PairWise: A time hopping Medium Access Control protocol for Wireless Sensor Networks

Kwan-Wu Chin
University of Wollongong, kwanwu@uow.edu.au

PairWise: A time hopping Medium Access Control protocol for Wireless Sensor Networks

Abstract

The availability of low cost, multifunctional embedded devices have become ubiquitous due to their wide ranging application that includes monitoring Radio Frequency Identification (RFID) tagged objects to ambient conditions such as temperature and humidity. Also, these devices can be equipped with a camera and then deployed in hostile environments. A key characteristic of these devices is that they can communicate wirelessly and form an ad-hoc, Wireless Sensor Network (WSN). Devices in a WSN then collaboratively monitor environment factors or objects, and forward data back to one or more central nodes. A key challenge in WSN is ensuring nodes operate in the order of months as they have limited energy resource, and it is impractical to replace their batteries due to their large numbers, and they may be deployed in inaccessible terrains. Hence, it is critical that these devices or nodes employ energy efficient protocols. To this end, we present PairWise, a novel, low power, time division multiple access (TDMA) based protocol for use in WSNs. PairWise is easily deployable in large scale WSNs as nodes are not synchronized globally. Instead, they synchronize and establish a pair of channels with each of their neighbors independently. Each channel hops pseudo randomly in time according to a seed and maximum rendezvous period (MRP). Hence, nodes using PairWise experience very minimal to no collision during communications. Apart from that, higher layer protocols are able to control the MRP of each channel such that a node's duty cycle matches the observed traffic load. We have implemented PairWise in the ns-2 simulator, and compared it to Sensor Medium Access Control (S-MAC) and a TDMA MAC. Our results show PairWise to have very low power consumption whilst ensuring packets have minimal delays. Moreover, PairWise has a high goodput with increasing node density, where goodput is defined as the number of packets that are transmitted by each node pair successfully over a give- n total number of packets.

Keywords

era2015

Disciplines

Physical Sciences and Mathematics

Publication Details

K. Chin, "PairWise: A time hopping Medium Access Control protocol for Wireless Sensor Networks," IEEE Transactions on Consumer Electronics, vol. 55, (4) pp. 1898-1906, 2009.

PairWise: A Time Hopping Medium Access Control Protocol for Wireless Sensor Networks

Kwan-Wu Chin

**Abstract — *The availability of low cost, multifunctional embedded devices have become ubiquitous due to their wide ranging application that includes monitoring Radio Frequency Identification (RFID) tagged objects to ambient conditions such as temperature and humidity. Also, these devices can be equipped with a camera and then deployed in hostile environments. A key characteristic of these devices is that they can communicate wirelessly and form an ad-hoc, Wireless Sensor Network (WSN). Devices in a WSN then collaboratively monitor environment factors or objects, and forward data back to one or more central nodes. A key challenge in WSN is ensuring nodes operate in the order of months as they have limited energy resource, and it is impractical to replace their batteries due to their large numbers, and they may be deployed in inaccessible terrains. Hence, it is critical that these devices or nodes employ energy efficient protocols.*

To this end, we present PairWise, a novel, low power, time division multiple access (TDMA) based protocol for use in WSNs. PairWise is easily deployable in large scale WSNs as nodes are not synchronized globally. Instead, they synchronize and establish a pair of channels with each of their neighbors independently. Each channel hops pseudo randomly in time according to a seed and maximum rendezvous period (MRP). Hence, nodes using PairWise experience very minimal to no collision during communications. Apart from that, higher layer protocols are able to control the MRP of each channel such that a node's duty cycle matches the observed traffic load. We have implemented PairWise in the ns-2 simulator, and compared it to Sensor Medium Access Control (S-MAC) and a TDMA MAC. Our results show PairWise to have very low power consumption whilst ensuring packets have minimal delays. Moreover, PairWise has a high goodput with increasing node density, where goodput is defined as the number of packets that are transmitted by each node pair successfully over a given total number of packets.[1]

Index Terms — Medium access control, wireless sensor networks, energy efficiency.

I. INTRODUCTION

The advances in Micro-Electro-Mechanical Systems (MEMS) have spurred the development of nodes with sophisticated processing, sensing, and communication technologies. Given the low cost and size of these nodes, one can then form novel wireless sensor networks (WSNs) that can be used for surveillance [1], environmental sampling [2], and even object tracking [3]. To realize these applications, nodes must operate in an ad-hoc and un-attended manner, and most importantly, have a lifetime in the order of months or years. Also, in some applications, it is impractical to replace nodes' battery. These requirements create challenging problems for network protocol designers, and have a direct impact on the practical and commercial potential of WSN applications.

Energy efficient Medium Access Control (MAC) protocols play an important role in realizing these applications. They dictate a sensor node's duty-cycle, and hence their energy consumption. Specifically, they determine the lifetime of WSNs. To date, researchers have proposed numerous MACs. For example, contention based approaches such as [4] and [5] are simple and easy to implement on sensor platforms. Unfortunately, they suffer from collisions, overhearing, control packet overhead, and idle listening; all of which cause significant energy expenditure. Henceforth, researchers have proposed TDMA based approaches [6][7][8][9][10], where each sensor node is allocated a slot for transmission or reception. In the former, each node is assigned a transmission slot. However, in the latter approaches, nodes are assigned a receive slot. This means nodes with a packet for the same receiver must contend with each other. Moreover, broadcasting is non trivial as a sender needs to transmit the same packet multiple times. The limitations of TDMA protocols include the following: (i) the need for global synchronization, (ii) transmission/reception schedule is dependent on topological information, either globally or locally, (iii) transmission/reception schedule is not flexible to varying traffic loads, (iii) finding a collision free slot becomes prohibitively expensive as the number of nodes grows, and (iv) changes in topology result in large control message overheads.

