This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Secon 2010 proceedings.

MCMAC: An Optimized Medium Access Control Protocol for Mobile Clusters in Wireless Sensor Networks

Majid Nabi*, Milos Blagojevic*†, Marc Geilen*, Twan Basten*† and Teun Hendriks†
*Department of Electrical Engineering, Eindhoven University of Technology, the Netherlands
Email: {m.nabi,m.c.w.geilen,a.a.basten}@tue.nl
†Embedded Systems Institute, Eindhoven, the Netherlands
Email: {milos.blagojevic,teun.hendriks}@esi.nl

***Abstract*—Wireless sensor networks (WSNs) are developing into a promising solution for many applications, for example in healthcare. In many scenarios, there is some form of node mobility. The medium access control (MAC) mechanisms should support the expected kind of mobility in the network. Mobility is particularly complicating for contention free MAC protocols like TDMA-based protocols, because they dedicate unique slots to every node in a neighborhood. In scenarios such as body-area networking, some clusters of nodes move together, creating further challenges and opportunities. This paper proposes MCMAC (Mobile Cluster MAC), a TDMA-based MAC protocol to support mobile clusters in WSNs. The proposed protocol does not need adaptation time after movement of clusters. Several optimization mechanisms are proposed to decrease power consumption. Simulation results show that the optimizations decrease power consumption of nodes around 70% without increasing latency of data transmission compared to the non-optimized version.**

I. Introduction

Wireless sensor networks are going to be used as a key solution for many applications. Healthcare, wild life monitoring, disaster management, and agriculture automation are some fields in which WSNs can help a lot to deal with their respective problems. Wireless sensor nodes are expected to develop into tiny, light and inexpensive devices to be deployed easily on a large scale. These nodes have limited energy resources. Thus, power consumption is a real challenging problem in designing an efficient WSN architecture.

Wireless technology makes the sensor nodes easy to deploy and gives them the opportunity to be mobile. The ability of sensor nodes to move provides more interesting applications for WSNs. However, network protocols should take mobility into consideration and have appropriate mechanisms for managing mobility. The MAC layer is the layer that is primarily responsible for managing mobility as it should control the access to the shared media. The problem is most prominent in schedule-based protocols such as TDMA-based MAC protocols. In these protocols, a specific time slot is dedicated to every wireless node for accessing the media of the wireless channel. To have a contention-free protocol, the dedicated time slot has to be unique in its neighborhood. So assigning time slots to sensor nodes should be done taking the location of nodes and also the range of wireless transmitters into account. When a node moves, it is clear that there should be some dynamic adaptation mechanism to assign new access time slots to mobile nodes in a new neighborhood.

The L-MAC protocol [1], [2] is a TDMA-based protocol with a dynamic mechanism for occupying time slots. Every node receives some meta-data from its neighbors detailing which slots are occupied by the one-hop neighborhood of the sending node and based on that information, it selects a free transmit slot. Thus, the selection is done in a distributed and dynamic way. There is also some method for detecting new nodes in a neighborhood that are interfering with other nodes. The problem here is that detecting movement and occupation of new slots takes time, which strongly affects the quality of service (QoS) of a running application. In fact, the L-MAC protocol is appropriate for occasional mobility of nodes, but not for continuous mobility. In [3], the authors present a protocol to support mobile nodes in a TDMA-based MAC protocol. In this method, the communication with mobile nodes is done with a contention-based mechanism while other nodes communicate with each other using a schedule-based mechanism. This method is designed for supporting mobility of singular nodes that send data when an event occurs.

In this paper, we address a specific kind of mobility in WSNs in which several nodes are moving together as a cluster (group). There are many examples of such kind of mobility in reality. A wireless body area network (WBAN) is a prominent example. A group of nodes is moving together in the network while there is also individual mobility of nodes inside the cluster. In such cases, managing the mobility is really important. The method has to be fast because the nodes are assumed to move like humans do. On the other hand, application-level constraints on QoS metrics like latency, reliability, and life time (power consumption) have to be met. MAC protocols proposed in [2], [3] do not have a direct support for this form of mobility.

We categorize the sensor nodes in two major classes. The first class, the *static network*, contains the static nodes. These nodes are supposed to be static after deployment or have limited mobility. The latter means that these nodes are normally static but they can be relocated, after which it may take some time to become stable at the new location and communicate to other nodes. The current algorithms of MAC protocols like L-MAC are able to deal with this kind of

978-1-4244-7151-5/10/$26.00 ©2010 IEEE

mobility. The second class contains the *mobile cluster* nodes. There can be several mobile clusters, each with multiple nodes, in a network. Clusters should be able to communicate with other nodes while moving without requiring recovery time.

The next section describes a concrete application scenario for our proposed protocol, although the scope of the protocol is not limited to this specific scenario. A mobility model for modeling the movement behavior of clusters is introduced in Section III. This model is used in our evaluation. Section IV describes the basic MCMAC (Mobile Cluster MAC) protocol covering the proposed method to deal with the mobile clusters in the network. Optimization methods for both static nodes and cluster nodes are explained in Section V. The protocol evaluation result are given in Section VI. Section VII concludes.

II. Application Scenario

There are several potential applications in which clusters of nodes are moving within a static wireless sensor network. Healthcare applications are important examples. Scenarios are envisioned in which patients are equipped with several special sensors on their body, yielding a wireless body area network (WBAN). Body sensors measure specific biological parameters of the body. Care workers can then monitor the status of the patient and consequently take required action. It is also possible to provide some useful online information for the patient, and the WBAN may be augmented by a static network of sensors. The sensors and the exact scenario are selected according to the condition that a patient may have.

We specifically consider elderly care or patient care with some special chronic diseases in the home situation. Chronic obstructive pulmonary disease (COPD) is, for instance, a disease in which the patient needs intensive attention about the amount of performed activity in a specified amount of time. In this scenario, the patient's home is equipped with ambient sensors, that are wireless for ease of installation and for deployment on for example furniture. These sensors are installed on the walls to measure ambient parameters like temperature and humidity. Sensors may be installed on chairs and beds to monitor the activity pattern of the person. These sensor nodes may move but with limited mobility.

