17-10-2005

ArDeZ: a low power asymmetric rendezvous MAC for sensor networks

K. Chin
University of Wollongong, kwanwu@uow.edu.au

Mohammed Raad
University of Wollongong, mraad@uow.edu.au

ArDeZ: a low power asymmetric rendezvous MAC for sensor networks

Abstract

We present a rendezvous based medium access control (MAC), culled ArDeZ, for use in sensor networks. ArDeZ is a TDMA based medium access scheme that does not rely on strict time slot positioning and assignments, making ArDeZ easily deployable in large sensor networks without having to adhere to strict time slot boundaries. ArDeZ establishes two independent peer-to-peer time channels between nodes, and these channels do not necessarily have the same duty cycle. Each channel has a set of rendezvous periods associated with it that are generated from a seed exchanged during the setup process. Further, the duration and frequency of each channel's rendezvous periods are easily changed by an application for each link on a given path, thereby allowing the application to balance traffic and energy requirements. We have implemented ArDeZ in the ns-2 simulator, and our results show that ArDeZ is capable of very low power consumption.

Disciplines

Physical Sciences and Mathematics

Publication Details

This article was originally published as: Chin, KW & Raad, R, ArDeZ: a low power asymmetric rendezvous MAC for sensor networks, Proceedings 14th International Conference on Computer Communications and Networks (ICCCN 2005), 17-19 October 2005, 99-104. Copyright IEEE 2005.

ArDeZ: A Low Power Asymmetric Rendezvous MAC for Sensor Networks

Kwan-Wu Chin and Raad Raad
Telecommunications Information Technology Research Institute
University of Wollongong, Northfields Avenue, Australia 2522
{*kwanwu, raad*}*@uow.edu.au*

***Abstract*— We present a rendezvous based medium access control (MAC), called ArDeZ, for use in sensor networks. ArDeZ is a TDMA based medium access scheme that does not rely on strict time slot positioning and assignments, making ArDeZ easily deployable in large sensor networks without having to adhere to strict time slot boundaries. ArDeZ establishes two independent peer-to-peer time channels between nodes, and these channels do not necessarily have the same duty cycle. Each channel has a set of rendezvous periods associated with it that are generated from a seed exchanged during the setup process. Further, the duration and frequency of each channel's rendezvous periods are easily changed by an application for each link on a given path, thereby allowing the application to balance traffic and energy requirements. We have implemented ArDeZ in the *ns-2* simulator, and our results show that ArDeZ is capable of very low power consumption.**

I. Introduction

Advances in engineering have enabled the development of small low-cost sensor nodes that are typically equipped with a transceiver, processor, memory and one or more sensors. Each sensor node establishes connections with other sensor nodes thereby forming a network capable of relaying sensed data back to a collection point. As a result, sensor networks have wide ranging applications such as precision agriculture [1] and bush fire monitoring [2]. However, the deployment of sensor nodes face many challenges due to tight resource constraints on the sensor platforms, one of them being battery life. Battery life is a key consideration because it affects the operational lifetime of the sensor node and in turn the sensor network, and replacing batteries is impractical when we consider networks with thousand of sensor nodes or when they are deployed in harsh, hard to reach environments.

The problem of prolonging a sensor node's battery life is being attacked from multiple directions where researchers have developed energy efficient solutions at different layers of the protocol stack. In this paper, we are interested in energy efficient medium access control (MAC) protocols. To date, many sensor or low-power MACs [3][4][5][6][7] exist, however these MACs are primarily based around contention-based channel access which need to overcome challenges such as collisions, overhearing, control packet overhead, and idle listening [3].

To address the aforementioned challenges, we outline a novel MAC protocol, called ArDeZ, for use in sensor networks that has tight integration with applications and routing protocols. Sensor nodes using ArDeZ experience very low packet collisions due to the use of pseudo-random time slots or rendezvous periods, and the ability to learn and avoid other sensor nodes' rendezvous periods. Further, ArDeZ does not require global synchronization, and does not have hard time slots boundaries like conventional TDMA MACs (e.g., [8]). As a result, ArDeZ does not require continuous assignment and distribution of time slots to sensor nodes nor require any clustering of nodes that share the same schedule. In fact, ArDeZ maintains dynamic and pseudo-random schedules that are negotiated when a sensor network is formed. Further, the frequency and duration of these schedules can be changed dynamically by higher layer protocols to meet changing traffic requirements.

