An Application-Tailored MAC Protocol for Wireless Sensor Networks

S. Chatterjea, L.F.W. van Hoesel and P. Havinga
Department of Computer Science, University of Twente
P.O. Box 217, 7500AE Enschede, the Netherlands
{supriyo, hoesel, havinga}@cs.utwente.nl

Abstract

We describe a data management framework suitable for wireless sensor networks that can be used to adapt the performance of a medium access control (MAC) protocol depending on the query injected into the network. The framework has a completely distributed architecture and only makes use of information available locally to capture information about network traffic patterns. It allows nodes not servicing a query to enter a dormant mode which minimizes transmissions and yet maintain an updated view of the network. We then introduce an Adaptive, Information-centric and Lightweight MAC (AI-LMAC) protocol that adapts its operation depending on the information presented by the framework. Our results demonstrate how transmissions are greatly reduced during the dormant mode. During the active mode, the MAC protocol adjusts fairness to match the expected requirements of the query thus reducing latency. Thus such a data management framework allows the MAC to operate more efficiently by tailoring its needs to suit the requirements of the application.

1. Introduction

Environmental monitoring is one of the primary applications driving wireless sensor network (WSN) research. WSNs will allow scientists to obtain measurements at increased spatial and temporal resolutions that are currently not attainable using existing monitoring technologies.

It is a well known fact that WSNs are application-specific networks. This makes it necessary to design communication protocols that are specifically tailored for a particular application in order to improve its level of efficiency. A protocol should be able to take advantage of certain inherent behavioural characteristics of the application being considered by adapting its operation within the constraints of the application.

The primary focus of this paper is the design of a data management framework that runs on every sensor node. The framework collects and stores information *about* data (i.e. metadata) that is flowing through the node itself. It has two modes of operation – dormant and active - that allow the collection of metadata using different levels of granularity. A node switches from dormant to active mode during the period it services a query. The dormant mode attempts to minimise energy consumption by offering a more granular view of the metadata. The gathered metadata can then be used by other components of the sensor network architecture (e.g. MAC, local query processor, routing scheme, etc.) which can subsequently adapt their operation accordingly. This would result in a more "application-aware" WSN architecture. In this paper, we focus specifically on the adaptation of the MAC based on the metadata collected.

We illustrate our idea by introducing a novel *adaptive and information-aware* medium access control (AI-LMAC) protocol for wireless sensor networks (WSNs) that is based on the Lightweight Medium Access Protocol (LMAC) [1]. AI-LMAC is different from other existing WSN MAC protocols [2, 3, 4, 9] that operate independently of the queries injected into the network. In this case, AI-LMAC adapts its operation to fit the information presented by the data management framework. Unlike LMAC where every node in a network is assigned only one slot, AI-LMAC assigns multiple slots to nodes that need to transmit more data.

Section 2 describes the specific application we are focussing on. In Section 3 we give an overview of our data management framework. Next, in Section 4 we describe details of the main mechanism of our MAC protocol. Section 5 explains how the MAC protocol adapts its operation using the data management framework presented in Section 3. We then present our results in Section 6 and related work is mentioned in Section 7. We conclude the paper and discuss future work in Section 8.

2. Details of application

We are setting up a WSN for environmental monitoring. Currently environmentalists use satellites to monitor parameters such as temperature and rainfall over large geographical areas. This data is not always accurate due to adverse weather conditions. Also data can only be obtained when the satellite flies over the area being monitored. Thus data from WSNs will be used to verify satellite data and also to obtain readings with higher spatial and temporal resolution.

2.1 Assumptions and implications

We assume a dense and relatively static network of heterogeneous nodes (i.e. not all nodes are expected to have the same set of attached sensors). New nodes/sensors (e.g. temperature, humidity, pressure, light, etc.) may be added at any time.

