Crankshaft: An Energy-Efficient MAC-Protocol for Dense Wireless Sensor Networks

G.P. Halkes and K.G. Langendoen

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, The Netherlands
{g.p.halkes, k.g.langendoen}@tudelft.nl

Abstract. This paper introduces Crankshaft, a MAC protocol specifically targeted at dense wireless sensor networks. Crankshaft employs node synchronisation and offset wake-up schedules to combat the main cause of inefficiency in dense networks: overhearing by neighbouring nodes. Further energy savings are gained by using efficient channel polling and contention resolution techniques.

Simulations show that Crankshaft achieves high delivery ratios at low power consumption under the common convergecast traffic pattern in dense networks. This performance is achieved by trading broadcast bandwidth for energy efficiency. Finally, tests with a TinyOS implementation demonstrate the real-world feasibility of the protocol.

Keywords: Wireless Sensor Networks, MAC Protocol, Dense Networks.

1 Introduction

In Wireless Sensor Networks (WSNs) energy efficiency is a major consideration. Sensor nodes are expected to operate for long periods of time, running of batteries or ambient energy sources. Because the biggest consumer of energy is the radio, many researchers have focused on creating energy efficient MAC protocols [1,2,3,4,5,6,7].

Recent experiences with real-world deployments [8] have shown that the number of neighbours in WSNs can be higher than 15, which exceeds the 5–10 that MAC protocol designers have typically assumed. The "smart-dust" vision of WSNs also incorporates these dense deployments. Dense deployments have their own specific challenges, due to the high connectivity. Below we list the most important problems in current MAC protocols arising from dense networks:

Overhearing. Overhearing of messages destined for other nodes is a source of energy waste in any deployment. However, in dense deployments there are more neighbours that will overhear a message which exacerbates the problem. Furthermore, having more neighbours also means there may be more messages to overhear.

Communication grouping. Several protocols, like S-MAC [1] and T-MAC [2], group communication into active parts of frames. This is done to allow the network to go to sleep during the inactive parts of frames. The approach has a significant drawback: the grouping increases contention and collisions. Collisions cause retries, which in turn increases the traffic load. Furthermore, for adaptive protocols like T-MAC the increased traffic will keep nodes awake longer without increasing useful energy consumption.

Over-provisioning. TDMA protocols like LMAC [6] schedule send-slots for participating nodes. However, if a node has nothing to send, the slot goes unused. In a dense deployment, a frame has to be split into many slots to allow all nodes to participate in the network. Most of these slots go unused, but as a node has to wait for its send slot before it is allowed to send, latency increases and throughput decreases. Also, all the non-sender nodes will have to listen for at least a short amount of time to check if the scheduled sender is actually using the slot, and to check if they are being addressed by the sender.

Neighbour state. Protocols that save neighbour state as for example PMAC [7] and WiseMAC [9] do, also run into problems in dense deployments. Because in a dense deployment each node has many neighbours, the MAC protocol will have to either maintain a lot of state or discard some of the neighbours. Maintaining state for over 20 neighbours is undesirable as it uses precious RAM. However, discarding neighbour information means that communication with certain nodes is not possible or at least severely hindered. Furthermore, the routing protocol also maintains a neighbour list. If the neighbour list of the routing layer contains different nodes than the neighbour list of the MAC protocol, considerable problems arise.

This paper introduces the Crankshaft MAC protocol, which is specifically designed to perform well in dense deployments. It reduces overhearing and communication grouping by letting nodes power-down their radios alternately, rather than simultaneously. It does not keep per-neighbour state and receive-slot scheduling ensures that over-provisioning is bounded. The trade-off is that the maximum throughput is reduced, especially for broadcast traffic. For many applications however, this trade-off is acceptable.

The rest of this paper is organised as follows: First we will present related work. Then, in Section 3 we discuss the design of the Crankshaft protocol. In Section 4 we discuss the setup of our simulations, followed by the simulation results in Section 5. In Section 6 we present our results with our TinyOS implementation. Finally in Sections 7 and 8 we provide more discussion of the results and finish with our conclusions and future work.

2 Related Work

Many MAC protocols have been designed for WSNs. Below we present a selection of protocols that have relevance to our new Crankshaft protocol.

One of the earliest proposals is Low Power Listening [3] (LPL). LPL uses a simple Data/Ack scheme to ensure reliability. This is combined with efficient channel polling to reduce the energy spent on listening for incoming messages.

Instead of simply turning the radio on, LPL periodically checks the channel. To ensure that messages are properly received the preamble of each message is stretched to include an additional poll period. The B-MAC [4] protocol is an evolution of LPL, whereby the application can tweak the poll period depending on its bandwidth usage.

The channel polling mechanism of LPL has been further refined in the SCP-MAC [5] protocol. It uses channel polling, but it synchronises all nodes to poll at the same time, essentially implementing a slotting mechanism. This allows potential senders to do contention resolution before the intended receiver wakes up. Furthermore, the message preambles do not have to be stretched for a complete poll period because the poll moment is known. Crankshaft employs a mechanism of channel polling very similar to the SCP-MAC protocol.

