A Low-Latency, Information-Centric Medium Access Protocol for Wireless Sensor Networks

L.F.W. van Hoesel, S. Chatterjea and P.J.M. Havinga
University of Twente
Department of Electrical Engineering, Computer Science and Mathematics
P.O. box 217, NL-7500 AE, Enschede, The Netherlands
Email: {l.f.w.vanhoesel, s.chatterjea, p.j.m.havinga}@utwente.nl

Keywords—**Wireless sensor networks, medium access protocol, energy efficiency, time division multiple access.**

Abstract—**In this paper we present a novel TDMA-based medium access control (MAC) protocol for wireless sensor networks. Unlike conventional MAC protocols which function independently of the application, we introduce an Adaptive, Information-centric and Lightweight MAC(AI-LMAC) protocol that adapts its operation depending on the requirements of the application. We also present a completely localised data management framework that helps capture information about traffic patterns in the network. This information is subsequently used by AI-LMAC to modify its operation accordingly. We present preliminary results showing how the MAC protocol efficiently manages the issues of fairness, latency and message buffer management.**

I. Introduction

It is envisioned that wireless sensor networks will bring about a paradigm shift in environmental monitoring techniques that will allow scientists to obtain measurements at increased spatial and temporal resolutions that are not attainable using existing monitoring technologies. As such networks will be made up of hundreds or even thousands of highly resource-constrained nodes that would be required to function for extended periods of time, it is imperative to ensure that we design protocols that adapt autonomously.

It is a well known fact that WSNs are application-specific networks [1]. Thus it is important to investigate the possibility of designing communication protocols that are specifically designed for a particular application in order to improve its level of efficiency. In other words, a protocol should be able to take advantage of certain inherent behavioural properties of the application being considered.

Judging from the previous statement, the reader may be inclined to think that following such a strict "application-specific" approach would result in a protocol that is completely static and is unable to adapt to any changes. We would like to emphasize however, that we believe it is very important to design a protocol that is dynamic. But we also feel that instead of designing a protocol that is able to adapt from one application to another, the developed protocol should be able to adapt within the constraints of the application being considered.

The primary focus of this paper is the design of a novel "information-aware" medium access control (MAC) protocol for wireless sensor networks (WSNs) that is based on the Lightweight Medium Access Protocol (LMAC) [2]. This is very different from other existing MAC protocols for WSNs [2], [3], [4], [5], [6], [7] that operate independently of the information queries injected into the network or the kind of data that flows within the network. However, apart from just describing the details of a new MAC protocol we also describe the design of a data management framework that resides on every node. This framework helps to capture data about the data flowing through the network. The information captured by the framework is then utilised by the MAC protocol to adapt its operation accordingly.

In the following section we present current research related to the area of MAC protocols designed specifically for wireless sensor networks. Section III describes the specific application we are focussing on. In Section IV we give an overview of our data management framework. Next in Section V we describe details of the main mechanism of our MAC protocol. Section VI explains how the MAC protocol adapts its operation using the data management framework presented in Section IV. We then present our preliminary results in Section VII and finally conclude the paper and discuss future work in Section VIII.

This work is performed as part of the NWO funded CONSENSUS project [8] and the European EYES project (IST-2001-34734) [9] on self-organising and collaborative energy-efficient sensor networks. It addresses the convergence of distributed information processing, wireless communication and mobile computing.

II. Related work

There has been a number of MAC protocols developed for WSNs over the past few years [2], [3], [4], [5], [6], [7].

We only highlight some of the more prominent protocols here due to space limitation. Sensor-MAC (SMAC) [3] reduces energy consumption by reducing the duty cycle of a sensor node. This is done by following a periodic active/sleep schedule that is fixed. Nodes turn off their radio during sleep periods and turn it on while receiving or transmitting. As SMAC uses a fixed duty cycle, it is unable to adapt its operation to varying traffic rates. Timeout-MAC (TMAC) [4] improves on SMAC by using an adaptive duty cycle thus adapting automatically to traffic fluctuations. However, due to its aggressive power-down policy, nodes often go to sleep too early, thus decreasing throughput and increasing latency [10].