We have implemented ArDeZ in the *ns-2* simulator and have experimented with various aspects of ArDeZ in different scenarios. Our results show ArDeZ to be very promising in terms of its ability to adapt to changing traffic needs where higher layer protocols configure communication frequencies and thus battery usage of nodes by adjusting the mean rendezvous period (MRP) values of nodes' channels to suit a given delay and throughput. Further, we find ArDeZ to have very low power consumption where a sensor node can remain functioning for months due to the fact that ArDeZ has low probability of collisions and suffers little impact from problems such as overhearing, and idle listening that have hindered contention-based schemes.

This paper has the following structure. Section II describes ArDeZ's channel setup procedure, and how sensor nodes transmit and receive sensed data to/from a sink node. We then present our analysis of parameters used by ArDeZ in Section III before describing our simulation environment in Section IV. This is then followed by simulation results in Section V. After that we review related work in Section VI before presenting our conclusions and future work in Section VII.

II. ArDeZ: Asymmetric Rendezvous MAC

A. Overview

ArDeZ requires each pair of nodes to establish two channels using pseudo-random seeds, one designated as uplink and the other downlink. We allow applications to set each channel to have a different duty cycle to better reflect the flow of information in a sensor network. For example, an application may inform ArDeZ to schedule more uplink rendezvous periods so that more sensed data can be transmitted back to the sink node.

Figure 1 shows a sensor network consisting of three nodes and a sink, and also the time slots for the corresponding uplink and downlink channels. We see that each pair of nodes has pre-defined time slots for communications and these slots are spread across time meaning these slots will have a small chance of overlapping. We will show later how users are able to control the occurence of these slots based on a sensor node's pre-determined duty cycle where a low duty cycle means that a channel's time slots will not appear often hence is less likely to collide with another node's time slot.

B. Channel Establishment

In the following sections we show how ArDeZ establish the uplink and downlink channels.

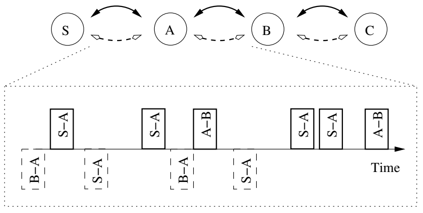

Fig. 1. *ArDeZ* Overview. Time slots at the top and bottom of the time axes correspond to uplink and downlink slots respectively.

1) Node Initiation: A sensor node's first task upon startup is to associate with as many neighbors as possible until a given threshold is met, or until the expiry of the *WaitNeighbor* timer. The expiry of the *WaitNeighbor* timer indicates insufficient neighbors or the sensor node could be the only node in the given geographical area. Note that a sensor node that has acquired too many neighbors will spend a significant amount of time communicating with each of these neighbors. Therefore, higher layer protocols must strike a balance between energy usage and traffic requirements.

To find neighbors, a node remains awake and waits for an *Invite* message, see Figure 2. The frequency of *Invite* messages is determined by the invite message's MRP at each node. Later in Section II-C we will present an algorithm that makes use of the MRP value to determine the frequency of rendezvous periods or in this case invite messages. After an invite message is sent a node will wait for a channel request message (CRM) for a given number of slots. Note that, the number of available slots will be dictated by a node's current neighbors count. For example, an application designer may want to set a limit on the number of neighbors that a node acquires, therefore he/she will reduce the number of slots after each new channel is established. Once a node has acquired enough neighbors it does not need to have an invitation period, or has its invite MRP set to a large value.

The length of an invite period is determined by Equ. 1 which shows the slot size and the number of slots following an *Invite* message.

$$InvitePeriod = \frac{size(CRM) + size(CAM)}{Bandwidth} \times \Psi \qquad (1)$$

where CRM and CAM correspond to the CRM and channel acknowledgment message (CAM) respectively, and lastly the parameter Ψ corresponds to the *RequiredNeighborsCount*. As will become clear later both these messages are used by a node to request and acknowledge the creation of asymmetric channels.

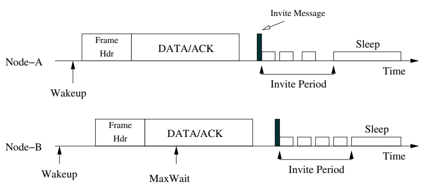

Fig. 2. Transmitting and receiving time frame. Also shown are invite slots for channels setup.