Several TDMA protocols have also been developed. A good example is the LMAC [6] protocol. Contrary to many other TDMA protocols, the LMAC protocol uses a completely distributed slot assignment mechanism. Each slot owner sends at least a packet header in the slot the node owns. Neighbouring nodes listen to the start of each slot, and detect which slots are free. However, for a TDMA protocol it is required that a slot is not reused within a two-hop neighbourhood. The LMAC protocol therefore includes a bitmap with all the slots assigned to a node's neighbours in the header. By combining the bitmaps of all neighbours, a node can determine which slots are free within a two-hop neighbourhood. Crankshaft also uses frames and slots, but schedules receivers rather than senders. However, the mechanisms employed by LMAC to achieve synchronisation, framing and slotting are used in Crankshaft.

Pattern MAC [7], or PMAC, also divides time into frames. Each frame consists of two parts: the Pattern Repeat part and the Pattern Exchange part. Both parts are divided into slots. During the Pattern Repeat part, nodes follow the sleep/wake pattern they have advertised. Nodes also wake up when a neighbour for which they have a packet to send has advertised it will be awake during a particular slot. Nodes advertise their chosen patterns during the Pattern Exchange part. Each slot in the Pattern Exchange part is long enough to send a node's pattern information and nodes have to contend for these slots. To enable all nodes to send their pattern information, the Pattern Exchange part has as many slots as the maximum number of neighbours a node is expected to have. The sleep/wake patterns are adapted to the traffic going through a node, to achieve maximal energy savings. The PMAC protocol is similar to the Crankshaft protocol in that it also schedules nodes to be awake for reception on a slot basis. However, the PMAC protocol requires nodes to exchange and store schedules.

Although unrelated to the Crankshaft protocol, we also briefly introduce the S-MAC and T-MAC protocols, as they have become a standard benchmark for WSN MAC protocols. The S-MAC [1] protocol attacks the idle listening problem by introducing a coarse duty-cycling mechanism. It divides time into frames with an active part and a sleeping part. All communication between nodes is performed in the active part. In a later paper [10] the S-MAC protocol was extended with "adaptive listening". This incarnation of the S-MAC protocol uses the same idea as the T-MAC [2] protocol: group all communication at the start of the active period, and go to sleep if no more activity is sensed. This adapts the length of the active period to the available traffic.

When employed in dense networks all the above protocols suffer from one or more of the problems signalled in the introduction. This results in suboptimal energy use.

3 Crankshaft

Having signalled the problems MAC protocols face in dense wireless-networks, we designed the Crankshaft protocol. The basic principle of the protocol is that nodes are only awake to receive messages at fixed offsets from the start of a frame. This is analogous to an internal combustion engine where the moment a piston fires is a fixed offset from the start of the rotation of the crankshaft. Allowing different nodes to wake up for reception at different offsets from the start of the frame means that there are fewer nodes overhearing messages and spreads out the communication between unrelated receivers. Below we detail the working of the Crankshaft protocol.

The Crankshaft protocol divides time into frames, and each frame is divided into slots. There are two types of slots in the Crankshaft protocol: broadcast slots and unicast slots. During a broadcast slot all nodes wake up to listen for an incoming message. Any node that has a broadcast message to send contends with all other nodes to send that message. A frame starts with all the unicast slots, followed by the broadcast slots.

Each node also listens for one unicast slot every frame. During that slot a neighbouring node can send a message to that node, provided it wins the contention. The slot a node listens to is determined by the node's MAC address. Therefore, a node wanting to send a message knows precisely in which slot the destination wakes up. Crankshaft uses a Data/Ack sequence for unicast messages, and the slot length is such that it is long enough for the contention period, maximum-length data message and acknowledgement message. If the sender does not receive an acknowledgement, the protocol is set to retry each message three times in subsequent frames. However, to reduce contention when retrying the transmission the node will only retry in the next frame with a probability of 70%. Otherwise it will wait for another frame.

Special provisions are made for base-station or sink nodes. Sink nodes will listen to all unicast slots. The rationale for this is that the sink is the destination for most traffic in the network and therefore requires more receive bandwidth. Furthermore, the sink is typically connected to either a much larger battery or mains power, which will allow it to spend more energy. To allow other nodes to determine whether a neighbour is a sink node, sink nodes use specially reserved addresses. For example, in TinyOS node 0 is always considered a sink node.

Although many complicated methods of slot assignment are possible (e.g. Time Division Hashing [11]), we have chosen to use a simple mechanism to limit the amount of processing power required. Each frame has n unicast slots, and the slot assignment is performed by calculating *MAC address* modulo n. Using a static slot assignment like this may result in two neighbours being assigned the same slot. To allow such neighbours to communicate, nodes are allowed to act as senders in their own receive slot. A node that acts as a sender in its own receive slot will revert to receive mode if it loses contention.

