EFFICIENT MANAGEMENT OF CLOCK DRIFT IN PREAMBLE SAMPLING MAC PROTOCOLS FOR WIRELESS SENSOR NETWORKS

C. E. Tönsing* and G. P. Hancke**

** Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, 0002, South Africa, E-mail: christoph.tonsing@gmail.com*
*** Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, 0002, South Africa, E-mail: g.hancke@ieee.org*

Abstract: In this paper, the design of an energy-efficient Medium Access Control (MAC) protocol for low traffic Wireless Sensor Networks (WSNs) is presented. The protocol, Dynamic Preamble Sampling MAC (DPS-MAC), is based on the existing preamble sampling MAC layer solutions for WSNs. Unlike its predecessors, DPS-MAC does not cater for the worst case clock drift but rather, it dynamically adjusts its operation to the clock drift experienced between any two communicating nodes. In this way, the energy previously expended on overcoming the problem of clock drift is reduced, leading to longer node and network lifetimes.

Key words: Clock drift, MAC protocol, Preamble Sampling, Wireless Sensor Networks.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) form an area of interest that has rapidly expanded in the recent past. A WSN is a collection of many small devices called sensor nodes. Each node is generally capable of performing some type of environment sensing (e.g. temperature, humidity etc.), processing the sensed data and wirelessly communicating with neighboring nodes, usually for forwarding data in a hop by hop fashion from one node to the next towards a sink node or base station node. At the base station/sink node, all the data received from the surrounding nodes can be further processed or stored as necessary.

Typical applications of WSNs include battlefield surveillance, intruder detection, wildlife observation, monitoring of agricultural processes, wildfire detection, home automation and many more. Essentially all of the application areas of WSNs have the common requirements that the sensor networks should be void of fixed infrastructure, should be dynamically deployable, should operate autonomously without human assistance/intervention, and should remain active for long periods of time. To meet these requirements, a WSN node must be autonomous itself, carrying with it a supply of energy (e.g. a battery) or the ability to scavenge energy from its surrounding environment (e.g. a solar panel). It is thus imperative that each node be able to survive for as long as possible with a highly limited supply of energy. The decisive design challenge underlying all types of WSNs is therefore energy efficiency. Other challenges such as small size, low cost etc. may or may not be crucial depending on the specific application. However, without energy efficiency, most of the other design challenges can never be met. A node cannot be made small if it needs a large battery or solar panel to operate. Also, a WSN cannot be made low cost if the maintenance of the network requires battery replacements.

The challenge of making WSNs energy efficient is a task that needs to be dealt with at every level of the network and every step of the design process. Raghunathan *et al*. [1] give an excellent overview of energy efficiency in WSNs. The authors make the important observation that the main consumer of energy in a node with current state of the art technology is the active wireless transceiver. The use of the transceiver must therefore be meticulously controlled so as to prevent energy wastage from prematurely rendering a node useless in the field.

It is generally accepted that the method of energy conservation is to have the transceiver of a node active for only very short periods of time for transmission and reception. During the remainder of the time, the transceiver, or even the entire node, is powered down. This method of operation is referred to as duty cycling and the process of sleep and wakeup cycles is controlled and coordinated among nodes by the Medium Access Control (MAC) layer of the communications protocol stack. The MAC protocol is thus a crucial part of an energy-efficient WSN design.

This paper deals with the design of a new energy-efficient MAC protocol with the goal of extending the useful lifetime of a WSN. The rest of this paper is organized as follows. Section 2 deals with related work in the existing literature. A general overview of existing WSN MAC protocols is given, followed by a closer analysis of the specific protocols that form the design basis for the new protocol. The design of the new MAC protocol is then discussed in Section 3. Section 4 gives the results of the protocol simulation and finally, Section 5 concludes the paper.

2. RELATED WORK

2.1 WSN MAC Protocols Overview

From the current literature pertaining to the topic of WSN MAC protocols, it is clear that there are two main types of WSN MAC protocols, namely schedule-based protocols (e.g. Time Division Multiple Access (TDMA)-type protocols) and contention-based protocols (e.g. Carrier Sense Multiple Access (CSMA)-type protocols). Since it has become clear that WSNs of varying kinds exist and will be devised in the future, it must be kept in mind that different applications will likely require very different hardware and protocols. This is expressly pointed out in a number of articles, such as the extensive WSN overview by Sadler [2].

As a very general classification, it can be stated that the properties of schedule-based MAC protocols lend them suitable for applications where the traffic patterns are deterministic and the topology static. Furthermore they mostly require tight time synchronization among nodes and are thus more suitable for medium to high traffic applications where a relatively constant stream of data is available to keep node clocks synchronized. Lastly they are suitable to be used in networks where the nodes gather into a clustered topology, with the cluster-head being in control of the scheduling functions. A few examples of schedule-based MAC protocols are LEACH [3], TRAMA [4], DE-MAC [5], and EMACS [6].