We envision the scenario where the network will be used as a tool for gathering various types of data. Thus different users can be expected to inject different queries that are distributed both spatially and temporally. Queries running simultaneously may overlap partially in terms of space, time and the parameters that need to be measured. This would imply that at a certain instant, various areas of the network can be assumed to have different levels of activity (i.e. some parts generate large amounts of data while other parts are relatively dormant).

Thus it is essential that various components of the WSN architecture are able to adapt continuously to incoming queries to limit the usage of resources (e.g. energy, bandwidth, etc.) only to areas which require it. For example, instead of having a MAC with static duty cycles throughout the entire network, a node should be able to dynamically change its duty cycle depending on the amount of data that is flowing through it in response to the cumulative requirements of the simultaneously running queries in the system.

3. Description of data management framework

The data management framework offers two modes of operation, dormant and active, that offer a dual-granular view of the network. A node switches to active mode the moment it receives a query and reverts back to the dormant mode once the query has expired. The main objective of the dormant mode is to keep all nodes updated regarding the current characteristics (e.g. range of sensor readings) of the network to allow the efficient dissemination of future queries. However, this mode only provides a *coarse-grained* view of the network since message transmissions are kept to a minimum to reduce
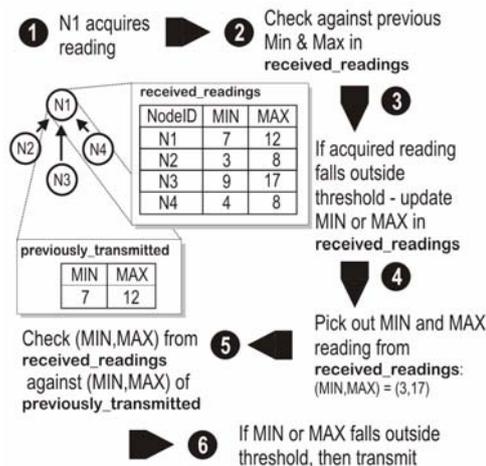

Figure 1. Operation during dormant mode

energy consumption. Upon receiving a query, a node switches to the active mode and begins collecting information about the data messages passing through the node, i.e. metadata. This mode provides a fine-grained view of the network due to the large number of messages traversing through the node. It is important to note that at any single instant, a network could consist of multiple dormant and active regions. In other words node activity is not uniform throughout the network.

The main objective of the dormant and active modes is to allow a node to maintain the range of sensor values that may be present in its subtree. Using this range information eliminates the need to flood the entire network when a new query is first injected into the network. It can also be used for adapting the MAC and efficient query processing. The latter however, is not discussed here as it is outside the scope of this paper. The only difference between the two modes is that they both perform the same task using slightly different mechanisms which result in different levels of energy consumption and accuracy.

3.1 Dormant mode

Initially, once a tree structure has been setup during network start-up and before any query has been injected into the network, all the nodes operate in the dormant mode. As shown in Figure 1, every node maintains a `received_readings` table which stores the latest *minimum* and *maximum* sensor readings that were generated by the node itself and also transmitted by its immediate children. (*Note that the readings received from immediate children may not have been generated at the nodes themselves but somewhere deeper in the tree.*) Once a node samples a reading from its sensor, it checks to see if this reading differs from its previous reading in the `received_readings` table by more than a certain pre-defined threshold (specified during
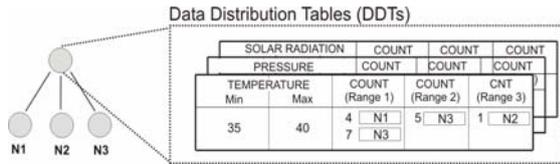
Figure 2. Format of Data Distribution Table

network deployment). If it does, the newly acquired reading replaces the previous entry in the `received_readings` table. The following step is to pick out the *minimum* and *maximum* sensor readings from the `received_readings` table and check both the values against the previously transmitted readings. Once again, if either one of the values differs from the previously transmitted readings by more than the threshold, the node transmits the minimum and maximum values to its parent node.