Henceforth, in Section II, we propose a new TDMA based MAC, called PairWise, in order to address the aforementioned limitations. Instead of a fixed frame with transmission/reception slots, node pairs establish two channels with each other; i.e., each node establishes a pair of channels, and hence the name "PairWise". Each channel is characterized by pseudo random time hoping rendezvous periods (RPs). Nodes wake up at each RP to transmit or receive, and more importantly, they use each period to maintain synchronization. Hence, nodes using PairWise are synchronized on a per-neighbor basis as opposed to globally.

[1]K-W Chin is with the School of Electrical, Computer, and Telecommunications Engineering, University of Wollongong, NSW, Australia (e-mail: kwanwu@uow.edu.au).

Contributed Paper
Manuscript received August 28, 2009

Apart from that, we show in Section II-B.1, how nodes use a dynamic frame during the invite process to reduce collisions. As a result, nodes are able to establish pair wise channels quickly and in an energy efficient manner. Another key feature of PairWise is that higher layer protocols are able to control a node's duty cycle via a single tuning knob called maximum rendezvous period (MRP). Hence, the RPs of nodes can be configured to match any traffic load. In particular, given the convergecast nature of WSN traffic, nodes near a sink can be configured to have a higher duty cycle; i.e., low MRP value.

The rest of the paper is structured as follows. Section III analyzes the frequency in which the RPs of nodes overlaps with one another. Note that overlapping RPs do not imply collisions because nodes perform carrier sense before transmission and they are able to learn the RP schedule of nodes up to two hops away. Moreover, as shown in Section II-B.6, nodes are able to derive a new pseudo-random hopping sequence that overlap minimally with their neighbors. As a result, nodes using PairWise experience very minimal packet collisions. In Section IV, we outline our simulation methodology. Our extensive simulation studies involving different node densities and traffic load show PairWise to have very low energy consumption without causing significant delays to packets. Moreover, PairWise has a high goodput with increasing node density. The experiments supporting these conclusions are presented in Section V. We then review related works in Section VI before presenting our conclusions in Section VII.

II. PairWise

A. Overview

PairWise is a time hopping MAC that requires a node to form a separate channel for transmission and reception with each of its neighbors. Note, in this paper, the term “channel” refers to a time period as opposed to frequency or code. The resulting channels or RPs hop randomly in time, and nodes are only required to be awake during these periods to transmit or receive packets to/from a given neighbor. Hence, given that these periods occur randomly, node pairs’ communications are less likely to collide. As we will see later, nodes can pre-compute their own RPs schedule and compare that against their neighbors' RPs schedule in order to avoid collisions.

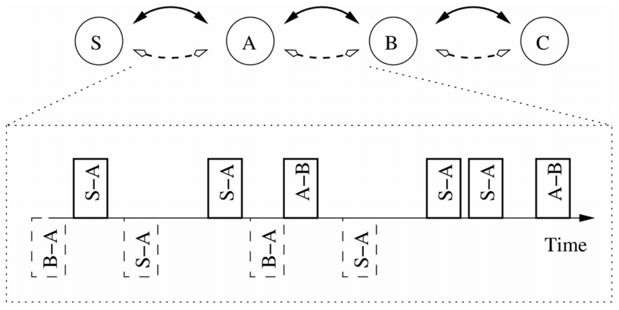

Fig 1. *PairWise MAC overview*. The time slots at the top and bottom of the axis correspond to an uplink or downlink channel respectively.

Figure 1 shows how four sensor nodes use PairWise to communicate. Each pair of nodes has two channels, one designated as {\it uplink} and the other as *downlink*. The former channel is used to forward data to node-S, which in this example is acting as the sink node. On the other hand, the *downlink* channel is used to transfer request messages from the sink to nodes in the WSNs. Notice that each link/channel in the WSN occurs randomly, and ideally, avoids each other. Moreover, nodes know exactly when they are supposed to rendezvous with a neighbor, and whether a given period is used for transmission or reception. Lastly, the duty cycle of channels, i.e., the frequency of RPs, can be configured to match the rate of data flows. For example, nodes may lower the duty cycle of their downlink channel if the sink node only communicates with them occasionally.

B. Protocol Details

The following sections present key aspects of PairWise.

1) *Neighbor Discovery*: By design, each node can only have up to N_{max} neighbors, where N_{max} is a system wide parameter. This is because a node's wake-up frequency is directly proportional to the number of neighbors. Hence, the more neighbors a node has, the more frequent it has to wake up to transmit or receive, and hence have higher energy expenditure as compared to nodes with lower degrees of connectivity. Note, a high degree of connectivity may be necessary to ensure that an optimal path exist between any pair of nodes. Therefore, higher layer protocols must strike a balance between energy usage and traffic requirements.

At startup, a node monitors the channel for *Invite* messages; see Table I. If after waiting for *WaitNeighbor* seconds and no *Invite* message arrives, the node broadcasts an *Invite* message after waiting for a random period of time, and waits a further *WaitNeighbor* seconds. Within this period, if the node receives a channel request message (CRM), it proceeds according to the steps presented in Section II-B.2. Otherwise, it continues to monitor for *WaitNeighbor* seconds again, and repeats the aforementioned process up to *MaxInviteLimit* times before going back to sleep.

Table I
Fields in the Invite Message.

Field	*Description*
Node address	The address of the node that originated this invite message.
N_I	Number of slots following this message.
Sink’s address	This indicates that the node has a path to the sink with this address.
$Sink_{HOP}$	The number of hops to the sink identified in the previous field.
Battery life	A node’s remaining battery life.
MRP	Maximum RP; i.e., the maximum interval between RPs.
Timestamp	This invite message’s sending time.
U_s	Seeds for uplink channel.
D_s	Seeds for downlink channel.
B_s	Broadcast seed
C_a and C_b	Constants used for calculating RPs.