As far as contention-based MAC protocols are concerned, their properties make them generally suitable for use in bursty traffic scenarios due to the fact that they do not follow a schedule. They are also suitable for use in low traffic scenarios due to their ability to operate at low duty cycles. Traditional contention-based protocols such as the IEEE 802.11 MAC protocol are useful for low latency scenarios. However, such protocols have high energy consumption. To counter this problem, the duty-cycled MAC protocol was introduced. The classical and extensively quoted WSN MAC protocol in this category is S-MAC [7] which has resulted in a number of derivatives and adaptations (e.g. T-MAC [8] or DSMAC [9]). Among contention-based protocols, another group of duty-cycled protocols are those employing preamble sampling to establish communication between two nodes. Some prominent examples of this protocol group are WiseMac [10], CSMA-MPS [11] and B-MAC [12].

2.2 Design Basis

It is apparent from the literature that the specific area of low traffic WSNs (i.e. WSNs where nodes transmit data relatively infrequently) has not been dealt with exhaustively. The preamble sampling family of MAC protocols shows suitability for such WSNs as these protocols can easily operate at low duty cycles.

Low duty cycled contention-based MAC protocols generally operate as follows. Each node in the network has a short listen slot during which time its transceiver is in listening mode to determine whether there is any incoming data from a neighboring node. If not, the transceiver is switched off again until the next listen slot. If there is incoming data, the node continues to receive until the data packet is fully delivered and then switches off the transceiver again. The time between successive listen slots, T_w, is the same for all nodes in a network; however, the listen slot start times are not synchronized among nodes. They are in fact randomly distributed. Therefore, if node A wants to send a data packet to a neighboring node B, it needs to know when this destination node will have its next listen slot. To do this, node A maintains a table with information of its one-hop neighbors. This table contains each neighboring node's address, as well as the time stamp since last communication with that node. From this time stamp and T_w, node A can calculate the time at which node B will have its next listen slot. In an ideal scenario, the sender will thus know the exact point in time at which the intended destination node will wake up and send the data at that instant. The destination node will wake up to listen and immediately start receiving the data. The reason why the ideal scenario cannot be achieved is summed up in the term *clock drift*. There is not a single clock which can maintain perfect time. This applies especially to cheap sensor nodes with cheap quartz crystals as frequency source. A crystal oscillator frequency tolerance of $\pm\Theta$ can cause timing errors on the order of $\pm\Theta L$, where L is the time since last clock resynchronization. Values of Θ for crystal oscillators can be in the region of 10-100 ppm (parts per million).

Preamble sampling protocols overcome the problem of clock drift by sending a preamble before the actual data packet to make sure that the destination's listen slot is "hit". In other words, if node A wants to send a packet to a neighboring node B, it must send a preamble until it is sure that node B has switched on its transceiver and is listening for incoming data. Node B thus notices the incoming preamble during its listen slot and continues to listen until the actual data packet is delivered immediately after the preamble. In this way, the preamble may be viewed as waking up the destination node, enabling the actual data to be delivered reliably.

Some preamble sampling protocols like B-MAC do not keep track of neighboring nodes' listen schedules. In this case, node A starts sending a preamble as soon as data becomes available for transmission to node B. The preamble is sent for T_w seconds, followed by the actual data. In this way, node A is sure that node B had a listen slot during the preamble transmission and continued listening until the data was delivered.

In order to shorten the preamble sending time and thus

save energy, the WiseMac protocol keeps track of neighboring nodes' listen schedules. Node A can thus calculate an estimation of B's next listen slot start, t_{est}, using the time stamp of last communication and T_w as mentioned above. If the maximum frequency tolerance of the nodes' crystal oscillators is given by $\pm\Theta$, the authors of the WiseMac protocol show in [10] that the estimated listen slot start, t_{est}, will be in error by a maximum of $\pm 2\Theta L$ seconds. Therefore, node B may in fact start its listen slot $2\Theta L$ seconds before or after t_{est}. To ensure that node B's listen slot is "hit", node A should start sending the preamble $2\Theta L$ seconds before t_{est} and continue to send it for $2\Theta L$ seconds after t_{est}, a total of $4\Theta L$ seconds. Of course, if the time since last communication, L, becomes large enough so that $4\Theta L > T_w$, then the preamble need only be sent for T_w seconds. In summary, the maximum preamble length in seconds is given by $T_{preamble} = \min(4\Theta L, T_w)$. For appropriate values of L, nodes using the WiseMac protocol can thus save significant energy compared to using a MAC protocol that does not keep track of neighboring nodes' listen schedules.

