3, 15]) 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 [24]. 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 [17]). 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 communica-

SenSys'05, November 2–4, 2005, San Diego, California, USA.

tion range (this phenomenon is known as *interference irregularity* [25]). 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 synchronization 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 so that 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* [18], an efficient scalable channel scheduling algorithm. DRAND is a distributed implementation of RAND [17], 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 time 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. Our analysis shows that even in the case when clocks are completely unsynchronized and some degree of slot assignment failure occurs, its performance becomes comparable to that of CSMA.

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

2. RELATED WORK

S-MAC [24] and T-MAC [20] 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 [20] improves the energy efficiency of S-MAC by forcing all the transmitting nodes to start transmission at the beginning of each active period.

Polastre et al. [15] develop a lightweight MAC protocol called B-MAC which is used as the default MAC for Mica2. B-MAC allows application to implement its own MAC through a well-defined interface. They also adopt LPL (low power listening) [10, 7] and engineer the clear channel sensing (CCA) technique to improve channel utilization. B-MAC is shown to have higher throughput and better energy efficiency than S-MAC and T-MAC.

CSMA/p^* [19] 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 [13] adapts CSMA/p^* [19] 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 mobile 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 [1, 11, 14, 16] are made for TDMA

in sensor networks. Unfortunately, these protocols still fail to address the fundamental difficulties that stand-alone TDMA schemes face.

Seamlessly adapting the MAC behavior between TDMA and CSMA according to the level of contention was previously explored by Ephremides and Mowafi [8] 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).

3. DESIGN OF Z-MAC

Z-MAC has the 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 does 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 compensated 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 the main transmission control of Z-MAC.

3.1 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 [18], a distributed implementation of RAND [17], 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 [18].

3.2 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 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 2^a time frame (is slots are $l \cdot 2^a + s_i$, for all $l = 1, 2, 3, \ldots$).

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

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

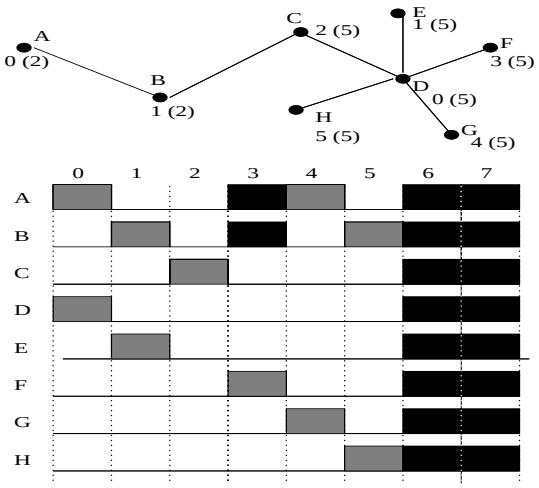

Figure 1: An example of the TF rule. The top figure shows a network topology and the numbers indicate the slot numbers assigned by DRAND and the numbers in parenthesis are F_i. The bottom figure shows the slot schedule of all nodes (shaded slots are ones where each node transmits, and dark slots are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

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 [9], only once at the beginning. After the initial synchronization, each node runs a low-cost local synchronization protocol discussed in Section 3.6.

3.3 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 might create too many fragmentations in time slots reducing the channel utilization. Since the slot size should be big enough to handle more than one packet, HCL slots will give more chance to an owner to succeed without hidden terminals.

Specific values of T_o and T_{no} have performance impact. We select these values based on stochastic analysis to maximize effective throughput. We omit the mathematical analysis because of the space constraint, but instead provide an intuitive explanation for our methodology to pick these values. 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 technique, we set T_o to 8 contention window slots (also a power of 2 for efficient implementation). We

set T_{no} to 32 slots since it is commonly used as the initial contention window size in IEEE 802.11 [5].

The transmission rule of Z-MAC is different from that of PTDMA [8]. Unlike PTDMA, the owner and non-owner access probabilities of Z-MAC ("a" and "b" in Eq. 1 in [8]) 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 the dependency between "a" and "b".

3.4 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).

