Y-MAC: An Energy-efficient Multi-channel MAC Protocol for Dense Wireless Sensor Networks

Youngmin Kim, Hyojeong Shin, and Hojung Cha
Department of Computer Science
Yonsei University
Seodaemun-gu, Shinchon-dong 134, Seoul 120-749, Korea
{ymkim, hjshin, hjcha}@cs.yonsei.ac.kr

Abstract

As the use of wireless sensor networks (WSNs) becomes widespread, node density tends to increase. This poses a new challenge for Medium Access Control (MAC) protocol design. Although traditional MAC protocols achieve low-power operation, they use only a single channel which limits their performance. Several multi-channel MAC protocols for WSNs have been recently proposed. One of the key observations is that these protocols are less energy efficient than single-channel MAC protocols under light traffic conditions. In this paper, we propose an energy efficient multi-channel MAC protocol, Y-MAC, for WSNs. Our goal is to achieve both high performance and energy efficiency under diverse traffic conditions. In contrast to most of previous multi-channel MAC protocols for WSNs, we implemented Y-MAC on a real sensor node platform and conducted extensive experiments to evaluate its performance. Experimental results show that Y-MAC is energy efficient and maintains high performance under high-traffic conditions.

1. Introduction

Sensor nodes are typically battery powered and operate in unattended environments. Therefore, maximizing the energy efficiency of the nodes is important in order to prolong network lifetimes. Since the radio module is a major energy consumer in a sensor node, much research has been devoted to designing energy efficient MAC protocols.

S-MAC [1] uses several techniques to reduce energy consumption of sensor nodes. Neighboring nodes form a virtual cluster to auto-synchronize their sleep schedules. Nodes periodically sleep and wake up to reduce idle listening overhead. S-MAC also implements RTS/CTS in order to reduce collisions and avoid overhearing.

Low Power Listening (LPL) [2] combines the low-level carrier sense technique with CSMA, and Preamble Sampling [3] also proposes a similar algorithm. Nodes are duty-cycled through periodic channel sampling. By stretching the preamble of a message so that it is longer than the sleep interval, senders are able to wake up receivers. WiseMAC [4] and B-MAC [5] are advanced versions of LPL. WiseMAC avoids long preambles by learning the sampling schedule of neighboring nodes. B-MAC supports run-time reconfiguration to reduce duty cycle and minimize idle listening. In order to save energy at non-target receiver nodes, X-MAC [6] uses a series of short preamble packets containing target address information. To shorten the preamble length, the receiver sends an early acknowledgement to the sender, in the short pause time between preamble packets. These contention-based MAC protocols can flexibly adapt to diverse traffic conditions by adjusting the duty cycle. However, they sacrifice energy during in the contention period.

To guarantee collision-free communication, several TDMA-based MAC protocols have been proposed. PEDAMACS [7] uses an access point to schedule node transmission and reception. The access point explicitly schedules all the nodes, based on its knowledge of the topology of the whole network. LMAC [8] uses a distributed time slot selection mechanism. Each node is able to send out a message collision-free since it owns an exclusive time slot in a two-hop neighborhood. However, all the nodes have to wake up at every time slot in order not to miss incoming messages. While LMAC schedules senders, Crankshaft [9] schedules receivers, allocating time slots to the nodes for data reception. Because each node wakes up for data reception at a different offset from the start of the super frame, the number of nodes overhearing unrelated messages is reduced.

Since the aforementioned MAC protocols focus on the efficient use of energy, they have difficulties in handling bursty traffic. Typical sensor applications, such as habitat/environmental monitoring and infrastructure diagnostics, require low data rates and their communication patterns are periodic. With the widespread use of sensor applications, however, the node density in WSNs becomes higher. Moreover, some of latest operating systems for WSNs enable sensor nodes to run multiple applications. This leads to higher packet density on the network, and thus handling bursty traffic has become a major issue in MAC design. To address this issue, Funneling-MAC [10] and Z-MAC [11] propose hybrid approaches, combining the advantages of contention-based protocols and time-slotted protocols. SCP-MAC [12] suggests not only synchronous channel polling to reduce energy wastage, but also a multi-hop streaming scheme to handle bursty traffic towards base stations.

Recently, several researchers have explored the possibility of using multiple channels to overcome the limitations of single channel MAC protocols. Ansari, Zhang and Mähönen [13] propose a multi-radio MAC protocol running on a sensor node platform equipped with two radio transceivers. This approach is not an economical solution for WSNs, hence devising a multi-channel MAC protocol using a single radio transceiver would be a better solution. Most commercial radio devices, such as the CC1000 [14] and CC2420 [15], already provide the basic functions required to support multiple channels.

This paper proposes an energy-efficient multi-channel MAC protocol, Y-MAC, for wireless sensor networks. The protocol fully describes our previous demonstration [16]. The main contributions of this paper are as follows:

- We propose a light-weight channel hopping mechanism. Y-MAC avoids redundant channel assignment by not allocating fixed channels to the nodes. Initially, messages are exchanged on the base channel. When a traffic burst occurs, a receiver and potential senders hop to one of the other available channels, according to the hopping sequence. Since these messages are carried over additional channels, each node is guaranteed to receive at least one message on the base channel.

- Most of the abovementioned multi-channel MAC protocols for WSNs have only been evaluated through simulation experiments. To validate the practicality of the proposed algorithm, we implemented Y-MAC in the RETOS operating system [17], and compared it with other published MAC protocols using a set of TmoteSky [18] sensor nodes.

The rest of this paper is organized as follows. In Section 2, we discuss related research. Section 3 describes the design of Y-MAC. Section 4 details its practical implementation and some related issues. In section 5, we evaluate Y-MAC through extensive experiments. We conclude the paper in Section 6.