The CSMA-MPS protocol is based on WiseMac and took some ideas from STEM [13], [14] as well. The authors of CSMA-MPS point out in [11] that the WiseMac protocol sends out a continuous preamble and the preamble length, $T_{preamble}$, is fixed prior to sending the preamble. Thus, even if the destination node, node B, starts its listen slot right at the beginning of the preamble, it still has to listen to the entire length of the preamble to receive the actual data straight afterwards. The STEM protocol takes a different approach, as its preamble consists of short alternating transmit and receive slots. During the transmit slot, a packet containing the destination node's address is sent out. During the receive slot, the source node listens for a preamble acknowledgement (ACK) packet. If no preamble ACK packet is received, the next preamble packet is sent out, and so forth. When node B wakes up for its listen slot and detects an incoming preamble packet from node A, node B responds with a preamble ACK packet to acknowledge its readiness to receive data from node A. Node A can thus stop sending the preamble and instead send the actual data. The CSMA-MPS protocol uses this same preamble format and on average halves the preamble sending time compared to WiseMac. In this way, CSMA-MPS improves on the energy efficiency and network lifetime of WiseMac.

It was noted that even though protocols like WiseMac and CSMA-MPS directly address the problem of clock drift, this problem has always been represented as a single value, Θ, the maximum frequency tolerance of the oscillator hardware. These protocols thus always assume worst case clock drift, whereas the properties and characteristics of the Θ value have not been investigated further. The DPS-MAC protocol discussed in this paper builds specifically on the CSMA-MPS protocol as the most energy efficient existing preamble sampling MAC protocol. Some basic properties of the frequency tolerance value are exploited so as to shorten preamble sending times and allow DPS-MAC to achieve improved energy efficiency compared to its predecessor.

2.3 Clock Drift

Overview: So far, the quality of a clock source has been described by a single value, Θ, the frequency tolerance of the clock source. However, the quality of a clock in fact consists of two distinct concepts, namely *accuracy* and *stability* [15]. The degree to which a crystal oscillator is *accurate* is the degree to which the average frequency at which it oscillates is close to its specified resonant frequency. On the other hand, the degree to which an oscillator is *stable* is the degree to which the frequency output of the oscillator varies from its average value over time. Thus, if a microcontroller uses a quartz crystal oscillator, the timing errors, or clock drift, that can be expected by the microcontroller consists of two parts. Firstly, a fixed frequency offset will cause its time to drift from actual time at a fixed rate, say Θ_{ave}. This is the clock *inaccuracy* and is caused mainly by variations in the exact shape and cut of the quartz crystals at manufacturing time. Secondly, the crystal's frequency output may vary over time due to various sources of *instability*, causing uncertainty in the microcontroller's time reference at any point in time.

A fixed frequency offset and the resulting timing errors that increment at a fixed rate can easily be compensated for if the average drift rate, Θ_{ave}, is known. Therefore, the uncertainty caused by frequency instability presents a greater problem to clock synchronization and the magnitude of this problem is now investigated more closely.

Clock instability: Intuitively, the procedure to evaluate a clock's quality would be to repetitively measure the frequency of the clock and then analyze the data. The average value of all the measurements would indicate the inaccuracy of the clock (fixed offset) and the standard deviation would indicate the instability of the clock (random fluctuations). There is a problem with this approach though, since clock frequency cannot be measured instantaneously, but needs an averaging interval τ in which to count the number of clock cycles so as to calculate the frequency for that interval τ. It turns out that the standard deviation is ill-defined for such a set of measurements as it shows dependency on the length of interval τ and in certain situations, the standard deviation value does not converge to a sensible value [16]. To solve this problem, different measures have been proposed to characterize clock stability. The specification which has been recommended by the IEEE to characterize clock stability in the time domain is referred to as the Allan deviation which is the square root of the Allan variance [17]. It is denoted by $\sigma_y(\tau)$. A detailed explanation of what it represents can be found in [16].

The reason why the Allan deviation is such a useful measure is because it is straightforward to convert the

Allan deviation to the timing errors that can be expected from an oscillator. This is achieved as follows. If a clock is perfectly synchronized and set at a point in time, then τ seconds later, the expected time error of the clock is simply calculated as $\tau\sigma_y(\tau)$ [17]. The two coordinates of a point on the $\sigma_y(\tau)$ plot are thus multiplied together and this gives the remaining time error that can be expected once the average time error, caused by a fixed frequency offset, has been removed. In other words, this value represents the expected time error standard deviation. Using the above procedure, a graph of the expected time error standard deviation for quartz crystal oscillators is generated as shown in Figure 1, by using the data from the $\sigma_y(\tau)$ plot in Figure A1 of [18].