On the other hand, the patient is equipped with a WBAN. The sensors are placed on several positions on the body based on the sensor type to measure different vital biological signals. A temperature sensor (for fever), light sensor, and also a microphone can be installed on the head. ECG sensors are mounted on the body to measure the heart status. A blood pressure sensor can be installed on the arms, and the SpO2 respiration sensor may be installed on the hand. A GSR (Galvanic skin response) sensor may also be mounted on the hand to monitor the emotional status of the patient. Accelerometer sensors are used to measure the movement as well as the tilting of the body. Fig. 1 shows a typical WBAN. A subset of these sensors may be selected based on the condition that patients are facing. The sensor nodes are forming a heterogeneous mix with different sampling time requirements. Some sensor types (like an ECG sensor) need a data sampling rate that is higher than the feasible delivery rate

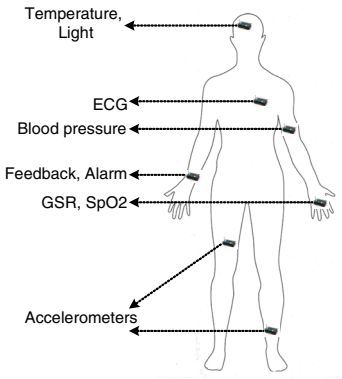


Fig. 1. A typical wireless body area network (WBAN).

in a typical multi-hop network. In such a situation, the sensor node performs some processing and then sends the conclusion in a periodic manner.

Normally, WBAN nodes are tiny and light, and so the power consumption should be very low. Wireless transmission range is set as low as possible to reduce power consumption. Moreover, it is assumed that there is not a more powerful node in the WBAN for doing computation and communication. All nodes in the WBAN are assumed to have a similar battery capacity and computation capability.

Some sink nodes in the static part of the network collect and process information of ambient sensors as well as data from body sensors. Sink nodes can also send information to a medical center through a wired network, receive feedback, and inform the patient. The number of sinks can vary based on the house or building. A reason for having more than one sink is to have better latency and reliability. In a typical deployment, we can put for example a sink on every floor of the building.

We use a gossiping strategy [4] as a routing protocol for multi-hop data transmission to the nearest sink node. In this protocol, every node has a cache that can store several data items from different sensor nodes in the network. In every round, a node randomly selects some data items from its cache plus its own sensed data and then propagates the packet to its neighbors. Once a node receives a data item, it is pushed into the cache if the time stamp is newer than the existing one. Gossiping mechanisms provide a reliable data dissemination because of the built-in redundancies. With gossiping, the network does not rely on some specific routing nodes for data dissemination. For instance, after a node failure, the network will remain alive because other nodes are also participating in data routing. To run a gossiping process, we need a fitting MAC layer (see Section IV).

III. Mobility Model

The mobility model plays an important role in the accuracy of simulations for wireless ad hoc and sensor networks with mobility in the network. Mobility models try to mimic the behavior of the mobile nodes in reality through characterizing stochastic patterns of the node movement. The right mobility model strongly depends on the application and the type of wireless nodes.

In our application, we have several nodes in the WBAN that move together in the fashion of a cluster. We need a mobility

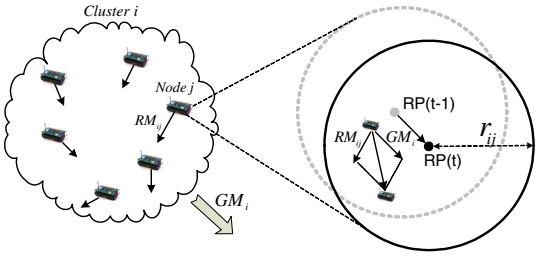


Fig. 2. Reference point group mobility model.

model that is able to model the cluster movement as well as individual movement of nodes within the cluster. Reference Point Group Mobility (RPGM) [5] is a mobility model that has these features and can be adapted for many applications. In this model, every group of nodes has a logical center (LC), the motion of which defines the entire group movement. Every group i has a group motion vector $\overrightarrow{GM}_i$ that determines the motion of the group's LC.

A reference point (RP) is assigned to each node in the cluster that follows the group motion. The reference point of every node moves with the group motion vector $\overrightarrow{GM}_i$. Every node j is moving within a predefined area (circle with radius r_{ij}) around its reference point with a random motion vector $\overrightarrow{RM}_{ij}$. So the reference point allows independent motion of individual nodes in the group while the logical center provides the group movement. The location of every node in each time step is defined by adding the group motion and the random motion vectors. Fig. 2 illustrates the situation for a group.

The definition of RPGM is quite general. By selecting a proper group motion vector, we can model the human movement, while setting the random motion vectors ($\overrightarrow{RM}_{ij}$) defines the motion of individual sensors installed on various positions on the body. For moving the logical center of a cluster, we use an adapted version of the Random Waypoint Mobility (RWM) model [6], [7] which is a commonly used mobility model for individual movement in ad hoc wireless networks. In each step, a random position is selected uniformly within the deployment area. A velocity value v_i is selected and the cluster moves towards the destination with the selected velocity. We assume a minimum v_{min} and a maximum v_{max} possible velocity value, which is based on the human walking speed. Another important decision is that, once the logical center of the cluster reaches the destination point, a uniformly chosen random waiting time w_i is selected, considering a maximum waiting time w_{max}. So the cluster stays for a while in its location, selects the next destination and corresponding velocity, and starts to move again. Algorithm 1 shows the movement of the logical center of a cluster in the network. Notation Δt stands for the time steps for moving the logical center. $U(x, y)$ denotes a uniformly randomly selected value between x and y. $U(deployment\ area)$ represents a uniformly random position selected from the deployment area of the network.

For individual movement of sensor node j in cluster i, we can set two parameters, which are the velocity (v_{ij}), and the radius (r_{ij}) of the circle around its reference point. The decision is made based on the location of that node. For example, the sensor nodes on the chest have no individual mobility and so the velocity is set to zero while the sensor nodes on the hands have the most movement.