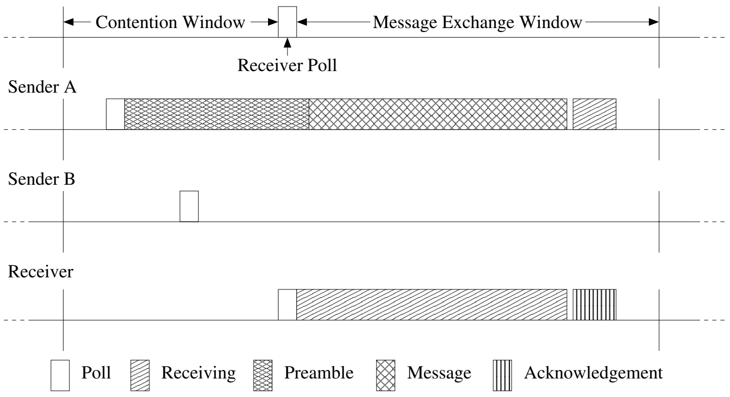

Fig. 1. Contention and message exchange in the Crankshaft protocol

Clearly, using frames requires that nodes are synchronised. Synchronisation can be achieved both through a reference node (i.e., the base-station or sink node), or through a distributed algorithm like GSA [12]. This synchronisation can also be used to achieve increased energy savings. Nodes need not wake up for an entire slot, but only for a small amount of time at a fixed offset to the start of the slot (see Figure 1). The period between the start of the slot and the moment the listening node turns on its radio is used to resolve contention. A node that wants to send a message in a particular slot chooses a moment in the contention window. The sending node listens for a short amount of time just prior to its chosen moment to detect other nodes contending for the same slot. If no other nodes are sending, the sending node starts sending a preamble to notify other nodes of its intention to send. Shortly after the receiving node is known to wake up, the sending node transmits the start symbol and the actual message. This is similar to the channel polling mechanism in the SCP-MAC protocol [5]. Note that during the contention window only the contending senders are awake, and only the winner of the contention is awake for more than a short poll. This results in very energy efficient contention resolution.

To improve contention resolution, the Crankshaft protocol also employs the Sift distribution [13] for choosing the moment to start sending. The Sift distribution is essentially a truncated geometric distribution, which results in fewer collisions than using a uniform distribution. Using Sift also reduces the average amount of time between the start of sending and the wake up moment of the receiver, saving even more energy.

The header for Crankshaft packets consists of a one byte length field, two byte *to* and *from* addresses, a one byte message type field, three bytes of clock synchronisation information, and two CRC bytes. For broadcast messages, the type field is set to broadcast and the *to* address is omitted. In our simulations, the synchronisation information contains the number of hops to the reference node, and the current (estimated) clock at the reference node.

Although not implemented in our simulations or real-world implementation, the Crankshaft protocol can use address filtering to reduce overhearing. A node would then simply turn of its radio after receiving the *to* address if the message is for another node.

4 Simulation Setup

To evaluate the Crankshaft protocol we created an implementation in our OMNeT++ [14] based simulator called MiXiM. The simulator contains a model of the EYES wireless sensor node, which includes an RFM TR1001 radio. The radio model is an SNR-based model, on top of a simple path-loss propagation model. For timing the nodes use a 32 KHz crystal, and the nodes are powered by two 1.5 V AA batteries supplying 3 V.

For our experiments we use the layout of a real-world potato-field experiment [8]. This setup includes 96 nodes on a field of approximately 90×50 meters. The simulated nodes have a radio range of 25 meters, which is similar to the radio range in the real-world experiment. The base station is situated near a corner. Average connectivity in the network is approximately 17.3.

In our simulations we have included five protocols: Low Power Listening (LPL), T-MAC, LMAC, SCP-MAC*, and of course Crankshaft. The SCP-MAC* protocol is our variation of the SCP-MAC protocol. Instead of using two contention periods, one of which the receiver overhears, it uses the Sift distribution for a single contention period before the receiver wakes up. This way the SCP-MAC protocol can easily be implemented as a variation of the Crankshaft protocol, where each slot is a receive and broadcast slot at the same time, and acknowledgements are disabled.

We have focused our simulations on traffic patterns we consider most important for wireless sensor networks: convergecast and broadcast flood. The convergecast pattern is the pattern used in most monitoring applications. All nodes periodically send data to a sink node, which then processes the data or stores the data for further processing. Broadcast floods are typically found in routing protocols and in distributing queries over the network.

Table 1 lists the simulation parameters. Each simulation lasts 200 seconds of simulated time and the results are the average of 20 runs with different random seeds. Routing is done using a static routing table. Therefore there is no routing traffic exchanged during the simulation.

Table 1. Simulation parameters

General		LMAC	
Message payload	25 bytes	Maximum data length	64 bytes
		Slots per frame	80
Radio		Packet header	20 bytes
Effective data rate	61 kbps		
Preamble + start byte	433 μs	SCP-MAC* and Crankshaft	
Transmit	12 mA	Contention window	9.15 ms
Receive	3.8 mA	Poll length	300 μs
Sleep	0.7 μA	Maximum data length	64 bytes
		Sift nodes parameter	512
LPL		Packet header (max.)	11 bytes
Sample period	300 μs		
		Crankshaft specific	
T-MAC		Unicast slots	8
Frame length	610 ms	Broadcast slots	2
Contention window	9.15 ms		
Packet header	8 bytes		
Maximum data length	250 bytes		
Activity timeout	15 ms		

5 Simulation Results

Below we present our simulation results. In Section 5.1 we present the results for the convergecast pattern followed by the results for the broadcast flood pattern in Section 5.2. In Section 5.3 we revisit the convergecast pattern results and show how the latency of the Crankshaft protocol can be improved.