Data-gathering MAC (DMAC) [5] also uses an adaptive duty cycle where the wake up schedule depends on the depth of a node in the data gathering tree. Additionally, it provides a low source-to-sink latency. However, it does not take fairness into account. Thus the duty cycle assigned to a certain child node may not be proportional to the amount of data it needs to transmit when compared to another child of its parent node, i.e. a sibling node.

[6] does take fairness into consideration by introducing a rate control mechanism. It defines fairness as giving every node in the network an equal opportunity to transmit its data. We however, are not interested in having all nodes transmit data simultaneously. Since we are considering a heterogeneous network where queries injected will be distributed both in the spatial and temporal sense, we define fairness as follows: *"Only nodes which are able to service an incoming query or nodes which have children which can service a particular query should be given the chance to transmit their data. Nodes which are not involved should be given lower priority. Thus the priority given to a node is directly proportional to the amount of data a node is expected to transmit.*" This is only a general definition of how we interpret fairness. More specifically, our mechanism uses what is known as "2-Dimensional" fairness.

Unlike [6] which uses the analogy of *"metering traffic onto a freeway where each node generating data is like cars trying to enter"*, our mechanism on the other hand dynamically changes the "number of lanes" in the freeway to accommodate varying data traffic rates. Like AI-LMAC, the traffic-adaptive medium access protocol (TRAMA) [7] uses a TDMA-based communication scheme that is able to adapt to actual traffic conditions. However, although TRAMA achieves high channel utilisation, it does so at the expense of considerable latency and high algorithmic complexity. It also fails to address the issue of fairness.

While some of the protocols mentioned above are able to adapt their operation to varying data traffic rates, none of them makes use of any knowledge of the actual application they are being used for. In other words the MAC layer is completely independent of the application running above it. Recently however, members of the database research community have realised the importance of influencing the MAC protocol using information from queries injected into the network. TAG [11] for instance performs some sort of communication scheduling using inputs from the query that helps to reduce the burden placed on the underlying MAC. The scheduling mechanism decides when the MAC should be operational, i.e. listening or transmitting. At other times the radio is put to sleep. [12] describes a Data Transmission Algebra (DTA) that uses query scheduling to reduce collisions at the MAC layer. However, in the event that the schedule is unable to avoid collisions, they are handled at the MAC layer.

In both instances, there is a scheduling mechanism based on an incoming query or data that decides when to turn the MAC **on** or **off**. The MAC continues its normal operation during its active period, e.g. it could still be prone to collisions. However, the likelihood of collisions occurring is reduced. While we have encountered communication schemes which basically only turn the MAC on or off, we have not encountered any schemes in the current literature that will actually modify the operation of the MAC itself.

We introduce a framework that improves on current cross-layer optimisation methods by going a step further than simply switching the MAC on or off by allowing the MAC itself to adapt its own operation depending on the query and data flowing within the network.

III. Application considered

As we take an application-specific approach to protocol design, we give an overview of the application we are currently considering. This would enable us to justify certain assumptions that we make about the network in view of the application at hand and we also describe the related implications in terms of design requirements.

Our primary focus is to setup a WSN that will be used for environmental monitoring. Environmentalists would like to deploy sensor nodes over a large geographical area in order to monitor certain parameters of the physical environment such as temperature, solar radiation, air pressure and rainfall. At the same time, they would be taking satellite images of the area during specific times. As information obtained from satellite images may be distorted at times due to unfavourable weather conditions, the idea is to collect ground data using the sensor nodes. This ground data can subsequently be used to verify readings obtained from the satellite. Also as satellite images are only taken a

couple of times a day it is impossible to obtain readings throughout the whole day. Thus instead of performing interpolation to estimate intermediate conditions, sensor nodes would be used to obtain readings of higher temporal resolution.

We assume a dense network where all node locations are fixed during deployment. The network consists of heterogeneous nodes where different nodes may have different sensors attached to them. Nodes (with even new sensor types) may be added to or removed from the network at any time.

We envision the scenario where different environmentalists with slightly different research interests access the same network. Thus, they inject different queries as they are interested in different data sets. In other words the sensor network is primarily used as a tool for gathering data. This means that there could be some overlaps of certain parts of multiple queries (both in the spatial and temporal sense) that may be injected into the network at any single point of time. An implication would be that even though data may not be generated at very high rates (e.g. certain parameters may change very gradually) a large number of queries could be expected to be active within the network simultaneously. Also it may be possible that certain queries generate very high data rates in certain parts of the network while other parts can be relatively inactive.

