

Minimum Delay Routing for Wireless Networks with STDMA

Fikret Sivrikaya
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180
Email: sivrif@cs.rpi.edu

Bülent Yener
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180
Email: yener@cs.rpi.edu

Abstract—STDMA emerges as a promising channel access technique for providing QoS guarantees in multi-hop ad hoc networks such as community mesh and sensor networks. The contention-free channel access combined with spatial reuse of the channel provide significant benefits in the energy/throughput trade-off. On the other hand, the time-multiplexed communication introduces extra delay on the packets when relayed by intermediate nodes. Hence in large wireless sensor networks or mesh networks, where data is routed over several hops before reaching the data sink, STDMA protocols may introduce high end-to-end latency due to the reservation-based access policy. We argue that a suitable routing protocol specifically designed for reservation-based MAC protocols can alleviate their high-latency drawback. Following this argument, we propose first such routing algorithms working on top of a generic STDMA MAC protocol. First, we consider routing with data fusion and present our GreenWave routing idea. We show that our algorithm significantly reduces the end-to-end delay when compared to routing over the shortest-hop paths. Second, we consider routing without data fusion, by taking into account the effect of congestion along the paths on the end-to-end delays. We provide a QIP formulation of the problem, and present a lower bound and a heuristic algorithm to bound the optimal solution. Based on the centralized heuristic algorithm, we propose a distributed, dynamic routing protocol *GWCF (GreenWave routing with Congestion and Flow control)*, which uses a novel congestion and flow control technique utilizing the underlying contention-free protocol. We show by simulations that *GWCF* routing significantly improves the end-to-end delay while increasing the network throughput when compared to routing over shortest paths.

Keywords: *Community Mesh Networks, Sensor networks, QoS, Routing, STDMA*

I. INTRODUCTION AND BACKGROUND

Wireless sensor networks receive significant research focus as advances in technology made feasible the design

and deployment of tiny sensors with radio communication capabilities. Many application scenarios for sensor networks are projected such as environment monitoring, disaster recovery, emergency and military applications. The characteristics of a sensor network may vary greatly according to the specific application scenario. However most sensor networks share some basic features. First, the lifetime of the network depends on the limited energy available at individual sensor nodes. Second, cheap sensor nodes are usually deployed in large amounts. Third, the sensor devices may fail frequently due to their low-cost nature. From the designer's perspective these features give rise to the issues of energy saving, scalability, and fault-tolerance.

Community mesh networks are also gaining increased popularity as an alternative for providing broadband Internet access in urban areas, while facilitating local communications and information sharing [1], [2]. A mesh network is typically a form of multi-hop ad hoc network consisting of mesh routers and mesh clients. Mesh routers have minimal mobility and form the mesh backbone for mesh clients [3]. MIT's *Roofnet* [4] is a project under development for providing broadband Internet access to local communities, with experimental deployments currently in Cambridge, MA and Portland, OR. In such community networks, the multi-hop router backbone forwards data between the clients and the Internet gateway. The gateway acts like the data sink in a sensor network, and the clients can be thought of as event sources, whose data are delivered to the sink by mesh routers (acting like the sensor nodes in a sensor network). Hence in order to unify the notation throughout this paper, we use the terms *sensor* and *sink* nodes, representing both community mesh and sensor network architectures.

Medium Access Control (MAC) protocols determine

how nodes in a wireless network access the shared wireless medium and they play a crucial role on the overall throughput and energy utilization in the network. Collision of radio packets in the wireless medium has usually a destructive effect on the packets and necessitates retransmission at the senders. Severe energy wastage may be caused by frequent collisions in a wireless network. A natural objective of a MAC protocol is thus to avoid or reduce collisions for better energy and channel utilization.