5.1 Convergecast

Figure 2 shows the results of our convergecast experiments. It is clear that the LPL protocol has the highest delivery ratio (top graph) except for high message rates, but at the expense of consuming a lot of energy (middle graph). The high energy consumption is due to the the high connectivity which causes many nodes to overhear each transmission. As nodes using LPL can send their message any time, i.e. they do not have to wait for a slot or frame to start, the latency remains low until the network is saturated (bottom graph).

The delivery ratio of the SCP-MAC* protocol is clearly adversely affected by the lack of acknowledgement messages for low message rates. Even for low message rates SCP-MAC* does not achieve perfect delivery. However, the lack of acknowledgements also means that there are fewer messages to send. For high messages rates the lack of acknowledgements is beneficial. For protocols using acknowledgements the increasing collisions at high message rates will induce more retransmissions, which in turn increase the network load. This effect can be seen in that the SCP-MAC* curve does not drop like the LPL and Crankshaft curves do. SCP-MAC*'s energy consumption is much better than LPL, but for high message rates increases quickly as well.

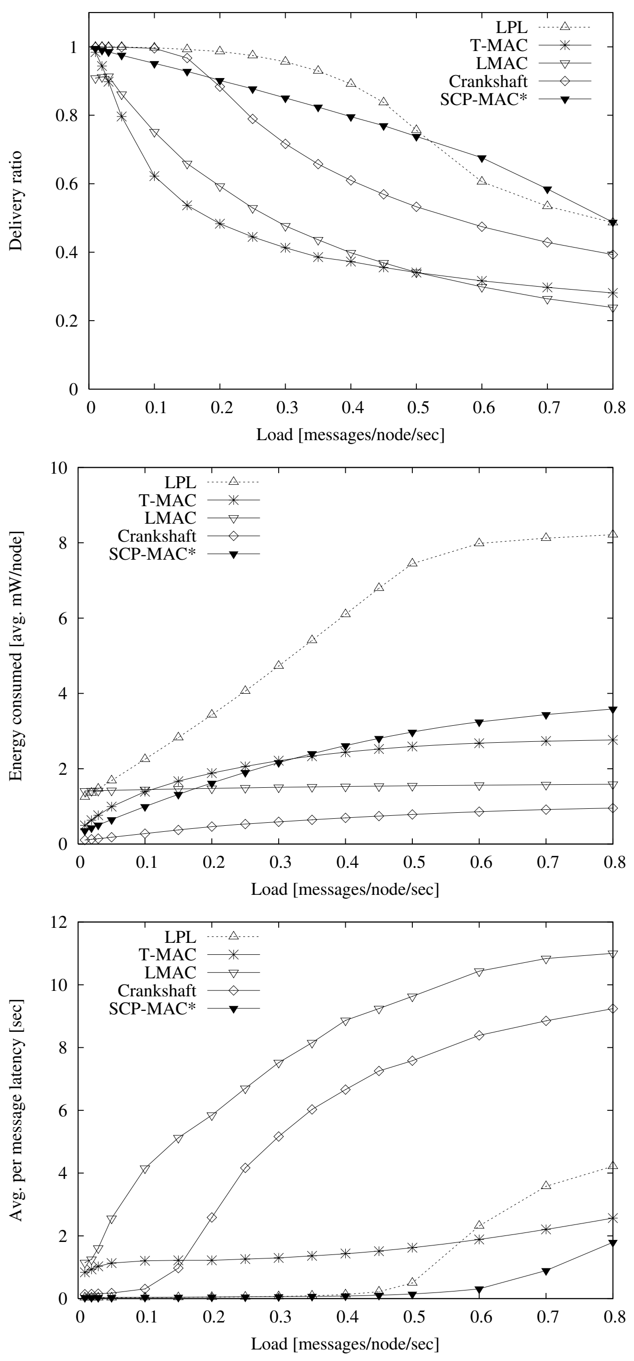

Fig. 2. Performance under convergecast traffic: delivery ratio (top), energy consumption (middle) and latency (bottom)

Compared to the LPL protocol, Crankshaft's delivery ratio starts to drop at lower message rates. This is caused by limiting the per-node receive bandwidth through selecting only one receive slot per frame. For low message rates Crankshaft does manage perfect delivery. The bandwidth limitation does mean that for medium and high message rates the Crankshaft protocol cannot achieve the high delivery ratios of LPL and SCP-MAC*.

The Crankshaft protocol is very energy efficient. It consumes a factor of 3.5 less energy than SCP-MAC*. There are two factors which contribute to the energy efficiency: firstly, because only a subset of the nodes is awake in each slot overhearing is reduced. Secondly, contention is reduced. When using SCP-MAC* and node A wants to send to node B and node C wants to send to node D, and A and C are within communications range, nodes A and C contend for the right to send. However, when using Crankshaft nodes B and D generally wake up in different slots, automatically resolving the contention between nodes A and C.

LMAC suffers from the need to assign a contention-free slot to all nodes in the network. To allow such an assignment to exist 80 slots are required. Even at low message rates the LMAC protocol does not achieve a perfect delivery ratio. Network congestion caused by low per-node bandwidth prevents this. The energy consumption remains nearly constant, because the LMAC protocol already saturates at low message rates. The small increase in energy consumption is due to more messages that are sent one hop, only to be discarded because of full message queues at the receiver. The large number of slots also induces a high message latency for the LMAC protocol.