Algorithm 1: The movement of the logical center (LC) of cluster i.

```
1  while True do
2      v_i = U(v_min , v_max);
3      dest = U(deployment area);
4      Ns = ||dest − LC_i|| / (v_i × Δt);   step = (dest − LC_i) / Ns;
5      while Ns > 0 do
6          LC_i = LC_i + step;
7          wait for Δt sec.;
8          Ns = Ns − 1;
9      end
10     wait for w_i = U(0 , w_max);
11 end
```

We implemented this mobility model as an add-on to the mobility framework of MiXiM [8] on top of the OMNeT++ 4 [9] network simulator. This implementation allows to have any number of clusters by instantiating an adequate number of RPGM modules. The number of nodes inside each cluster can also be set and the movement parameters can be adjusted individually. Moreover, the mobility pattern of every cluster is logged. Accordingly, the implementation has the option to read the mobility pattern from the logged file instead of using Algorithm 1. Doing so, we can run various configurations of the network with the same mobility pattern to have a more precise comparison.

IV. The MCMAC Protocol

The Dutch company Chess has developed a WSN architecture ([10]) using an epidemic (gossip) communication model. The MAC layer, gMAC [11], has been designed based on the L-MAC protocol. It deviates from the L-MAC only in its addressing scheme. In gMAC packets, no destination address is included and packets are broadcast to all receiving neighbors. This makes the protocol suitable for running the gossip-based routing mechanism. We use this TDMA-based MAC protocol as the basic for our protocol. In this section, we propose our basis MCMAC protocol that incorporates a method to deal with the mobility that we have in our application scenario. Optimizations are discussed in the next section.

A. Basic MAC protocol

In the gMAC protocol, each TDMA frame is divided into equal sections called slots. Nodes are active in the active part of frames and are in sleep mode in other slots to reduce power consumption. A node transmits its data in exactly one slot and listens in other slots in the active part of the frame. Each node occupies a transmit (Tx) time slot to send its data packets. If the selected slot is unique in the node's two-hop neighborhood, the protocol is contention free. Fig. 3 depicts a time frame in this MAC protocol.

Wireless nodes are synchronized using a decentralized frame synchronization mechanism [12]. In every frame, each node computes its phase difference to any of its direct neighbors and based on that adjusts its clock. A guard time is inserted at the start and the end of every transmit slot to resolve

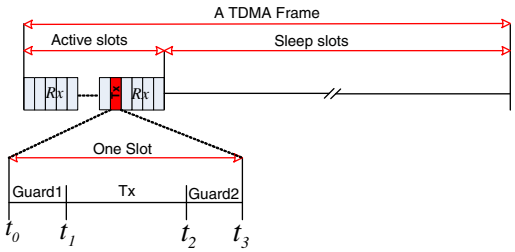


Fig. 3. A TDMA frame in the basic gMAC protocol.

small phase differences (Fig. 3). Every receiving node starts to listen to the slot from t_0 until it receives the whole packet or reaches the end of the slot time (t_3). The transmitter sends its packet between t_1 and t_2.

Tx slot selection is done dynamically in each node using the method introduced in the L-MAC protocol [1]. The MAC header of every data packet contains a bit-set detailing which slots are occupied by the one-hop neighborhood of the sending node. A new node in a neighborhood can detect the free slots by OR-ing the received occupancy bit-sets in a whole frame. The node then randomly selects a slot among the free slots and claims it by broadcasting its occupancy bit-set. When a node moves, it causes interference until the nodes in the new neighborhood detect the collisions and try to find a proper slot for transmission. There are some methods for detecting collisions. For instance, every node can occasionally stop transmitting in its Tx slot and listen to that slot. If the node receives data in its slot, it means that the slot is being used by other nodes.

In this way the gMAC and L-MAC protocols already support mobility. However, the detection of movement (collisions) and then occupying the new Tx slot takes time and the mobile nodes interfere with other nodes for a while in between. So the current protocol is suitable only for mobile nodes with a (very) limited mobility. For instance, in our application, the sensor nodes that are mounted on furniture can move and use this method to update their Tx slot selection. The current method can definitely not deal with a moving cluster like a WBAN in which nodes are mobile (a human) and several nodes move together at once. This is what we address in this paper. We need a method for reliable communication between wireless sensor nodes both in the static part of the network and in mobile clusters. The method needs to have a very short adaptation delay when a cluster moves.

B. The basic MCMAC protocol supporting mobile clusters

The main idea behind supporting mobile clusters in our Mobile Cluster MAC (MCMAC) protocol is to dedicate a part of the active slots to the mobile clusters existing in the network. We split the active slots of every frame into two separate parts. The first one are the static active slots (*SAS*) which are used by the static nodes to communicate with each other. These slots are occupied dynamically by the static nodes with the same strategy used in the basic gMAC protocol. The number of slots in this part ($|SAS|$) depends on the density of the network because every node has to have a unique Tx slot in its two-hop neighborhood.

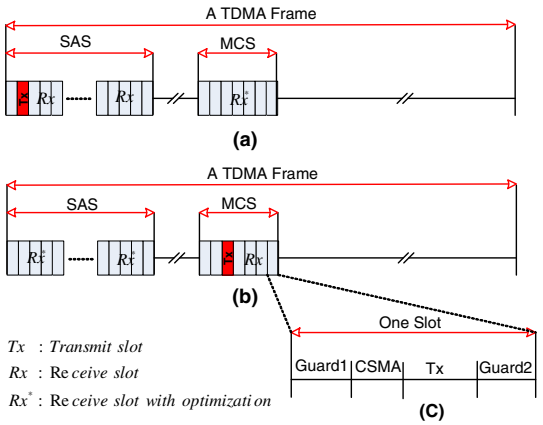


Fig. 4. (a) A TDMA frame for the static nodes (b) A TDMA frame for the cluster nodes (c) The architecture of a time slot in the *MCS* part.