3.2 Active mode

A node changes from operating in the dormant to active mode the instant it receives a query that either the node itself or its children can service. Every node maintains its own set of Data Distribution Tables (DDTs), Figure 2. Once a node receives a query it looks up its DDTs to deduce how many of its children are going to respond to the particular query. Each DDT corresponds to a sensor type that is present within a node's set of child nodes. A DDT is built over time by monitoring the data that flows through a particular node. Thus statistics about the data flowing through the network is collected without incurring any extra overhead. It is simply based on the data that already needs to flow through a node. As the DDT does not actually store any of the data that flows through it, the size of the DDT is independent of the total number of children a node has within the tree. It does however, depend on the different sensor types, regions defined within the network and the number of ranges sensor readings are classified into.

Upon receiving a reading from a child node, the parent node updates the entry in the appropriate DDT depending on a number of variables: the type of sensor that originally generated the reading and the range within which the received reading falls. Additionally, the DDT also keeps track of which particular immediate neighbour sent it the reading.

It is important to highlight that the numbers or rather the "count" found in the DDT is not simply a count of the total number of children a node has. It only includes the number of active children, i.e. the number of nodes that are actually involved in servicing a particular query.

Using multiple DDTs and categorising the incoming data into ranges allows users to formulate complex query dissemination schemes that would prevent the need to flood the entire network and limit query dissemination to only relevant parts of the network.

An example of a query parsing multiple DDTs would be: *Get solar radiation readings in sector B only if the temperature reading for the sector is above 35°C and the air pressure readings are above 1007mb.* Not only would this help in creating highly specific query dissemination schemes but this can also help in localised query processing.

Also, when a system component (e.g. MAC) needs to adapt its operation based on the information presented by the DDT, the architecture of the DDT allows decisions to be made based on not just the requirements of a single query but based on the net requirements of all queries running simultaneously at that instant.

4. Overview of AI-LMAC

The Adaptive and Information-aware, Lightweight Medium Access Protocol (AI-LMAC) is a TDMA-based protocol that is an adaptive and information-aware version of the LMAC protocol [1]. In this section, we describe some of the common aspects between the two protocols.

Time is divided into *time slots*, which nodes can use to transfer data without having to contend for the medium or having to deal with energy wasting collisions during transmissions. A time slot consists of two parts: the Control Message (CM), which is always transmitted by a node in the time slot(s) it controls and the Data Message (DM), which contains the higher protocol layer data. The CM has a fixed length and specifies the ID of the time slot controller, distance of the node to the gateway for simple routing and the intended receivers of the DM. The DM can have a length up to the end of the time slot or can even be omitted when the node has no data to send. For energy-efficiency reasons, nodes will turn-off their energy consuming transceiver when they are not the intended receivers of a DM, or when the transmission of a DM has finished before the end of the time slot. To be able to maintain the protocol, nodes will always listen to CMs of their neighbouring nodes. When a node is not addressed in that message or the message is not addressed as a broadcast message, the nodes will switch off their power consuming transceivers only to wake up at the next time slot.

To limit the number of time slots necessary in the network, we allow time slots to be reused at a non-interfering distance. Unlike traditional TDMA-based systems, the time slots in AI-LMAC are not divided among nodes by a central manager. Instead, the nodes use an algorithm that is based on local information only to choose time slots to control. Every node transmits a table in the CM that specifies
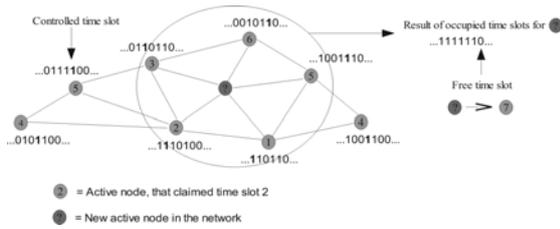
Figure 3. A new active node in the network can pick a time slot when it has discovered all its neighbour nodes

which time slots the node considers to be occupied by itself and its one-hop neighbour nodes. This information can be efficiently encoded by a number of bits equal to the number of time slots in a frame. A node can occupy the appropriate slots when the required number of slots is considered to be free by all its neighbours. This method ensures that a time slot is only reused after at least three hops and that no collisions will occur.