2. Related work

Several multi-channel MAC protocols have been studied for general ad-hoc networks [19] [20] [21] [22]. The protocols are, however, not suitable for WSNs because of their limitations in terms of resources, computing power and cost. Recently, a few multi-channel MAC protocols for WSNs have been proposed.

MMSN [23] is the first multi-channel MAC protocol which takes into account the restrictions imposed by WSNs. The protocol suggests four strategies for assigning different frequencies to the nodes. Each node is assigned a physical frequency for data reception. With the assigned frequencies, nodes cooperate to maximize parallel transmission among neighboring space.. Although MMSN achieves increased network throughput, the fixed channel allocations limit channel utilization. In general, the amount of data passing through a node changes depending on its current position in the routing path. Moreover, the routing topology varies dynamically because WSNs are susceptible to changes in the surrounding environment. Therefore, MAC protocols for WSNs should flexibly allocate network resources, such as frequencies and time slots, to the nodes in the network. In order to avoid static channel assignment, Xun et. al. [24] did not allocate a fixed channel to each node, and assumed all the nodes in the network to be clustered. In their protocol the cluster head collects request messages from the cluster members, and then distributes channels to both the source and the destination nodes. After receiving a schedule from the cluster head, node pairs communicate on the designated channels. Although this coordinator-based mechanism is able to increase the total sleep time of the nodes, the maximum network throughput of the cluster is limited by the number of request packets which can be managed by the cluster head. Moreover, cluster heads consume more energy than standard nodes, which can be a serious problem if all the nodes are physically homogeneous. Durmaz Incel, Dulman, and Jansen [25] propose a multi-channel MAC protocol based on LMAC. Nodes communicate on the basic channel at first, and once all the time slots on the basic channel are exhausted, new channels are introduced. Since nodes are scattered over several channels, they are linked through bridge nodes. One disadvantage of this protocol is that if two nodes positioned within a one-hop distance are located on different channels, they still have to communicate via a bridge node. This results in increasing packet latency and additional energy consumption by bridge nodes.

While these multi-channel MAC protocols were evaluated only through theoretical analysis and simulation results, McMAC [22] has been evaluated by implementing a simplified version of the protocol on a general sensor node platform. McMAC pairs a random channel hopping algorithm with a rendezvous scheme. However, its performance in WSNs is in doubt because the scheme was originally designed for general wireless networks, such as 802.11a/b/g.

3. The Y-MAC protocol

This section describes the detailed design of Y-MAC, and related issues.

3.1. Frame architecture

Y-MAC is a TDMA-based multi-channel MAC protocol. In general, TDMA-based MAC protocols allocate a time slot to each node in the network. The allocated time slot is used for data transmission or data reception according to the protocol. We define a time slot for data transmission as a send time slot, and a receive time slot is defined accordingly.

If each node has an exclusive send time slot in two-hop neighborhood, collision-free access to the medium is guaranteed. Such a scheme is thus able to reduce energy wasted by contention and collisions. However, all nodes must wake up at every time slot so as not to miss incoming messages. This results in energy wastage due to idle listening and overhearing. In the case of the latest commercial radio transceivers for WSNs, energy consumption while receiving is even greater than while transmitting due to the sophisticated de-spreading and error correction techniques [26]. Therefore, scheduling receivers is more energy efficient than scheduling senders under light traffic conditions, because each node samples the medium only in its own receive time slot. Although potential senders compete to seize the medium in a CSMA fashion, the contention level is relatively low since contention among senders which have different destination nodes is eliminated. To achieve low energy consumption under light traffic conditions, we adapted this scheme.

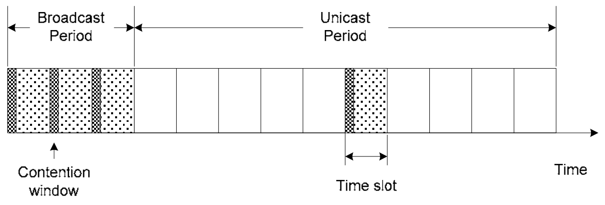


Figure 1. Frame architecture of Y-MAC

Figure 1 illustrates the frame architecture of Y-MAC. Time is divided into several fixed-length frames, and each frame is composed of a broadcast period and a unicast period. Since the wake up times for nodes are dispersed, every node must wake up at the start of the broadcast period to exchange broadcast messages. If there are no incoming broadcast messages, each node turns off its radio until its own receive time slot to save energy. Determining the number of time slots is important because there is a tradeoff between the number of time slots and the delivery latency. The more time slots we have, the more nodes we can allocate exclusive time slots to, but delivery latency increases due to the prolonged length of the frame period. One alternative approach is to increase the number of possible time slots using multiple channels. This requires complex operations.

3.2. Time synchronization

Since several channels are available, a sender and a receiver have to agree on the communication channel as well as the transmission timing. This necessitates time synchronization algorithms for typical multi-channel MAC protocols. Although some time synchronization techniques are available [27] [28] [29] [30], we use a simple time synchronization technique to decrease synchronization overhead. In our protocol, sensor nodes synchronize their upcoming timer events by exchanging the time remaining in the current superframe period, not just agree on a common clock. This scheme can be easily implemented by adjusting the expiration times of timer events.

3.2.1. Initial time synchronization. Time synchronized nodes periodically broadcast the information required for time synchronization. This consists of the time remaining to the start of the next frame period, and the sequence number originated from the sink node. We assume that there is only one sink node in the network to simplify the explanation. Our protocol is able to accommodate several sink nodes if additional memory space is allocated.