In the following we refer to the node that sent an invite message as an inviter and a neighboring node who wishes to reply as an invitee. After receiving an invite message, an invitee responds to the inviter if all three of the following criteria are met: (1) number of allowable neighbors or total MRPs is below a given threshold, (2) no channels with the inviter exist, and (3) the inviter meets a given criterion, for example has a path to a sink. If these criteria are met, the invitee then initiates the channel set up process described next.

The node first checks whether any of the seeds proposed by the inviter (see Table I) or their resulting rendezvous periods clash with existing channels it has established with other neighbors. There are many ways a node can use to check whether seeds, say S_u and S_d from the set of U_s and D_s, are suitable. For example, the node could check whether S_u and S_d match any initial seeds of existing neighbors. Alternatively, the node may generate and compare x rendezvous periods using the initial seeds of existing neighbors, and also S_u and S_d. By comparing the generated periods and if they overlap significantly, the chosen seeds are rejected. If the proposed seeds are acceptable, the invitee sends a CRM (see Table II for message format) to the inviter, requesting for a pair of channels to be set up using seeds S_u and S_d.

Fields	*Description*
Source Address	Invitee's address
Time Slot Duration	This allows an application on the invitee to negotiate the duration of both uplink and downlink channels rendezvous durations. For example, an application may increase the duration of its rendezvous periods but reduces the MRP if the sensed data is delay insensitive.
Timestamp	This message's sending time
S_u	Selected uplink seed
S_d	Selected downlink seed

TABLE II
FIELDS IN THE CRM

Upon receiving a CRM, the inviter checks whether it has an existing connection to the invitee, and ensures that the chosen seeds have not been taken by another neighbor. If the seeds have been taken, the inviter sends the invitee a negative acknowledgment message informing the invitee reasons why the setup process failed. On the other hand, if there is an existing connection, meaning the invitee has lost synchronization, the inviter deletes any information pertaining to the invitee before proceeding to generate its first rendezvous period with the invitee using the algorithm described in Section II-C. Finally, the inviter sends a CAM back to the invitee confirming that the channels have been set up. The invitee then executes the same algorithm in Section II-C to generate its first rendezvous period with the inviter.

C. Generating Rendezvous Periods

Recall that the CRM message contains seeds, S_u and S_d, that have been proposed by an invitee for establishing both the uplink and downlink channels. If the seeds are acceptable, the inviting node or inviter proceeds to compute rendezvous slots with the invitee using those seeds as follow:

1) The values S_u and S_d are generated randomly by an inviter, and they have a range between 0 and 255. We will elaborate more on the range of random numbers in Section III. Initially,

$$U^i_{seed} = S_u \qquad (2)$$

$$D^i_{seed} = S_d \qquad (3)$$

2) The initial seeds for the uplink and downlink channels are:

$$S^i_u = (C_a U^i_{seed} + C_b) \,\%\, 255 \qquad (4)$$

$$S^i_d = (C_a D^i_{seed} + C_b) \,\%\, 255 \qquad (5)$$

where C_a and C_b are constants, and % is the mod operator.

Fields	*Description*
Node address	The node that originated the invite message.
Sensor capabilities	This variable allows heterogeneous sensors to advertise their capabilities where sensor nodes may use these to influence their channel establishment policies.
Sink address	This indicates that the node has a path to the sink. Having the sink's address is useful, because it enables applications to have a joining policy that dictates which sink a node has a path to. Further, connecting to a node only if it has a path to the sink ensures that sensor nodes are connected to a tree rooted at a sink. Otherwise, if nodes connect amongst each other aimlessly islands of nodes may form, resulting in a segregated sensor network.
Number of hops to Sink	Similar to the sink address field this enables a node to associate the sink with a node that has the shortest path to a sink.
Battery life	A node's battery life, where a joining policy could be that nodes establish channels only if a neighbor has a battery life over a given threshold.
MRP	The mean rendezvous period that determines a node's duty cycle
Timestamp	The invite message's sending time. This value will be used by a neighboring node to account for the difference in clock values.
U_s	Seeds for uplink channel.
D_s	Seeds for downlink channel.
B_s	This node's broadcast seed.
C_a and C_b	Constants used in calculating rendezvous periods.

