A Practical Multi-Channel Media Access Control Protocol for Wireless Sensor Networks [*]

Hieu Khac Le, Dan Henriksson, and Tarek Abdelzaher
Department of Computer Science, University of Illinois at Urbana-Champaign
201 N Goodwin Ave., Urbana, IL 61801
{hieule2, danhenr, zaher}@cs.uiuc.edu

Abstract

Despite availability of multiple orthogonal communication channels on common sensor network platforms, such as MicaZ motes, and despite multiple simulation-supported designs of multi-channel MAC protocols, most existing sensor networks use only one channel for communication, which is a source of bandwidth inefficiency. In this work, we design, implement, and experimentally evaluate a practical MAC protocol which utilizes multiple channels efficiently for WSNs. A control theory approach is used to dynamically allocate channels for each mote in a distributed manner transparently to the application and routing layers. The protocol assumes that sensor nodes are equipped with one half-duplex radio interface which is most common in current hardware platforms. The protocol does not require time synchronization among nodes and takes the channel switching cost of current hardware into account. Evaluation results on a real testbed show that it achieves a non-trivial bandwidth improvement using 802.15.4 radios in topologies which are typical in WSNs. The MAC protocol was implemented in TinyOS-2.x and packaged as a software component to enable seamless use with existing applications.

1. Introduction

This paper presents a practical design, implementation, and evaluation of a multi-channel Media Access Control (MAC) protocol for Wireless Sensor Networks (WSNs). There has been a lot of MAC protocols introduced for WSNs that use only one channel for communication. However, with the new radio capabilities of WSN motes which can communicate on multiple frequencies, this is a great source of inefficiency. The very high density of current WSNs inevitably results in physical bandwidth limitations and heavy collisions on a single channel.

There is previous work on multi-channel MAC protocols for WSNs [16, 9, 19, 4, 23]. Some of these MAC protocols assume that the time to switch between two channels is negligible, whereas others require fine-grained time synchronization among nodes. Some assume that nodes have a multi-radio interface or can listen on different channels simultaneously. Most of these MAC protocols have only been evaluated in simulation, and the rest require devices with fully fledged multi-radio interfaces. To the best of our knowledge, all previous multi-channel protocols for sensor networks have at least one of the aforementioned limitations.

Our work is the first multi-channel MAC protocol which is implemented for MicaZ motes with only one half-duplex radio interface and with long channel switching times. In [13] a multi-channel MAC protocol was developed for collection WSNs, and was also implemented on MicaZ motes. There has been similar work for dissemination in [18] and [21]. While these efforts implemented multi-channel solutions for specific applications in sensor networks (such as data collection or dissemination), the MAC protocol described in this paper is the first general purpose MAC protocol which is designed and implemented on sensor motes with no specific assumptions on the application.

The main idea of the protocol is to assign a *home frequency* to each node such that network throughput is maximized. We call each different frequency available to the network, a *channel*. All nodes in the network start on the same channel. When this channel becomes overloaded, some nodes migrate to other channels to spread the communication load across non-interfering frequencies. Migration to another home frequency does not entail loss of connectivity with nodes that remain on the old home frequency. Instead, our protocol involves a mechamism whereby a node can send messages to another node that is on a different home frequency. Briefly, when a node needs to send messages to another on a different home frequency, it switches to the

[*]This work has been supported in part by the Wenner-Gren Foundations and the Hans Werthén Foundation, Sweden and in part by NSF grants CNS 05-54759, CNS 06-15318, CNS 06-26342, and CNS 06-26825

home channel of the destination node enough to send the message. Obviously, communication between nodes on the same home channel incurs less overhead. This motivates formulating the network throughout optimization problem in a manner similar to a clustering problem in a graph, whereby nodes that communicate frequently are clustered into the same channel, whereas those that do not communicate much (but are within each other's interference range) are separated into different channels.

Our protocol solves the problem in a distributed manner where nodes locally compute their "edge costs" and make migration decisions independently. Towards that end, nodes exchange state information about messages received and degrees of estimated communication success probability. If the estimated success probability is low, a node may switch to another channel. The switching is done based on a probability such that while alleviating congestion we avoid having all nodes jump to the new channel.

The algorithm design is reduced to two main problems. First, the graph clustering algorithm conceptually attempts to find a minimum K-way cut in the graph given by the network topology in a distributed manner in order to minimize costly inter-channel communication. Second, a configuration control problem is formulated to compute the probabilities of home channel switching. These probabilities are chosen such that the network self-configures into using just the right number of channels without excessive fluctuation among channels and without being too slow to respond to changes in load. The first problem is NP-complete, which is thus approached by an efficient distributed heuristic. For the self-configuration problem we propose a feedback control approach to compute the switch probabilities.

Our main contribution is to show that sensor nodes with a single half-duplex radio interface can actually benefit from channel diversity. The protocol is simple and light-weight enough to be implemented on MicaZ motes. Evaluation on an actual testbed shows that it works efficiently. The experimental results in both simulation and on a real test-bed show that the new MAC protocol not only achieves higher bandwidth, but also adaptively alleviates network congestion and avoids channels with high interference due to external sources (e.g., nearby 802.11 connections). In this work, the MAC is independent from the routing layer. Power management in the presence of multiple channels is not in the scope of this work.

The implemented MAC protocol component works on top of the existing single channel MAC protocol in TinyOS-2.x and exposes the same interface as Packet Protocols in [14]. Hence, applications developed for TinyOS-2.x can be adapted to the new MAC protocol seamlessly. The code for MicaZ motes on TinyOS-2.x can be downloaded from `http://cs.uiuc.edu/homes/hieule2/IPSN08/`.

The rest of this paper is organized as follows. Section 2 gives an overview of related work. Section 3 describes architecture considerations and underlying analytical foundations. Section 4 presents the protocol design and Section 5 describes the implementation details of the channel selection algorithm on the MicaZ motes. Section 6 presents an evaluation both on a real testbed and in simulation. The testbed experiments work as proof-of-concept and are used to validate the simulation results. The simulations evaluate the MAC protocol on a larger scale. Finally, the paper concludes with Section 7.

2. Related Work

The idea of multi-channel MAC protocols is not new in the wireless network research community. In the adhoc network domain, there have already been some practical systems utilizing multiple channels for communication. However, most of them have a very simple network topology, assuming only one base-station with the remaining nodes communicating with the base-station within only one hop. Others use heavy-weight protocols which are not suitable for sensor network devices [24].