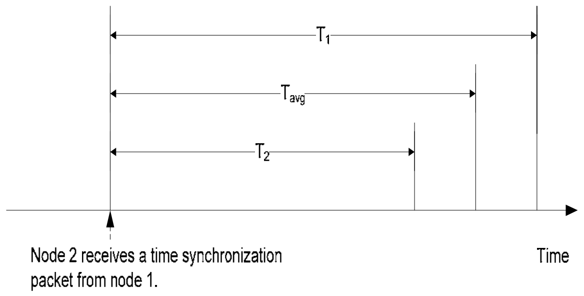


Figure 2. Time synchronization technique

The sink node starts normal operation soon after being booted, and periodically broadcasts control messages to initiate the network. A node which is trying to join the network turns on its radio transceiver to receive this timing information. We set a wait time equivalent to the time interval between control messages. Once a node receives the first control message, it sets its time remaining to the next frame period to equal that of the sender. This aligns the superframe periods of the two nodes.

3.2.2. Error compensation. Since the crystal clocks used for general sensor node platforms are typically cheap and inaccurate, all nodes have to communicate periodically in order to compensate for time synchronization errors resulting from clock drift. An example of error compensation is shown in Figure 2. T1 and T2 represent the time remaining to the next frame period for node 1 and node 2. When node 2 receives the time synchronization information from node 1, it averages the time remaining and adjusts the expiration time of its timer event. As a result, the starting points for the next frame period of these two nodes get closer. To lessen the control overhead for time synchronization, the timing information is included in control messages that every node periodically broadcasts to maintain network connectivity.

3.2.3. Network partition detection and reassociation. Ad-hoc wireless networks are often partitioned. This can be caused by a number of factors, including node failure, flat batteries, and the presence of obstacles. If a node has not received any control messages with fresh sequence numbers during a predefined time, it is considered to be detached from the network. The waiting time for such control messages should be carefully determined, since reliable broadcasting is hard to achieve in WSNs. Our protocol separates broadcast traffic from unicast traffic. This makes broadcasting is more reliable than in the other MAC protocols that do not separate different types of traffic. In our implementation, the wait time is three times as large as the control message interval. This empirical value was sufficient to detect a network partition in our implementation.

2 bytes	1 byte	2 bytes	1 byte	2 bytes
Node address	Occupying time slot	Slot allocation vector	Sequence number	Time remaining of the current frame period

Figure 3. Control message in Y-MAC

If a node detects a network partition, it goes into the sleep mode to save energy. To reassociate with the network, the radio is periodically turned on to perform the bootstrap phase described earlier. Unless it receives any control messages, the radio is turned off again. Initially, the sleep interval is short. Whenever a node wakes but fails to rejoin the network, the interval doubles to save energy consumption from periodic listening. However, a maximum value for the sleep interval is defined to guarantee a reasonable reassociation time.

3.3. Time slot assignment and retrieval

Time slot assignment algorithms are classified as either coordinated methods or distributed methods. To distribute control overhead evenly, we adapted a distributed method based on algorithms proposed in LMAC [8] and MMSN [23]. Every node maintains a slot allocation vector, storing occupied time slots within its own one-hop neighborhood including itself. Each bit in the slot allocation vector is set if the time slot at the same position is occupied. This information about time slot assignment is broadcast with the control messages. After completing its initial time synchronization, each node collects information about occupied time slots in its two-hop neighborhood by OR-ing slot allocation vectors upon receiving a control message. The collection time must be greater than the control message interval to allow for the possible loss of control messages. If there are many available time slots, one of them is randomly selected, otherwise several nodes should share the same time slot. The total number of time slots should therefore be determined considering the estimated node density.

The next issue to be considered is node removal. If a node runs out of battery, or is removed from the network, the time slot that has been used by that node must be released for the future use. A node is considered to have been removed when its control message has not arrived during a predefined time. Neighboring nodes update their slot allocation vectors once the removal of the node is noticed.

Figure 3 shows the structure of the control message used in Y-MAC. The message consists of a node

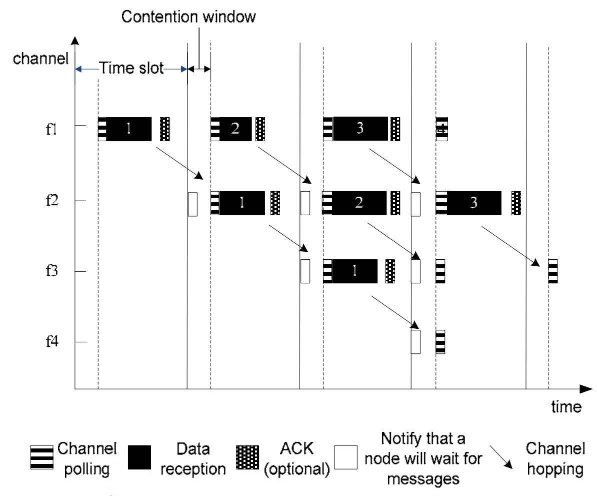


Figure 4. A light-weight channel hopping mechanism

address, its own time slot, a slot allocation vector, a sequence number originated from the sink node, and information for time synchronization.

3.4. Medium access design

The medium access design of Y-MAC is based on synchronous low power listening. We define the time slot length to be long enough to receive one message. Therefore, only the contention winner can transmit a message to the destination node. Contention between potential senders is resolved in the contention window. The node wishing to send a packet sets a random back-off value within the contention window. When the back-off timer is fired, the node wakes up and checks the medium for a certain amount of time. If the channel is clear, a preamble is transmitted until the end of the contention window to suppress competing transmissions. The receiver wakes up at the end of the contention window to receive the data part of the message. If it receives no signal from neighboring nodes, the receiver node turns off its radio device. This scheme can be easily implemented in bit-stream oriented RF transceivers such as the CC1000. In the case of packet oriented RF transceivers, however, additional techniques, proposed in [6] [12] [31], are needed.