TABLE I
FIELDS IN THE INVITE MESSAGE

3) We then compute the next wakeup offset as

$$U_i^{wake} = \frac{S_u^i}{255} \times (2 \times MRP) \tag{6}$$

$$D_i^{wake} = \frac{S_d^i}{255} \times (2 \times MRP) \tag{7}$$

where MRP is the mean rendezvous period. The MRP is an adjustable value and can be viewed as a node's duty cycle.

4) Therefore, a node's next uplink and downlink times to node-i are,

$$Uplink_i = (IM_{rx} + U_i^{wake}) \tag{8}$$

$$Downlink_i = (IM_{rx} + D_i^{wake}) \tag{9}$$

where IM_{rx} is the time stamp value, invite message's send time, found in the Inviter's invite message, see Table I.

5) Set,

$$U_{seed}^{i+1} = S_u^i \tag{10}$$

$$D_{seed}^{i+1} = S_d^i \tag{11}$$

After determining the rendezvous periods, the following tuple, <U_{seed}^i, D_{seed}^i, Node_ID, RendezTime, MRP > is inserted into a table called *Neighbors_Tbl*, that records all information pertaining to a given neighbor. Note that, the first rendezvous period is relative to the timestamp in the Invite message which was used to establish the channels. In other words, if the *Invite* message was received at time t and a rendezvous period is calculated to be k seconds, then a rendezvous period will occur at time $t + k$. Subsequent rendezvous periods are then calculated based on the end time of the last rendezvous period.

After setting up a channel, the inviter may choose to remain awake and wait for more neighbors to respond. The invitee on the other hand could come across another neighbor node with a shorter path to the sink or may have a path to a different sink. In the former, the invitee will remove its association with the neighbor with the longer path and re-associate with the new neighbor. Note that, a node does not need to explicitly tear down its connection with a neighbor. This is because a neighbor is considered to have disassociated if it has not sent any packets in the last z rendezvous periods.

A node's duty cycle dictates the amount of energy expended, hence directly impacts a node's lifetime. An application must configure the MRP parameter (see Table I) according to its traffic requirement whilst ensuring that a node wakes up infrequently. Bare in mind that a node may lie on the shortest path to the sink, thus will require a significant amount of energy to be expended in order to get data back to the sink. Therefore, an application designer must ensure that the total MRP of a node is sufficiently low in order to prolong battery life.

The channels we have created thus far are peer-to-peer and cannot be used for broadcasting packets. However, many routing protocols rely on a broadcast channel in order to propagate route information. To address this shortcoming, ArDeZ designates a seed, B_s, specifically for broadcast. Neighboring nodes learn of each other's broadcast seed which they then use to derive each other's broadcast rendezvous periods where at these periods there is only one node broadcasting, and other nodes will be awake to listen only. As we can see this maintains the pseudo-random nature of ArDeZ and also reduces collisions due to broadcast traffic.

D. Transmitting and Receiving

Once a node has one or more neighbors it schedules itself to wake up at the earliest rendezvous period, before going into sleep mode. Upon waking up, assuming the period is for an uplink channel, the node transmits its head-of-line packet to the corresponding node without having to contend for the channel. Figure 2 shows the transmit and receive processes between nodes A and B, where in this example node-A has a packet for node-B. Notice that node-B wakes up earlier than node-A to account for any synchronization errors that may have been caused by clock drifts. Apart from transmitting and receiving, at each rendezvous period a sensor node repeats the steps shown in the previous section in order to generate the next rendezvous period.

To conserve energy each sensor node will wait for a *MaxWait* time before concluding that a neighboring node has no data pending. However, if a sensor node has not transmitted any packets after z periods it will transmit a dummy packet to the receiver in order to keep the channel up and also to make sure clock drifts are corrected.

We like to point out that although there is a possibility that rendezvous periods between nodes overlap, collisions only occur when packets are transmited. To ensure that packet collisions are kept to a minimal a node can perform carrier sense upon wakeup to determine whether the channel is busy before transmitting. If the channel is busy the node schedules itself to wake at the next rendezvous period and goes back to sleep.

E. *Synchronization*

ArDeZ does not require global clock synchronization. A node only needs to record the position of its clock relative to each neighbor it has connection to. This means nodes do not need to synchronize to each of its neighbors nor synchronize to a common clock. However, each node needs to maintain a separate clock drift between it and each of its neighbors, and uses that to calculate the next rendezvous period with a neighbor. This clock drift is updated at each rendezvous period using the timestamp included in each data packet. If there are no data packets, a node will send a dummy packet once every z rendezvous periods. The dummy packet also serves as a "heart-beat" because if a node does not hear from the peer node after a given number of rendezvous periods have passed, the peer is deemed to have left, ran out of battery or that the rendezvous period overlaps with other nodes' periods hence is losing packets due to collisions. As a result, any of these cases will result in the peer's channel information being removed from the *Neighbors_Tbl*.