There has also been work based on channel hopping [5, 20]. These schemes require synchronization and frequent switching among channels even when there is no need for communication. Unless supported by the physical layer, switching channels in sensor devices costs a great deal of time and energy which makes these schemes unattractive for WSNs. In our experiments with MicaZ motes [1], the time to switch between two channels and wait until the frequency synthesizer stabilizes is roughly equal to the time to transmit one packet. Therefore, nodes which frequently switch channels run out of battery faster than other nodes. Although the radio technology is advancing, it is nontrivial to reduce the cost of channel switching in such strictly constrained devices as sensor nodes. There exist efforts applied for devices with multi-radio interfaces [4, 12], or with specially designed interfaces which can listen simultaneously to different channels [9, 16]. However, as we have observed, hardware equipped with one half-duplex radio interface are much more popular.

There are also efforts in industry which utilize multichannel radios. TSMP [3] maintains synchronization among nodes. Nodes employ frequency hopping according to a shared pseudo-random schedule. TSMP requires both sychronization and frequent channel switching. There is also ongoing effort on the SP100.11a standard [11]. This standard uses a simplified 3-channel hopping scheme. Sexton and others have documented narrow band fading problems and promote multi-channel communication as added diversity[1].

[1]We thank one of our reviewers for pointing this out

Finally, there are results in both ad-hoc multi-hop networks and WSNs focusing on systems with only one half-duplex radio interface [19, 6, 17]. Unfortunately, all of them are done in simulation and most of them assume that the time to switch channels is negligible. There are also protocols that have a working implementation showing significant improvements in total network throughput. However, these protocols are limited to a specific traffic pattern such as data collection and aggregation [13], or data dissemimation [18, 21]. Networks with arbitrary traffic patterns are not yet covered.

3. Theoretial Analysis

Our algorithm is based on three observations. First, a new channel should be allocated only when needed. If there is no serious interference or collisions in the neighborhood, nodes should not switch to other channels. This reduces the cost of inter-channel communication. Second, by design, some nodes should be more likley to initiate channel switches than others. The more global is the view (of communication) that a node has, the more informed it is, and the better equipped it is to make the right move. Third, nodes with a more limited view should act locally to minimize cross-channel communication. The best local action is to follow a node with a better view.

In this section, we present solutions and analysis to the problems of minimizing inter-channel communication and choosing the channel switching probabilities consistently with the above observations. The former is formulated as a distributed clustering problem and the latter is approached using feedback control theory. Below, we first describe our fundamental mechanism for communication between neighboring nodes. We then provide solutions to clustering and feedback control to maximize communication throughput.

Since cross-channel communication introduces extra cost both due to channel switching times and due to retransmissions caused by the deafness problem (when a node sends messages to another but uses the wrong channel), it is desirable to minimize cross-channel communication and maximize same-channel traffic. This is directly related to the objective of the K-way cut problem in graph theory and is used as an inspiration for our channel allocation protocol.

3.1. The K-way Cut Problem

Our approach is to partition the nodes in the network into different sets, each assigned a separate home channel, such that two types of constraints are met. First, communication within each set is limited to local capacity. Second, communication across sets is minimized. In a graph where each node is a communication device and where link costs represent the amount of communication, this corresponds to solving a K-way cut problem of minimum K that respects the capacity constraints on each cluster. There have been several centralized approaches to solve versions of the K-way cut problem. An optimal algorithm [7] was proposed for a k-way cur with fixed K with $O(n^4)$ complexity for $K = 3$ and $O(n^9)$ complexity for $K = 4$. A more efficient algorithm [10] was subsequently proposed, which has $O(n^3)$ complexity for $K = 3$ and $O(n^4)$ complexity for $K = 4$, which is the fastest optimal algorithm for a fixed K. Some centralized approximation algorithms for undirected graphs were also developed [22]. Distributed heuristic algorithms were proposed for undirected graphs.

These algorithms require a-priori knowledge of K and are quite heavy-weight. The graphs in our problem are directed, weighted graphs without a fixed K, which makes the problem harder and more complex. Due to the constraints on the sensor devices, any algorithm with a high polynomial complexity will lack scalability.

In the following, we will describe our adaptive algorithm which takes only $O(n^2)$ computation time and $O(n)$ memory and provides reasonably good performance within the scope of our MAC protocol.

3.1.1 The Algorithm

With the above intuitions in mind, channels are organized as a ladder, starting with the lowest channel, F_0, up to the highest channel F_N, with N being the number of channels available in the network. Whenever a node first joins the network it starts at channel F_0 (hence, initially all nodes are the same "cluster". Once a node figures out that there are lots of messages lost due to collisions and interference (i.e., the local cluster capacity constraint is violated), the node considers switching channels. The switching decision is based on how serious the collisions and interference are and on the role of the node in contributing traffic to the network.

To measure the effect of a crowded spectrum, each node periodically broadcasts a tuple $< s, f >$, where s is the total number of times the node successfully acquires the channel, and f is the number of times the node is unsuccessessful (at acquiring the channel). Periodically, every node i receives a set of tuples from its neighbors j. Based on that, node i estimates the probability that any of its neighbor nodes can successfully attempt to access the channel:

$$\alpha_i = \frac{\sum_j s_j}{\sum_j (s_j + f_j)}. \qquad (1)$$

If α_i is too low, the channel must be too crowded around node i [2]. Hence, if α_i is less than a configurable value α_{ref},

[2]Notice that α_i also reflects interference at i. If interference exists

node i will consider switching from its current home channel F_c to the next higher channel F_{c+1} (unless $c = N$), with a probability that depends on channel conditions. The use of switch probabilities will reduce fluctuations between channels that may otherwise result if many nodes switched channels at the same time. In the next subsection, we describe how a control scheme is used to make the channel switching process stable. For now, let's assume that given the history and current status of home channel c of node i, the node has a probability to switch from channel c to the next channel $c+1$, denoted by $\beta^i_{c,c+1}$. In our algorithm this probability increases with the difference in quality between the source and destination channels (i.e., as the source becomes substantially worse than the destination).

In summary, as appropriate of a K-way cut heuristic (with a variable K), we operate by dividing existing clusters that exceed capacity repeatedly until capacity constraints are met. Key to the design of this heuristic is to determine the mechanism for splitting clusters and the boundaries across which splitting must occur. This reduces to two questions: who should initiate the split and who should follow into the new cluster? The solution should be distributed and obey the goal of minimizing communication across clusters.

To answer these questions, observe that in a wireless sensor network, nodes are usually not equal in contributing to network load. Hence, they should act differently in terms of channel switching probability. Consider two extreme examples. The first example is a node that only *sends* messages (to its neighbors) but does not receive. This pattern is consistent with that of data sources. The second example is a node that only *receives* messages (from its neighbors) but does not send. This pattern is consistent with data sinks in wireless sensor networks. Channel congestion typically occurs at sinks. Hence, sinks have a more global view of traffic than sources. As such, sinks are better positioned to make decisions on channel allocation.