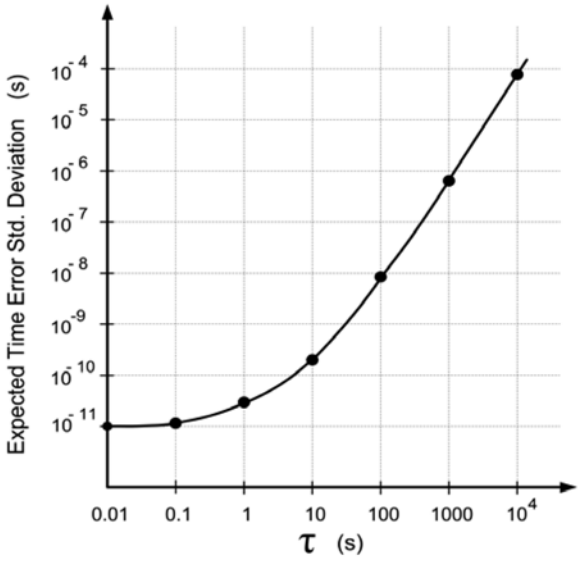

Fig 1: Expected time error standard deviation vs. synchronization interval τ for quartz crystals.

To demonstrate these observations, take as example a microcontroller which implements a clock function. Suppose that the microcontroller's crystal oscillator has an average frequency inaccuracy given by Θ_{ave} = 20 ppm = 20×10^{-6}. At time zero the microcontroller's time is synchronized perfectly to a hypothetical perfect time reference. After τ = 100 seconds, the time error caused by the fixed frequency offset in the crystal is given by $\Theta_{ave} \times \tau = 2\times10^{-3}$ seconds. On the other hand, using the graph in Figure 1 for τ = 100 seconds shows that the time error caused by clock instability is only approximately 9×10^{-9} seconds. In this case, the random time error caused by clock instability is more than five orders of magnitude smaller than the time error caused by the fixed frequency offset. In fact, as seen in Figure 1, up to a resynchronization interval of τ = 1000 seconds, the time error due to instability is on the order of less than 1 µs. As τ increases, processes such as random frequency walk, component aging as well as temperature changes may increase the effects of frequency instability [17].

The observations that have been made regarding clock drift are exploited in the design of the DPS-MAC protocol as discussed next.

3. PROTOCOL DESIGN

As has been mentioned, the Dynamic Preamble Sampling MAC (DPS-MAC) protocol is based on the CSMA-MPS MAC protocol. The three operational states of the DPS-MAC protocol are as follows, keeping in mind that the first two of these states are the same as in the CSMA-MPS protocol.

3.1 State 1 - Unsynchronized

When a node is first powered up, it has no knowledge of the listen schedules of its surrounding nodes. Each node simply begins its own listen schedule upon startup i.e. it enters a short listen period every T_w seconds. For each node, the scheduling of this listen slot is independent of any of its surrounding nodes and there is no synchronization of listen schedules amongst nodes. When such a node, say node A, needs to transmit data to one of its neighboring nodes, it starts the transmission process immediately. The transceiver is first switched on, entering a wake-up (WU) mode until it is fully functional. Then, a carrier sense (CS) operation is done to determine whether the medium is busy or not. If the medium is detected as busy, the transmission attempt is rescheduled after a short random time in the interval $[T_w/2, T_w]$. If the medium is found idle, node A starts sending the preamble. After a maximum of T_w seconds, the destination node, node B, will have had its listen slot and in response sent a preamble ACK packet, telling node A to send the actual data. After receiving the data, node B responds with a final ACK packet. Node B measures the time difference, $\Delta t_{l\text{-}r}$, between when its listen slot started, $t_{listenB}$, and when it actually received the first valid preamble packet, $t_{rxpreambleB}$. This value is returned to node A in the preamble ACK packet. In this way, node A knows exactly the time at which node B started its listen slot, and stores this value with node B's entry in the neighbor table as the last time of communication, t_{lastB}. The next time a communication must occur with node B, node A can operate in the second state.