Under low transmission rates, even if we increase the number of senders, we find that 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* [6], the ratio of co-variance of the two metrics

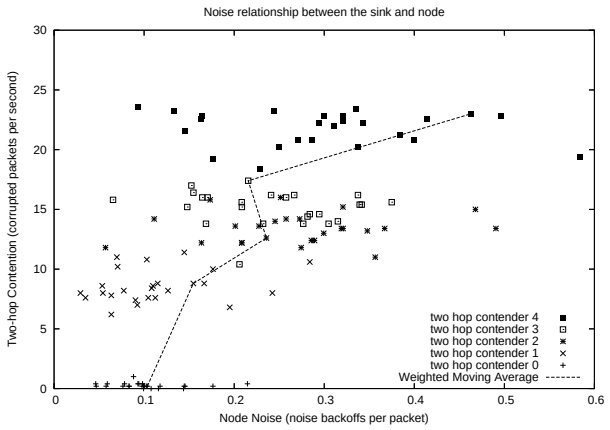

Figure 2: The correlation between the noise level and two-hop contention.

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. Figure 3 shows an example of ECN forwarding.

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

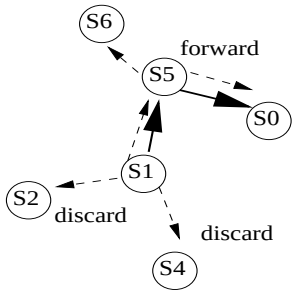

Figure 3: An example of ECN forwarding. When a node $s1$ experiences heavy packet losses, it sends a one-hop ECN to $s5$. The thick arrows indicate the routing path and the dotted arrows indicate ECN messages. Since nodes $s2$ and $s4$ are not the destination, they simply discard the ECN. Node $s5$ broadcasts the ECN to its one-hop. Once HCL is enabled, nodes $s6$ and $s0$ cannot compete during the time slot of $s1$, while nodes $s5$, $s2$ and $s4$ can compete as one-hop neighbors with lower priority than $s1$.

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[21]. 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.

3.5 Receiving Schedule of Z-MAC

DRAND defines only the transmission schedule of nodes. In Z-MAC, a node can transmit in any slot. As Z-MAC is implemented on top of B-MAC, it uses LPL by default wherein each node maintains listening duty cycle separated by a *check period* and each transmission is preceded by a preamble as large as the check period. 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 also is 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. The application designers have to evaluate the trade-offs and find the slot size that fits their needs.

3.6 Local Time Synchronization

Using carrier-sensing and congestion backoffs, Z-MAC becomes more robust to clock errors than TDMA. Even under no synchronization, its performance falls back to that of CSMA. Furthermore under low contention, Z-MAC works like CSMA with or without synchronization. Thus, Z-MAC requires clock synchronization under high contention to implement HCL. An important feature of Z-MAC is that synchronization is required only among neighboring senders and also when they are under high contention. These points offer 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) [2]. 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 $40\mu s$ [9]). 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}: $C_{avg} = (1 - \beta_t)C_{avg} + \beta_t \cdot C_{new}$.

In Mica2, to maintain 1 ms clock accuracy with 40 μs 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.

4. PERFORMANCE EVALUATION

4.1 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 [15] 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 [15]. Since Z-MAC is implemented on top of B-MAC, we use the same packet format as B-MAC (shown in Table 4 in [15]). The default initial and congestion backoff window sizes of B-MAC are 32 and 16 slots respectively (each slot is 400 μs). Except the throughput tests where we vary the backoff window sizes to see the impact of window sizes on channel utilization, we keep the default window sizes. The default values of Z-MAC parameters are shown in Table 1.

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

One-hop benchmark. This benchmark is reproduced from [15] where n nodes are placed equidistant from a receiver in a circle. Each node transmits as quickly as possible with the full transmission power. Before each run of experiment, 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. Since B-MAC has the same test, we can directly compare their result to ours. 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.