Finally, T-MAC is unable to cope with the flood of messages directed towards the sink node. The aggressive sleep policy is causing nodes to go to sleep too often, which hinders throughput. Although latency seems low for high message rates, this is caused by only the nodes near the sink being able to reach the sink. The aggressive sleep policy does provide low energy consumption, even though the T-MAC protocol suffers from communication grouping and overhearing.

Of all the protocols compared, SCP-MAC* has the lowest latency. Again this is caused by the lack of acknowledgements and retries. Messages are not kept in queues waiting for retransmissions of other messages, which means latency is kept down. Crankshaft's latency rises quickly as the delivery ratio drops. This is partly caused by messages having to wait in queues due to congestion, and partly caused by messages having to wait a full frame (150 ms) when contention is lost. However, Crankshaft's latency can be improved, as we will show in Section 5.3.

5.2 Broadcast Flood

With our second experiment we investigate the performance under broadcast flood traffic. In this experiment the sink node initiates the floods. The delivery ratio is calculated as the total of unique flood messages received by all nodes, divided by the total of unique messages that should have been received by all nodes. To allow nodes to determine the uniqueness of a message, each message contains a serial number. Nodes compare the serial number in the message with the highest serial number received so far. If the serial number in the message is lower or equal, the message is discarded. Note that this may result in messages not previously received being discarded, if a message with a higher serial number has already been delivered. For this to happen, collisions or link errors must have resulted in the failed reception of a lower numbered message first.

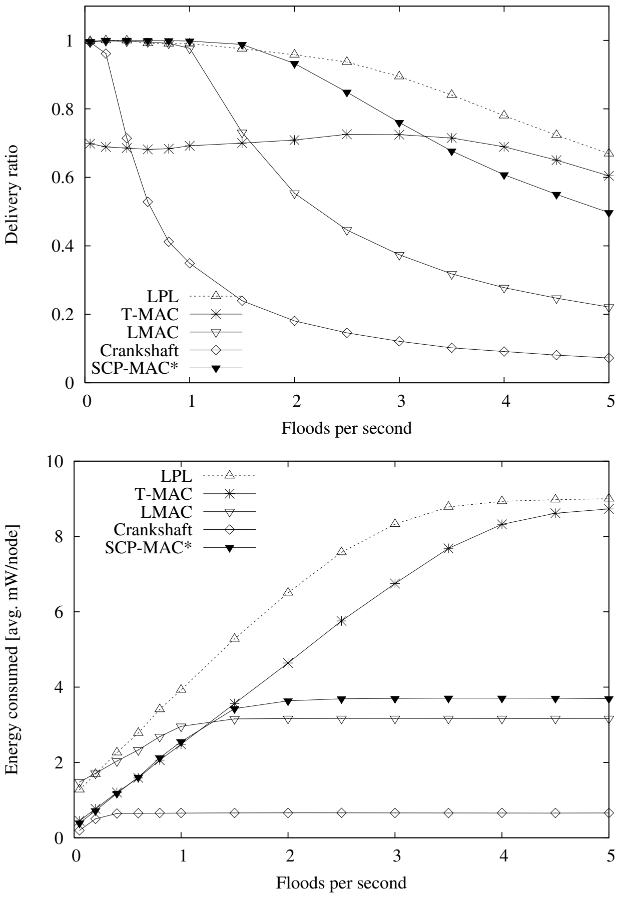

Fig. 3. Performance under broadcast-flood traffic: delivery ratio (top), energy consumption (bottom)

Figure 3 shows the delivery ratio (top) and energy consumption (bottom) for all protocols under the broadcast flood traffic pattern. LPL outperforms all other protocols in terms of delivery ratio for high message rates, but at the expense of high energy consumption. Again, because it does not use any slotting or framing it can quickly move messages through the network. Although it uses much energy, the energy is mostly put to good use.

The T-MAC curve shows a remarkable artifact. As the amount of traffic in the network increases, so does the delivery ratio. The cause is again T-MAC's aggressive sleep policy. In low traffic conditions, most of the network is asleep after the sink initiated the broadcast flood. Although all nodes that heard the sink's broadcast repeat the message, the sleeping part of the network does not receive most of these. Depending on the collisions and back-off, parts of the network may not receive any of these messages at all. Increasing the traffic load will keep more nodes awake for more time, increasing the number of nodes receiving re-sends. However, it also increases the chance of messages arriving out of order, thereby causing nodes to disregard previously unseen messages arriving later.

LMAC's performance is not hampered by the 80 slots in this scenario, because all nodes need to send the same number of messages. Therefore, there is little over-provisioning. Protocols like LPL and SCP-MAC* can support more simultaneous transmissions than LMAC, because they do not require all nodes in a two-hop neighbourhood to remain quiet. Of course this leads to collisions, but because every message is sent multiple times, although by different nodes, this is not a problem. If a node does not receive the first transmission, it probably receives a second or third. Hence, the LPL and SCP-MAC* protocols outperform LMAC.