The second part is dedicated to mobile clusters, the mobile cluster slots (*MCS*). This part can be consecutive to the *SAS* part or somewhere else in the time frame. Nodes in the mobile cluster communicate with each other via the slots in the *MCS* part. Each slot in this part is assigned to exactly one node in the cluster for transmission. So the number of slots is equal to the maximum number of nodes in the clusters. As the considered clusters are rather small (human body), all sensor nodes in a cluster are typically within each other's one or two-hop neighborhood, so assigning a fixed slot to every node is appropriate.

The two parts of the network need to communicate. Static nodes should receive data from cluster nodes to forward it to a sink node as fast as possible. This is the most important aspect of the network in our application scenario. This can be done by static nodes through listening to the *MCS* part of the frame as well. Thus, a static node listens to the *SAS* part to communicate with other static nodes and to the *MCS* part for receiving data from cluster nodes. Notice that static nodes just transmit in one occupied slot in the *SAS* part. Fig. 4(a) shows a TDMA frame for a static node.

It is needed for cluster nodes to receive information from static nodes as well. Receiving commands or alarms from sink nodes are examples. Also, to remain synchronized, nodes in the clusters need to listen to the static nodes. Furthermore, by receiving information from static nodes, the overall latency of the network can in same cases be improved because mobile clusters can piggy back information while they are moving. The cluster nodes receive data from static nodes by listening to the *SAS* part as well as the *MCS* part. Fig. 4(b) shows a time frame from a cluster node perspective.

When there is more than one cluster in the network, they are sharing the *MCS* part, possibly causing interference when two clusters are in a neighborhood. Having more than one *MCS* part in a TDMA frame causes power consumption overhead. Therefore, we have just one *MCS* part in every frame. The problem is limited when clusters are moving independently, but when several clusters stay together for a while, it may cause reduction of bandwidth.

We use a hybrid contention-based and schedule-based channel access mechanism to deal with the presence of multiple

clusters in the network. There are some protocols in the literature like Z-MAC [13] that use such a hybrid approach, but none of these is designed to support mobile clusters. In our MCMAC protocol, besides that every cluster node has a dedicated transmit slot in the *MCS* part, it performs a simple carrier sense multiple access (CSMA) mechanism for accessing the radio channel in its slot. The duration of a time slot is extended to include a CSMA period. This CSMA period is used for carrier sensing and is located exactly after the guard time (Fig. 4(c)) of *MCS* slots. Every node in the cluster selects a random time in this period and then senses the channel. If the radio channel is idle, the node starts to transmit data. So the real transmission starts from somewhere in the CSMA period. Note that there is a switching delay from carrier sensing mode to the real transmission. So in this period, there is a possibility that other competing nodes sense the channel and find it clear while another node is preparing for transmission. Collisions can also occur due to the well known hidden terminal problem. These are actually native problems of CSMA protocols.

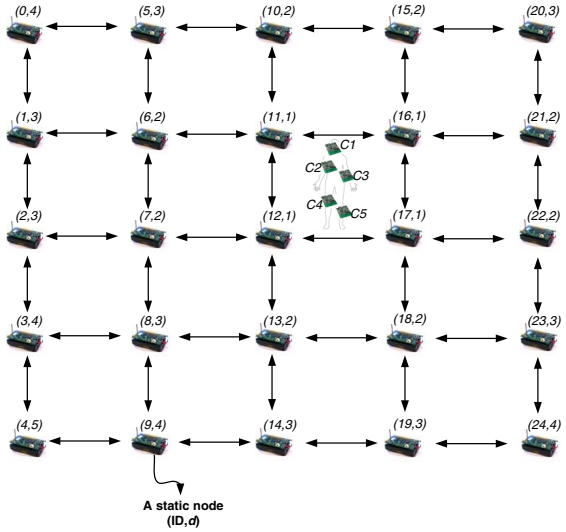


Fig. 5. A 5×5 node static wireless sensor network with one mobile cluster including 5 cluster nodes.

V. Quality of Service Optimizations

A. Optimization for static nodes

Based on our method for supporting moving clusters in our TDMA-based MCMAC protocol, every static node has to listen to some extra slots in every TDMA frame. By this, static nodes receive data from sensor nodes in the clusters and then they route the data to the sink nodes. On the other hand, a very important challenge in designing protocols for WSNs is power consumption because typically sensor nodes are tiny with limited battery capacity. Suppose that we have 16 slots in the *SAS* part of each TDMA frame ($|SAS| = 16$) and there are 5 slots in the *MCS* part ($|MCS| = 5$). In this situation the power consumption overhead for cluster mobility support will be around 30% which is really challenging. We propose an optimization algorithm that considerably decreases the power consumption overhead.

The idea is that the static nodes should listen to the MCS part of the frame just when there is a mobile cluster in their neighborhood. When there is none, listening is just a waste of energy. We propose a method with which the static nodes can estimate the hop distance (d) to the nearest cluster. The value d is added to every packet. We dedicate four bits for this parameter in every data packet and so the maximum value (d_{max}) is 15. This maximum bound is big enough for typical networks sizes. Cluster nodes always send zero as the hop distance. So every static node that receives zero in a round, detects that there is already a cluster in its neighborhood. Moreover, every node receives several d values in a round from its neighbors and then selects the minimum value plus one as its own d. If a node did not receive any packet in a round, the last d value is considered. In the next round, the value d will be added to the packet as the hop distance and sent to the neighbors. Using this method, every node can also predict the movement of clusters in term of the variations of the distance d. Fig. 5 shows the value of d for all nodes in a given 5×5 static network with a mobile cluster.