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 μs
ECN refresh period (t_{ECN})	10 seconds
Averaging weight for time synchronization (α_{synch})	0.25
Maximum clock drift rate (r_{drift})	40 μs
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 1: The default settings of Z-MAC parameters

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. 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 4 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 [23], the default routing protocol of TinyOS. Figure 5 shows the tree routing paths we used out of 30 nodes in the testbed. We use node 36 as the sink. Thicker lines indicate links carrying more traffic.

4.2 Throughput

In this experiment, we measure and compare the effective channel utilization of B-MAC and S-MAC and Z-MAC. For throughput measurement, we measure only data throughput as done in [15] 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).

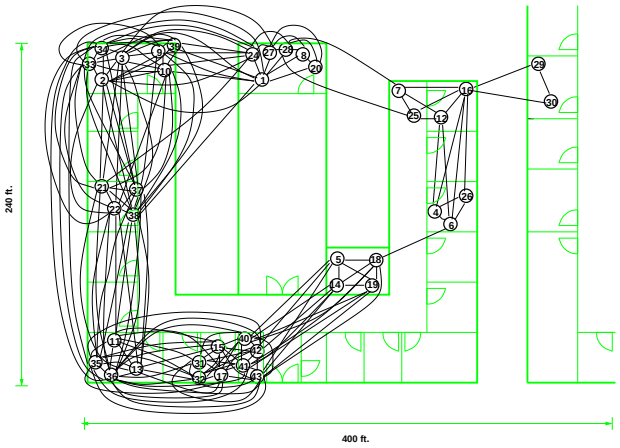

Figure 4: NCSU testbed with 42 Mica2 nodes.

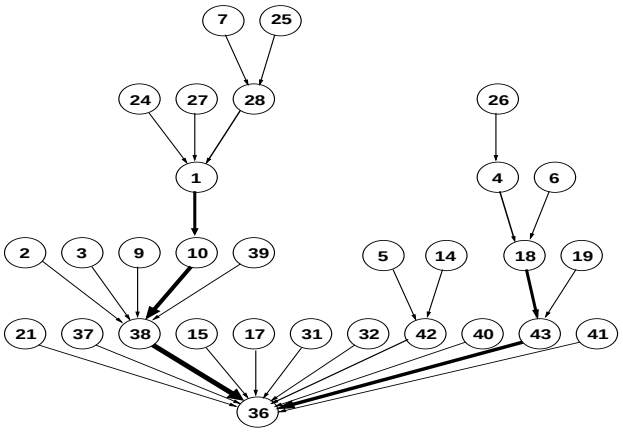

Figure 5: Routing paths used in the multi-hop Mica2 benchmark.

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 6 shows the data throughput of B-MAC and Z-MAC from Mica2 one-hop benchmark. Unfortunately, we are not able to reproduce the same performance of B-MAC as shown in [15]. Our result is significantly less than what they report ([15] 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. The performance of B-MAC with congestion window size 64 shows 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

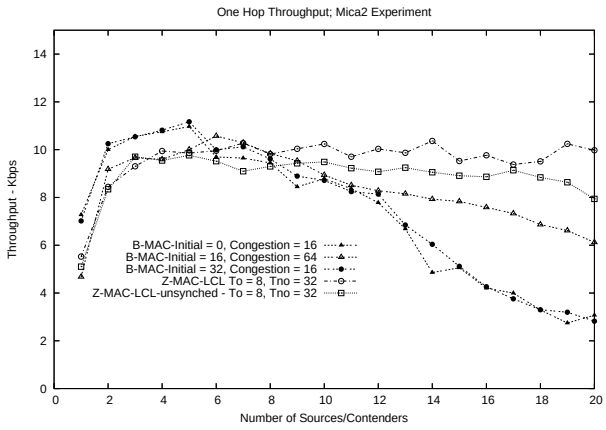

Figure 6: The data throughput comparison in the one-hop Mica2 benchmark.

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 [15].

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 the performance of Z-MAC even with unsynchronized clocks is quite comparable to that with synchronized clocks. This is because T_o is sufficiently large to handle multiple owners within a slot.