The Crankshaft protocol shares the SCP-MAC* characteristics in the broadcast case. However, because only two out of every 10 slots are broadcast slots Crankshaft can cope with roughly one sixth of the traffic that SCP-MAC* can. It only uses approximately one sixth of the energy as well. Crankshaft's broadcast flood performance can be tuned to the amount of broadcast flood traffic is expected, by increasing the number of broadcast slots. Of course the broadcast performance is then traded for unicast performance, as unicast traffic will receive a smaller share of the available time.

5.3 Crankshaft Latency

The average message latency for the Crankshaft protocol under convergecast traffic increases quickly as the number of messages per node per second exceeds 0.15. However, the design of the Crankshaft protocol leaves an option to improve the latency. Recall that the cause of the quickly increasing latency is that a node that loses contention or does not receive an acknowledgement has to wait for an entire frame to retry. Given that a dense network provides many alternate paths to the sink, a node that loses contention or does not receive an acknowledgement could also try to send its message to another next hop.

Of course the routing layer has to cooperate with the Crankshaft protocol to make this work. To this end the interface between the routing layer and the MAC protocol is augmented. First of all, the routing layer can query Crankshaft about which of a list of neighbours wakes up first. Secondly, Crankshaft will return a message that it could not deliver in the first try to the routing layer. That is, it will not block and try at some later moment as most MAC layers do. The routing layer will then decide if and to whom to retry sending the message.

Figure 4 shows the latency (bottom) for the Crankshaft protocol with the low-latency option, marked *Crankshaft LL*. For reference the results for the regular Crankshaft protocol and the SCP-MAC* protocol are repeated.

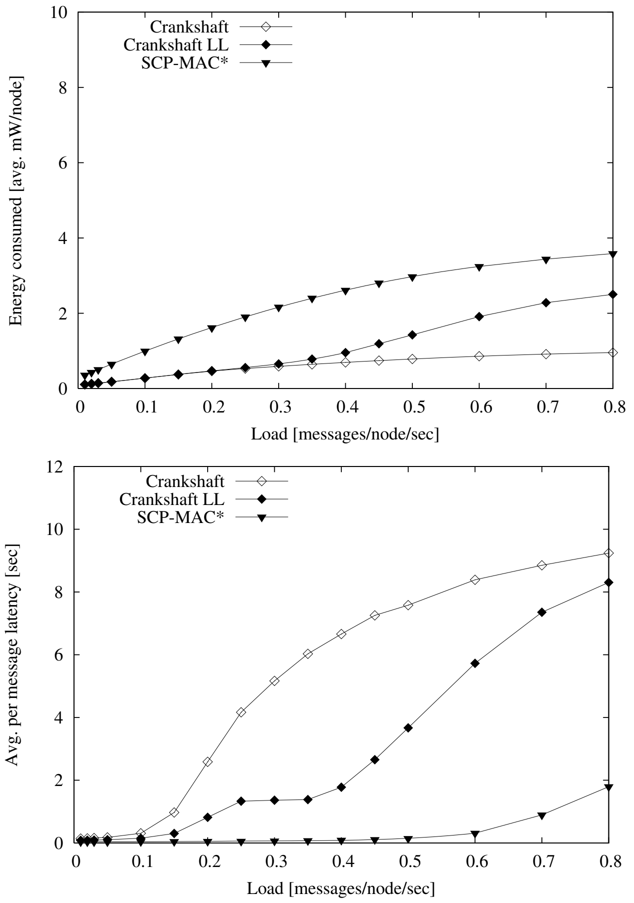

Fig. 4. Crankshaft energy consumption (top) and latency (bottom) with low-latency option

Even for Crankshaft's low-latency version, average latency starts to increase noticeably at approximately 0.15 messages per node per second. The latency is however reduced to approximately 37% of the regular Crankshaft protocol's latency. The increase at 0.15 is almost entirely due to a small group of 15 nodes that are in the corner opposite the sink. These nodes do not have many next hop neighbours. As the network load increases these nodes are having increasingly more difficulty in sending their messages to the next hop. Average latency for the messages for these nodes increases to some eight seconds and more. From 0.25 through 0.35 messages per node per second the increase in latency stops. On this interval the number of messages that arrive at the sink from the 15 distant nodes decreases in such a way that it compensates for the increased latency. Most of the other 80 nodes still have average per message latencies in the range of 0–0.7 seconds. Using the regular Crankshaft protocol, approximately half of the nodes already have average per message latencies (much) larger than one second. For message rates larger than 0.35 the other 80 nodes are also seriously affected by the increasing network load. This is reflected by the increasing latency.

As can be seen from the top graph in Figure 4, Crankshaft's energy consumption does not change for message rates below 0.35 messages per node per second. At higher rates the energy consumption does increase significantly, to 2.7 times the energy consumption without the low-latency option at a rate of 0.8 messages per node per second. The congestion in the network is causing increasing numbers of retransmissions, which consume much energy. The delivery ratio is affected very little by the low-latency option. For message rates between 0.15 and 0.5, at most 7 percentage points are gained, while at 0.8 approximately 10 percentage points are lost.