III. Analysis

A. *Effect of Random Numbers Range*

In the algorithm presented in Section II-C we assumed a random number range of $0 \ldots 255$. However, depending on processor capability application designers may want to use a higher range such as $0 \ldots 65536$. Figures 3 and 4 show the collision ratios using different random number ranges in a sensor network with different node densities. We calculate, out of a total of 100 rendezvous periods for each node, the total number of overlapping periods over the total number of rendezvous periods in the sensor network, for both uplink and downlink channels. In all experiments we investigated the effect of two different random number generators that are capable of generating numbers in different ranges. In Figure 3 we see that as the node density increases to 20 sensor nodes, close to 60% of the rendezvous periods overlap. However, this is not the case in Figure 4 due to the wide ranging variation in random numbers. Hence, depending on the sensor network topology, application designers using ArDeZ need to consider the capability of the on-board processor in generating wide ranging random numbers, otherwise sensor nodes will spend a significant amount of time avoiding overlapping rendezvous periods by going back into sleep mode. It is also important to note here that overlapping wakeup periods will not nessccarely result in a collision. A collision will result if the periods overlap and there is data to trasmit at the same time.

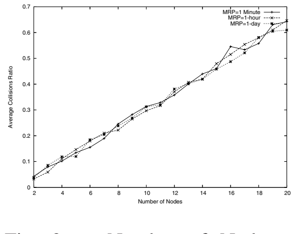

Fig. 3. Number of Nodes vs. Collision Ratio. Range: 0 to 255.

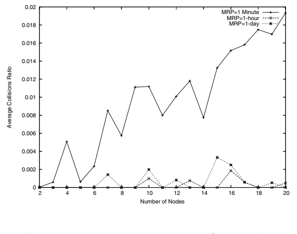

Fig. 4. Number of Nodes vs. Collision Ratio. Range: 0 to 65536.

B. *Channel Setup Process*

ArDeZ's channel setup delay is dependent on the number of slots that are available after each rendezvous period and also the number of nodes that are contending for these slots. The following shows the results of our analysis of the channel setup process. The details of our analysis which led to Figure 5 are outlined in [9].

Figure 5 shows the mean time a node will take in order to complete establishing channels with all its neighbors. The MRP is set to 300 seconds; hence the rendezvous period is every five minutes on average. The number of invite slots is set at eight. We assume that all packet lengths are equal to 50 bytes, and the baud rate is set at 10 kb/s.

Figure 5 shows that as long as the number of neighbors is small, the mean period for channel setup is small and equivalent of the transmission to one IM. As the number of neighbors approaches that of the number of slots, it takes a number of IM messages before channel setup occurs. As the number of neighbors increases, the mean time to setup all the channels increases significantly. The above curve shows that the process of setting up degrades slowly and not dramatically as the number of neighbors increases. This is because, at each successive attempt, the number of nodes that require to setup a channel is being reduced, and hence any successive attempts have a higher chance of being successful.

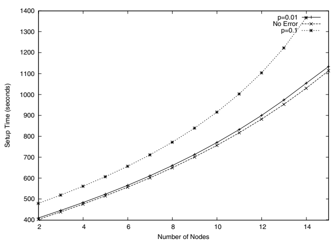

Fig. 5. Mean Channel Setup Time

Figure 5 also shows the effects errors on the setup process. The bottom line shows the performance when there are no errors on the channel. As expected, an increase in the error rate increases the average time to complete channel setup. But what is clear is that the performance degrades slowly and is still proportional to the number of neighbors.

IV. Simulation

We have implemented ArDeZ in the *ns-2* [10] (ns-allinone-2.28) simulator. Our simulation is meant to reflect precision agriculture applications whereby sensor nodes are deployed in a farm to monitor variables such as soil moisture. In a typical farm, nodes remain static and they are either placed randomly or in a grid of size $n \times n$. To simulate sensed data we have each node generate a constant bit rate (CBR) stream of packets of size 50 bytes every 30 minutes back to a sink node which represents the farmer's house. To ensure that sensor nodes know the path to the sink node we require all nodes establish channels with a neighbor that has the shortest path to the sink, thus sensor nodes only need to transmit sensed data onto their uplink channel in order to get the data back to the sink node. Also, the reason channels are establish to neighbor nodes with a path to the sink is to ensure that nodes do not form islands in the sensor network. Thus, the channel setup process is iterative in that the sensor nodes will have to wait for one of their neighbors to obtain a path to the sink node before they themselves can start the channel setup process.