Thus it is important that the architecture adapts continuously to incoming queries and data e.g. parts of the network which are generating more data should work more actively than other inactive parts so that data can be relayed to the root node with lower latency. For example, we illustrate in Section VII that it would not be a good idea to have a MAC where nodes have static duty cycles. Instead a node should be able to dynamically change its duty cycle depending on the amount of data that is flowing through the network and when and where it is flowing.

In order for the architecture to adapt dynamically, it is essential to know the expected data that will be generated for a particular query that is injected into the network. To handle the task of processing queries and making more sense of the queries and data flowing through the network, we propose the creation of a data management framework as shown in the following section.

IV. Description of Data Management Framework

In this section we introduce a Data Distribution Table (DDT) built into the framework that helps make deductions about the kind of data traffic that can be expected depending on the query injected into the network and the distribution of the data that is being generated.

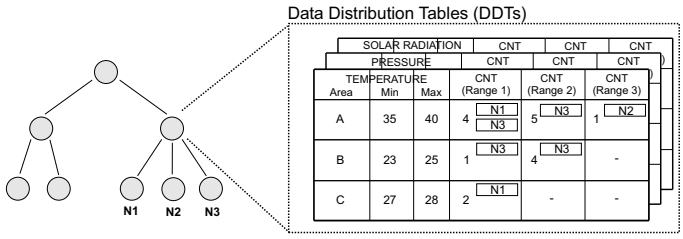

Fig. 1. Format of the Data Distribution Table

Once a node receives a query it looks up its Data Distribution Tables (DDTs) to deduce how many of its children are going to respond to the particular query. Every node maintains its own set of DDTs each of which is a table representing a particular type of sensor that is present within its set of child nodes. So if for instance a node and its children possess temperature and air pressure sensors, then the node would have two separate DDTs. A DDT is built over time by monitoring the data that flows through it. In other words, 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 from the leaf nodes to the root. Thus statistics are collected using only locally available information.

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, the region where the reading originated from and the value of the reading itself. Additionally, the DDT also keeps track of the maximum and minimum readings obtained and also which particular immediate neighbour sent it the reading. As shown in Figure 1, apart from the first three columns of a DDT, the remaining columns contain the number of readings received that fall within a particular range.

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. By active children, we are solely referring to the number of nodes that are actually involved in servicing a particular query. Knowledge of the number of active children for a certain query would help in making rough estimates on how much data can be expected and this information can then be used to adapt the various components of the WSN architecture accordingly.

Using the information collected in the DDT, components of the WSN can also change their operation to adapt to varying query dynamics. Query dynamics refer to the situation when the number of queries passing through a particular node at any point of time varies. We mentioned earlier that we expect multiple queries to be injected

into the network distributed both spatially and temporally. Thus the MAC protocol can adapt its operation based on a not just a single query but based on the net requirements of all queries being served at a certain point of time.

V. Design 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 [2].

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– (see Section V-A) and the Data Message (DM), which contains the higher protocol layer data. The CM has a fixed length and contains control information. The DM can have a length up to the end of the time slot or can even be ommitted 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.

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 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. In fact, most MAC protocols ensure a simular distance between simultaneous transmissions. For example, in the SMAC protocol, this distance is assured by the exchange of RTS- and CTS-messages [3].

Fig. 2 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 consequetive 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.

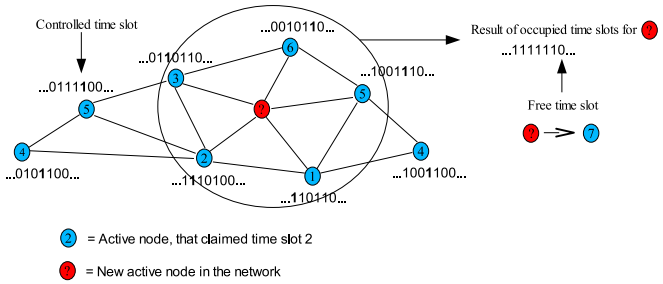

Fig. 2. A new active node in the network can pick a time slot when it has discovered all its neighbour nodes

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.