3.2 State 2 – Slot Estimate Available

Having had a previous successful communication with node B, node A can roughly estimate the time at which node B will have its next listen slot, t_{estB}, by simply adding T_w repetitively to t_{lastB} until a value greater that the current time is obtained. Each node in the network also knows the maximum frequency tolerance of the crystal oscillators used on the nodes, Θ_{max}. As explained previously for the WiseMac protocol, node A will thus start sending the preamble $2\Theta_{max}L_B$ seconds before t_{estB}, where $L_B = t_{estB} - t_{lastB}$. In fact, CSMA-MPS adds a short random time, t_{rand}, to prevent similarly synchronized neighbors from sending data to node B at the same time, causing a collision. This random time is given as $t_{rand} = kT_{rx\text{-}tx}$, where k is a uniform random integer in the interval [0, N], N being the number of entries in the neighbor table, and $T_{rx\text{-}tx}$ is the receive-transmit switching time of the transceiver hardware, a constant. Thus, the

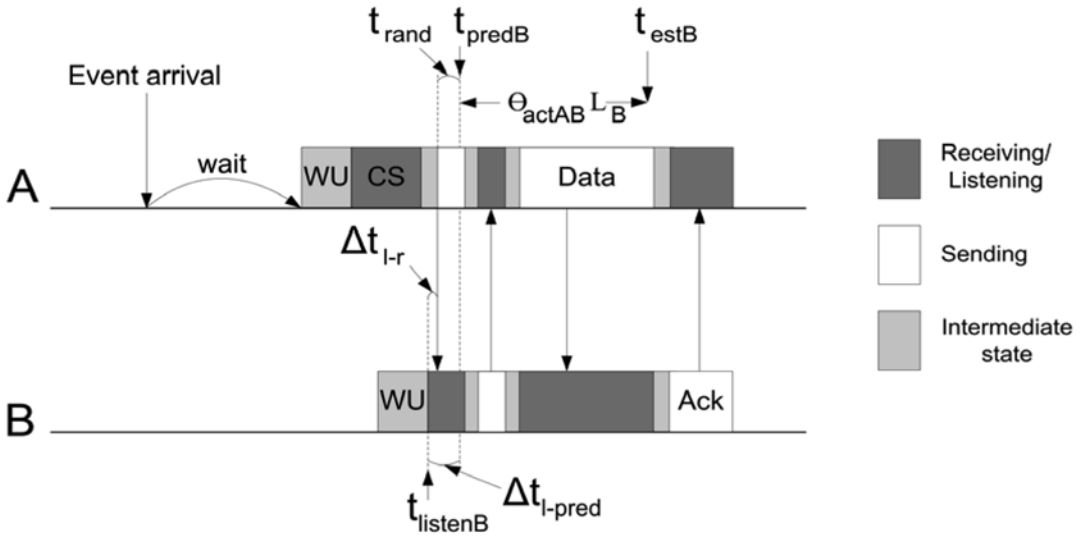

Fig 2: The operation of the DPS-MAC protocol in the third state where a clock drift estimate to the destination node is available.

time at which node A actually starts sending a preamble to B is given by $t_{txA} = t_{estB} - 2\Theta_{max}L_B - t_{rand}$. If before starting the transmission, node A detects the medium busy during the CS operation, the transmission is rescheduled for node B's next listen slot. If the medium is idle, node A starts sending the preamble. Upon reception of the first valid preamble packet, node B again measures the time difference $\Delta t_{l\text{-}r}$ and includes this value in the preamble ACK packet. From this value, node A again calculates the time at which node B started its listen slot, $t_{listenB}$, and stores this again as the last time of communication with node B, t_{lastB}. Furthermore, node A also knows the time at which it originally estimated node B would start its listen slot, t_{estB}. From this point on, DPS-MAC extends the functionality of CSMA-MPS. Node A calculates $\Delta t_{l\text{-}est} = t_{listenB} - t_{estB}$ as the difference in time between when node B actually started listening and when node A estimated node B would start listening. From this, node A can calculate the actual clock drift that occurred between the two nodes in the time since the last communication as $\Theta_{actAB} = \Delta t_{l\text{-}est}/L_B$. This value is stored in node A's neighbor table entry for node B and the next time a communication must occur with node B, node A can operate in the third state.