Now, a static node does not always listen to the MCS part. Algorithm 2 illustrates the behavior of a static node in the MCMAC protocol. The function *StaticOpt()* decides whether the node will be listening to the *MCS* part based on the calculated hop distance to the nearest cluster (d). Every node has a variable listening interval time T_l that is adjusted according to the value of parameter d in that node. A node listens to the MCS part (*StaticOpt()* returns *true*) just for one round in every T_l round(s). The process of adjusting the listening interval is important because it affects the latency of data items from cluster nodes, which is a vital constraint in our application scenario. The adjustment process should balance two aspects. First, to minimize idle listening when there is no mobile cluster around. Second, to minimize the packet loss and latency caused by not listening to the MCS part, when there is a cluster in the neighborhood. While it is important to minimize the latency of data items from the cluster nodes, the actual goal is to adjust the listening interval time to reach an optimal trade-off considering two QoS objectives; latency and power consumption. Our experiments show that care is needed for this process. Otherwise, it will worsen the latency of the data items from cluster nodes. With the implemented method, we decrease power consumption overhead by 70% without exacerbating the latency compared to our basic protocol without optimizations.

We use the Additive Increase Multiplicative Decrease (AIMD) [14],[15] mechanism for adjusting the value of T_l. To perform the AIMD, three parameters are considered; a limited history of d, the current value of d, and its last change (Δd). Because of the varying packet reception ratio of the radio channel, there are variations for parameter d over time even when a cluster is not moving. To reduce the effect of such variations and to separate this effect from a real change of the value of d caused by the movement of a cluster, we use an exponentially weighted average of a limited history of H TDMA rounds of the parameter d in every node (Eqn. 1). In fact, the used method for adjusting the time

Algorithm 2: Behavior of every static node.

```
for every TDMA frame do
    Lst ← StaticOpt(d);
    for i = 1 to |SAS| do
        if i ≠ ownTxSlot then
            RxPacket ← Listen() ;
            if Received any then Rxd[i] = RxPacket.d;
        else
            TxPacket.d ← d;
            Transmit(TxPacket);
        end
    end
    if Lst then
        for i = 1 to |MCS| do
            Listen();
            if Received any then Rxd[0] ← 0;
        end
    else
        Go to the sleep mode until the end of the MCS part;
    end
    d ← min{Rxd[i] | i : 0 ≤ i ≤ |SAS|} + 1;
    Go to sleep mode until the next frame
end
```

Algorithm 3: Behavior of a cluster node.

```
for every TDMA frame do
    if U(0,1) < N_i^l / N_i then
        for i = 1 to |SAS| do Listen()
    else
        Go to the sleep mode until the start of the MCS part;
    end
    for i = 1 to |MCS| do
        if i ≠ ownTxSlot then
            Listen();
        else
            t ← Random time in CSMA period;
            Sense the channel until time t;
            if SenseChannel() = idle then
                Transmit();
            else
                Listen();
            end
        end
    end
    Go to sleep mode until the next frame;
end
```

interval is proposed based on several experiments in various circumstances and with taking the variations of parameter d over time into account.

$$avg(t) = \frac{\sum_{k=1}^{H} a_k d(t-k)}{\sum_{k=1}^{H} a_k} \quad , \; a_k = (\frac{1}{\alpha})^k \quad , \; 1 \leq \alpha \leq 2 \quad (1)$$

$$\delta_d = d(t) - avg(t) \quad (2)$$

$$\Delta d = d(t) - d(t-1) \quad (3)$$

where $avg(t)$ stands for the weighted average of d over H previous rounds before round t and $d(t)$ is the received d value at round t. δ_d returns the change of d with respect to the weighted average. The last variation of d is Δd, that gives the change with respect to the previous round. Eqn. 4 and 5 show the AIMD method of adjusting the listening interval based on the above parameters. T_l' is the newly calculated listening interval.

$$T_l^* = \begin{cases} T_l + d(t) & \Delta d > 0 \\ \lfloor \frac{T_l}{2^{\lceil \delta_d \rceil}} \rfloor & \Delta d < 0 \\ T_l + 1 & \Delta d = 0 \end{cases} \quad (4)$$

$$T_l' = min\{T_l^* \; , \; T_l^{max}[d]\} \quad (5)$$

We consider a maximum value $T_l^{max}[d]$ for the listening interval when there is a cluster in a d-hop neighborhood. The proper value for the maximum values depends on the density of the network and the average speed of the clusters. The $T_l^{max}[1]$ is set to one so that a node listens to the *MCS* part in all rounds when there is a cluster around. Moreover, assume we want to add a new cluster to a running network. By considering the maximum value, in the worse case, it takes $T_l^{max}[d_{max}]$ rounds for the network to detect the new cluster and to start listening to its nodes.

B. Cluster node optimizations

In our basic MCMAC protocol, cluster nodes have to listen to the *MCS* part for communication with other nodes in the cluster as well as listen to the *SAS* part to receive information from the static part of the network. It is possible to reduce the power consumption of cluster nodes while satisfying latency constraints. To do so, we reduce the listening to the *SAS* part of the frame. The considerations here are firstly to fairly distribute the overhead of listening to the *SAS* part among all existing nodes in the cluster. Secondly, to have some nodes listening to the *SAS* part in every round. In fact, we want to have a percentage of nodes in the cluster to listen to *SAS* part in every round and we rotate this task among all nodes to distribute power consumption overhead among all.

The literature on WSNs describes some methods for cluster head election. The LEACH method [16] is a commonly used method which tries to elect cluster heads in every round. The goals are similar to our goals and the algorithm is fully distributed. In every round, a node decides to be head in the next round with a probability taking the history of being head in previous rounds into account. In LEACH, however, there is no guarantee of having at least one head in every round. Furthermore, a big difference to our problem is that one round in our problem is a TDMA frame, whereas in LEACH, a round is a relatively long time to the next reconfiguration of the network.