3.4.1. Broadcast messages. Broadcast messages are exchanged only within the broadcast period. At the beginning of the broadcast period, every node tunes to the base channel. If a node has a pending broadcast message, it takes part in the contention process described above. Only the contention winner transmits the data part of its message, and the other nodes listen to it. If a node did not receive or send a broadcast packet in the broadcast time slot, the radio transceiver is turned off to conserve node energy. To prioritize to control messages, Y-MAC sets their back-off timer values shorter than for general broadcast messages.

3.4.2. Unicast messages. Unicast messages are initially exchanged on the base channel. Each node changes its frequency to the base channel at the beginning of its own receive time slot, and potential senders also do so. After receiving a unicast message from the contention winner, the receiver sends an acknowledgement to the sender to confirm delivery success, if the acknowledgement request flag was set in the message. If reliable transmission is critical, the application is able to set this flag to request an acknowledgement. Otherwise, the flag should be cleared to reduce communication overhead. We use one of the reserved fields in the IEEE 802.15.4 MAC header to indicate an acknowledgement request.

This scheme is energy efficient under light traffic conditions, since every node polls the medium only during the broadcast time slots and its own unicast receive time slot. Under heavy traffic conditions, however, many unicast messages may have to wait in the message queue, or are dropped due to the limited per-node bandwidth. To address this problem, we propose a light-weight channel hopping mechanism, exploiting multiple channels to reduce the packet delivery latency. This mechanism is shown in Figure 4. We assume that four channels are available. F1 is the base channel. If a node receives a unicast message on the base channel, it hops to the next channel to receive the following message. The next channel is calculated by the hopping sequence generation algorithm. Any nodes that have pending messages destined to the same receiver also hop to the same channel and compete again. In this way, bursts of messages ripple across channels, and only one node uses the base channel at any one time. To guarantee per node fairness, we give a penalty to the contention winner by limiting the range of its back-off timer value for the next transmission.

Although the channel number increases sequentially in this figure, other algorithms can be used to generate a hopping sequence. In order to space out ripples, however, the sequence number generation algorithm must guarantee that there should is only one node among one-hop neighbors on any particular channel. The algorithm should also be deterministic, so as not to require state exchange to arrange the rendezvous.

The question here is how to notify the contention losers whether the receiving node will wait during the next time slot or not. There are two options. The first is to insert a flag into the acknowledgement, and the second is to transmit a small and independent packet at the start of the time slot. With the first option,

Algorithm 1. A node in the unicast time slot

```
if it received a unicast message in the previous time slot
    hops to the next channel and notifies that it will
    continue to receive in current time slot then{
   if there is an incoming message then
       receives a message;
   else turns off the radio;
}
else if it took part in contention in the previous time
   slot and still has messages destined to the previous
   destination node then{
       hops to the same channel which the receiver
       locates in;
       if it has been informed that the destination node
           will continue to receive in the current time slot
           then
               participates in the contention;
           else turns off the radio;
}
else if it has some pending messages in the current time
slot then{
    hops to the base channel and participates in
    contention;
}
else if the current time slot is its own time slot then {
   hops to the base channel;
   if there is an incoming message then
       receives a message;
   else turns off the radio;
}
else Turns off the radio;
```

contention losers must overhear the acknowledgements. Since receivers send acknowledgements to senders selectively, we chose the second option.

Algorithm 1 states the control flow of a node in the unicast time slot. A node may also be a sender in its own receive time slot if two neighbors hold the same time slot due to lack of available time slots.

4. Implementation

We implemented Y-MAC in the RETOS operating system on the TmoteSky motes. RETOS is an operating system for WSNs developed at Yonsei University. The network architecture of RETOS is layered. Figure 5 illustrates the overall network architecture of RETOS. It consists of three layers: the dynamic networking layer (DNL), the networking supporting layer (NSL), and the medium access control (MAC) layer. User applications need only to interact with the DNL to deliver messages, and different routing modules can be selected taking into account the characteristics of user applications. The NSL is responsible for neighborhood management, and maintains neighborhood information in the neighbor table. New nodes are added to the neighbor table, and nodes are removed from the table if their periodic control messages have not arrived for a predefined period of time. The bottom layer is the MAC layer. We break the MAC layer into five components: back-off timer, access control, radio device driver, SPI bus control, and message queue. For our implementation of Y-MAC in RETOS, we also separate the broadcast message queue and the unicast message queue. All of the Y-MAC parameters, including the queue size, are determined at compile-time. The most important of these four components is the access control component which coordinates medium access by nodes. MAC designers can easily implement new MAC protocols by rewriting this component.

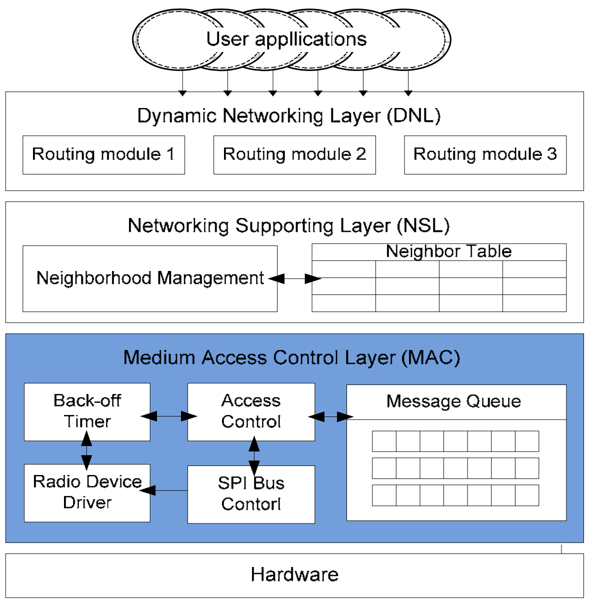


Figure 5. RETOS network architecture

General TDMA-based MAC protocols maintain neighbor information that includes assigned time slots. Since the NSL is in charge of neighborhood management, MAC implementers can concentrate on the access control component, keeping the source code simple and concise.