6 TinyOS Implementation

To demonstrate the real world feasibility of the Crankshaft protocol, we have created an implementation of Crankshaft for the TinyOS operating system. We have tested our implementation by running a convergecast scenario on our 26 node testbed consisting of mica2 clones named TNOdes. The nodes in our testbed can all hear one another, but not all pairs of nodes have good links. We have created a static routing table in which 12 of the 26 nodes use other nodes to forward messages to the sink. All nodes, with the exception of the sink, repeatedly send messages to the sink at fixed intervals. For comparison, we have included our implementations of T-MAC and LMAC, and the default MAC protocol for mica2 nodes, i.e. BMAC.

Table 2 shows the parameters for the experiment and the protocols. We have strived to make the parameters match those of our simulations as closely as possible, but the hardware used does dictate many of the protocol timings. Also, the physical layers in our TinyOS framework include extra header fields, and the time synchronisation fields take five bytes instead of three. Therefore the packet headers are longer than in simulation. Furthermore, the BMAC protocol as implemented in TinyOS does not provide for retries, where the simulated LPL protocol does. The Crankshaft protocol is implemented without the use of the Sift distribution. A standard uniform distribution is used instead for ease of implementation.

The experiments only show the results of a single run, due to time constraints. Therefore it is not possible to draw firm conclusions from these results. However, the results do provide an indication of the performance.

Figure 5 shows our results. The first thing to note is that even for low message rates, none of the protocols achieves perfect delivery. This is due to the use of

Table 2. Real-world experiment parameters

Experiment		LMAC	
Time per message rate	5 min	Slots per frame	32
Message payload	10 bytes	Packet header	24 bytes
Maximum data length (all protocols)	29 bytes		
		Crankshaft	
		Contention window	5.8 ms
Radio		Poll length	4 ms
Effective data rate	19 kbps	Packet header	18 bytes
Preamble + start byte	2.1 ms	Unicast slots	8
		Broadcast slots	2
T-MAC			
Frame length	610 ms	BMAC	
Contention window	5.8 ms	Sample period	20 ms
Packet header	18 bytes		
Activity timeout	18 ms		

non-perfect radio links. The LMAC protocol copes best, because only one node will try to send in each slot, ensuring the best possible link quality.

The LMAC protocol also shows the same sudden drop in delivery ratio as our simulation results. Again this is due to the bandwidth limitations imposed by the TDMA structure of the protocol. The T-MAC protocol shows erratic behaviour for low message rates. This is probably due to the aggressive sleep strategy used in the T-MAC protocol. As in the simulations, the T-MAC protocol shows it does not deal well with high load situations.

The Crankshaft protocol and the BMAC protocol show similar performance. For low message rates, the use of retries in the Crankshaft protocol means its delivery ratio is higher than BMAC's. For high message rates the difference between Crankshaft and BMAC is too small to draw any conclusions.

7 Discussion

The simulations results show that the Crankshaft protocol can provide good convergecast performance at low energy consumption in dense networks. Latency can be a problem in the unoptimised protocol. However, by spending a little more energy latency can be kept down if the routing layer is modified as well. Furthermore, message rates of more than 0.5 messages per node per second are infrequent in monitoring applications.

The trade-off is that broadcast bandwidth is significantly reduced. This means that to use the Crankshaft protocol effectively, broadcast flooding must be minimised. Flooding is used mostly for two purposes in WSNs: collecting routing information and pushing queries into the network. For the former purpose broadcast flooding is required to operate correctly. The latter can be achieved by more bandwidth efficient broadcasting schemes, especially in dense networks. This would eliminate the need for high broadcast throughput. Also, sustained flood rates of more than one flood per second are probably excessive.

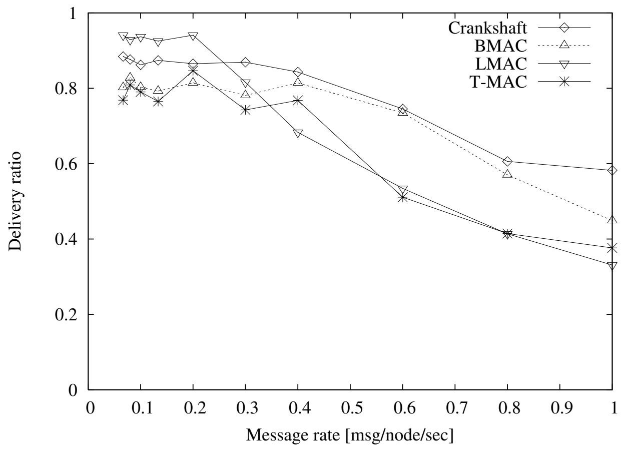

Fig. 5. Delivery ratio for convergecast with TinyOS implementation

A point to consider with respect to the implementation of the Crankshaft protocol is the recent trend towards packet-based radios. To implement Crankshaft variable preamble lengths are required. Packet radios in general do not provide sufficient flexibility to implement this directly. However, the creators of the WiseMAC protocol [9] already provide a solution: repeatedly sending the packet for the required duration. Although this may seem wasteful, one has to consider that sending is the infrequent operation. Letting the sender spend more energy to allow multiple (overhearing) receivers to save energy is usually more energy efficient on balance.

The Crankshaft protocol can also be tuned to the specific application and network density. By increasing the number of unicast slots the consumed energy can be brought down further, although maximum bandwidth will be decreased and latency will go up. By decreasing the number of unicast slots, the maximum per node bandwidth can be increased and the latency decreased, at the expense of more energy consumption and more collisions. Similar considerations hold for the number of broadcast slots.