Here we use a method in which the decision is done based on uniform probabilities. Suppose that we have N_i nodes in the i^{th} cluster and the desired number of heads that listen to the *SAS* part is N_i^l. So in every round that $U(0,1) < \frac{N_i^l}{N_i}$, the node listens to the *SAS* part. By selecting a proper value for N_i^l, we can achieve an acceptable latency. In fact, there is again a trade-off between the latency and power consumption in decision about the value of N_i^l. The distribution is fair in

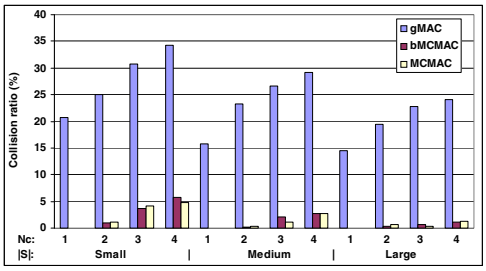
the long run because of the identical probabilities. Although the method does not guarantee at least one listening node in every round (like in LEACH), but it is very easy to implement without requiring any extra information from other nodes and so it is fully distributed. Algorithm 3 shows the behavior of a cluster node.

VI. Protocol Evaluation

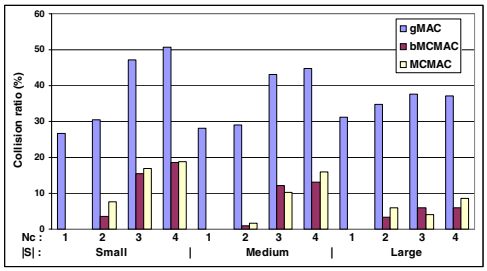
We implemented the MCMAC protocol in the MiXiM [8] framework on top of the OMNeT++ 4 [9] network simulator and ran several simulations with different network setups to evaluate the protocol. We considered three static network sizes, 6×6, 8×8, and 10×10, referred to as small, medium, and large networks, respectively. For each network size, we randomly distributed static sensor nodes in a square area. To ensure an even distribution across the area, we placed the nodes with a 10% variance around fixed grid points. While scaling the number of static nodes, the deployment area was scaled accordingly, such that the density of static nodes was equal for all networks. In every network, we assumed the static nodes in the corners were sink nodes. We also considered various numbers of clusters ($N_c = 1, 2, 3, 4$). Each cluster contains five sensor nodes ($N_i = 5$) and uses the mobility model described in Section III. The parameters for the mobility model were set to mimic human movement.

Each simulation was run for 2000 rounds (TDMA frames). To have statistically more reliable results, every experiment was repeated 10 times with different seeds for the random generator and the shown results are the average over all runs. Moreover, we use the same mobility pattern for all simulation runs of the same network sizes. Our implementation of the mobility model provides such facility to have fairer comparison among different protocols. In all figures in this section, gMAC, bMCMAC, and MCMAC refer to the simulation results for the gMAC protocol, our basic MCMAC protocol without optimizations, and the MCMAC protocol with QoS optimizations, respectively.

Since we are setting up experiments with *MyriaNed* sensor nodes [10], we used the specification of those nodes for the physical layer characteristics in our simulations. *MyriaNed* uses a *Nordic* chip as transceiver, which broadcasts radio packets of 32 bytes with a 2.4GHz ISM-band carrier frequency and data rate up to 2Mbps. Twelve bytes of every transmitted data packet are reserved for the MAC layer (including four bits for parameter d) and twenty remaining bytes are used for payload. So every data packet includes five data items (four bytes for each data item). Among these five items, we dedicate one item for the sensor node's own sensed sample and the others are selected randomly from the memory. As it is mentioned in Section II, in the gossip layer on top of the MAC layer, every node stores a certain number of data items from other sensor nodes in the network. In our implementation, we assumed that every node has enough memory capacity to store one data item from every sensor node existing in the network. For every transmitted data packet, two data items are selected from static nodes and two items from cluster nodes to ensure an acceptable latency for cluster nodes.



(a) Collision ratio counted in static nodes

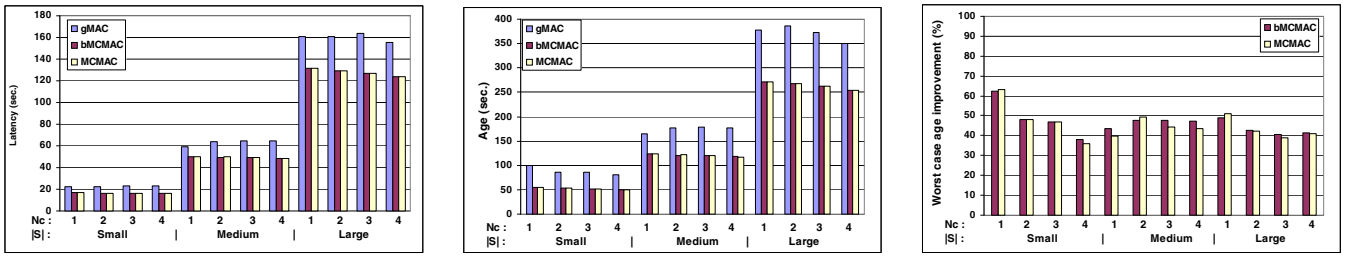


(b) Collision ratio counted in cluster nodes

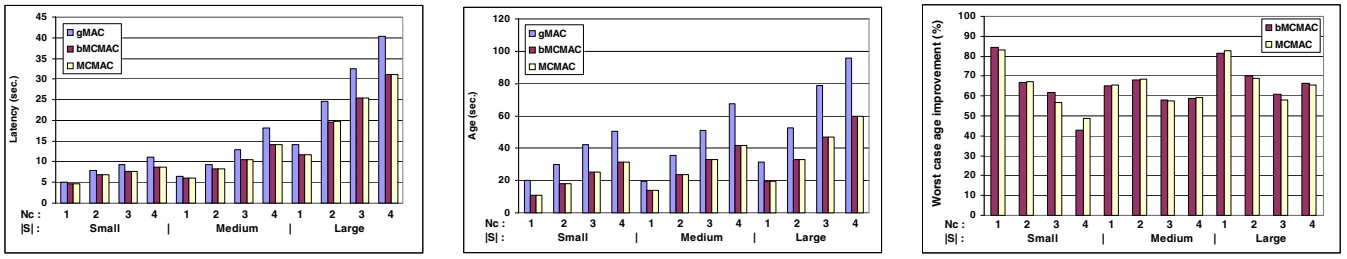
Fig. 6. The average percentage of packets that are lost due to a collision (referred to as collision ratio).