Figure. 3 gives an example of how a new node in the network can pick a time slot after it has discovered all its neighbours. When a node picks a time slot to control, it will control the same time slot in consecutive frames. Currently, we are considering frames of 32 time slots. Note that nodes will only use their own time slots to transmit data to their neighbouring nodes.

In the AI-LMAC protocol, nodes are allowed to control multiple time slots in a frame. To ensure a connected network, every node controls a minimum of one time slot. For further details regarding network setup and routing to the gateway, we refer the reader to [1].

5. Adapting AI-LMAC to the application requirements

We now describe how AI-LMAC can adapt its operation using the information provided in the DDT. Unlike LMAC, which allows every node within the network to own only one slot [1], AI-LMAC allows a node to own multiple slots. Also, AI-LMAC is able to vary the number of slots a particular node owns depending on the amount of data that is expected to flow through it. This ensures fairness in the sense that the bandwidth allocated to a node corresponds to the traffic it is expected to encounter. Note that AI-LMAC uses a single-slot in the dormant mode and multiple-slots in the active mode.

In AI-LMAC, we assume that a parent-child relationship exists between all the nodes in the network, such that the root of the network can be considered to be the highest parent in the hierarchy. Using the DDT, every node would know how much

"importance" to give every one of its immediate children.

Using the DDTs a node cannot decide by itself, how much importance it should give itself to transmit. This is because the DDTs only contain information about a node's active child information. A node is not aware of the data generated by the other children of its own parent node as they may not be within transmission range. Thus, the parent is the only node that has knowledge of the proportion of data that will be contributed by each of its immediate children. The idea here is that if a node realizes that a subset of its immediate children is going to transmit large quantities of data, then more attention needs to be paid to this particular subset of child nodes. In this case, when we say more attention, we actually refer to assigning multiple slots to a particular child.

However, even though a parent node knows which child node deserves more slots to be assigned to it, it cannot send such a rigid instruction to its children as in LMAC. This is because in LMAC, when a node performs slot assignment, it has knowledge of the slot ownership of its first and second order nodes. In this case, the parent node would not know slot ownership information about the slot assignments of its child node's second order nodes since they are three hops away.

Thus, the responsibility of the parent node is simply to "advise" the child, i.e. the parent node sends a message to every one of its children indicating the ideal number of slots that a particular node should take up under the current conditions. It is then up to the child node to follow the advice as closely as possible. This naturally depends on the number of empty slots available.

The process of giving advice starts at the root node of the tree when a query is first injected into the network. This process then percolates down the branches of the tree towards the leaf nodes. If however, the process of giving advice started at an intermediate node, this would increase the chance of performing unfair slot allocations. This is because a node assigning slots would not be aware of the bandwidth requirements of all its sibling nodes which are not within its direct range. From this argument, it is obvious that if we apply this rule repeatedly, the root node is the only node which can assign slots fairly at the beginning. We term this as horizontal fairness as the mechanism ensures that all sibling nodes (i.e. at the same level) under a certain parent are allocated slots fairly.

Apart from establishing a horizontal relationship between nodes, we also introduce a mechanism to include vertical fairness. In order to prevent buffer overflow problems, our mechanism ensures that that the total number of slots assigned to the immediate children of a certain parent node, does not exceed the
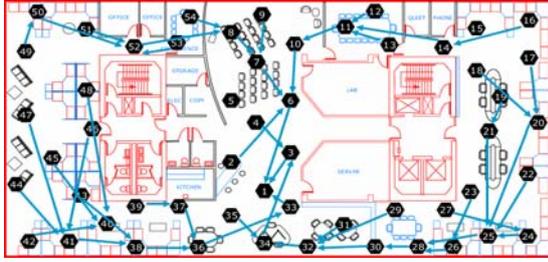
Figure 4. Multihop network with Node 1 as the root [7]

Figure 5. Total number of messages transmitted per hour by all 54 nodes

number of slots owned by the parent. This reduces the likelihood of data packets being dropped due to lack of bandwidth. Furthermore, leaf nodes are prevented from being allocated excessive bandwidth using this mechanism.