In our algorithm, nodes that behave predominantly as sinks have preference to switch channels first (i.e., initiate the cluster split). This has the desirable side-effect of creating well isolated clusters. A node that acts predominantly as a sink does not send much traffic by definition, and hence has a low-cost outgoing link, making it appropriate to cut (by the K-way cut algorithm). Nodes that communicate heavily with those who switched, follow them into the new cluster. This works well for aggregation topologies, which is the predominant case in data collection networks. Finally, to communicate across clusters, a sender on one home channel simply switches to the home channel of the receiver temporarily to send messages to the latter.

More specifically, the probability that a node initiates a cluster-split by switches channels from c to $c+1$ is given by

$$P^i_{c,c+1} = MAX\left(0, sink\_factor_i \times \beta^i_{c,c+1}\right) \qquad (2)$$

where $\beta^i_{c,c+1}$ increases with the difference in quality between channel c and $c+1$ (in favor of $c+1$), and $sink\_factor$ is an indicator showing how closely a node resembles a sink. It is computed from

$$sink\_factor_i = \frac{In_i - Out_i}{In_i + Out_i}, \qquad (3)$$

with In_i and Out_i being the total number of messages received and sent by i at its home channel, respectively. If the node is a true sink, $sink\_factor = 1$. If it is a pure source, $sink\_factor = -1$. An intermediate node in the network might sink some traffic and forward some. Its $sink\_factor$ will thus have some intermediate value. For example, a pure router that simply forwards all traffic will have $sink\_factor = 0$. An aggregator that summarizes the traffic and forwards the summary will have a $sink\_factor$ closer to 1.

By encouraging splitting when the current channel is much worse than the target channel, our cluster splitting mechanism guarantees that a network will allocate more channels when it gets congested hence preserving cluster capacity constraints. By letting sinks initiate the split with a higher probability, we ensure that the split starts across a low-cost link. Finally, by letting neighbors who send much traffic to those who switched follow them to the new channel, we present a natural way to grow a new cluster in a way that minimizes the communication across different clusters. We call this phase *channel expansion*. When a channel is no longer congested, nodes on this channel invite those from the next (higher) channel in the ladder to switch to the underutilized frequency. As before, sink-like nodes initiate such transitions with a higher probability. Other nodes follow. We call this phenomenon *channel shrinking*.

3.2. The Self-Configuration Problem

The self-configuration problem considers the dynamics of channel expansion and channel shrinking. In particular, it is important that such transitions are stable. Otherwise, nodes may incur a significant overhead switching between channels. Both the channel expansion and channel shrinking mechanisms are designed using feedback control theory, where the control signal is the probability for a node to switch channel. The use of probabilities takes the distributed nature of the control system into account and prevents all nodes from switching at the same time, which would not improve the situation.

The control laws for channel expansion and channel shrinking have been designed to be intuitive and easy to

around i, node i will be able to sense the signal and cease to access the channel and that affects the value of α_i

implement. However, we will also present an analysis that shows how to choose the controller gain parameters to achieve a good trade-off between fluctuations and performance. The analysis is based on restricting the fraction of nodes that are allowed to switch channel during a certain time interval related to the time delay in the system. The restriction ensures that the loop is stable in a control-theoretic sense.

3.2.1 Channel Expansion

We propose the following feedback control scheme for the channel expansion. The probability for a node i to switch from its current channel c to the next channel $c+1$ if $\alpha_c^i < \alpha_{ref}^{up}$ is computed as

$$\beta_{c,c+1}^{i}(k) = \beta_{c,c+1}^{i}(k-1) + K_r^{up}\left(\alpha_{ref}^{up} - \alpha_c^i(k)\right), \quad (4)$$

where k denotes the sampling interval (i.e., the time between consecutive updates of the switch probabilities). The controller is in integral form (i.e., its output is proportional to the integral of inputs), where the switch probability is increased for each sample as long as $\alpha_c^i < \alpha_{ref}^{up}$. Similarly, as $\alpha_c^i \geq \alpha_{ref}^{up}$ we decrease the switch probability with a faster rate as

$$\beta_{c,c+1}^{i}(k) = \beta_{c,c+1}^{i}(k-1) - \hat{K}_r^{up}\left(\alpha_c^i(k) - \alpha_{ref}^{up}\right), \quad (5)$$

where $\hat{K}_r^{up} > K_r^{up}$

3.2.2 Channel Shrinking

Nodes switch to higher channels when their current channel gets congested. We also need a mechanism by which nodes may switch back to lower channels once the traffic is less busy. This will reduce the cost of cross-channel communication. Analogous to the case of advancing channels, this scheme has nodes inviting nodes from higher channels once the success rate, α, is above a given threshold, α_{ref}^{down}. The invitation probability for a node i at channel $c+1$ to switch down to the current channel c if $\alpha_c^i > \alpha_{ref}^{down}$ is given as

$$\beta_{c,c-1}^{i}(k) = \beta_{c,c-1}^{i}(k-1) + K_r^{down}\left(\alpha_c^i(k) - \alpha_{ref}^{down}\right) \quad (6)$$

As before we decrease probability with faster rate when we have $\alpha_c^i \leq \alpha_{ref}^{down}$ as

$$\beta_{c,c-1}^{i}(k) = \beta_{c,c-1}^{i}(k-1) - \hat{K}_r^{down}\left(\alpha_c^i(k) - \alpha_{ref}^{down}\right), \quad (7)$$

The key element in both channel expansion and shrinking is to accurately set the controller gain K^{up} and K^{down}, which determine how aggressively or conservatively switching occurs. (Higher K implies a higher switching probability or more aggressive switching.)

3.2.3 Choosing the Controller Gains

The integral controller increases the switch probability by a small fraction at a time until enough nodes have switched to improve the quality of the current channel. However, depending on the topology of the network, there may be a substantial delay before the effect of a channel switch of one node has propagated to others on the old channel. It is well-known from basic feedback control theory that delay decreases the stability of a feedback control system, since it takes longer for the effects of control actions to become measurable. As a result, there is an interesting trade-off to consider when choosing the switching probability. If it is too big, switching is aggressive and nodes may oscillate among channels, moving back and forth excessively and causing overhead. On the other hand, if it is too small, it will take longer for the network to re-configure to new channels upon load changes. This trade-off is captured by the controller gain parameters K_r^{up} and K_r^{down}.

In the following, we will provide an insight into how to choose the gain parameters such that we avoid excessive channel fluctuations in the presence of delay. The analysis will be based on computing the worst-case switching frequency at network delay, d (measured in controller samples). The main reason for the delay comes from propagating the domino effect of switching from a sink back through intermediate nodes to the source. During this time, nodes close to the source still experience the same measured poor performance α as before the switch. The main effect of the delay is thus increased possibility that more nodes than necessary will switch to the new channel not knowing that someone else has already switched and that soon performance will consequently improve.