A. Control Message of AI-LMAC

The Control Message has a fixed size and is used for several purposes. It carries the ID of the time slot controller, it indicates the distance of the node to the gateway in hops for simple routing to a gateway in the network, it addresses the intended receiver(s) of the Data Message, it reports the length of the DM and it carries acknowledgements to successfully received messages.

The control data will also be used to maintain synchronisation between the nodes and therefore the nodes also transmit the sequence number of their time slot in the frame. The transmission of the control data is carefully timed by the nodes, although we do not assume that the nodes have clocks with high accuracy. We assume that the clock drift is negligible in a single frame, even for clocks with low accuracy.

All neighbouring nodes will ensure they receive the control messages 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.

B. Network setup

When nodes are powered on, they are all unsynchronised. Also nodes are unaware of the number of slots they need to control. In order to get synchronised, the gateway takes the initiative to start controlling a time slot. The control messages of the gateway are received by its one-hop neighbours. These neighbours then synchronise their clocks to the gateway. After one frame, the one-hop neighbours are aware of all time slots that are owned by possible multiple gateways in their reception range. Next, the recently synchronised nodes will pick a random time slot. Slots already occupied will be excluded from this random

slot selection process.

Once the nodes have been synchronised, the next step is to assign the right number of slots to a node depending on the expected traffic of a particular query as described in the followingsection.

For details of the support for routing to the gateway node, we refer the reader to [2]. Note that the protocol also supports ID-based routing techniques, without adoptation of the protocol.

VI. Adapting AI-LMAC Using The Data Management Framework

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 [2], 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. For example, it would be pointless to allocate a large number of slots to a particular node that is not generating or relaying significant amounts of data.

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 in 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 realises 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. a 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 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 and it will try to take up as many slots as advised. To ensure a balanced slot allocation between children nodes, nodes will increment the number of controlled slots in turns at a rate of one slot per frame.

VII. Experimental Analysis

Our framework provides a mechanism to assign more bandwidth 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 the number of messages which need to be buffered in the nodes can be substantially reduced. Figure 3 presents

TABLE I
MAXIMUM NUMBER OF BACK LOGGED MESSAGES (WORST CASE)

Scenario	Maximum messages
1 slot	105
2 slots	63
4 slots	34
8 slots	32
12 slots	54
16 slots	56

these measurements for the single slot scenario. These results are obtained by simulation using the discrete event simulator OMNeT++ [13]. Results are averaged over five different network topologies consisting of 49 nodes and one root. Ten different runs were carried out per topology.

In Figures 4-8, the advice for the maximum number of allowable time slots is varied from 2 to 16.

The results clearly indicate that latency is proportionally reduced with the the maximum number of controlled slots. However, this holds true only until eight slots (Fig. 6). For the twelve and sixteen slot scenarios, 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.

In this MAC protocol, nodes are able to receive up to 32 (=number of time slots) messages per frame. The number of messages that can be transmitted per frame is dependent on the number of time slots the node controls. When the number of incoming messages exceeds the number of messages that can be transmitted during a frame, the additional incoming messages have to be buffered. For each of the scenarios, we have collected data about the maximum number of messages back logged in the worst case (Table I). These results reflex the same trend as Figures 3-8. Note that in real-life implementations, the capacity to hold back logged messages will be limited due to scarce memory resources in the sensor nodes.

VIII. CONCLUSION AND FUTURE WORK

In this paper we have described a novel MAC protocol that adapts its operation depending on the type of query injected into the network and also the type of data that flows through it. Thus unlike other conventional MAC protocols which run independently of the application, AI-LMAC is application dependent. We have also described a data management framework that helps gather data about the queries and data flowing through the network and how it is subsequently used by the MAC layer to adjust its operation.

Using a mechanism which depends on two-dimensional fairness, we have illustrated how AI-LMAC reduces both latency and back logged messages. It also handles the issue of fairness. Thus only parts of the network that need to respond to a certain query increase their level of activity. Other sections of the network remain relatively inactive.

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 including other information in the data management framework such as data generation rates and degree of aggregation.

REFERENCES

[1] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva, "Directed diffusion for wireless sensor networking", *IEEE/ACM Trans. Netw.*, Vol. 11, No. 1, 2003, pp. 2-16.