In our simulations we keep the physical layer parameters of the 914 MHz Lucent WaveLAN DSSS radio interface, and set the channel rate to 20 Kb/s. The length of each rendezvous period is 30 milliseconds, i.e., sufficient time to transmit one packet of 50 bytes in size. The values of C_a and C_b for generating the MRP for the random number generator are set to 10 and 20 respectively, and we use a random number range of $0 \ldots 255$ corresponding to a sensor platform with a 8-bit micro-controller. Note, in *ns-2* all nodes are globally synchronized, therefore we only need to account for transmission delays and we did not simulate clock drift. To model energy usage we assume a sensor node consumes 12 mAh for transmission, 1.8 mAh for reception, and 5 μAh when asleep. Further, sensor nodes are powered by a AA battery that provides 2200 mAh of power.

V. RESULTS

Figure 6 shows the average channel setup time. In this experiment, the number of invite slots is set to eight, MRP set to 300, and the packet error rate set to zero, $\frac{1}{100}$ or $\frac{1}{10}$. We see that the performance of the join process is dependent upon the number of slots where once there are more than eight neighbors, the time to setup a channel increases exponentially. Further, we see that a PER of $\frac{1}{100}$ does not have a significant impact on channel setup time.

Figure 7 shows the network setup time of a farm consisting of 36 [1] sensor nodes. We record and average time over 50 runs when the last node establishes a channel to a neighboring node that has a path to the sink. We see that the frequency of invite periods as dictated by the invite MRP plays an important role in determining when a network is fully setup. Note, all nodes have the same MRP value for this experiment. Another parameter that has an effect on network setup time is the number of neighbors allowed as dictated by the number of invite slots. A low number means nodes will have to find other neighbors when a neighbor node has met its maximum neighbors requirement. Another consideration is the frequency of nodes' invite MRP. In this experiment we have assumed that all sensor network nodes are switched on at the same time, therefore a farmer may want them to form a connected network quickly. However, in a farm sensor nodes may be deployed iteratively with the farmer placing sensor nodes strategically around his/her farm, therefore having a larger invite MRP that coincides with the deployment process will be more energy efficient.

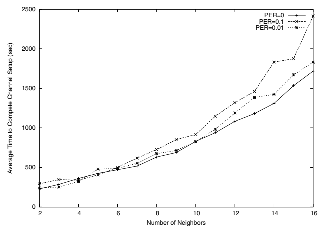

Fig. 6. Channel Setup Time

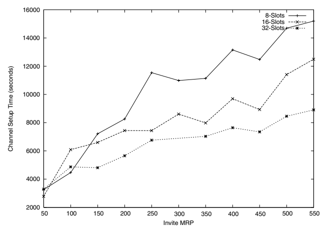

Fig. 7. Network Setup Time - 36 nodes.

Figures 8, 9 and 10 show the impact of MRP values on throughput and delay experienced by nodes. Here, the throughput is determined by the number of packets received over a 300000 seconds period where each node transmits a maximum 100 packets at a rate of one packet every 30 minutes. At the end of the simulation run we record the number of packets that have arrived at the sink, and calculate the average number of packets (out of 100) received over all nodes. From the figures we see that as the MRP gets bigger the impact on delay becomes greater since nodes nearer to the sink will become bottlenecks. Similarly, with increasing number of neighbors or network size the number of packets that get through before the simulation finishes become less, and needs a higher MRP value to accommodate for the increase in load. Figure 10 shows the effect different loads have on the end to end delay. We see that both the load and MRP play a key role in determining the end to end delay of a sensor network.

From the above, it is important that application designers adjust the MRPs of the uplink and downlink channels to values that match sensing or data generation rate and also a node's position within the sensor network. Thus, the routing protocol plays a critical role in balancing delay and battery life. The design of such a routing protocol is currently an on-going effort, and will be presented in the near future.