Thus introducing two-dimensional fairness ensures that the number of slots taken up by a node does not only depend on its siblings but on its parent as well.

Once a node has received the ideal number of slots it should take up, it checks to see which slots are free within its 2nd order neighbourhood. Just like in LMAC, once a node decides to take up a certain slot, it "marks" the slot using a "1" to indicate that the slot has been taken up.

6. Experimental analysis

Simulations for the dormant mode were based on a dataset acquired from Intel Research Lab at Berkeley [7] which was generated from the trace of 54 sensors that were placed within the laboratory. We have used readings of the temperature sensors over a period of 24 hours. A multihop network was set up using node 1 as the root, Figure 4. Figure 5 shows the total number of messages that were transmitted over 24 hours using different threshold values.

A total of three peaks can be observed from the graphs – the first peak occurs during the first hour which is when the nodes initially start up. This is due to the first few minutes during which a large proportion of the sensors generated highly erratic readings (e.g. oscillating between 20˚C and 120˚C). Such erratic readings resulted in a large number of range changes and this in turn translated into a large number of transmissions. Also, even if the errors occurred in sensors deep in the communication tree, the "swing" of the readings caused the errors to propagate up towards the root, thus affecting all the intermediary nodes as well. While in this case this is an undesirable effect, it is also important to note that the same mechanism is ideal for "event detection" using minimal message transmissions. We however, plan to solve the problem of minimising erratic sensor readings in the future using statistical prediction techniques.
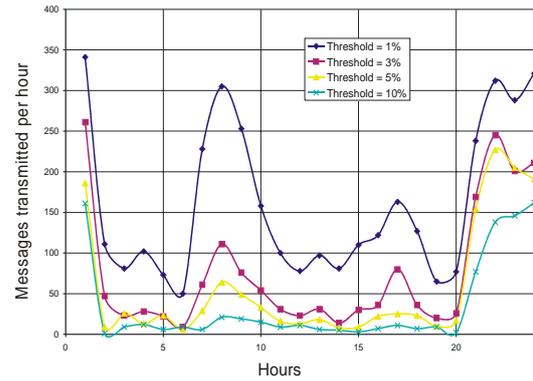
The effect of minimising message transmissions using a larger threshold had a greater effect when there was a gradual rise of temperature in the day. This is evident from the second peak. The third peak was caused by temperature fluctuations by up to 7 degrees. However, it is important to note that even when the transmissions peaked at 341 messages during the first hour (Threshold = 1%), the average number of message transmissions per node, per hour was only slightly higher than 6 messages.

Figure 6 illustrates how the number of child nodes a particular node has affects the numbers of transmissions. When collecting raw sensor readings, it is a common occurrence to have nodes with greater number of children transmitting a larger number of messages. This will result in premature network partitioning. Also, when performing aggregation as described in [6], the number of messages transmitted by a node is independent of the number of child node it has. In this case, due to coarse-grained aggregation being performed, the number of messages transmitted is dramatically reduced (by up to a factor of nearly 35) as the number of children increases. It should be added that while the average number of transmissions by a leaf node may appear to be substantially higher than other nodes, on average, a leaf node only transmits 4.3 messages every hour.
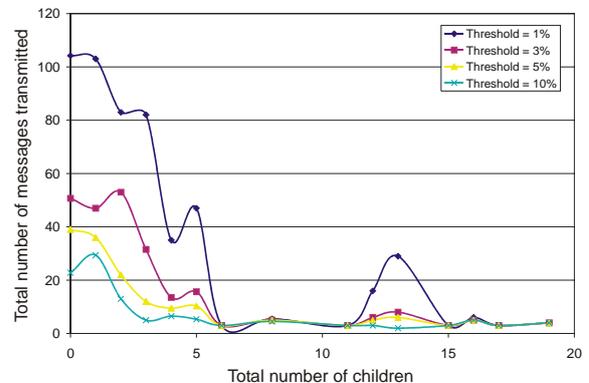