The main difficulty in implementing any TDMA-based MAC protocol is time synchronization. Every node in the network periodically communicates with other nodes to compensate for synchronization errors caused by clock drift. To timestamp the time remaining in the current frame period in the control message, we used a technique proposed in ETA [32]. Besides errors from clock drift, another problem is caused by the timer queue. TDMA-based MAC protocols should maintain an accurate time interval between two consecutive frame periods. Like other operating systems, RETOS provides a common timer queue.

Kernel developers and application developers can register timer events through the set_timer() system call which takes an upcoming timeout as a parameter. Since RETOS implements a common timer queue as a variable timer, the kernel has to reprogram the timer tick rate for every timer request. We initially implemented Y-MAC using this variable timer. When a new frame period is invoked, Y-MAC registers the next frame period as a timer event, but delays in the registration and deletion of timer events caused jitters. Moreover, activated timer requests should wait in the bottom half to be executed for a while. For these reasons, we added a new timer queue dedicated to Y-MAC. This saves the starting time of the previous frame period. By adding this value and the event interval, Y-MAC maintains a constant time interval between timer events.

5. Evaluation

For comparison purposes, we also implemented LPL [2] and Crankshaft [9] in RETOS. Our version of LPL was based on the technique proposed in [31], and Crankshaft was implemented by modifying some features of Y-MAC.

We evaluated performance in terms of three metrics: energy efficiency, delivery latency and reception rate. We used duty cycle as an indicator of energy efficiency, because accurately measuring the energy consumption of sensor nodes is difficult.

5.1. Time synchronization error

Since precise time synchronization between nodes is crucial to implementing multi-channel MAC protocols, we conducted some preliminary experiments. Twenty TmoteSky motes were used, with attenuated radio transmission power to construct a multi-hop environment.

Table 1. Time synchronization experiments

Number of nodes	20
Avg. number of neighbors	4.5
Spacing between nodes	2m
Time synchronization message interval	8 sec.
Avg. error with neighbors	1.65 ticks

Table 1 summarizes the parameters and the results of the time synchronization experiments. The experiments were conducted for one hour. Every node broadcast its time remaining in the current frame period every 8 seconds. Each node recorded the time difference between nodes when it received time synchronization messages from the neighboring nodes. The average time difference was reported to the sink node. The results show that the overall synchronization errors are acceptable for implementing a multi-channel MAC protocol in general sensor node platforms. We used an auxiliary clock sourced from a 32.768 kHz crystal oscillator, hence 1 tick is equal to 1/32 ms.

Table 2. Experimental parameters

Low Power Listening	
Sleep interval	300 ms
Channel polling duration	4 ms
Crankshaft and Y-MAC	
Contention window	15 ms
Message exchange window	15 ms
Channel polling duration	4 ms
Number of broadcast slots	2
Number of unicast slots	8
Packet header length	16 bytes
Payload length	32 bytes
Control message interval	15 secs
Maximum retransmission	2
Y-MAC specific	
Number of channels	5
Wait for a notification message	3 ms

5.2. Experimental setup

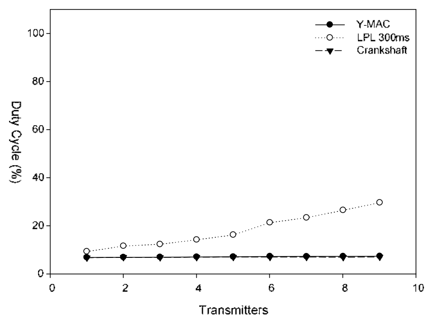
The parameter settings of our experiments are shown in Table 2. To compare the three MAC protocols fairly, we set the sleep interval for LPL to equal the frame length of Crankshaft and Y-MAC. RETOS is a multi-threaded operating system, so when a node receives a message it wakes up a blocked thread. Taking into account this wake-up delay, and the time to handle a message, we set the message exchange window to 15 ms.

5.3. Performance in single-hop environments

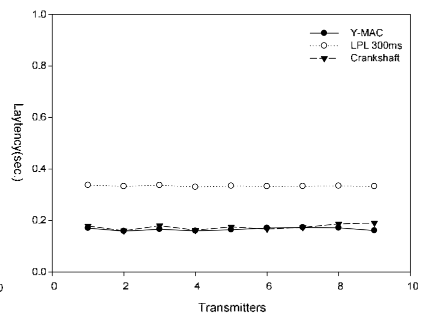
We evaluated the basic performance of the three MAC protocols in single-hop environments. Every node periodically transmits a message to a single receiver. We varied the number of transmitters to analyze the relationship between performance and node density.

In Crankshaft, the sink node listens for whole frame period because it is connected to the base station and powered from it. However we assumed the single receiver to be a general sensor node in this experiment, thus it only listens during its own receive time slot.