From here on, we refer to the node that sent an invite message as an *inviter*, and a neighboring node that wishes to reply as an *invitee*.

Upon receiving an Invite message, an invitee responds if the following criteria are met: (i) the number of established channels is less than N_{max}, and (ii) the inviter is new. Note that an invitee can also consider whether the inviter has a path to the sink, and the number of hops an invitee has to the sink before sending a CRM; see Table II.

Each *Invite* message is followed by a frame of N_I slots, each of duration ω. In addition, each message has a number of seeds which are used by the invitee and inviter to compute their RPs; see Section II-B.2. A neighboring node intending to establish a channel with the inviter selects two seeds and two constants from the *Invite* message and a slot randomly. The invitee then transmits its CRM in the chosen slot. Note, the invitee will select seeds that yield minimum overlapping RPs for a given time frame; i.e., the invitee computes the next *n* RPs for each seed, and selects one that overlaps the least with its existing channels.

Table II
Fields in CRM.

Field	*Description*
Source address	Invitee's address.
Time slot duration	This allows an invitee and inviter to negotiate the duration of both uplink and downlink periods. For example, an inviter may want to set the length of each RP such that it can send 10 packets.
Timestamp	This message's sending time.
S_u	Selected uplink seed.
S_d	Selected downlink seed.

Upon receiving a CRM, the inviter checks whether it has an existing channel with the invitee. If there is, the inviter deletes any information pertaining to the invitee before proceeding to compute the first RP with the invitee according the algorithm in Section II-B.2. Note, an invitee needs to re-establish a new channel if it has lost synchronization with an inviter. Another reason is when the invitee fails to receive a channel acknowledgment message (CAM). In both cases, the inviter deletes the staled invitee state and calculates the first RP with the invitee using the new seeds and constants. Finally, the inviter sends a CAM back to the invitee to confirm the creation of a channel. The invitee then computes the first RP with the inviter as per the algorithm in Section II-B.2.

An inviter must ensure invitees do not use the same set of seeds, as this would increase RP collisions significantly. Therefore, if the invitee has chosen the same seeds as a previous node, the inviter sends a negative acknowledgment message (NAM) containing unallocated seeds to the invitee, thereby requesting the invitee to choose different seeds in its next CRM attempt. Note, a possible optimization here is to include the set of unreserved seeds in each CAM, and have nodes listen to CAMs, thereby allowing them to adjust their chosen seeds accordingly. This, however, requires nodes to be awake at each slot to receive CAM. Alternatively, inviters can include a large number of seeds in their CRMs to reduce the probability of two or more invitees selecting the same set of seeds. This is the approach we used in our implementation. Specifically, each Invite message contains a range of seeds that an invitee can choose from; i.e., each Invite message contains a minimum and maximum seed value.

A key problem during neighbor discovery is collisions; two or more nodes transmitting in the same slot. This problem is particularly problematic when there are more neighbors than the number of available slots. To address this problem, N_I is initially set to $N_I^{\min} = 8$. If half of the slots experienced collisions, the inviter increases the number of slots to $N_I = MIN(N_I \times \sigma, N_I^{\max})$, where $N_I^{\max} = 64$. For example, if four out of eight slots have a collision, the inviter sets N_I to 16 in the next Invite message. On the other hand, if less than half of the slots experience collisions, the inviter sets the number of slots to $MAX(\frac{N_I}{\tau}, N_I^{\min})$.

As mentioned, after receiving an Invite message, a node selects a random slot to transmit its CRM. If the node experiences a collision, the node waits for the sender to send another Invite message. This means a node is only allowed to transmit once in each frame. Note that we define collision as having occurred when an invitee did not receive a CAM after transmitting a CRM.

A node, say K, may receive one or more Invite messages from different neighbors. This means node-K is in the overlapping region of two neighboring nodes. In other words, the nodes that transmitted an Invite message are hidden from each other. In this scenario, node-K's transmission is likely to interfere with these neighbors' reception. Hence, node-K is not allowed to transmit after receiving more than one Invite message. Instead, it defers its own Invite message for $(N_i \times \omega) + r$ seconds, where N_i is the latest Invite message's frame size, and r is a random number in the range [0 … 100]. For example, if node-K received an invite message with a frame size of 16, node-K would select a random number, say 33, and transmits its Invite message $(16 \times \omega) + 33$ seconds later. Note, the range [0 … 100] can be adjusted according to a WSN's node density.

2) *Channel establishment:* Recall that in every Invite message, there is a set of uplink and downlink seeds. Moreover, there are constants C_a and C_b. A sender decides on these parameters before broadcasting its Invite message. Upon receiving an Invite message, a node selects an uplink (S_u) and downlink (S_d) seed, and constant C_a and C_b. These

parameters are then stored in a table called *Neighbor_Tbl* along with the start time of the last RP and the MRP. Using these parameters, the first RP is calculated as follows:

a) Set,

$$U = S_u$$
$$D = S_d$$
$$T_u^{Base} = T_d^{Base} = T_j^{Invite}$$

where T_j^{Invite} is the timestamp value found in node-*j*'s Invite message; see Table I.

b) Next, compute the initial seeds.

$$S_u = C_a U + C_b$$
$$S_d = C_a D + C_b$$

c) The wake up offset for both channels are

$$U_{wake} = \frac{S_u \quad \% \quad 255}{255} \times MRP$$
$$D_{wake} = \frac{S_d \quad \% \quad 255}{255} \times MRP$$

where % is the modulo operator.

d) The node then wakes up at the following times to transmit and receive respectively.

$$T^{up} = T_u^{Base} + U_{wake}$$
$$T^{Down} = T_d^{Base} + D_{wake}$$

e) The next RP is then calculated by setting

$$U = S_u$$
$$D = S_d$$
$$T_u^{Base} = T^{up}$$
$$T_d^{Base} = T^{Down}$$

and repeating steps 2 to 5.