We will compute K_r^{up} and K_r^{down} such that the fraction of nodes that may switch during the propagation delay, d, is below a given threshold, γ. For channel expansion, the worst-case situation occurs if $\alpha_c^i = 0$, in which case the increase in the switch probability of Equation (4) from sample to sample is equal to $K_r^{up} \cdot \alpha_{ref}^{up}$. During the first sample, an average fraction $K_r^{up} \cdot \alpha_{ref}^{up}$ of the nodes will leave and $(1 - K_r^{up} \cdot \alpha_{ref}^{up})$ will remain at the channel. In the second sample, the switch probability will increase to $2 \cdot K_r^{up} \cdot \alpha_{ref}^{up}$ and the fraction of the original nodes that leave in this sample is equal to $(1 - K_r^{up} \cdot \alpha_{ref}^{up}) \cdot 2 \cdot K_r^{up} \cdot \alpha_{ref}^{up}$. The fraction of nodes, $\Gamma(d, K_r^{up}, \alpha_{ref}^{up})$, that switch channels during a time interval of d samples can, thus, be computed as

$$\Gamma(d, K_r^{up}, \alpha_{ref}^{up}) = \sum_{m=1}^{d}\left(m \cdot K_r^{up} \cdot \alpha_{ref}^{up} \cdot \prod_{j=1}^{m-1}\left(1 - j \cdot K_r^{up} \cdot \alpha_{ref}^{up}\right)\right) \quad (8)$$

For given values of α_{ref}^{up}, d, and γ (which all can be assumed to be available off-line), we may compute K_r^{up} from the relation $\Gamma(d, K_r^{up}, \alpha_{ref}^{up}) = \gamma$. As an example, Figure 1

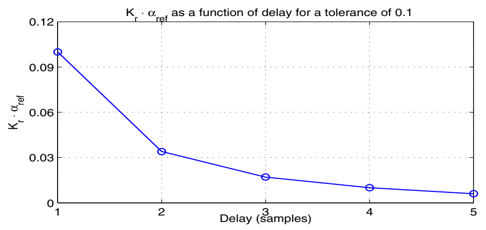

Figure 1. $K_r^{up} \cdot \alpha_{ref}^{up}$ as a function of the network delay for a tolerance of $\gamma = 0.1$.

shows $K_r^{up} \cdot \alpha_{ref}^{up}$ as a function of delay for the case $\gamma = 0.1$.

The same analysis applies for the case of channel shrinking, with the exception that the worst-case switch probability is given by $K_r^{down}(1 - \alpha_{ref}^{down})$.

With this analysis, we have provided a more intuitive design parameter than choosing the controller gains K_r^{up} and K_r^{down}. Specifying γ can be interpreted as choosing the worst-case fraction of nodes moving to a new channel. To prevent sustained oscillations, this fraction has to be less than 1. Smaller fractions have a larger stability margin (in a control-theoretic sense) but fractions that are too small cause a sluggish system response to load changes.

3.2.4 Channel Overflow

Once a home channel gets overcrowded, nodes switch to the upper channel. When the bandwidth of the available channels is sufficient, nodes in the network will be distributed from channel F_0 and up to the number of channels needed to accommodate the traffic. However, in the worst-case, there is still a chance that N channels are not sufficient to avoid network overload. Channel overflow also happens when a closely connected set of nodes (e.g. a sink and its followers) does not fit into one channel.

We propose a scheme to choose the threshold α_{ref}^{up} for channels, which solves the channel overflow problem. The idea of the scheme is that the higher a channel is, the lower its threshold should be; and the threshold of the highest channel F_N should be zero. This make nodes become more conservative in switching every time they go up one channel. Nodes stop considering switching channels once they get to the highest boundary channel. Based on that, the threshold α_{ref}^{up} is chosen as follows

$$\alpha_{ref,c}^{up} = \begin{cases} \alpha_{ref,0}^{up} - c \cdot \epsilon & \text{if } c < N \\ 0 & \text{if } c = N \end{cases} \quad (9)$$

with $\alpha_{ref,0}^{up}$ is the threshold at channel F_0 (where everynode starts to function at) and $\alpha_{ref,c}^{up}$ is the threshold at channel F_c, and ϵ is a chosen constant.

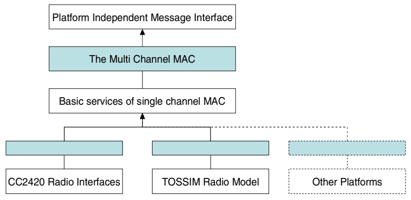

Figure 2. Component structure of the multi-channel MAC

4. Protocol Design

In this section we will describe the design of the multi-channel MAC protocol and how it extends the existing components of TinyOS-2.x.

4.1. Component Structure

The protocol is packaged in a component which exposes the same interface found in the TinyOS-2.x Packet Protocol. This will allow new applications developed for TinyOS-2.x to use the new MAC without any porting effort. Furthermore, since the multi-channel MAC protocol works on top of a single-channel MAC, we also decouple the services provided by the basic MAC from the multi-channel MAC so that it is independent of the platform-specific implementation. Figure 2 shows how the interfaces among layers are split to facilitate seamless integration with both virtual platforms provided by TOSSIM and an actual MAC provided for the CC2420 ChipCon radio in the MAC of the MicaZ motes. The white boxes are interfaces and the solid boxes are implemented. Other platforms can be integrated similarly by adding a thin layer on top of their MACs and provide the interface used by the multi-channel MAC.

4.2. Algorithm Design

There are important design decisions which need to be taken to ensure that the algorithm will be simple and efficient enough for WSNs applications. Following we describe these decisions.

Time-triggered Activity

The MAC protocol is designed to work in a time-triggered manner. In other words, rate control is achieved explicitly using an interval timer as opposed to implicitly by receipt of send done notifications. Most message types are queued and served periodically (except a few types of messages as will be explained later). Special messages used in the protocol have places reserved in the network message queue so that data messages do not occupy the whole queue and prevent protocol messages from being sent.

Channel Status Updates

Nodes periodically broadcast their perceived home channel conditions. The information broadcasted out by a node i is a pair $< s_i, f_i >$ where s_i is the number of times the MAC layer succeeds in accessing the channel and f_i is the number of failed attempts. This pair together with the current home channel of node i will be put on the same message called a Channel Update Message, and is enqueued to the same queue as with normal messages.

Nodes collect channel update messages and use that information to estimate the channel acquisition probability as described earlier. From channel update messages, nodes also are able to know the up-to-date home channel of their neighbors.