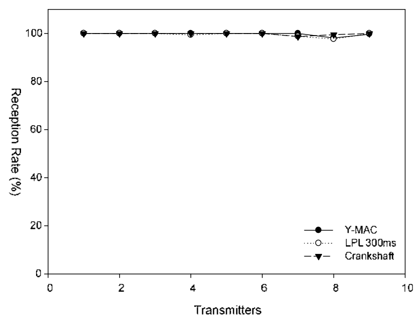
Figure 6 (a) shows the average duty cycles of the three MAC protocols when every node generates a message every 10 seconds. When node density is



(a) Average duty cycle of node

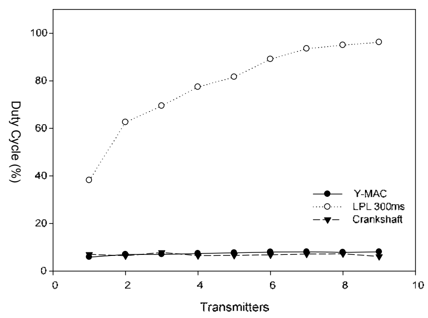


(b) Average delivery latency

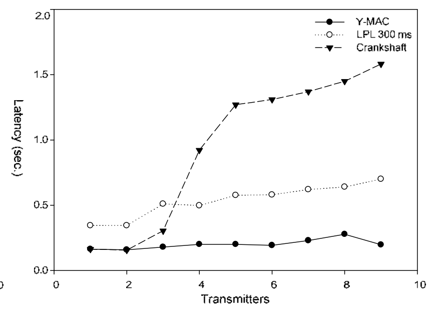


(c) Average data reception rate

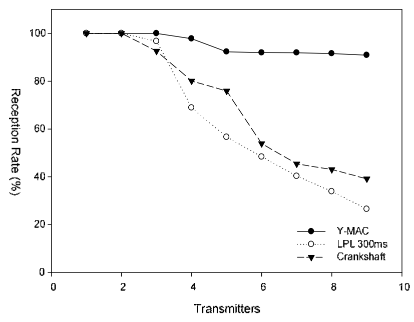
Figure 6. Performance in single-hop environments, 1 packet per 10 seconds.



(a) Average duty cycle of nodes



(b) Average delivery latency



(c) Average data reception rate

Figure 7. Performance in single-hop environments, 1 packet per second.

relatively small, LPL achieves low duty cycle. As the number of transmitters increases, however, duty cycle increases because senders waste energy overhearing unintended messages. In Y-MAC and Crankshaft, the average duty cycle remains low because the overhearing problem is reduced by allocating receive time slots to the nodes.

The average message delivery latency is shown in Figure 6 (b). The latency of LPL is significantly higher than the other protocols, since a sender has to send a preamble longer than the sleep interval to wake up a receiver. For Y-MAC and Crankshaft, delivery latency is slightly higher than the theoretical average delivery latency of half of the frame length. Delay from insertion into the message queue causes the gap between these two latency values. Figure 6 (c) shows that all three protocols achieved good reception rates. However, Y-MAC and Crankshaft achieved good reception rates and low duty cycles at the same time.

Figure 7 (a) shows the average duty cycles of the three MAC protocols when every node generates a message once every second. The average duty cycle of LPL increases with the number of transmitters because senders not only transmit messages but also overhear the messages from other senders. Y-MAC and Crankshaft still exhibit the same trend shown in figure 6 (a). Figure 7 (b) shows the average delivery latency. The delivery latency of Crankshaft increases dramatically once the number of senders exceeds 3 due to the limited per node reception bandwidth. As a receiver receives only one message during in the frame period, pending messages must wait in the message queue to be retransmitted in a subsequent frame period. In Y-MAC, however, the delivery latency remains steady since a contention loser can retry in the next time slot on the next channel. Meanwhile, Figure 7 (c) shows that Y-MAC achieves good reception rate even under high traffic conditions, while the other single channel MAC protocols suffer due to limited reception bandwidth.

5.3. Performance in multi-hop environments

To validate practicality in real environments, we constructed a multi-hop network consisting of 15 TmoteSky sensor nodes. These were deployed in a hall 20 meters long and 15 meters wide. We attenuated the transmission power of the nodes to construct a multi-hop environment. The experiment site is shown in Figure 8. Each node reports local information to the sink node, including the parent node in the routing path, the number of neighbors, and battery power remaining. We used a sensor network monitoring tool, RMon [33], to observe the current state of the deployed network.