Figure 7 shows ns-2 simulation results for one-hop involving PTDMA, Sift, B-MAC and Z-MAC. For PTDMA and Z-MAC, we set the number of stations M to be 21. For PTDMA, we vary "a", the access probability of owners, from 0.5 to 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 32 slots as recommended in [13]. The results presented in this figure are also verified by our stochastic analysis. The analysis can be found in [22]. 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 [8] as well). The B-MAC simulation result closely

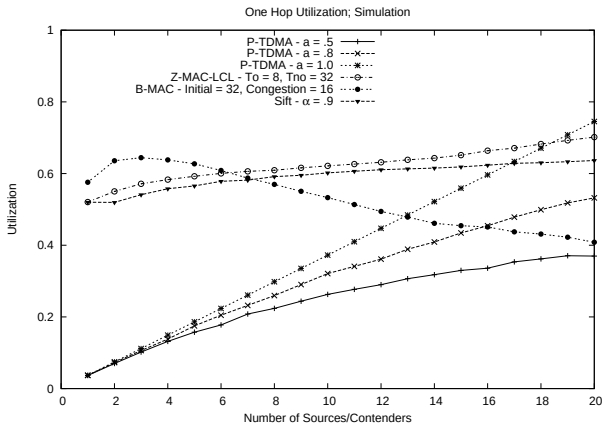

Figure 7: The data throughput comparison in the one-hop ns-2 benchmark.

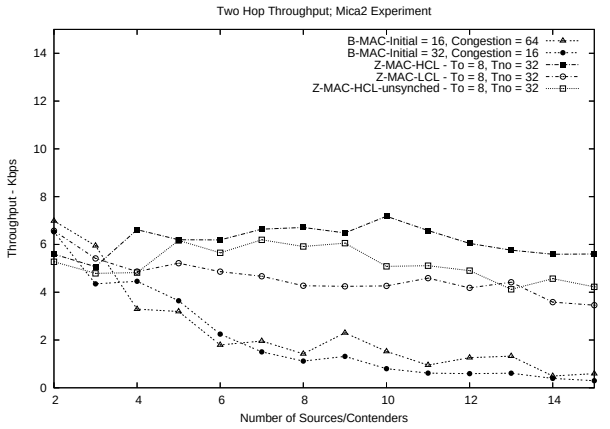

Figure 8: The data throughput comparison in the two-hop Mica2 benchmark.

follows that in [15]. Z-MAC is the top performer in this test, but only by a small margin from Sift. The good performance of Sift is because all nodes are one-hop and data are always available for transmission so the senders can be easily synchronized with each other to the boundary of each packet transmission.

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

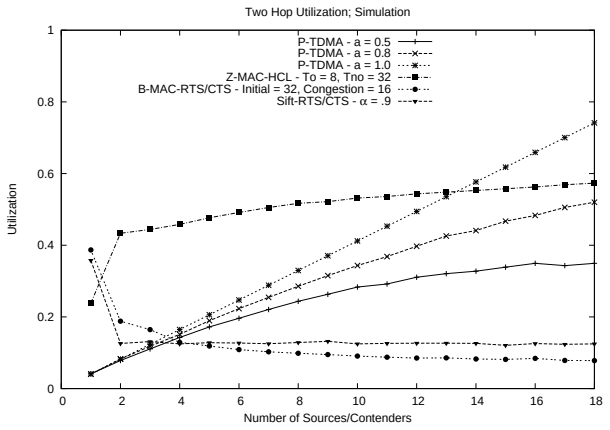

Figure 9: The data throughput comparison in the two-hop ns-2 benchmark.

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 [15], 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. We also run Z-MAC-HCL under unsynchronized clocks. Its performance drops a little but is still better than B-MAC.