[2] L.F.W. van Hoesel and P.J.M Havinga, "A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing Preamble Transmissions and Transceiver State Switches", In: *First International Conference on Networked Sensing Systems*, Tokyo, 2004.

[3] W. Ye, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks", In: *INFOCOM*, 2002.

[4] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks", In: *Proc. First International Conference on Embedded Networked Sensor Systems*, Los Angeles, California, USA, 2003.

[5] G. Lu, B. Krishnamachari and C. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in sensor setworks", In: *4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks* , 2004.

[6] A. Woo and D.E. Culler, "A transmission control scheme for media access in sensor networks", In: *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking*, Rome, Italy, 2001.

[7] V. Rajendran, K. Obraczka and J.J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks", In: *Proc. First International Conference on Embedded networked sensor systems*, Los Angeles, California, 2003.

[8] CONSENSUS homepage: http://www.ctit.utwente.nl/research/projects /national/NWO/consensus.doc/.

[9] EYES homepage: http://eyes.eu.org.

[10] K. Langendoen and G. Halkes,"Energy-Efficient Medium Access Control", *Embedded Systems Handbook*, Embedded Systems Handbook, CRC Press, 2004.

[11] S. Madden, R. Szewczyk and M. Franklin and D. Culler, "Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Net-

works", In: *Proc. of 4th IEEE Workshop on Mobile Computing and Systems Applications*, 2002.

[12] V.I. Zadorozhny, P.K. Chrysanthis, P. Krishnamurthy, "Framework for Extending the Synergy between MAC Layer and Query Optimization in Sensor Networks", In: *Proc. of First International Workshop on Data Management for Sensor Networks*, Toronto, Canada, 2004.

[13] OMNeT++ discrete event simulator, http://www.omnetpp.org

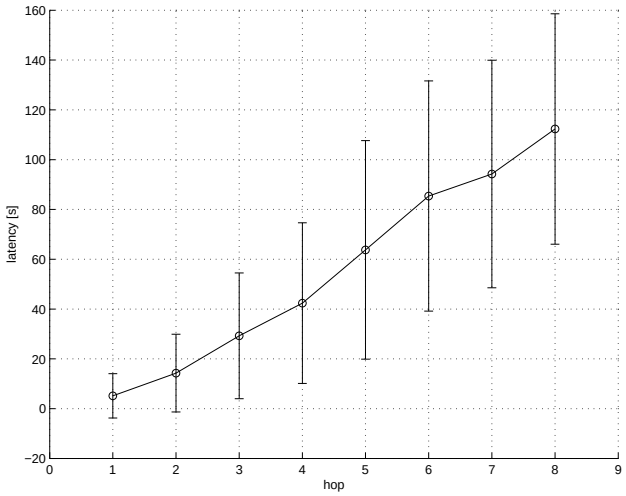

Fig. 3. Average latency and standard deviation when all nodes control one time slot