As an example, consider node-*i* establishing an uplink channel with node-j. Assume the required parameters have the following value: C_a=10, C_b=20, S_u=35, and MRP=1000. Also, $T_j^{Invite} = 0$. Using these parameters, node-i determines its first rendezvous with node-j is at t=450. After that, the next RP is t=1038, followed by t=1998, and so forth.

A key PairWise feature is that nodes have the flexibility to select different MRPs for each channel. This is beneficial because it allows nodes to adapt to changing traffic load. For example, nodes can reduce the MRP of channel(s) leading to the sink in order to minimize delay. Conversely, nodes with a high number of neighbors can choose to increase the MRP of its channels in order to conserve energy.

3) *Broadcast support*: The channels we have created thus far are for unicast traffic. To support broadcast, a node can perform multiple unicast transmissions. Unfortunately, doing so creates unnecessary delays. To support broadcast, each node advertises a broadcast seed B_s, which is then used by its neighbors to calculate a channel dedicated to broadcast traffic. This means each node has a unique sequence of RPs which it uses to send broadcast messages. Conversely, each node knows the broadcast periods of its neighbors, and hence knows when to wake up to receive broadcast packets from a neighbor.

4) *Time Synchronization*. PairWise does not require nodes to be synchronized. Hence, PairWise does not need a time synchronization protocol. Instead, a node maintains the clock drift between it and each of its neighbors in a table. To ensure this table is up to date, all packets have a timestamp field, which is then used by each node to calculate the time difference between it and the packet's sender. Specifically, node-j calculates the clock drift with node-*i* as follows: $d_i = t_j - (t_i + \upsilon)$, where d_i corresponds to the clock drift for node-*i*. t_i and t_j are the timestamp for node *i* and *j*, and υ is the propagation delay, respectively.

A continuous flow of packets is required to keep the table of clock drifts up to date. Otherwise, nodes may lose synchronization. This is especially problematic given that sensor nodes have poor clock accuracy. To prevent this from happening, a node transmits a dummy packet to neighbors it has not transmitted to in the last *n* RPs. Note, a channel is considered down if a corresponding neighbor fails to respond with any packets within *2n* RPs.

5) *Transmitting and Receiving.* Once a node has at least one neighbor, it schedules itself to wake up at the earliest RP. Upon waking up, assuming an uplink channel, the node does a carrier sense, and if the channel is idle, it transmits the head-of-line packet to the corresponding neighbor. Otherwise, it goes back to sleep. In other words, the node will try to transmit the head-of-line packet again in the next RP.

Figure 2 shows the transmit/receive RP of nodes A and B, where in this example node-A has a packet for node-B. Notice that node-B woke up earlier than node-A to account for any synchronization errors that may have been caused by clock drifts. Hence, nodes using PairWise are not required to maintain accurate clock drifts. After transmitting/receiving, a sensor node calculates the next RP for the corresponding node according to Section II-B.2. In order to conserve energy at each RP, a node goes back to sleep if no packet arrives after waiting for *MaxWait* seconds. From the figure, we can see that a node only needs to be awake for a fraction of the RP.

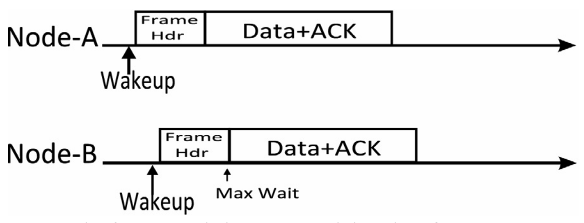

Fig. 2. Transmitting and receiving time frame.

To reduce packet delays, PairWise allows a sender to transmit a burst of packets at each RP. This is achieved by setting the "more" bit in data packets, and including an estimated transmission time of the next packet in the burst. At the receiver, if the "more" bit is set, the receiver checks whether the reception time of the next packet overlaps with its next RP. If not, the receiver sets the "more" bit in its acknowledgment to one. Otherwise, the "more" bit is unset, and the receiver goes back to sleep.

6) *Optimization*. The RPs of node pairs may overlap, thereby increasing the possibility of collisions. It is important to note that RPs are the wake-up times of node pairs, and do not necessarily correspond to packet collisions because nodes may have nothing to transmit. Moreover, nodes perform channel sensing to avoid an on-going transmission. Collisions only occur when there are hidden nodes. Unfortunately, RTS/CTS exchange cannot be used because nodes may be asleep when their neighbors transmit RTS/CTS messages. A potential solution is to transmit a busy tone, but this requires a secondary transceiver that consumes non-negligible amount of energy.

For these reasons, PairWise uses an automatic channel reconfiguration mechanism. That is, if a node fails to transmit a packet, either due to a busy channel or the absent of an acknowledgment, in *k* consecutive RPs, the node removes the channel and attempts to re-establish a new channel using a different seed and constants.

RP collisions can be reduced further by allowing nodes that are within two hops range of each other to exchange seeds. This means an Invite message from a node would contain seeds being used by its neighboring nodes. For example, in Figure 1, if node-B includes seeds communicated to it by node-C, node-A will learn of these seeds from node-B's Invite messages. Using these seeds, an invitee can then compute the RPs of all nodes within its two hops range and determine which seeds yield the fewest RPs overlap. In our example before, node-A will be able to calculate the RPs of node-C's neighbors, and ensures its own RPs do not overlap; at the very least, overlap rarely. For this optimization to work, a node must be aware of the clock drift of its two hops neighbors. Therefore, a node attaches its clock drift relative to each of its neighbors in its Invite message. Using our earlier example, if node-B informs node-A that its clock is 2μs slower than node-C's clock, and if node-A's clock is 2μs faster than node-B's clock, then both node A and C's clock is synchronized.