3.3 State 3 – Clock Drift Estimate Available

From the previous protocol state, node A was able to calculate the actual clock drift that it observed with node B, Θ_{actAB}. As was shown in the section on clock instability, the biggest portion of Θ_{actAB} is in fact a stable value, namely two fixed frequency offsets, one each in node A and node B's crystal oscillators. The Θ_{actAB} value will thus remain stable enough so as to accurately predict node B's next listen slot. To do this, node A again calculates $L_B = t_{estB} - t_{lastB}$, the time since last communication with node B. Node A then predicts node B's next listen slot start as $t_{predB} = t_{estB} + \Theta_{actAB}L_B$ as shown in Figure 2. Note that Θ_{actAB} has a negative value in the example in Figure 2, therefore t_{predB} occurs before t_{estB}. The unstable portion of the Θ_{actAB} value is the sum of the instabilities in node A and node B's crystal oscillator output. As was shown, the uncertainty in each node's oscillator output due to instabilities is on the order of $\tau\sigma_y(\tau)$ seconds. In the worst case, the sum of the uncertainties of both nodes' oscillators will thus be on the order of $2\tau\sigma_y(\tau)$ seconds. As seen in Figure 1, up to a synchronization interval of τ = 1000 seconds, $2\tau\sigma_y(\tau)$ is less than 2×10^{-6} seconds. Therefore, even if nodes A and B last communicated 1000 seconds (approximately 16 minutes) ago, node A can predict the start of node B's next listen slot, t_{predB}, as described above, and only be in error by approximately 2 µs. In terms of a transceiver operating at a bit rate of 1 Mbps, 2 µs is the equivalent of 2 bits being transmitted. It is not expected that WSNs will use transceivers operating at bit rates much higher than 1 Mbps. In fact, most practical implementations so far have operated at much lower bit rates. In this context, a time slot start prediction error of 2 µs can be viewed as insignificant. As τ increases above 1000 seconds, the uncertainty in the listen slot prediction grows larger, due to component aging, random frequency walk etc. and thus, DPS-MAC requires a node to send a very short keep-alive packet to a neighboring node if no communication has occurred with that node for more than 15 minutes.

As is seen in Figure 2, node A again adds a short random time, t_{rand}, to its prediction of node B's listen slot start. The magnitude of t_{rand} is the same as in the second state. Furthermore, when node B has received the first valid preamble packet, it again calculates the time difference $\Delta t_{l\text{-}r}$ and includes this value in the preamble ACK packet sent back to A. Node A calculates the time at which node B started listening, $t_{listenB}$, and node A also knows t_{predB}, the time at which it predicted node B would listen. From these values, A calculates $\Delta t_{l\text{-}pred} = t_{listenB} - t_{predB}$. Since node A already has a drift estimate for node B, the value

of $\Delta t_{l\text{-}pred}$ includes the short term instability of the two nodes' oscillators on the one hand, but also any long term changes in node A and B's oscillator outputs due to component aging, environmental changes etc. and thus, it cannot be ignored. Node A uses the $\Delta t_{l\text{-}pred}$ value to update its clock drift estimate for node B by calculating $\Theta_{actAB} = \Theta_{actAB} + \Delta t_{l\text{-}pred}/L_B/2$. This calculation essentially sets the next value of Θ_{actAB} to the average of its current value and the newly measured value. In this way, DPS-MAC easily deals with long term changes in the crystal oscillators' frequency outputs.

3.4 Summary of Protocol Features

From the above basic operational description, it can be seen that DPS-MAC builds on the existing CSMA-MPS protocol and extends this protocol's energy savings by dynamically adjusting to the clock drift experienced between any two neighboring nodes. No additional protocol overhead is required to sustain this functionality. In fact, DPS-MAC is designed to reduce the length of preambles compared to its predecessors. Furthermore, DPS-MAC is fully distributed and operates autonomously at each node. No clustering is required and nodes are not grouped hierarchically. Also, the protocol does not rely on complex calculations and can be implemented on resource constrained hardware of different kinds.

4. SIMULATION AND RESULTS

The DPS-MAC protocol was compared to its predecessor, CSMA-MPS, using simulation. The comparison was done on the one hand to verify the DPS-MAC protocol design, and on the other hand to demonstrate the energy savings of DPS-MAC over CSMA-MPS, if any. The details of the simulation as well as its results are discussed next.

4.1 Simulation Details

Simulations were performed using the OMNeT++ platform, version 3.2. On top of this, the Mobility Framework simulator, version 1.0-a-6 was used. The state machines of the DPS-MAC and CSMA-MPS protocols were fully implemented in this framework. Every single state in which the transceiver of a node may operate was modeled in the simulation, including startup and intermediate states, transmitting, active receiving, passive listening and off states. In this way, the total time spent by a node in any of the possible communication states at any point in time was accurately measured for each node in the simulation. From these timing values, the energy consumption of each node in the simulation network was calculated.

The simulated network consisted of 50 nodes, randomly distributed in a 400 m × 400 m square area. A sink node was located at the centre of the area, collecting the data from all the nodes in the network. Upon startup, the sink node sent a broadcast packet to all its neighbors, with a hop count (HC) set to zero. Each of the recipients incremented the received HC by one and then forwarded the packet to its own neighbors, away from the sink node. At the end of this startup procedure, each node knew its distance in hops from the sink node. During normal operation, whenever a node had a packet to transmit (either generated by itself or received from an upstream neighbor) it forwarded the packet to the downstream neighbor (a neighbor with smaller HC) whose next listen slot occurred soonest. In this way, data always travelled towards the sink node and routing loops were prevented.