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 [15]. 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

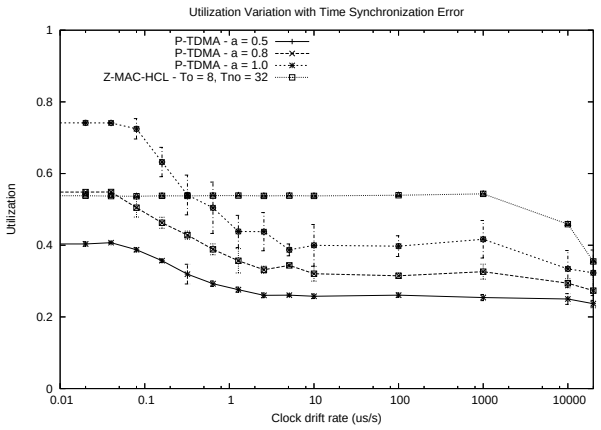

Figure 10: The data throughput in the two-hop ns-2 benchmark as we add clock drifts. The figure reports throughput when the number of senders is 18 as we vary the drift rate (shown in a logarithmic scale).

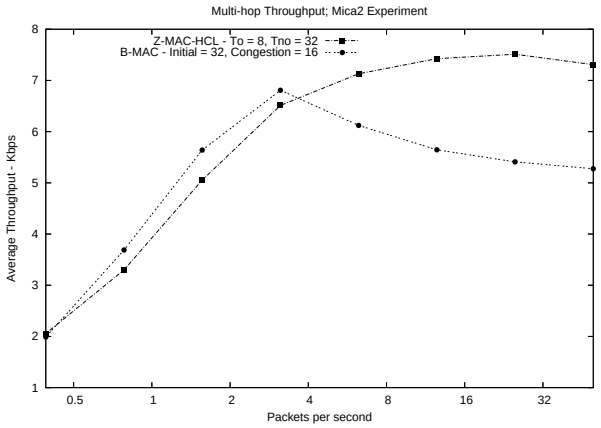

Figure 11: The data throughput in the multi-hop Mica2 benchmark as we vary the transmission rate of each sender. In this experiment, all nodes except the sink (node 36) are sending.

clock synchronization feature. The results are that PTDMA shows high sensitivity to even a small clock drift rate (1 μs/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. Note that the maximum drift rate of Mica2 is around 40 μs per second.

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 lower than 3.12 packets/sec, both protocols deliver all the packets and achieves 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 (which has 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.

4.3 Fairness

We measure the fairness index [12] 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 result of the two-hop Mica2 benchmark. The fairness indices of Z-MAC-HCL are within 0.7 and 1 while those of B-MAC drop to 0.2 to 0.3 as the number of senders increases. The high performance of Z-MAC comes from the TDMA aspect of the protocol as each sender will have eventually one chance to transmit without interference during one time frame. One reason that B-MAC does not perform well is that it does not handle two-hop contention. Thus two-hop nodes get less chance to deliver packets because of increased chance of collision.

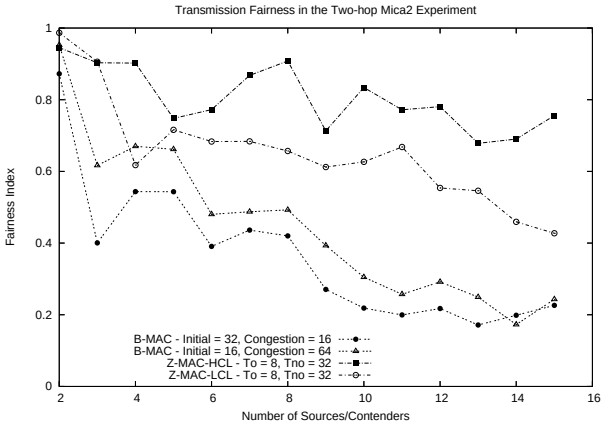

Figure 12: Fairness index from the two-hop Mica2 benchmark

Figure 13 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. The fairness index of Z-MAC still shows about 40% higher than that of B-MAC, under the full rate.