Neighbors' Home Channel Maintenance

When a node wants to send a message to another node, it needs to switch to the home channel of the receiver before transmitting. Hence, a node needs to know the home channel of its neighbors who it communicates with. When nodes first join the network, they assume that the home channels of other nodes are the same as their own, which is F_0. When the home channel information is out of date, communication fails and the node initiates a search for neighbors on all channels as will be described later.

In the following, we will describe the different message types used by the MAC protocol, how these messages are queued, and the management of neighbor tables. We then give the functional description of the algorithm.

4.2.1 Message Types

We begin by describing the message types used by our protocol for future reference. When a node first joins the network, it broadcasts a HELLO message at the home channel to inform its neighborhood that it has joined the channel. When a node needs to send a message to a neighbor but does not know its home channel, it sends out WHERE IS messages. CHANNEL UPDATE messages are sent out periodically by nodes and contain the pair $< t_i, s_i >$ of every node i. These messages are sent at the home channel of the sender. BYE message are sent out by nodes that decide to leave their current channel (because of channel expansion or channel shrinking). When the home channel is under-loaded (as described in Section 3) the node sends out INVITATION messages to the above channel to invite nodes to join its home channel. The last type of messages is DATA which constitutes any messages passed to the MAC by the upper layer via the component interface. The upper layer will be notified whenever a DATA message is sent successfully, or whenever the delivery failed for several transmission attempts.

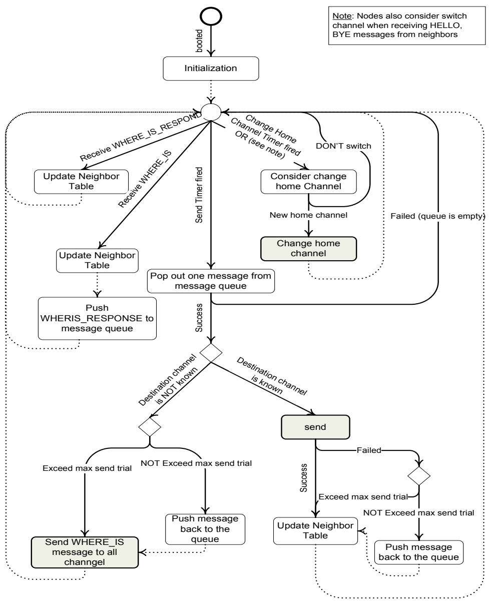
Note: Nodes also consider switch channel when receiving HELLO, BYE messages from neighbors
booted
Initialization
Receive WHERE_IS_RESPOND
Update Neighbor Table
Receive WHERE_IS
Update Neighbor Table
Push WHERIS_RESPONSE to message queue
Send Timer fired
Pop out one message from message queue
Change Home Channel Timer fired OR (see note)
DON'T switch
Consider change home Channel
New home channel
Change home channel
Failed (queue is empty)
Success
Destination channel is NOT known
Destination channel is known
send
Exceed max send trial
NOT Exceed max send trial
Send WHERE_IS message to all channgel
Push message back to the queue
Failed
Success
Exceed max send trial
NOT Exceed max send trial
Update Neighbor Table
Push message back to the queue

Figure 3. Overall state machine of the algorithm

Message Type	Included Information
WHERE IS	Home channel of the sender, Id of the requested node
WHERE IS RESPONSE	Home channel of the sender
HELLO	Old home channel
BYE	New home channel
CHANNEL UPDATE	$< t_i, s_i >$
INVITATION	$\beta_{c+1,c}$
DATA	Data

Table 1. Summary of the seven types of special protocol messages.

Table 1 gives a summary of the special network messages that are used by the MAC protocol.

The protocol traffic overhead happens in three cases: (i) updating channel status to nodes' neighbors, (ii) notifying neighbor about channel switching, and (iii) finding neighbors' home channels. In these overhead messages, the first type happens the most. In our implementation, channel update messages are sent periodically each one second. The length of this message is only 5 $bytes$. So, the overhead traffic is only around 5 $bytes/second/node$.

4.2.2 Message Queuing

Apart from WHERE IS, HELLO, and BYE messages, all types of messages – including WHERE IS RESPONSE, CHANNEL UPDATE, INVITATION, and DATA - are queued before being sent out. The MAC protocol periodically pops messages out of the queue and sends them to the corresponding destination. If the MAC fails to send a message, it will put the message back at the end of the queue. If the number of retry attempts exceeds a threshold, it will discard the message and notify the upper layer if the message is a data message.

Since the number of messages generated by the MAC (*protocol messages*) is small, and they play an important role in the behavior of the protocol, it is desirable for the queue to favor these messages over DATA messages from the upper layer. The queue implements this by not allowing DATA messages to fill up the whole queue. The remaining spaces are reserved for protocol messages. By doing this, the protocol messages will rarely be discarded because of queue overflow, even in heavy traffic conditions[3].

4.2.3 Neighbor Table Management

Neighbor tables are required to maintain information about neighboring nodes that a node communicates with. The neighbor table is designed as a simple hash table in which keys are the neighbor IDs. Since the number of entries is finite, new entries will replace the entry which is least recently used when the table is full.

4.2.4 Functional Description

The overall algorithm is captured by the state machine shown in Figure 3[4]. After initialization, each node goes to an idle state from which it executes different actions depending on messages it receives and the expiration of timers.

After receiving a WHERE IS RESPONSE or WHERE IS message, the neighbor list is updated. In the latter case, a WHERE IS RESPONSE message is also popped to the top of the message queue before the node returns to the idle state.

A decision to switch the home channel is made each time the change home channel timer fires. In case the node decides to switch channels it executes the steps shown in the *Change Home Channel* subsystem of Figure 4.

The actions performed by the node are, first, to send out a BYE message at its current home channel, then switch to the new home channel, and finally send out a HELLO message on the new channel. After switching, the node returns to the idle state.

[3]The fact the messages are queued before transmitted should work with most of the application. There are chances that this may affect special applications as Deluge [8]. However, further study need to be done to conclude this

[4]Circles represent states, rounded boxes are processes, solid lines are conditional transitions, dotted lines are unconditional transitions, and diamonds are condition checks. Filled boxes represent composite processes

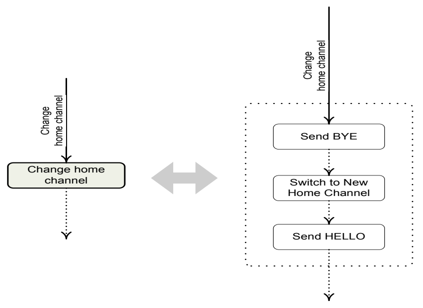

Figure 4. State machine subsystem for changing the home channel.