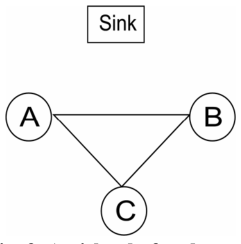

Fig. 3. An island of nodes.

Lastly, when a node has more neighbors than N_{max}, it is important that islands of nodes do not form, as nodes on these islands will not have a path to a sink node. In Figure 3, if N_{max} is two, node A, B and C may form a link with each other and not with the sink or a neighboring node with a path to a sink. As a result, all of them do not have a path to the sink. To prevent this from happening, Invite message contains a field that indicates whether a node has a path to the sink. Hence, a node will only establish channels with neighbors with at least one path to the sink.

III. Analysis

In the algorithm presented in Section II-B.2 we assumed a random number range of 0 … 255. However, depending on processor capability, application designers may want to use a higher range such as 0 … 65536.

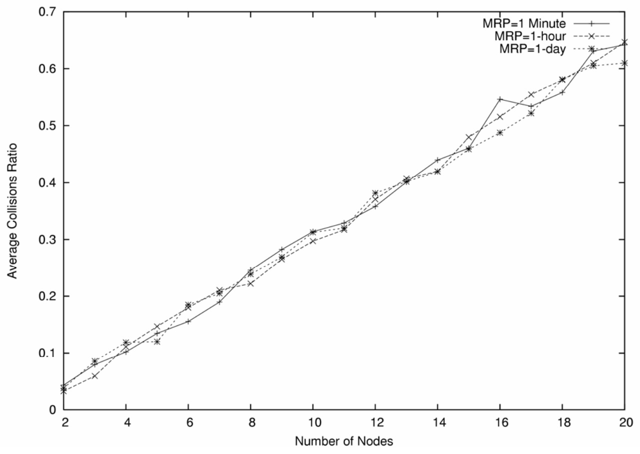

Fig 4. Number of nodes versus collision ratio. Range: 0 to 255.

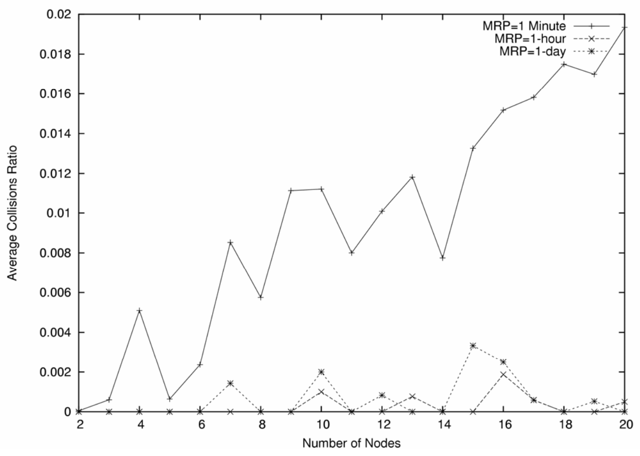

Fig 5. Number of nodes versus collision ratio. Range: 0 to 65536.

Figure 4 and 5 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 4, we see that as the node density increase to 20 sensor nodes, close to 60% of the rendezvous periods overlap. However, this is not the case in Figure 5 due to the wide ranging variation in random numbers. Hence, depending on the sensor network topology, application designers using PairWise 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.

IV. Simulation Methodology

We have implemented PairWise in ns-2 [11] (ns-allinone-2.28). We kept the physical layer parameters of the 802.11b radio interface, but set the channel rate to 250 Kb/s instead. We also calculate the signal strength of each packet in order to determine its error rate. Nodes are static, and they generate constant bit rate (CBR) traffic to their neighboring nodes. Each traffic flow has a maximum of 5000 packets, each 512 bytes in size. We set C_a and C_b to 10 and 20 respectively, and use a random number range of 0 … 65536. Moreover, we experimented with three RPs: see Section V. Besides that, given that nodes are synchronized in *ns-2*, we only need to account for transmission delay. To model energy usage, we assume nodes have a AA battery that provides 2200 mAh of power. Nodes also have a CC2500 [12] radio, which draws 22 mAh per transmission, 14 mAh per reception, 1.5 mAh when idle, and 200 nA when asleep.

We compare PairWise against two other MACs:

- S-MAC [4]. Nodes follow a predefined listen/awake cycle, where nodes wake and sleep together. During listen periods, nodes with a packet to send start contending for the channel using CSMA/CA; we disable the RTS/CTS handshake in our experiments because all nodes are within range of each other. Each listen period is 0.5 second in length, except for the experiment in Section V-D where we consider varying listening periods. In our experiments, we consider the following duty cycles or the time duration between listen periods: 10 and 100 seconds.

- TDMA. Each node is allocated a transmit slot in each frame. This means if there are six nodes, the frame length is $\omega \times 6$ seconds where ω is the slot duration; $\omega = \frac{512 \times 8}{250000}$. At the beginning of each slot, nodes that are not transmitting listen for $0.1 \times \omega$ seconds. Nodes go back to sleep if (i) no transmission is heard, or (ii) a packet is destined for another node. Nodes are awake at the start of every slot, and frames occur consecutively; i.e., nodes do not sleep between frames, thereby ensuring minimal delay between transmissions.

V. Results

In our simulation studies, we have investigated the (i) impact of node density on channel setup time, (ii) energy consumption and delay variance in different traffic load scenarios, and (iii) the goodput of all MACs with increasing traffic load.

A. Channel Setup Time

In this experiment, we study the benefits of using an adaptive frame length. Specifically, we analyze different values of σ and τ; the scalars used to increase and decrease the frame length in accordance with the number of observed collisions respectively.