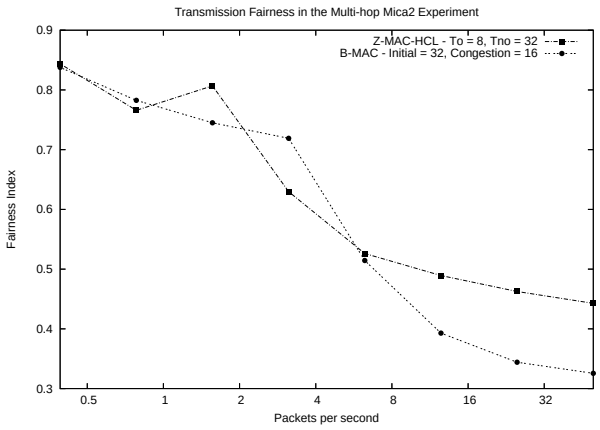

Figure 13: Fairness index from the multi-hop Mica2 benchmark

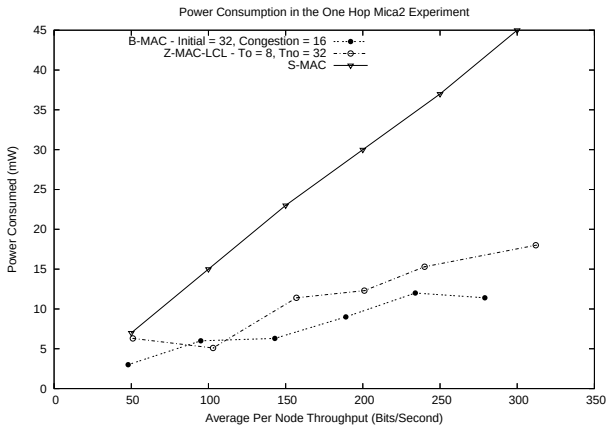

Figure 14: The power efficiency in low-data rate applications with low duty cycle.

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 2: Average energy consumption (in Joule) during the setup operations in the multi-hop Mica2 testbed.

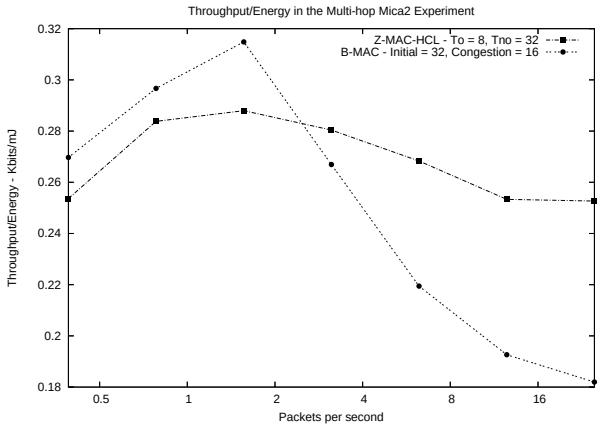

Figure 15: The power efficiency in the multi-hop Mica2 benchmark.

4.4 Energy Efficiency

Table 2 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 [15]). 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.

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 [15] using the one-hop Mica2 benchmark. As we vary the transmission rate, we compute the optimal check interval for the traffic pattern. As a reference point, we use S-MAC result in [15] and plot it along with the results of Z-MAC and B-MAC in Figure 14. 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, the data rates are so low that all nodes are in LCL. Thus, no overhead for ECN is incurred.

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, we find that 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. Figure 16 plots the average number of data transmissions per second in the testbed. We find that Z-MAC has substantially fewer transmissions under high transmission rates. This is because HCL forces senders not to transmit during the slots owned by two-hop neighbors and instead to sleep to save energy.

5. CONCLUSION AND FUTURE WORK

This paper presents a new MAC protocol, called Z-MAC,

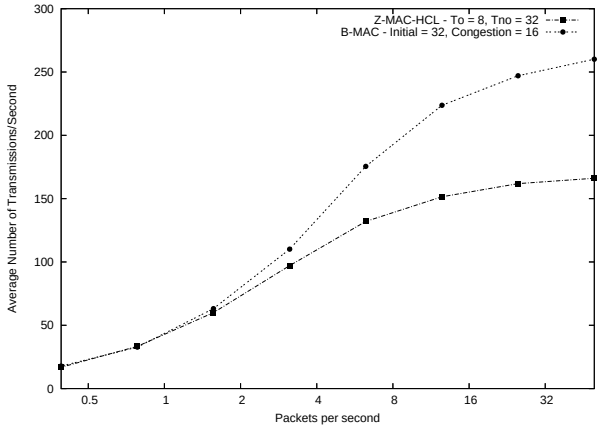

Figure 16: The average number of data transmissions per second in the multi-hop Mica2 experiment