The simulation provided no external synchronization amongst the nodes. Each node chose a random point in time to start its listen slot schedule upon startup. From then on, the communication process occurred as explained in the above description of the two protocols' states of operation. The listen slot period, T_w, was set at 10 seconds, whereas each node was simulated as generating a packet regularly every 15 minutes. The simulation duration was set to 1 day.

The transceiver model used in the simulations is based on the CC2400 radio from Chipcon [19]. The transceiver was simulated to operate at a bit rate of 1 Mbps.

4.2 Results

Numerous simulation tests were performed to demonstrate the performance of DPS-MAC in comparison to CSMA-MPS under various operating conditions. The test which demonstrates most clearly the difference between the two protocols is a simulation using various maximum frequency tolerance values, Θ_{max}, for the crystal oscillators of the nodes in the network. Values of Θ_{max} between 10 ppm and 80 ppm were simulated.

The graph showing the average transmitting time per node for the two simulated protocols for various values of the maximum frequency tolerance is shown in Figure 3 below.

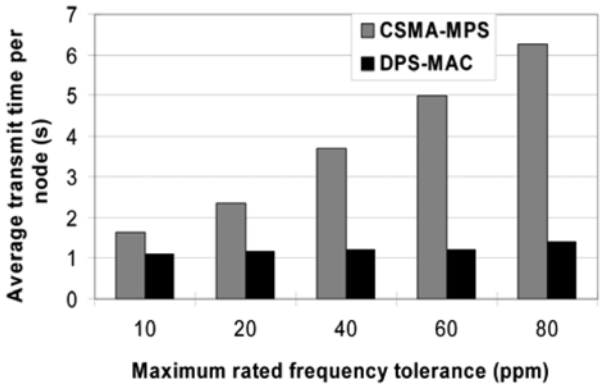

Fig 3: Comparison of the average transmit time per node for CSMA-MPS and DPS-MAC for various maximum frequency tolerance values. The simulation time is 1 day.

CSMA-MPS always starts sending a preamble $2\Theta_{max}L$ seconds before the neighbor's listen slot is estimated to start, where L is the time since last communication. Therefore, intuitively, the larger the value of Θ_{max}, the longer the preamble transmission will be. DPS-MAC on the other hand dynamically adjusts its predictions of the neighbor's listen slot to the actual observed clock drift and not the worst case clock drift. The transmission time is therefore largely independent of Θ_{max}. These expectations are confirmed by the results in Figure 3.

Taking into account the duration of the transceiver in each of the states, the average power consumption per node can be obtained as shown in Figure 4 below.

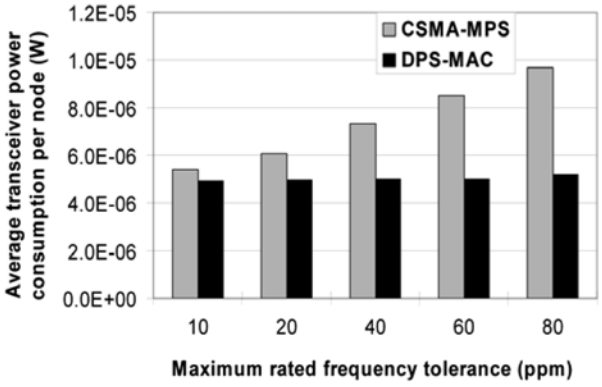

Fig 4: The average transceiver power consumption of DPS-MAC and CSMA-MPS for various maximum frequency tolerance values.

Finally, the transceiver energy savings of DPS-MAC over CSMA-MPS is given in Figure 5 below. It is clear that as the quality of the clock increases (i.e. the frequency tolerance rating decreases), the energy savings of DPS-MAC over CSMA-MPS decrease. However, even when using high quality crystal oscillators with frequency tolerance ratings of around 20 ppm, DPS-MAC still demonstrates an increase in transceiver energy savings of approximately 18 % over CSMA-MPS.

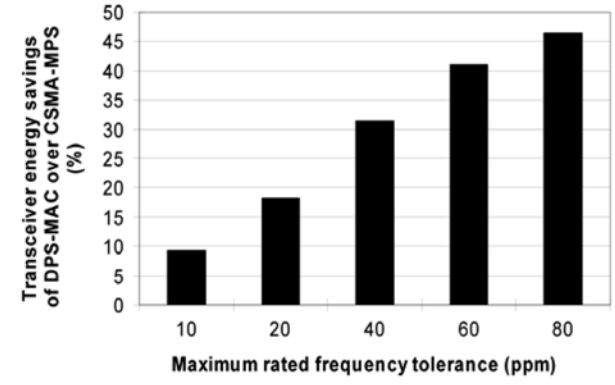

Fig 5: The transceiver energy savings of DPS-MAC over CSMA-MPS for various maximum frequency tolerance values.