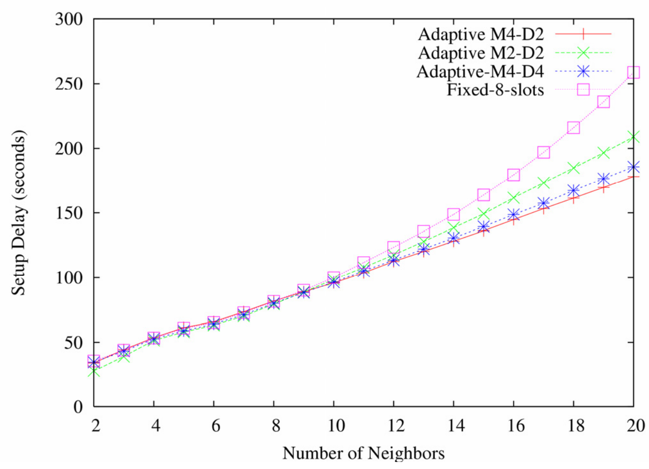

Fig.6. Fixed versus adaptive frame adjustment. The x and y value of the label Mx-Dy denotes the multiplier used to increase and decrease the invite frame length respectively.

Figure 6 shows the advantage of using an adaptive frame length during the invite process. In particular, the channel setup delay of PairWise grows almost linearly with increasing number of neighboring nodes as opposed to exponentially, as is the case when using a fixed frame length. Also, it is important that a node does not drastically reduce its frame length; viz. curve "M4-D2" and "M4-D4". Not doing so ensures collisions are kept low, and hence enables nodes to quickly form channels with each other.

B. Load vs. Energy Consumption

In this experiment, we set three pairs of nodes to communicate at varying rates. Specifically, we vary the packet generation interval (λ) from 10 to 100 seconds. After each experiment, we record the energy consumed by each node, and also the average packet delay of all flows.

Figure 7 shows the average energy consumed by all nodes for varying packet generation rates. We see that PairWise has the lowest energy consumption among all other MACs. TDMA and S-MAC with a 10 second duty cycle have the highest energy expenditure. This is because all nodes using S-MAC needs to wake-up periodically to determine whether they need to receive a packet. In other words, idle listening and overhearing are the key factors that result in S-MAC's poor performance. Similarly, nodes using TDMA suffer from the same problem. Nodes must wake up at the beginning of each slot to determine whether a neighbor has a packet for them. A key advantage of TDMA over S-MAC is that during high load, i.e., $10 \le \lambda \le 20$, nodes using TDMA do not experience collisions. Hence, packets are only retransmitted if they have bit errors.

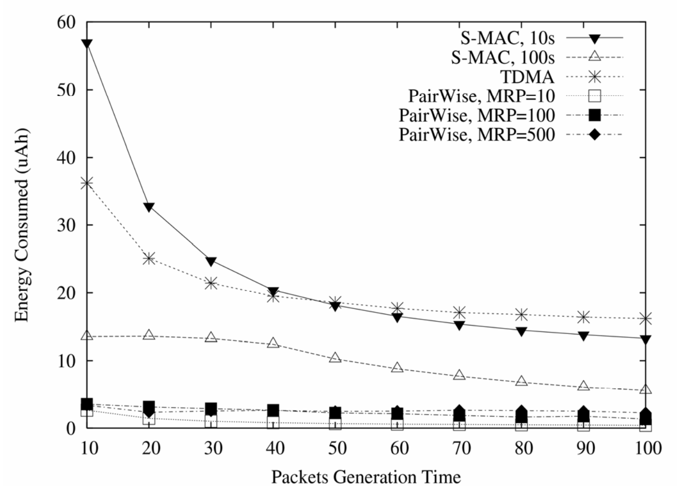

Fig. 7. Energy consumed by PairWise, SMAC and TDMA with varying traffic rates.

PairWise combines the advantages of S-MAC and TDMA. That is, a node wakes up briefly to check whether it needs to receive a packet from a neighbor. Otherwise, it goes back to sleep. This has the effect of minimizing the negative impact of idle listening and overhearing. Other nodes that are not scheduled to transmit or receive remain asleep. This is a marked difference from nodes using TDMA or S-MAC as all nodes need to wake up at the beginning of each slot or in each time interval to check for transmission, and also to ensure they remain synchronized with each other.

C. Load vs. Delay

Figure 8 shows the average delay of packets sent using various MACs. We used the same simulation parameters presented in the previous section. Nodes using TDMA experience the lowest delays. This, however, is at the expense of increased energy expenditure, see Figure 7. Packets transmitted using PairWise have lower delays than those transmitted using S-MAC. For example, PairWise with a MRP of 10 has a slightly lower delay than S-MAC with a duty cycle of 10 second. Recall that the MRP value of nodes corresponds to the maximum time between wake-ups. On the other hand, nodes using S-MAC are awake only every *t* seconds, where *t=10* or *t=100* in our experiments. As a result, nodes using PairWise, on average, experience lower delays.

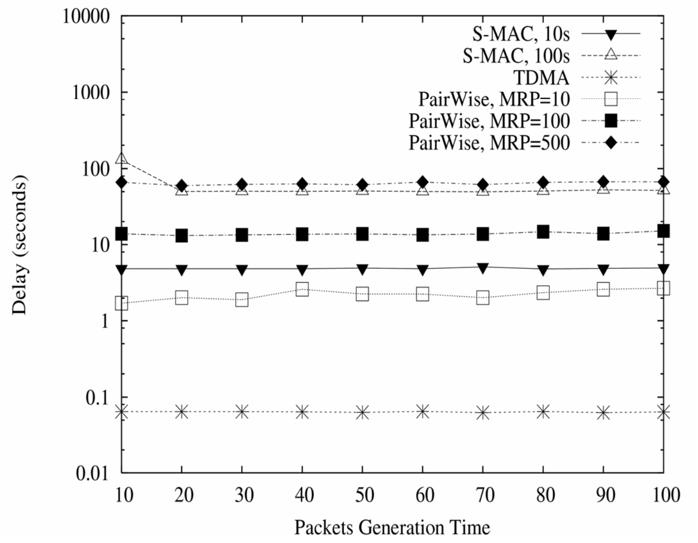

Fig.8. Average delay experienced by nodes using PairWise, SMAC and TDMA in varying traffic rate scenarios.