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. The performance results show that Z-MAC has advantage over B-MAC under medium to high contention while it shows competitive, but slightly less performance than B-MAC under low contention (especially in terms of energy efficiency). Z-MAC finds its utility in applications where expected data rates and two-hop contention are medium to high. PTDMA shows high sensitivity to timing and slot errors and further, low utilization when a subset of nodes are transmitting. Sift shows high performance under one-hop contention, but under two-hop contention, it needs to rely on RTS/CTS, hence incurring high overhead. We plan to extend Z-MAC to wireless mesh networks and mobile ad hoc networks. The TinyOS and ns-2 source code of Z-MAC and DRAND can be found in http://www.csc.ncsu.edu/faculty/rhee/export/zmac.

Acknowledgment

The work reported in this paper is financially supported in part by NSF-NOSS 0435157. We would like to thank Andrew Campbell (who helped shepherd this paper) and anonymous reviewers for their constructive comments. We'd also like to thank Kyle Jamieson for providing us with ns-2 implementation of Sift.

6. REFERENCES

[1] K. Arisha, M. Youssef, and M. Younis. Energy-aware TDMA-based MAC for sensor networks. In *IEEE Workshop on Integrated Management of Power Aware Communications, Computing and NeTworking (IMPACCT 2002)*, New York City,NY, May 2002.

[2] 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.

[3] A.Woo and D.Culler. A transmission control scheme for media access in sensor networks. In *ACM MobiCom 2001*, pages 221–235, 2001.

[4] H. Balakrishnan. Opportunities in high-rate wireless sensor networking. NSF NOSS Principal Investigator and Informational Meetings, October 2004.

[5] B. Crow, I. Widjaja, J. G. Kim, and P. Sakai. IEEE 802.11 wireless local area networks. *IEEE Communications Magazine*, 35(9):116–126, 1997.

[6] A. L. Edwards. The correlation coefficient. In *An Introduction to Linear Regression and Correlation*, pages 33–46. W. H. Freeman, 1976.

[7] A. El-Hoiydi. Spatial TDMA and CSMA with Preamble Sampling for Low Power Ad Hoc Wireless Sensor Networks. In *ISCC*, pages 685–692, July 2002.

[8] 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*, pages 52–61. IEEE, Jan. 1982.

[9] S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, November 2003.

[10] J. Hill and D. Culler. A wireless embedded sensor architecture for system-level optimization, 2001.

[11] T. Inukai. An efficient SS/TDMA time slot assignment algorithm. *IEEE Trans. Communications*, 27:1449–1455, 1979.

[12] R. Jain, D.-M. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Technical report, Digital Equipment Corporation, 1984.

[13] K. Jamieson, H. Balakrishnan, and Y. Tay. Sift: A MAC protocol for event-driven wireless sensor networks. Technical Report MIT-LCS-TR-894, MIT Laboratory for Computer Science, 2003.

[14] J. Li and G. Lazarou. A bit-map-assisted energy-efficient MAC scheme for wireless sensor networks. In *3rd Int. Symp. on Information Processing in Sensor Networks (IPSN04)*, pages 55–60, Berkeley,CA, April 2004.

[15] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore,MD, November 2004.

[16] V. Rajendran, K. Obraczka, and J. J.Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, November 2003.

[17] S. Ramanathan. A unified framework and algorithms for (T/F/C)DMA channel assignment in wireless networks. In *IEEE INFOCOM 1997*, pages 900–907, 1997.

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

[19] 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.

[20] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, November 2003.

[21] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: congestion detection and avoidance in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 266–279, New York, NY, USA, 2003. ACM Press.

[22] A. Warrier and I. Rhee. Stochastic analysis of wireless sensor network MAC protocols. Technical report, Computer Science Department, North Carolina State University, Raleigh, NC, 2005.

[23] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27, New York, NY, USA, 2003. ACM Press.

[24] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(3):493–506, 2004.

[25] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138, New York, NY, USA, 2004. ACM Press.