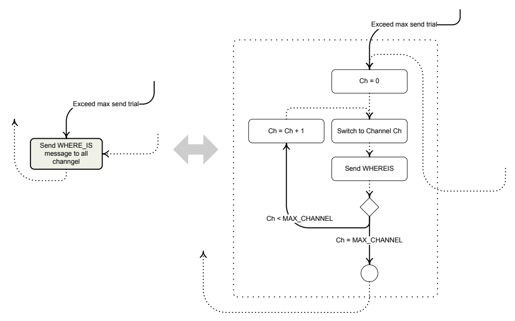

Figure 5. State machine subsystem for sending requests asking for the home channel of a neighbor.

The second timer used in the implementation is for sending messages. As this timer fires, the first message is popped from the message queue. If the queue is non-empty, the next action is to determine if the home channel of the destination is known.

If the destination channel is unknown, the message is pushed back to the message queue if the number of transmission attempts does not exceed the maximum. Thereafter, WHERE IS messages are sent out on gradually increasing channels as described by the subsystem in Figure 5. The messages are sent starting at channel F_0 followed by all higher channels up to F_N until an acknowledgment is received. After the WHERE IS have been sent, the node returns to the idle state.

If the destination channel is known, the message is sent. The sending of a message is described in more detail in the subsystem of Figure 6. If the destination is on a different channel, the sender needs to switch channels before sending the message. After determining if an acknowledgment was received, the sender switches back to its home channel.

If the transmission was successful, the neighbor table is updated and the node returns to its idle state. If the transmission failed, the message is pushed back to the message

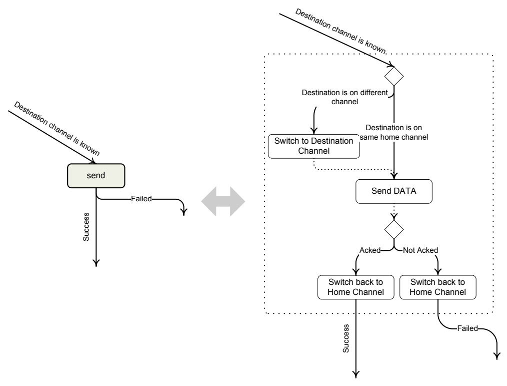
Destination channel is known
send
Failed
Success
Destination channel is known.
Destination is on different channel
Switch to Destination Channel
Destination is on same home channel
Send DATA
Acked
Not Acked
Switch back to Home Channel
Switch back to Home Channel
Success
Failed

Figure 6. State machine subsystem for sending a message.

queue if the number of transmissions is below the threshold.

5. Implementation

In this section we give a brief overview of the implementation of the multi-channel MAC protocol on the MicaZ motes.

5.1. Code Structure and Footprint

Software components are created to conform to the design described in Section 4.1. The multi-channel MAC is implemented in the nesC programming language for MicaZ motes with TinyOS-2.x. The code is structured so that the platform-dependent parts are separated from the core mechanisms of the protocol. Therefore, in the core mechanism implementation, there is no distinction between TOSSIM and MicaZ. One code base is used for both platforms.

The compiled code for the multi-channel MAC is 9544 bytes in ROM and 761 bytes in RAM. In future work, we will optimize the footprint. In the rest of this section, we will briefly present technical issues related to enabling multi-channel communication on MicaZ motes as well as in TOSSIM [15].

5.2. Adopting Multi-Channel Communication Capabilities

We need the capability to communicate on different channels *dynamically at runtime*. TOSSIM does not support this directly. Hence, we had to modify TOSSIM to adopt this feature.

Making TOSSIM support dynamic channel switching requires introducing new types of events in the event queue as well as changing the implementation of the radio model. The newly introduced event for switching channels also takes the experimental channel switching time from real motes into account, which make the simulation model accurately reflect the physical constraint.

6. Evaluation

In this section, we present an evaluation of the multi-channel MAC both in testbed experiments with MicaZ motes and in simulation using TOSSIM. While the testbed results show the performance in a small-scale network setting, the evaluation in TOSSIM (with the same code base as is run on the MicaZ motes) enables testing at a larger scale.

The evaluation settings focus on collection and aggregation traffic patterns which are most popular in WSN. Admittedly, our protocol favors this case. This is not a coincidental choice. We believe that random point-to-point traffic patterns are less popular in WSN. Hence, they are neigher targeted not evaluated in this paper and will generally result in poor performance of our protocol.

6.1. Experimental Testbed Evaluation

The following experimental evaluation will be run against simulations in TOSSIM for comparison and to validate the simulations with results from the real platform. The radio model is signal-strength-based. It follows previous literature [25], already supported in TOSSIM for TinyOS-2.x [2]. The network configuration files can be found at the link given at the end of the paper.

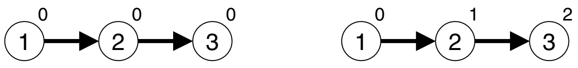

Figure 7. Setups used in the cross-channel communication evaluation.

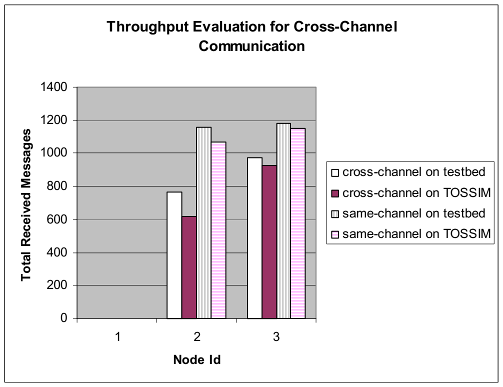

Figure 8. Throughput comparison in the cross-channel communication evaluation

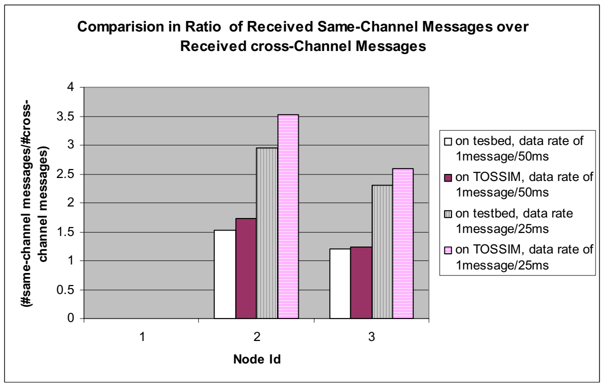

Figure 9. More comparision between TOSSIM and testbed results

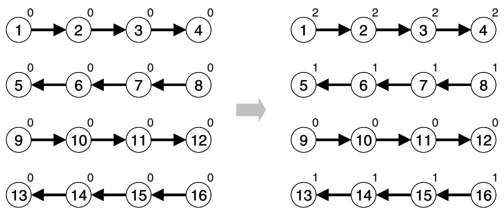

Figure 10. 16-node setup used for testbed experiments.