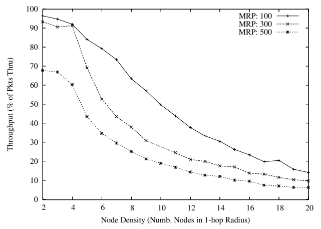

Fig. 8. Neighbor density vs. packet delivery ratio

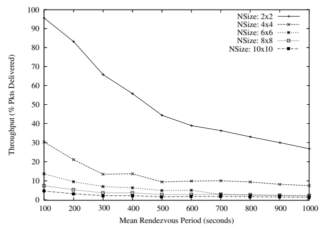

Fig. 9. MRP vs. Throughput for different sensor network sizes

Figure 11 shows the energy drain of three MRP values given different running time, ranging from two to eight months. In this experiment we consider the worst case scenario where a node has a packet to transmit at every rendezvous period. We can see that even at a MRP of 500 the energy left at the end of an eight month period is above 1800 mAh. Clearly, the application will have to tradeoff either delay or battery life.

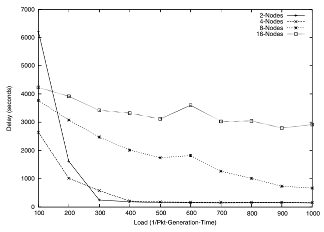

Fig. 10. Load vs. Delay. MRP is set to 200.

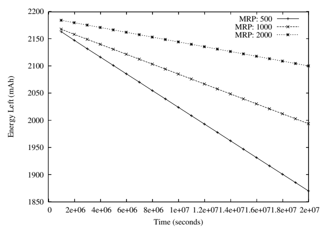

Fig. 11. Energy drain vs. MRP

VI. RELATED WORK

A scheme that uses pseudo-random numbers to establish TDMA channels was first proposed for low orbit satellite system presented by Binder et al. [11]. Each pair of satellites sets up a channel using a master and slave mechanism where a master satellite sends an invite message containing the sender's ID, current position and motion, local clock time, and an initialization seed on a special antenna and hence different physical link. A slave satellite then determines its rendezvous periods with the master satellite using the initialization seed and clock reference information within the invite message. We have extended Binder et al.'s protocol by applying some of their ideas for use in sensor networks, and also provided an algorithm for generating rendezvous periods and defined signaling messages for setting up channels which they have omitted from their paper.

There has been relatively few MACs that generate pseudo-random rendezvous or sleep/wake periods. In [12], Chlamtac et al. assign a distinct polynomial over a Galois field of a given order to each node. This enabled nodes to derive their time slots from roots of polynomials, and they showed each node is afforded at least

[1]Other results were omitted in the interest of page limit.

one collision free transmissions within a frame. Many aspects of their work and ours are similar where in their scheme nodes could exchange polynomials and work out which time slots a neighbor node will be transmitting or receiving in. However, their scheme is less flexibile in that applications cannot control the number of time slots used by a node, and it also requires nodes to be synchronized. In both [13] and [14] nodes exchange a probability variable or seed with their neighbors which is then used to generate a transmission probability or time which a node can use to avoid a neighboring node's transmissions. We see that these existing works have generally targeted the problem of reducing the collision probability of contention based approaches or noise floor. ArDeZ on the other hand is a loose TDMA scheme that does not need nodes to be synchronized globally nor require them to form clusters or assignment of time slots.

In [15], the authors briefly mentioned the advantage of a pure asynchronous rendezvous scheme where nodes have a wake-up radio and a sender simply transmits a wake-up signal whenever there are packets destined for a receiving node, thereby only waking the receiver when it is necessary to do so. However, they observed that wake-up radios are only feasible if they consume less than 50 μW when in standby mode. ArDeZ achieves asynchronous transmission/reception without the need for wake-up radios since nodes generate rendezvous periods using an initial seed agreed upon during channel setup.

TRAMA [8] has a contention period which is used by nodes to determine the nodes and their corresponding traffic schedule within a two hops range. The schedules are then used by each node to determine how time slots in the scheduled access period are going to be used, thereby achieving collision free access albeit with additional signaling overheads. Although in the current embodiment ArDeZ does not guarantee collision free channel access we believe ArDeZ to have a lower energy requirement due to less number of signaling messages emitted by nodes since we dot require access periods and time slots coordination, and does not require nodes to contend for pre-defined time slots when routing packets to the sink.