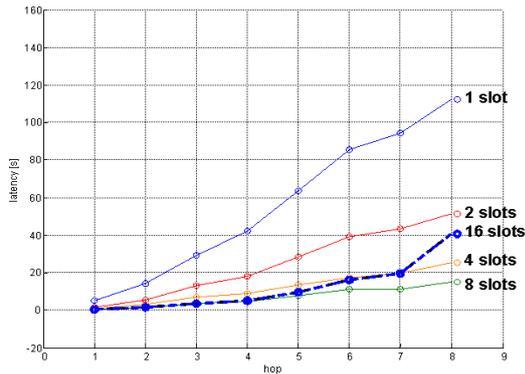Figure 6. Effect of number of children on message transmissions

Figure 7. Effect of number of children on message transmissions

Our framework provides a mechanism to assign more bandwidth to those parts in the network that encounter more data traffic than others. In fact, the assigned bandwidth is proportional to the expected traffic. Hence our framework is able to minimise the overall latency in the network and also the number of messages which need to be buffered in the nodes. Figure 7 illustrates how latency is reduced as the maximum number of allowable slots that can be owned by a node is increased from 1 to 16. These results are obtained by simulation using the discrete event simulator OMNeT++ [8]. Results are averaged over five different network topologies consisting of 49 nodes and one root. Ten different runs were carried out per topology.

The results clearly indicate that latency is proportionally reduced with the maximum number of controlled slots. However, this holds true only until eight slots. For the sixteen slot scenario, the number of free slots in the network rapidly decreases with every hop from the root and thus the nodes are not able to comply with the advice. Consequently, a bottleneck is created at a few hops (6 to 8) from the root, resulting in higher latency for messages created in those areas.

7. Related work

There are certain parallels between the data management framework presented here and semantic routing in [6]. However, semantic routing solely deals with the dissemination of queries. The primary difference is that no information is stored by the nodes about the number of active children that can be expected to respond to a certain query. Also, while TinyDB does have some form of communication scheduling [6], it only helps to reduce the burden on the underlying MAC protocol; it does not alter the operation of the MAC itself. There are also a host of MAC protocols described in the literature for sensor networks [1, 2, 3, 4, 9]. While some are able to adapt to varying traffic patterns, none has any knowledge of the queries injected into the network.

8. Conclusion and future work

Out results indicate that an "application-aware" MAC results in a substantial reduction of transmitted messages and allows it to adapt according to the application requirements.

Future research will look into methods that can be employed to improve the energy efficiency of the protocol. For instance, currently the same bandwidth is allocated for traffic both from and towards the root node. This can be improved on since data flow can be expected to be much higher than query flow. Also, instead of assigning slots randomly, we plan to minimise switching by assigning slots in a contiguous manner. We also plan to include other information in the data management framework such as data generation rates and degree of aggregation and also investigate various statistical techniques that may be employed to reduce energy consumption during data acquisition.

9. 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 *Proceedings of 1st International Workshop on Networked Sensing Systems*, 2004.

[2] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. 2002.

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

[4] K. O. V. Rajendran and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *Proceedings of the 1st internationalconference on Embedded networked sensor systems*, pages 181–192. ACM Press, 2003.

[5] Consensus homepage: *http://consensus.ctit.utwente.nl/.*

[6] Samuel Madden. The Design and Evaluation of a Query Processing Architecture for Sensor Networks. *PhD Thesis*, UC Berkeley, 2003.

[7] Intel Lab Data, http://berkeley.intel-research.net/labdata/.

[8] Omnet++ discrete event simulator, http://www.omnetpp.org.

[9] B. K. G. Lu and C. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in sensor networks. In *In Proceedings of 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks(WMAN 04)*, 2004.

[10] EYES homepage: *http://www.eyes.eu.org.*