To observe the behavior of the various MAC layers in the presence of mobile clusters, we calculate the average percentage of packets that are lost due to collisions. Fig. 6 shows this observation for all combinations of network sizes and cluster numbers. Fig. 6(a) shows the average collision ratio computed in static nodes whereas Fig. 6(b) shows the collision ratio counted in the cluster nodes. In general, the basic TDMA-based g/L-MAC protocols should be contention free. But when there are mobile clusters in the network, many collisions occur. The reason is that the protocols are not designed for such kind of group mobility. In some runs using the gMAC protocol, the average collision ratio is around 40-50% for cluster nodes. There is also a chance of collision in our MCMAC protocol when there are multiple clusters in a neighborhood. Remember that we use CSMA for accessing the radio channel in the *MCS* time slots. So collisions can still happen due to the switching time from channel sensing mode to transmission and also because of the hidden terminal problem. Note that we only consider packet loss due to the collisions caused by data transmission in the same slot. The real packet loss can be higher because of the quality of the radio channel and interference of other wireless devices.

To evaluate the MCMAC protocol for various aspects, we also calculate the application level QoS metrics of the network. We specifically consider power consumption of sensor nodes and the overall latency of the network. The latency of each data item from every node is calculated as the minimum latency to any one of the four corner sinks. Then the latency for every node is calculated as the average of the latency of all data items generated by that node during the whole simulation over all runs. Finally, we calculate the overall latency of the network as the average latency of all sensor nodes in the network. Eqn. 7 shows the formula for calculating the overall latency of data items from static nodes (Lat_S).



(a) Overall latency for data items from static nodes (b) Average age of the data items from static nodes at the sink nodes (c) Improvement percentage for worst case age of data items from static nodes



(d) Overall latency for data items from cluster nodes (e) Average age of the data items from cluster nodes at the sink nodes (f) Improvement percentage for worst case age of data items from cluster nodes

Fig. 7. The overall latency, the average age of the data items stored in the memory of sink nodes, and improvement of worst case age for various simulations.

$$L_k = \frac{1}{|I_k|} \sum_{i \in I_k} min\{l_i^{\hat{s}} \mid \hat{s} \in Sinks\} \quad , \; k \in S \tag{6}$$

$$Lat_S = \frac{1}{|S|} \sum_{k \in S} L_k \tag{7}$$

where I_k is the set of arrived data items from node k to the sink nodes. $l_i^{\hat{s}}$ denotes the latency of arrived item i to the sink node $\hat{s}$ and L_k stands for the average latency of data items from static node k to the sink nodes. S is the set of static nodes. As the latency constraints for data items from cluster nodes and data items from static nodes may be different, and to have a more precise evaluation of the behavior of the MAC layers, we calculate those latencies separately. So we use a similar formula to calculate the latency of cluster nodes (Lat_C). Fig. 7(a) and 7(d) show Lat_S and Lat_C, respectively. These values give the latency of arrived data items to at least one sink.

To have a more precise comparison, we also need to take the packet delivery ratio to the sink nodes into account. The *age* of the data items stored in the sink nodes can reveal the effect of both latency and the delivery ratio. The age of every data item at every moment is calculated as the difference between the current simulation time and the sample time attached to the existing data item in the memory of the sink node. Fig. 7(b) and 7(e) show the average age of the data items from static nodes and cluster nodes. The first observation here is that the age for cluster nodes using gMAC is around 56% higher on average than the age in our MCMAC protocol.

Consider that, besides requirements for the overall latency of the network, applications normally have constraints for the worst case latency of each sensor node or even each data item. Assume that a mobile cluster stays for a while in a certain position in the network. Using the gMAC protocol, some cluster nodes might be colliding with some static nodes in the neighborhood for that period of time. In such a situation, the instance latency of data items from those nodes may be too high. In contrast, such a situation never happens using our MCMAC protocol and the network becomes more stable and robust. Fig. 7(c) and 7(f) give the improvement that we achieve compared to gMAC using our protocol in terms of the worst case age of the data items from static nodes and from cluster nodes, respectively. To compute the worst case age, for every sensor node, we pick the minimum age at every round over all sink nodes. The maximum age over all simulation rounds is then selected for each node. Finally, the worst case age of each class of sensors is calculated as the maximum calculated age for those sensor nodes. The result is averaged over all 10 simulation runs per experiment.

The second observation in Fig. 7 is that our optimizations do not exacerbate the age or latency and so do not have a noticeable bad effect on the QoS metrics. But power-wise, the saving is considerable. To show the effect of the optimizations on power consumption, we computed the energy consumption of the sensor nodes. The current load of the transceiver (*Nordic*) is 11.3mA, 12.3mA, and 22μA in transmitting, receiving, and standby modes, respectively. Thus, the receiving mode is the most power consuming mode of the radio transceiver. Our optimization algorithms try to decrease idle listening in receiving mode. Eqn. 8 gives the average energy consumption per TDMA frame of the transceiver module of the sensor nodes in our simulations.

$$E = (I_{tx} \times T_{tx} + I_{rx} \times T_{slot} \times \theta) \times V_{bat} \tag{8}$$

I_{rx} and I_{tx} are receiver and transmitter current, respectively. T_{tx} stands for the real transmission duration. As the *Nordic* transceiver uses fixed length packets (32 Bytes), this parameter