7.1 Variations

The Crankshaft protocol design allows for some variations in the slot assignment mechanism. For example, we have tested using meta frames consisting of four frames. Each of the four frames would use a different assignment calculation, in our case different parts of the MAC address.

As nodes are usually numbered sequentially, simply using the higher bits of the MAC addresses would not give enough variation. Therefore the calculations were more complex than simple modulo calculation. We also disallowed sending in a node's receive slot, because there always is at least one of the four frames in which the slot assignment for two neighbours differs.

Although this assignment scheme resulted in a small performance increase, we felt the difference did not compensate for the added code complexity and processing requirements.

Other options we tested are Time Division Hashing [11], and using broadcast slots for unicast if two neighbours share a slot assignment. None of these options gave a significant performance benefit, so they are not included in the results.

8 Conclusions and Future Work

In this paper we have presented Crankshaft, a MAC protocol for dense wireless sensor networks. It employs receive slot scheduling to reduce overhearing and bases the schedule on MAC addresses to obviate the need to keep neighbour state. We have shown through simulation and real-world experiments that Crankshaft can provide good delivery ratios in convergecast scenarios with respect to Low Power Listening, T-MAC, LMAC and SCP-MAC*. Simulations also show that Crankshaft manages to do so while consuming very little energy.

Further simulations show that the Crankshaft protocol cannot provide good broadcast flood delivery. However, Crankshaft's energy consumption does not suffer from broadcast flooding either. Most applications only require sporadic flooding, for which Crankshaft provides adequate performance.

We conclude that the Crankshaft protocol is suitable for long-lived monitoring applications, where energy efficiency is key. The Crankshaft protocol can provide the required delivery ratios for convergecast traffic at extremely low energy consumption.

The Crankshaft protocol as proposed has a very rigid structure. This limits the applicability of the protocol. For future work we intend to look at adaptive scheduling of extra receive slots to facilitate more application types and mitigate bottlenecks that can occur near sink nodes and elsewhere in the network. Another research item is improving the broadcast capabilities of the Crankshaft protocol.

A final research direction for the Crankshaft protocol is leveraging multichannel radios. In principle this is as simple as assigning nodes a channel as well as a time within a frame.

Acknowledgements

We would like to thank the anonymous reviewers for their comments. Furthermore we would like to thank Otto Visser and Ivaylo Haratcherev for their work on the testbed and Tom Parker for his work on his TinyOS MAC-framework which made developing the TinyOS implementation so much easier.

References

1. Ye, W., Heidemann, J., Estrin, D.: An energy-efficient MAC protocol for wireless sensor networks. In: Proc. of the 21st Conf. of the IEEE Computer and Communications Societies (INFOCOM). Volume 3. (2002) 1567–1576

2. van Dam, T., Langendoen, K.: An adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proc. of the 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, CA (2003) 171–180
3. Hill, J., Culler, D.: Mica: a wireless platform for deeply embedded networks. IEEE Micro **22** (2002) 12–24
4. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: Proc. of the 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, MD (2004) 95–107
5. Ye, W., Heidemann, J.: Ultra-low duty cycle MAC with scheduled channel polling. Technical Report ISI-TR-604, USC Information Sciences Institute (2005)
6. van Hoesel, L., Havinga, P.: A lightweight medium access protocol (LMAC) for wireless sensor networks. In: Proc. of the 1st Int. Workshop on Networked Sensing Systems (INSS 2004), Tokyo, Japan (2004)
7. Zheng, T., Radhakrishnan, S., Sarangan, V.: PMAC: an adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proc. of the 19th IEEE Int. Parallel and Distributed Processing Symp. (IPDPS'05). (2005) 65–72
8. Langendoen, K., Baggio, A., Visser, O.: Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In: Proc. of the 14th Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), Rhodes, Greece (2006)
9. El-Hoiydi, A., Decotignie, J.D.: WiseMAC: An ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks. In: Proc. of the Ninth Int. Symp. on Computers and Communications, 2004 (ISCC 2004). Volume 1. (2004) 244–251
10. Ye, W., Heidemann, J., Estrin, D.: Medium access control with coordinated, adaptive sleeping for wireless sensor networks. IEEE/ACM Trans. on Networking **12** (2004) 493–506
11. Cheng, W., Lee, I.T.A., Singh, N.: Time division hashing (TDH): A new scheduling scheme for wireless ad-hoc networks. In: Proc. of the Int. Symp. on Advanced Radio Technologies (ISART). (2005) 91–100
12. Li, Y., Ye, W., Heidemann, J.: Energy and latency control in low duty cycle MAC protocols. In: Proc. of the IEEE Wireless Communications and Networking Conf., New Orleans, LA, USA (2005)
13. Jamieson, K., Balakrishnan, H., Tay, Y.C.: Sift: a MAC protocol for event-driven wireless sensor networks. In: Proc. of the 3rd European Workshop on Wireless Sensor Networks (EWSN). (2006) 260–275
14. Varga, A.: The OMNeT++ discrete event simulation system. In: Proc. of the European Simulation Multiconference (ESM'2001), Prague, Czech Republic (2001)