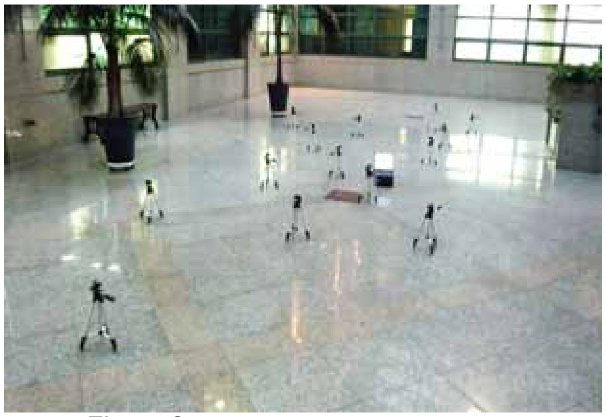


Figure 8. Experimental setup in the hall

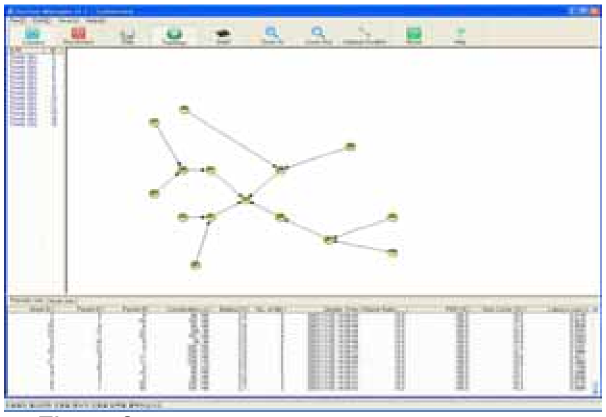
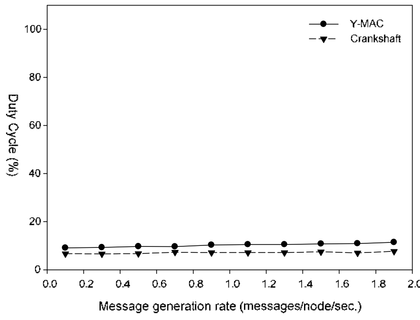
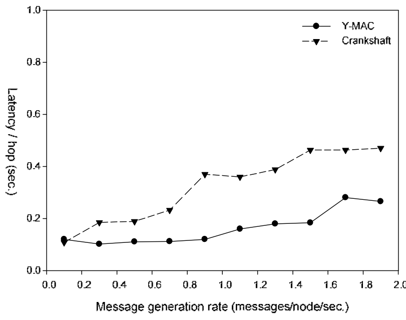


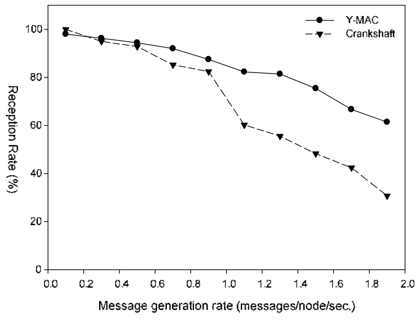
Figure 9. Sensor network monitoring tool : RMon



(a) Average duty cycle of nodes



(b) Average per hop delivery latency



(c) Average data reception rate

Figure 10. Performance in multi-hop environments

RMon displays the current topology of the network as well as the local information gathered from the nodes. As shown in Figure 9, RMon displays the topology of the network in a graphical user interface (GUI), with statistics in the bottom panel. For the experiments, we added additional performance indicators, such as duty cycle and message delivery latency. LPL was deliberately excluded from this experiment because it did not perform well under high traffic conditions.

Energy savings for the sink nodes are less important because this node is normally powered by the base station. In Y-MAC and Crankshaft, the sink node listens in every unicast time slot. The difference between the two protocols is that the sink node in Y-MAC continuously hops to the next channel so as not to interfere with data reception by other nodes. Channel hoping is determined taking into consideration the available number of channels.

We controlled the message generation rate to vary the traffic load in the network. Figure 10 (a) shows the average duty cycle of nodes with increasing message generation rate. Both Y-MAC and Crankshaft maintain a low duty cycle. The duty cycle of Y-MAC is slightly higher than Crankshaft because receivers send out a notification messages at the start of the next time slot in order to receive consecutive messages. Figure 10 (b) indicates that the average per hop delivery latency of Crankshaft rises faster as the traffic load increases. If a sender has lost the contention or did not receive an acknowledgement, it retries in the next frame period. The average per hop delivery latency of Y-MAC is steady because retransmission can be carried out in the next time slot on the next channel. Figure 10 (c) shows the average data reception rate at the sink node. Y-MAC outperforms Crankshaft in terms of data reception rates. Crankshaft does not provide good data reception rates under high traffic conditions because it cannot handle bursty messages. The average data reception rate of Y-MAC remains relatively high because a receiver is able to receive a series of messages by using multiple channels.

6. Conclusions

In this paper, we proposed a multi-channel MAC protocol for wireless sensor networks. Although existing energy efficient MAC protocols achieve low energy consumption, they sacrifice network throughput. When an important event occurs, sensor nodes around the event have to report their data to the base station as quickly as possible. To handle bursty messages effectively, Y-MAC exploits multiple channels.

Unlike most of the other multi-channel MAC protocols for WSNs, we implemented Y-MAC in the RETOS operating system running on TMoteSky motes. A serious challenge in implementing any TDMA-based MAC protocol is achieving accurate time synchronization. In our implementation, sensor nodes exchange the time remaining in the current frame period to synchronize their starting points for the next frame period. Our preliminary experiments show that the average time synchronization error among sensor nodes is acceptable for MAC designers to implement TDMA-based MAC protocols. We also proposed a light-weight channel hopping mechanism that enables multiple node pairs to communicate simultaneously on multiple channels. Sensor nodes hop to the next radio channel if they have additional pending messages for the receiver. This mechanism increases network throughput and reduces message delivery latency at the same time.

Throughout this paper, we have conducted extensive experiments to validate the practicality of the proposed Y-MAC protocol. Experimental results show that Y-MAC achieves low duty cycle under light traffic conditions, similar to other existing low power MAC protocols. We have proven that Y-MAC achieves effective transmission of bursty messages, under high traffic conditions, while maintaining low energy consumption.

We believe that our work demonstrates the practicality of adopting multi-channel MAC protocols in real life sensor node platforms. The use of multiple channels can definitely increase MAC protocol performance with low energy consumption.

7. Acknowledgements

This research was supported by the National Research Laboratory (NRL) program of the Korean Science and Engineering Foundation (No. M10500000059- 6J0000- 05910) and MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute for Information Technology Advancement)" (IITA-2008-C1090-0801-0015)

8. References

[1] W. Ye, J. Heidemann and D. Estrin, “An Energy-efficient MAC Protocol for Wireless Sensor Networks”, 21st Conference of the IEEE Computer and Communications Societies.

[2] J. Hill and D. Culler, “Mica: a wireless platform for deeply embedded networks.”, IEEE Micro, 2002.

[3] A. El-Hoiydi, “Aloha with Preamble Sampling for Sporadic Traffic in Ad Hoc Wireless Sensor Networks.”, IEEE International Conference on Communications (ICC), 2002.