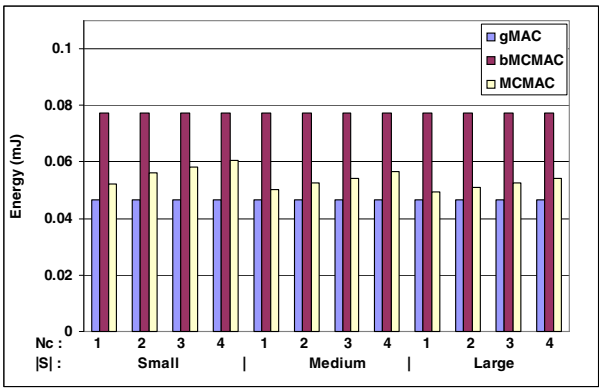


Fig. 8. Average energy consumption of sensor nodes per TDMA frame

has a fixed value. Parameter θ denotes the average number of slots to which the node listens in one TDMA frame. According to our optimizations, the number of Rx slots can vary in different TDMA frames based on the location of clusters. T_{slot} is the duration of one TDMA slot and V_{bat} is the voltage level of the battery, which is 2.4V for our nodes. As the power consumption in standby mode is much less than in other modes, we ignored that in our measurements. The energy consumption of sensor nodes per TDMA frame is shown in Fig. 8 for various circumstances. In general, our basic MCMAC protocol has 55% power consumption overhead compared to gMAC. After optimizations, we remove 70% of this overhead. So the final power overhead of our method is around 16% compared to gMAC. Since in our protocol the static sensor nodes just listen to the *MCS* part if there is a cluster in their neighborhood, the power consumption depends on the network size and the number of clusters.

The proposed MCMAC protocol has several parameters. Configuring them properly affects the performance of the protocol. The considered parameters for the history (H, α), the maximum value of the listening interval to the *MCS* slots for each value of d ($T_l^{max}[d]$), the desired number of cluster nodes that listen to *SAS* part in every frame (N^l), and the length of the CSMA period in *MCS* slots are some examples. We set these parameter values for the simulation according to our experiments ($H = 15$, $\alpha = 1.25$, $T_l^{max}[i] = (i-1) \times 10 + 1$, $N^l = 2$). The length of the CSMA period is set to four times the typical switching delay of the *Nordic* chip ($80\mu s$). However, the process of configuring the parameters to reach the optimal performance and analyzing the QoS trade-offs needs more investigation that we left as future work. Application requirements and constraints should also be considered for tuning these parameters.

VII. Conclusion

This paper proposes a schedule-based medium access control protocol, MCMAC (Mobile Cluster MAC), for supporting mobile clusters in wireless sensor networks (WSN). A specific application scenario is introduced consisting of an ambient wireless sensor network and some wireless body area networks (WBAN) as clusters of sensor nodes. A mobility model for mimicking the behavior of cluster movement is introduced which is used for simulation. As current TDMA-based MAC protocols do not have direct support for such form for mobility, the paper proposes a protocol which deals with cluster mobility. Several simulations were run for different network sizes and different numbers of clusters. The simulation results show that the proposed method gives a considerable improvement with respect to the existing protocols in terms of application level latency and reliability. On the other hand, the basic MCMAC protocol suffers from a higher power consumption of sensor nodes. The paper also proposes optimization methods that considerably decrease the power consumption overhead without worsening the latency. As future work, we will investigate the performance analysis and proper configuration of MCMAC for given application scenarios, and we plan to develop an experimental testbed for validation purposes.

Acknowledgment

This work was supported by the Dutch innovation program Point-One, through project ALwEN, grant PNE07007.

References

[1] L. van Hoesel and P. Havinga, "A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches." in *Proc. 1st Int'l Workshop on Networked Sensing Systems (INSS)*. Society of Instrument and Control Engineers (SICE), September 2004, pp. 205–208.

[2] ——, "Collision-free time slot reuse in multi-hop wireless sensor networks." in *Proc. Int'l Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE, 2005, pp. 101–107.

[3] ——, "Ideas on node mobility support in schedule-based medium access." in *Proc. Int'l Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE, 2008, pp. 539–544.

[4] D. Gavidia and M. van Steen, "A probabilistic replication and storage scheme for large wireless networks of small devices," in *Proc. 5th IEEE Int'l Conf. Mobile and Ad Hoc Sensor Systems (MASS)*. IEEE, 2008.

[5] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A group mobility model for ad hoc wireless networks." in *Proc. 2nd ACM Int'l Conf. on Modeling, analysis and simulation of wireless and mobile systems (MSWiM)*. ACM, 1999, pp. 53–60.

[6] J. Y. M. Liu and B. Noble, "Random waypoint considered harmful." in *Proc. of 22nd Annual Joint Conference of the IEEE Computer and Communications societies (INFOCOM)*. ACM, 2003, pp. 1312–1321.

[7] E. Hyytiä and J. Virtamo, "Random waypoint mobility model in cellular networks," *Wireless Networks*, vol. 13, no. 2, pp. 177–188, 2007.

[8] A. Kopke and et al, "Simulating wireless and mobile networks in OMNeT++ - the MiXiM vision," in *Proc. 1st Int'l Conf. on Simulation tools and techniques for communications, networks and systems and workshops*. ICST, Brussels, 2008.

[9] "OMNeT++ website. http://www.omnetpp.org."

[10] F. van der Wateren, "The art of developing WSN applications with MyriaNed," Chess Company, the Netherlands, Tech. report, 2008.

[11] P. A. Anemaet, "Distributed G-MAC: A flexible MAC protocol for servicing gossip algorithms," Master's thesis, Technical University of Delft, the Netherlands, 2008.

[12] F. Assegei, "Decentralized frame synchronization of a TDMA-based wireless sensor network." Master's thesis, Technical University of Eindhoven, the Netherlands, 2008.

[13] I. Rhee, A. Warrier, M. Aia, J. Min, and L. Mihail, "Z-MAC: a hybrid MAC for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 511–524, 2008.

[14] S. Cai, Y. Liu, and W. Gong, "Analysis of an AIMD based collision avoidance protocol in wireless data networks." in *Proc. 42nd IEEE Conference on Decision and Control*. IEEE, 2003, pp. 104–109.

[15] C. Liu and E. Modiano, "On the performance of additive increase multiplicative decrease AIMD protocols in hybrid space-terrestrial networks." *Comput. Netw. ISDN Syst.*, vol. 47, no. 5, pp. 661–678, 2005.

[16] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks." in *Proc. 33rd Annual Hawaii International Conference on System Sciences*. IEEE, 2000, pp. 1–10.