D. Delay variance vs. Duty Cycles

In this experiment, we investigate what impact duty cycles have on the minimum and maximum delay experienced by packets. By comparing Figure 9 and 10, we can see that packets transmitted using PairWise experience much less delay variance than S-MAC. The key reason is that PairWise nodes wake-up more frequently to exchange packets, but without incurring significant energy expenditure. Hence, a packet that is not transmitted in a given RP does not need to wait long for the next RP to occur; in the worst case, a node waits up to the MRP. However, in practice, most RPs occur sooner than the MRP.

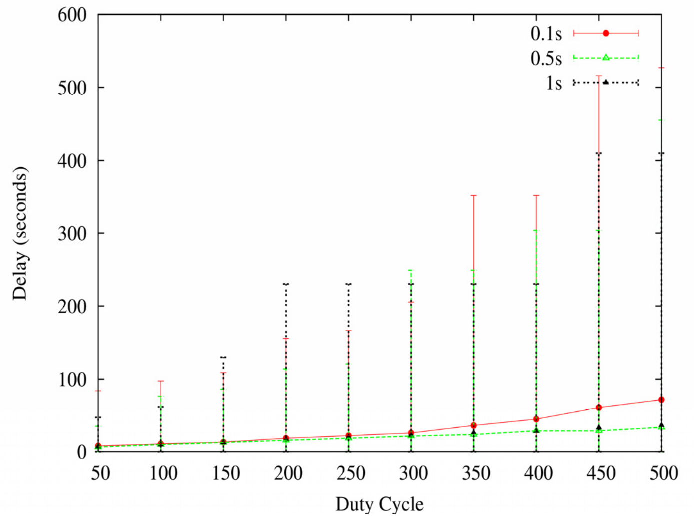

Fig. 9. Delay variance for PairWise. Each curve denotes a node's wake-up period. In other words, the duration in which the node remains awake to transmit and receive.

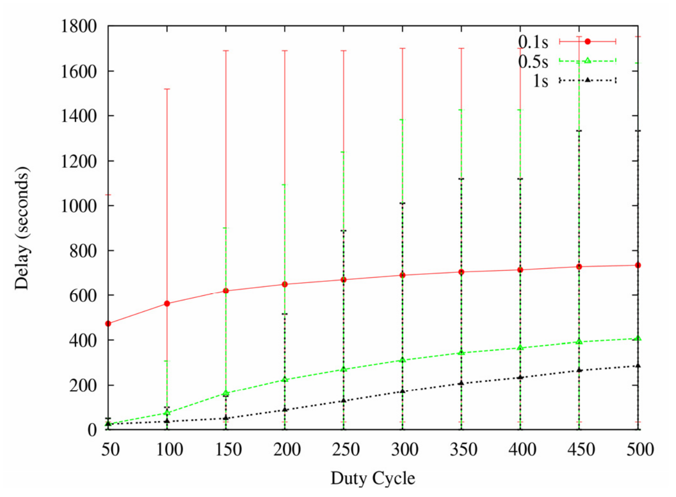

Fig.10. Delay variance for SMAC. Each curve denotes the duration in which all nodes wake up to transmit and receive.

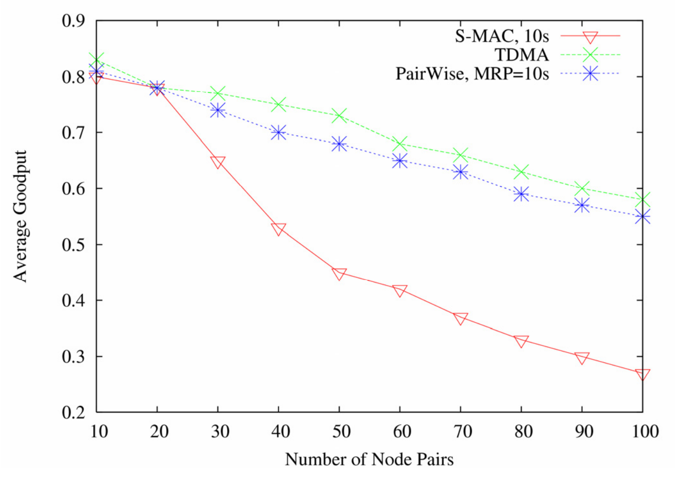

Fig. 11. Average Goodput. PairWise's MRP is set to 10 second. Nodes using S-MAC wake up every 10 second for one second in order to exchange packets.

To reduce variance, we also experimented with nodes' wake-up duration: 0.1, 0.5 and 1.0 seconds. Setting a longer wake-up period clearly helps reduce variance as doing so allows more packets to be transmitted in each period. This is particularly true for S-MAC. However, for PairWise, having a large wake-up duration does not provide any significant gains. This is because extending nodes' wake-up duration results in neighboring node having to wait longer before they are allowed to transmit or receive. Hence, packets experience prolonged delays as compared to a short wake-up duration because nodes occupy the channel for a shorter length of time.

E. Average Goodput

Lastly, we determine the average goodput obtained by PairWise, S-MAC and TDMA with increasing traffic load. That is, we vary the number of node pairs from 10 to 100, and set each pair to transmit 5000 packets at one packet per-second. At the end of the simulation, we calculate goodput as the number of packets that are transmitted by each node pair successfully over the total number of packets, i.e., 5000. We then average the result over all nodes for each of the 10 simulation runs.