[4] A. El-Hoiydi and J.-D. Decotignie, “WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks.”, First Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004), Lecture Notes in Computer Science, LNCS 3121, 2004.

[5] J. Polastre, J. Hill and D. Culler, “Versatile low power media access for wireless sensor networks.”, SenSys04, 2004.

[6] M. Buettner, G. Yee, E. Anderson and R. Han, “X-MAC: A Short Preamble MAC Protocol For Duty-CycledWireless Networks.”, SenSys06, 2006.

[7] S. Coleri-Ergen and P. Varaiya, “PEDAMACS: Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks.”, IEEE Trans. on Mobile Computing, 2006.

[8] L. van Hoesel and P. Havinga, “A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks.”. INSS04, 2004.

[9] G. Halkes and K. Langendoen, “Crankshaft: An Energy-Efficient MAC-Protocol For Dense Wireless Sensor Networks.”, EWSN07, 2007.

[10] Gahng-Seop Ahn, Emiliano Miluzzo, Andrew T. Campbell, Se Gi Hong and Francesca Cuomo, “Funneling-MAC: A Localized, Sink-Oriented MAC For Boosting Fidelity in Sensor Networks.”, SenSys06, 2006.

[11] I. Rhee, A. Warrier, M. Aia and J. Min, “Z-MAC: a hybrid MAC for wireless sensor networks”, SenSys05, 2005.

[12] W. Ye, F. Silva and J. Heidemann, “Ultra-Low Duty Cycle MAC with Scheduled Channel Polling.”, SenSys06, 2006.

[13] Junaid Ansari, Xi Zhang and Petri Mähönen, “Demo Abstract: Multi-Radio Medium Access Control Protocol for Wireless Sensor Networks.”, SenSys07, 2007.

[14] Chipcon, CC1000 Single Chip Very Low Power RF Transceiver. http://www.chipcon.com/files/CC1000Data Sheet2_2.pdf

[15] Chipcon, CC2420 2.4GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf

[16] Youngmin Kim, Hyojeong Shin and Hojung Cha, “Demo Abstract: A Multi-channel MAC Implementation for Wireless Sensor Networks.”, SenSys07, 2007.

[17] H. Cha, S. Choi, I. Jung, H. Kim, H. Shin, J. Yoo, C. Yoon, "RETOS: Resilient, Expandable, and Threaded Operating System for Wireless Sensor Networks.", The Sixth International Conference on Information Processing in Sensor Networks (IPSN 2007)

[18] Tmote Sky, http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf

[19] Shih-Lin Wu, Chih-Yu Lin, Yu-Chee Tseng, and Jang-Ping Sheu, "A New Multi-Channel MAC protocol with on demand channel assignment for mobile ad-hoc networks.", In Proc. International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '00)

[20] A. Tzamaloukas and J.J. Garcia-Luna-Aceves, "Channel-Hopping Multiple Access.", In Proc. IEEE ICC 2000, New Orleans, Louisiana, June 18-22, 2000.

[21] Jungmin So and Nitin Vaidya, "Multi-Channel MAC for AdHoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver.", ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), May 2004.

[22] Hoi-Sheung So, W. Walrand, J. and Jeonghoon Mo, "McMAC: A Parallel Rendezvous Multi-Channel MAC Protocol.", Wireless Communications and Networking Conference (WCNC), 2007.

[23] Gang Zhou, Chengdu Huang, Ting Yan, Tian He, John A. Stankovic and Tarek F. Abdelzaher, "MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks.", INFOCOM, 2006

[24] Chen Xun, Han Peng, He Qiu-sheng, Tu Shi-liang, Chen Zhang-long, "A Multi-Channel MAC Protocol for Wireless Sensor Networks.", In Proceedings of The Sixth IEEE International Conference on Computer and Information Technology (CIT), 2006.

[25] Ozlem Durmaz Incel, Stefan Dulman, and Pierre Jansen, "Multi-channel Support for Dense Wireless Sensor Networking.", EUROSSC, 2006.

[26] Hui Cao, Kenneth W. Parker and Anish Arora, "O-MAC: A Receiver Centric Power Management Protocol.", Network Protocols, 2006. ICNP '06.

[27] Kay R omer, "Time Synchronization in Ad Hoc Networks.", In Proceedings of MobiHoc 2001, Long Beach, CA, Oct 2001.

[28] Weilian Su and Ian F. Akyildiz, "Time-Diffusion Synchronization Protocol for Sensor Networks.", Technical report, Georgia Institute of Technology, Broadband and Wireless Networking Laboratory, 2002.

[29] Saurabh Ganeriwal, Ram Kumar, Sachin Adlakha, and Mani B. Srivastava, "Network-wide Time Synchronization in Sensor Networks.", Technical report, University of California, Dept. of Electrical Engineering, 2002.

[30] Hoi-Sheung Wilson So, Giang Nguyen, Jean Walrand, "Practical Synchronization Techniques for Multi-Channel MAC.", MobiCom'06, September 23–26, 2006, Los Angeles, California, USA.

[31] S. Moon, T. Kim, H. Cha, "Enabling Low Power Listening on IEEE 802.15.4-based Sensor Nodes", The 5th IEEE Wireless Communications & Networking Conference (WCNC 2007), Hong Kong, China, March 2007.

[32] Branislav Kusy, Prabal Dutta, Philip Levis, Miklos Maroti, Akos Ledeczi, and David Culler, "Elapsed Time on Arrival: A simple, versatile, and scalable primitive for canonical time synchronization services.", accepted for publication in Int. J. Ad Hoc Ubiq. Comput.

[33] I. Jung, H. Cha, "RMTool: Component-Based Network Management System for Wireless Sensor Networks," 2007 IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, January 2007.