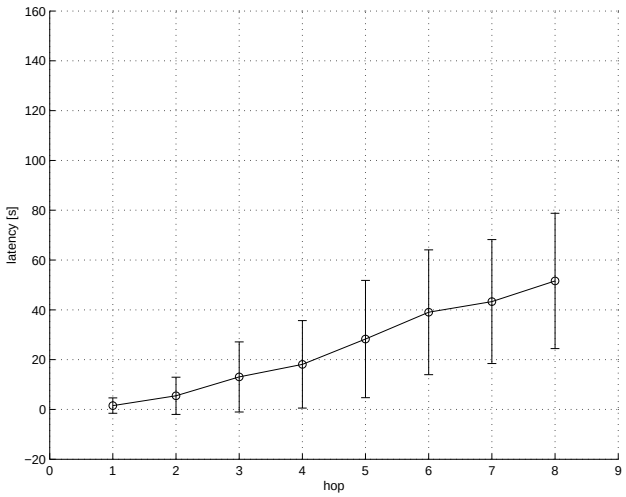

Fig. 4. Average latency and standard deviation; maximum given advice: 2 slots

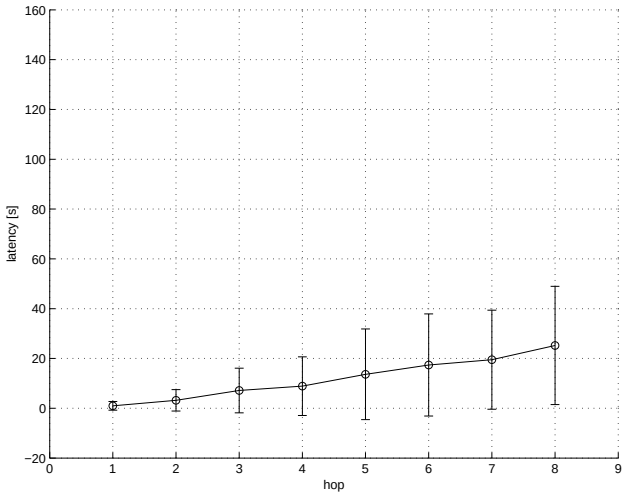

Fig. 5. Average latency and standard deviation; maximum given advice: 4 slots

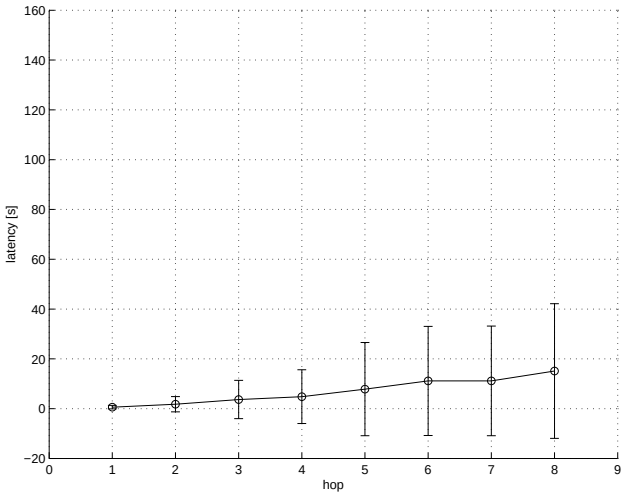

Fig. 6. Average latency and standard deviation; maximum given advice: 8 slots

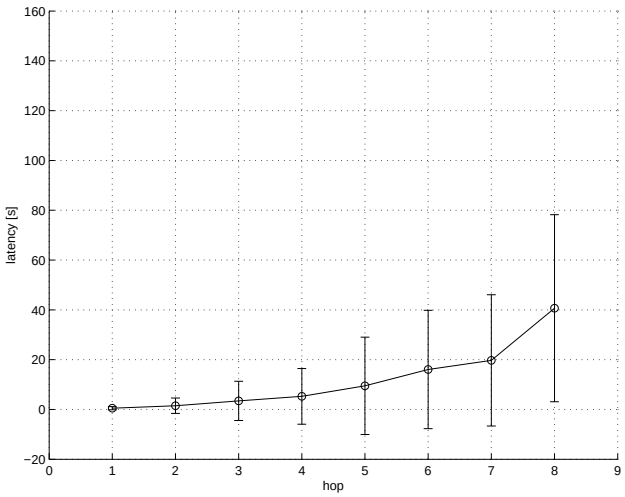

Fig. 7. Average latency and standard deviation; maximum given advice: 12 slots

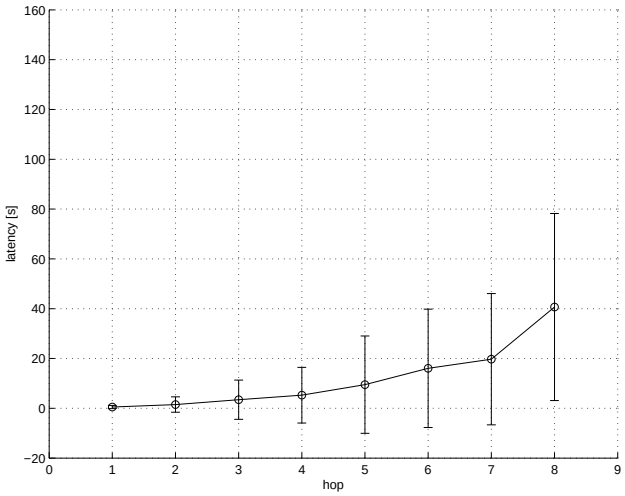

Fig. 8. Average latency and standard deviation; maximum given advice: 16 slots