5. CONCLUSION

A preamble sampling MAC protocol for WSNs, called DPS-MAC, has been presented in this paper. By exploiting some characteristics of the frequency tolerance ratings of crystal oscillators, DPS-MAC is able to cause a significant reduction in the duration of preamble transmission compared to its predecessors. DPS-MAC is aimed specifically at application in low traffic WSNs and can be used to achieve longer network lifetimes in such WSNs, in this way contributing to making such networks more feasible.

6. REFERENCES

[1] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivastava: "Energy-aware wireless microsensor networks," *IEEE Signal Processing Mag.*, vol. 19, no. 2, pp. 40-50, Mar. 2002.

[2] B. M. Sadler: "Fundamentals of Energy-Constrained Sensor Network Systems," *IEEE A&E Systems Mag.*, vol. 20, no. 8, pp. 17-35, Aug. 2005.

[3] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan" "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *2000 Proc. Ann. Hawaii International Conference on System Sciences*, pp. 1-10.

[4] V. Rajendran, K. Obraczka and J. J. Garcia-Luna-Aceves: "Energy-efficient, collision-free medium access control for wireless sensor networks," in *2003 Proc. SenSys Conf.*, pp. 181-192.

[5] R. Kalidindi, L. Ray, R. Kannan and S. Iyengar: "Distributed Energy Aware MAC Layer Protocol for Wireless Sensor Networks," in *2003 Proc. International Conf. on Wireless Networks*, pp. 282-286.

[6] L. F. W. van Hoesel, T. Nieberg, H. J. Kip and P. J. M. Havinga: "Advantages of a TDMA based, energy-efficient, self-organizing MAC protocol for WSNs," in *2004 Proc. IEEE Vehicular Technology Conf.*, vol. 3, pp.1598-1602.

[7] W. Ye, J. Heidemann and D. Estrin: "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *2002 Proc. INFOCOM Conf.*, pp. 1567-1576.

[8] T. van Dam and K. Langendoen: "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *2003 Proc. SenSys Conf.*, pp. 171-180.

[9] P. Lin, C. Qiao and X. Wang: "Medium access control with a dynamic duty cycle for sensor

networks," in *2004 Proc. IEEE WCNC Conf.*, vol. 3, pp. 1534-1539.

[10] A. El-Hoiydi and J.-D. Decotignie: "WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks," in *2004 Proc. ISCC Symp.*, pp. 244 – 251.

[11] S. Mahlknecht and M. Böck: "CSMA-MPS: a minimum preamble sampling MAC protocol for low power wireless sensor networks," in *2004 Proc. IEEE Workshop on Factory Comm. Systems*, pp. 73-80.

[12] J. Polastre, J. Hill and D. Culler: "Versatile low power media access for wireless sensor networks," in *2004 Proc. SenSys Conf.*, pp. 95-107.

[13] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava: "Optimizing Sensor Networks in the Energy-Latency-Density Design Space," *IEEE Trans. on Mobile Computing*, vol. 1, pp. 70-80, March 2002.

[14] C. Schurgers, V. Tsiatsis and M. B. Srivastava: "STEM: Topology management for energy efficient sensor networks," in *2002 Proc. Aerospace Conf.*, pp 3-1099 - 3-1108.

[15] J. R. Vig and A. Ballato: *Ultrasonic Instruments and Devices*. Academic Press Inc., USA, 1999, ch. 7.

[16] D. W. Allan: "Time and Frequency (Time-Domain) Characterization, Estimation, and Prediction of Precision Clocks and Oscillators," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. UFFC-34, no. 6, pp. 647-654, Nov. 1987.

[17] J. R. Vig: (2006, April). Quartz Crystal Resonators and Oscillators for Frequency Control and Timing Applications - A Tutorial (Rev. 8.5.2.3) [Online]. Available: http://www.ieee-uffc.org/freqcontrol/tutorials/vig2/tutorial2.ppt

[18] D. W. Allan, N. Ashby, and C. C. Hodge: "The Science of Timekeeping," Hewlett Packard Company, Application Note 1289, Jun. 1997.

[19] Chipcon AS. (2006, March). CC2400 datasheet (Rev. 1.5) [Online]. Available: http://www.chipcon.com