6.1.1 Cross-channel Communication

We first evaluate a cross-channel communication scheme and compare the results from both testbed and TOSSIM. Figure 7 shows the experimental setups. In both setups, node 1 sends messages to node 2 and node 2 sends messages to node 3. In the first setup, all three nodes work on the same channel. In the second setup, the three nodes are assigned to three different channels. Hence, node 1 has to switch to the channel of node 2 to send and node 2 has to switch to the channel of node 3 to send. The deafness effect will happen in the traffic from node 1 to node 2.

Figure 8 shows the number of messages received at each node for the same-channel and cross-channel setups in both experiments and simulation. As can be seen, there is a good match between values from the testbed and values from the simulation. The most significant difference between simulations and the testbed is in the number of messages received in the cross-channel case. The lower throughput in the simulation is due to an over-estimation of the channel switching time. This makes our simulation model more pessimistic, while still valid for a comparison in the more realistic cases studied in Section 6.2.

Figure 9 shows another view in which we compare the ratio of throughput in the same-channel case and the cross-channel case for each node in the testbed and TOSSIM environments. The results for both message-generating rates (1 message/25 milliseconds and 1 message/50 milliseconds)

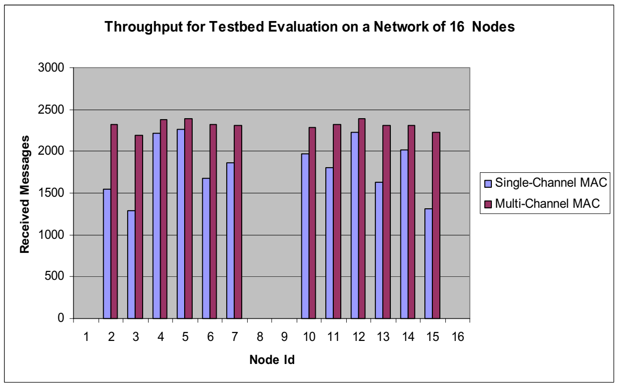

Figure 11. Testbed comparison between networks with a single channel MAC and with the multi-channel MAC for the 16-node setup.

again show a good match between the simulation and the real testbed.

6.1.2 Effect of Utilizing Multiple Channels

In this section, we evaluate how the multi-channel MAC improves throughput in a crowded network. A testbed with 16 nodes was used with the topology shown in Figure 10. The arrows show the traffic flows. The left figure shows the initial channel settings and the right figure shows the channel allocation after the network stabilizes. The experiment lasts for 10 minutes. Nodes reach the final channel allocation configuration on the right of Figure 10 in less than 3 minutes. The results in Figure 11 show that the case of using multiple channels on average outperforms the single channel case by about 30% in terms of throughput.

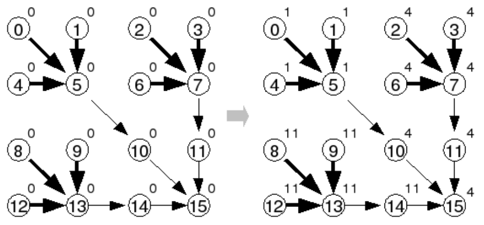

Figure 12. Network with tree lightly connected sub-networks

6.1.3 Network of Independent Sub-networks

We experiment with network traffic patterns shown in Figure 14. The network includes two separated sub-networks which form two different collection trees. The experiment was conducted in 10 minutes. After 3 minutes, the channel allocation was as shownon the right side of the figure. The two indepedent parts were located on two different channels. The throughput comparison is shown in Figure 15.

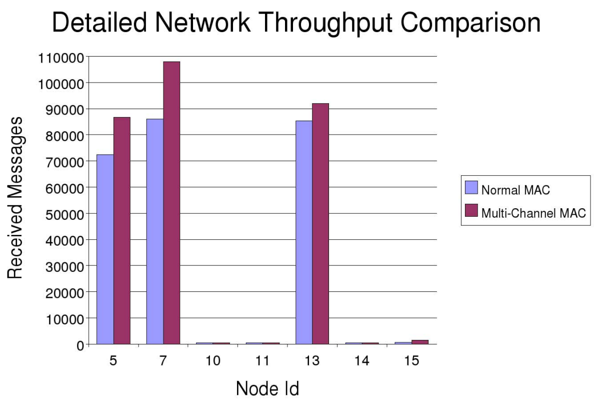

Figure 13. Testbed comparison between networks with a single channel MAC and with the multi-channel MAC for the 16-node setup.

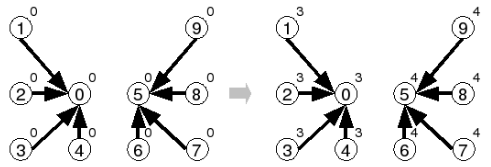

Figure 14. Networks with two separated sub-networks. The two sub-networks later are located on two different channels

We observe that the multi-channel MAC outperforms the single channel MAC by 31%. This experiment also shows the ability to avoid interference of the multi-channel MAC. Each sub-network can consider the other as a source of interference thus they end up at different channels.

6.1.4 Network of Lightly Connected Sub-networks

In this experiment, the network traffic is chosen as shown in Figure 12. This network is organized as an aggregation tree which includes three other sub-trees. Different data rates are also introduced in this evaluation (the thin arrows correspond to a data rate of 1 message / 1000 milliseconds, and the thick arrows have data rate of 1 message / 10 milliseconds). As shown in the throughput comparison in Figure 13, our MAC protocol out-performs the case of a single-channel MAC at all aggregation nodes. In particular, for node 16 the throughput improvement is roughly 25%.

6.2. Simulation and Scaling

The previous experimental evaluation in section 6.1.1 showed a good match between simulation and the real testbed. In this section, we scale the evaluation by simulation to a network of 36 nodes composed of two aggregation trees. The roots of the trees are placed next to each other so they interfere. The topology of the network is shown in Figure 16.

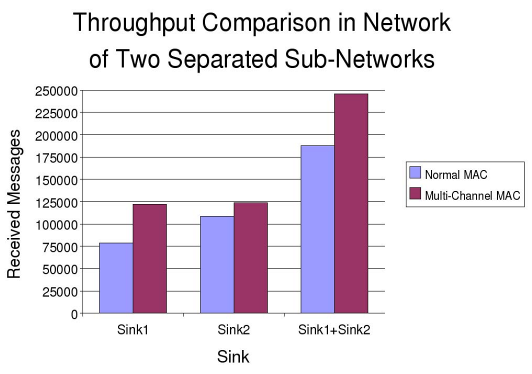

Figure 15. Testbed comparison in network with two separated sub-networks.