Figure 11 shows the average goodput obtained by each MAC. S-MAC's goodput deteriorates quickly as the traffic load increases. This is as expected given CSMA's poor performance in high load scenarios. Specifically, nodes experience more collisions with increasing node density. Moreover, the short wake-up duration bounds the number of packets that can be sent by nodes. In this respect, fairness is an important issue to consider because at each wake-up period, nodes contend for the channel again fairly without any regards for nodes with backlogged packets and those that have not had a chance to transmit in the previous wake-up period. TDMA has the best performance as nodes wake-up frequently to transmit, as opposed to periodically or pseudo-randomly. As a result, nodes obtain better throughput, and more importantly, nodes do not experience collisions. This performance, however, is at the expense of high energy consumption. Moreover, as the number of nodes increases, the frame length becomes longer, which reduces nodes' throughput.

PairWise has a lower goodput than TDMA. This is because nodes wake-up less frequently. Also, RPs of nodes may overlap, thereby preventing nodes from transmitting. Nevertheless, the goodput of PairWise is significantly higher than S-MAC because nodes do not experience persistent collisions. In addition, as PairWise nodes are able to transmit multiple packets in each RP, their goodput is comparable to nodes using TDMA.

VI. Related Works

There are only a handful of MACs that use pseudo-random communication strategies. In [13], a node broadcasts a seed to their neighbors which they then use to calculate the node's transmission and reception probability. Given that nodes are aware of their neighbors' communication state, they can schedule their own transmissions accordingly. Their MAC, however, is targeted at contention based MACs and requires global synchronization. Similarly, in [14], nodes in a spread spectrum system generate pseudo random schedules in order to reduce medium access interference. Apart from [15], which is targeted at satellite systems, existing schemes have mainly considered the use of pseudo-random time-slots in an effort to reduce collisions. For example, the MAC presented by Cao et al. [9] requires nodes to assign a unique seed to each of their neighbors. They then use the seed to select a random communication slot in a neighbor's frame. Their approach reduces collisions because neighbors choose a different slot in each frame. Moreover, a node knows the slots chosen by its neighbors in each frame. Thereby, allowing the node to set itself to wake-up at these slots only. Their scheme, however, does not support broadcast and requires node to be synchronized globally. In [16], the authors briefly mentioned the advantage of a pure asynchronous rendezvous scheme where nodes have a wake-up radio, and a sender only needs to transmit a wake-up signal whenever it wants to send a packet to a node; i.e., nodes are awaken only when it is necessary. However, they observed that wake-up radios are only feasible if they consume less than 50 μW when in standby mode. PairWise 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. However, once low power wake-up radios are realized, we believe sensor nodes will achieve better energy usage as compared to PairWise.

VII. Conclusion

This paper presents PairWise, a novel, simple TDMA based MAC that establishes a pair of pseudo-randomly occurring channel to control nodes transmissions, and hence their duty cycle. Nodes using PairWise experience minimal packet collisions as communication channels follow a pseudo-random hopping pattern. Moreover, nodes are able to learn the transmission schedules of nodes that are up to two hops away. These characteristics ensure nodes' energy is not wasted on unnecessary transmissions and collisions. In addition, nodes do not need to be synchronized globally; a key advantage over existing TDMA based MACs. Lastly, PairWise allows higher layer protocols to adjust a node's duty cycle via a single tuning knob; i.e., MRP. This is especially important for WSN applications. For example, when a WSN is used for object tracking, the MRP of nodes can be reduced temporarily whenever an object is detected so that one or more sink nodes are notified of the event quickly.

References

[1] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "Energy efficient surveillance system using wireless sensor networks," in *ACM MobiSys*, (Boston, Massachusetts, USA), ACM, June 2004).

[2] A. Cerpa, J. Elison, D. Estrin and 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), April, 2001.

[3] D. Klair, K.-W. Chin, R. Raad, and D. Lowe, "A spatially aware RFID-enhanced wireless sensor networks", in the *sixth IEEE International Conference on Pervasive Computing and Communications (IEEE PerCom): Google PhD Forum*, (Hong Kong, China), IEEE, Mar, 2008.

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

[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] Dust Networks, "Technical overview of time synchronized mesh protocol," 2008. White Paper, http://www.dustnetworks.com/.

[7] K. Arisha, . Youssef, and M. Younis, "Energy aware TDMA based MAC for sensor networks," in *IEEE IMPACT*, (New York, CA, USA), May, 2002.

[8] S. Coleri-Ergen and P. Varaiya, "PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks," *IEEE Trans. On Mobile Computing*, vol. 5 no. 7, pp. 920-930, 2006.

[9] H. Cao, K.W. Parker, and A. Arora, "O-MAC: A receiver centric power management protocol," in *IEEE ICNP*, (Santa Barbara, CA, USA), IEEE, Oct 2006.

[10] 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, CA, USA), ACM, November, 2003.

[11] S. McCanne and S. Floyd., "ns network simulator-2" http://www.isi.edu/nsname/ns/.

[12] Texas Instrument, "CC2500: Low-cost low-power 2.4 GHz RF Transceiver." http://focus.ti.com/lit/ds/symlink/cc2500.pdf.

[13] R. Rozovksky and P. R Kumar, "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 *ACM SIGCOMM'97*, (Cannes, France), Sept. 1997.

[15] 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.

[16] 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.

Dr Kwan-Wu Chin obtained his Bachelor of Science with first class honours from the Curtin University of Technology, Australia. He then pursued his PhD at the same university, where he graduated with distinction and the vice-chancellor's commendation. After obtaining his PhD, he joined Motorola Research Lab as a senior research engineer, where he developed zero-configuration home networking protocols and designed new medium access control protocols for wireless sensor networks. In 2004 he joined the University of Wollongong as a senior lecturer. His current research areas include medium access control protocols for wireless networks, routing protocols for delay tolerant networks, RFID anti-collision protocols, and multi-constrained routing algorithms. To date, he holds five US patents, and has published more than 35 articles in numerous conferences and journals.