Current MACs for sensor networks have focused mainly on a contention-based approach [3][4][5][6][7]. For example, S-MAC [3] forms a virtual cluster consisting of sensor nodes that share the same sleep/wake periodic pattern. Nodes that reside in the overlapping area between two clusters will have two sleep/wake patterns, therefore has twice the duty cycle. One of the key energy conserving problems in contention-based MACs is to reduce idle listening. In other words, listening for packets that may not arrive at all. For example, T-MAC [4] defines a threshold period which a node will wait for an activation event, for example arrival of a packet, and upon expiry goes into sleep mode. The idle listening problem has little impact on ArDeZ due to channels being peer-to-peer since at any rendezvous point there is only one peer node, thus a node only need to ensure that the peer node does not have any packets waiting for it. After waiting for a minimal period and if no packet appears a node simply schedules for the next rendezvous period and sets itself to sleep. Another issue that has minimal impact on ArDeZ is the hidden node problem because of the pseudo-random nature of the rendezvous periods.

VII. Conclusion

In this paper we have presented ArDeZ, a novel and simple TDMA based MAC that relies on pseudo-random rendezvous periods for use in sensor networks. ArDeZ has low packet collisions, and adjustable mean rendezvous period values for uplink and downlink channels, and overcomes problems typically associated with deploying conventional TDMA based protocols in sensor networks. One of ArDeZ's key features is the simple interface it provides to applications or a routing protocol in order to adjust a channel's rendezvous periods depending on traffic requirements, thus allowing an application to balance delay and energy usage. This is especially important when we consider the number of sinks and corresponding network topology where for example in a tree-based network topology nodes nearer to the sink will require a higher MRP value.

We are currently gathering more results for different network topologies with different number of sinks, and also comparing ArDeZ to other MACs for sensor networks. Finally, given the ability to control a node's mean rendezvous period, we are investigating various transport and routing protocols that will "tune" the MRP values at each hop depending on traffic requirements.

References

[1] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," in *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean'2001*, (Costa Rica), Apr. 2001.

[2] "The firebug project." http://firebug.sourceforge.net.

[3] W. Ye, J. Heidemann, and D. Estrin, "An energy efficient MAC protocol for wireless sensor networks," in *IEEE Proceedings of INFOCOM'2002*, (New York, USA), June 2002.

[4] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *IEEE SenSys 2003*, (Los Angeles, USA), Nov. 2003.

[5] S. Singh and C. Raghavendra, "PAMAS: Power aware multi-access protocol with signaling for ad-hoc networks," *ACM Computer Communications Review*, vol. 28, pp. 5–26, July 1998.

[6] A. Woo and D. Culler, "A transmission control scheme for media access in sensor networks," in *MOBICOM'2001*, (Rome, Italy), Oct. 2001.

[7] IEEE 802.15.4 Working Group, "Part 15.4: Wireless medium access control (MAC) and physical layer (PHY specifications for low-rate wireless personal area networks (LR-WPANs)." IEEE Std 802.15.4-2003, Oct. 2003.

[8] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy efficient, collision free medium access control for wireless sensor networks," in *ACM SenSys*, (Los Angeles, USA), Nov. 2003.

[9] K.-W. Chin and R. Raad, "ArDeZ: A low power asymmetric MAC for sensor networks." Technical Report, Jan. 2005. UoW-WTL-TechRpt-10, University of Wollongong.

[10] S. McCanne and S. Floyd, "VINT network simulator - ns (version 2)." Software: http://www.isi.edu/nsnam/ns/.

[11] R. Binder, I. G. Stephen D. Huffman, and P. A. Vena, "Cross-link architecture for a multiple satellite system," *Proceedings of the IEEE*, vol. 75, Jan. 1987.

[12] I. Chlamtac and A. Farago, "Making transmission schedules immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 23–33, Feb. 1994.

[13] R. Rozovksy and K. P. R, "SEEDEX: A MAC Protocol for Ad-Hoc Networks," in *ACM MobiHOC*, (Long Beach, CA, USA), Oct. 2001.

[14] T. Shepherd, "A channel scheme for large dense packet radio networks," in *SIGCOMM'97*, (Cannes, France), Sept. 1997.

[15] E. A. Lin, J. M. Rabaey, and A. Wolisz, "Power-efficient rendezvous schemes for dense wireless sensor networks," in *IEEE International Conference on Communications ICC'2004*, (Paris, France.), June 2004.