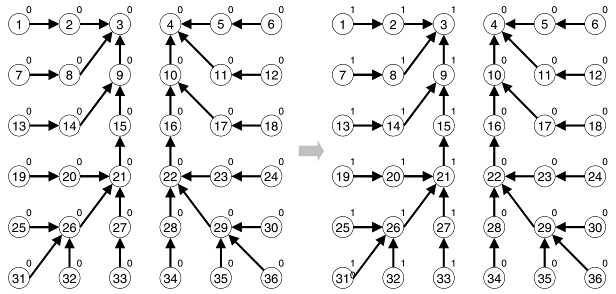

Figure 16. 36-node network with two aggregation trees placed close to each other.

At the end of the simulation, the nodes in each aggregation tree end up at different channels, which helps the throughput improve considerably at each node. The throughput increases by roughly 50% at the aggregation points as shown in Figure 17.

7. Conclusions

This paper presented a practical design, implementation, and evaluation of a multi-channel MAC protocol for wireless sensor networks (WSNs). The multi-channel MAC protocol constitutes the first real implementation that considers the hardware constraints (single half-duplex radio interfaces and non-trivial channel switching times) associated with commonly used WSN motes.

A distributed heuristic was proposed to partition nodes among channels in a way that keeps costly cross-channel communication to a minimum. Furthermore, a simple feedback control strategy was designed to oversee the partitioning process in a way that ensures stability and avoids congestion.

The MAC protocol is simple and light-weight and was evaluated on a proof-of-concept testbed with MicaZ motes.

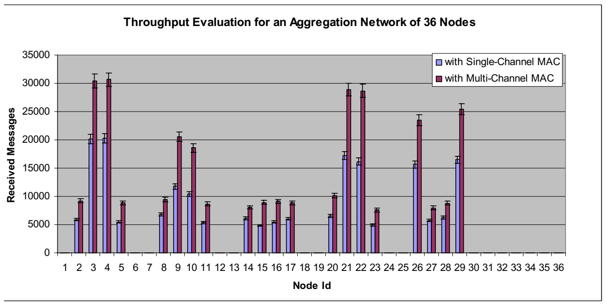

Figure 17. Throughput comparision for the 36-node aggregation network in simulation

The evaluation showed that the multi-channel protocol was successful in avoiding network congestion and achieved performance improvements compared to the single-channel case.

In this work, we focused on improving network throughput. Message delay, message loss, and power management issues are left for furture studies. Details and source code of the proposed multi-channel MAC protocol can be downloaded from:

`http://cs.uiuc.edu/homes/hieule2/IPSN08/`

References

[1] `http://www.xbow.com/Products/productdetails.aspx?sid=164`.

[2] `http://www.tinyos.net/tinyos-2.x/doc/html/tutorial/usc-topologies.html`.

[3] Technical overview of time synchronized mesh protocol (tsmp). TSMP White Paper, http://www.dustnetworks.com/, 2006.

[4] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Proceedings of IEEE Broadnets'04*, San José, CA, 2004.

[5] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proceedings of ACM MobiCom'04*, Philadelphia, PA, 2004.

[6] X. Chen, P. Han, Q.-S. He, S. liang Tu, and Z.-L. Chen. A multi-channel MAC protocol for wireless sensor networks. In *Proceedings of The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, Seoul, Korea, 2006.

[7] O. Goldschmidt and D. S. Hochbaum. Polynomial algorithm for the K-cut problem. In *IEEE 29th Annual Symposium on Foundations of Computer Science*, pages 444–451, 1988.

[8] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94. ACM Press, 2004.

[9] N. Jain, S. R. Das, and A. Nasipuri. A multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless networks. In *Proceedings of IEEE IC3N'01*, Scottsdale, AZ, 2001.

[10] Y. Kamidoi, S. Wakabayashi, and N. Yoshida. Faster algorithms for finding a minimum K-way cut in a weighted graph. In *1997 IEEE International Symposium on Circuits and Systems*, Hong Kong, 1997.

[11] P. Kinney and D. Sexton. Isa100.11a release 1 - an update on the process automation applications wireless standard. http://www.isa.org/isasp100/, 2008.

[12] P. Kyasanur and N. H. Vaidya. Routing and interface assignment in multi-channel multi-interface wireless networks. In *Proceedings of IEEE WCNC'05*, New Orleans, LA, 2005.

[13] H. K. Le, D. Henriksson, and T. Abdelzaher. A control theory approach to throughput optimization in multichannel collection sensor networks. In *IPSN 2007*, Cambridge, MA, 2007.

[14] P. Levis. Tep 116: Packet protocols. `http://www.tinyos.net/tinyos-2.x/doc/html/tep116.html`, 2006-06-27.

[15] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of ACM SenSys'03*, Los Angeles, CA, 2003.

[16] A. Nasipuri and S. R. Das. Multichannel CSMA with signal power-based channel selection for multihop wireless networks. In *Proceedings of IEEE VTC'00*, Boston, MA, 2000.

[17] J. A. Patel, H. Luo, and I. Gupta. A cross-layer architecture to exploit multichannel diversity with a single transceiver. In *INFOCOM Minisymp. 2007*, 2007.

[18] R. Simon, L. Huang, E. Farrugia, and S. Setia. Using multiple communication channels for efficient data dissemination in wireless sensor networks. In *MASS 2005*, 2005.

[19] J. So and N. H. Vaidya. A multi-channel MAC protocol for ad-hoc wireless networks. In *Proceedings of ACM Mobihoc'04*, 2004.

[20] A. Tzamaloukas and J. Garcia-Luna-Aceves. Channel-hopping multiple access. In *Proceedings of IEEE ICC'00*, New Orleans, LA, 2000.

[21] L. Wang and S. S. Kulkarni. *appa*: Gossip based multichannel reprogramming for sensor networks. In *DCOSS*, pages 119–134, 2006.

[22] L. Zhao, H. Nagamochi, and T. Ibaraki. Approximating the minimum K-way cut in a graph via minimum 3-way cuts. In *ISAAC '99: Proceedings of the 10th International Symposium on Algorithms and Computation*, pages 373–382, London, UK, 1999. Springer-Verlag.

[23] G. Zhou, C. Huang, T. Yan, T. He, J. A. Stankovic, and T. F. Abdelzaher. MMSN: Multi-frequency media access control for wireless sensor networks. In *Proceedings of the IEEE Infocom*, Barcelona, Spain, 2006.

[24] G. Zhou, J. Stankovic, and S. Son. The crowded spectrum in wireless sensor networks. In *Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets 2006)*, Cambridge, MA, 2006.

[25] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. USC Tech Report 04-823.