There is considerable work on MAC protocols for wireless ad hoc and sensor networks. Most of the initial attempts are based on the CSMA technique or are variations of the IEEE 802.11 protocol, trying to improve their power-saving characteristic by occasionally turning off the radio transceivers of sensor nodes. S-MAC [14] is the first among those, introducing the *low-duty-cycle* operation in sensor networks by letting nodes periodically *sleep*, i.e. turn off their radio. The T-MAC protocol [15] attempts to improve on S-MAC protocol's energy conservation by using variable duty-cycles as opposed to the fixed, pre-defined duty cycles in S-MAC. Polastre et al. proposed B-MAC in [16]. B-MAC is a light-weight protocol that provides an interface to the applications to implement their own MAC. By using a small amount of information from the applications, B-MAC minimizes idle listening to provide increased energy savings. B-MAC is shown to have higher throughput and better energy efficiency than S-MAC and T-MAC, and is used as the default MAC for Mica2 motes [17]. Z-MAC [18] is a hybrid protocol in the sense that it combines TDMA and CSMA. The communication is slotted as in TDMA protocols and each node is assigned a time slot, however nodes may also transmit during other slots in Z-MAC with less priorities than the slot owners. Z-MAC behaves like CSMA under low contention and like TDMA under high contention.

Many recent works argue that contention-free MAC protocols may be more suitable for the stationary and energy-constrained sensor networks, most of them providing channel access scheduling algorithms based on the spatial reuse time division multiple access (STDMA) technique. In [19], a set of such MAC protocols are proposed which are the first completely distributed, asynchronous, conflict-free MAC protocols for wireless ad-hoc networks. The basic approach is to use a randomized protocol in which nodes try to negotiate with neighbors for conflict-free time slots in a distributed manner. The fundamental property of those protocols is that they do not rely on a global time synchronization,

and they utilize only local communications, which make them scalable choices for large wireless sensor networks. TRAMA [20] is a TDMA-based MAC protocol based on the idea of Neighbor-Aware Contention Resolution (NCR) [21]. In this approach, each node calculates the priority of its 1-hop and 2-hop neighbors by applying an MD5 hash of the concatenation of the node id and the time slot t . The node with the highest priority is the owner of slot t . TRAMA assumes that adequate synchronization is attained in the network, such that all nodes have the same perception of time slots. Simulation results show that TRAMA exhibits higher throughput compared to S-MAC and 802.11, and significant energy savings (since nodes can sleep for up to 87% of the time). The BitMAC protocol [22] is based on the assumption that a receiver may hear the bitwise "or" of multiple transmissions from *synchronized* senders, which is the case when On-Off-Keying (OOK) signalling mode is used at the radio. BitMAC uses a spanning tree of the network with the sink at the root. Time-division multiplexing is used among each *star*, which consists of an internal node in the spanning tree and its immediate children. BitMAC assumes that the radio supports a sufficient number of communication channels, so that neighboring stars can be allocated different channels to avoid interference between them. Herman and Tixeuil [23] present a distributed TDMA slot assignment algorithm which is based on a fast clustering technique. The time slot allocation is done by the *leaders* in clusters. The DE-MAC protocol in [24] uses the TDMA technique together with periodic listen and sleep schedules to avoid major sources of energy wastage. In [25], energy-aware routing and MAC protocols are presented, which are cluster based algorithms controlled by the gateway node of each cluster. A self-organizing algorithm is given in [26] to schedule the activation of links in the network, which is a combined approach for constructing a connected network structure and conflict-free communication schedule. A synchronous deterministic self-stabilizing TDMA MAC protocol appears in [27]. In the more recent works [28] and [29], slot assignment strategies for STDMA in wireless ad hoc networks are investigated. A distributed, robust and scalable TDMA scheduling algorithm, DRAND, is proposed in [28]. The authors state that DRAND is ideal for wireless networks with limited mobility, such as wireless sensor and community mesh networks. In [29], link scheduling and node scheduling approaches for spatial reuse TDMA in wireless ad hoc networks are compared, and a new assignment strategy is proposed that combines the ad-

vantages of both.

Some multi-hop network applications utilize a resource saving technique called *in-network data fusion*, which is especially useful for better energy utilization in sensor networks. In-network data fusion is the idea of aggregating data from multiple sources/sessions before relaying to the next hop so that the total communication load, hence the energy dissipation, is reduced. However, it is not possible to utilize this technique in all multi-hop ad hoc network applications.

In this paper, we consider the problem of minimizing the end-to-end communication delays for both cases; routing with and without data fusion. Our contributions are three-fold. (i) First we propose *GreenWave routing*, an algorithm to construct end-to-end (sensor-to-sink) routes on a sensor network where data are fused along the routing paths. We show by simulations that GreenWave routing provides significant improvements in latency when compared to routing over shortest paths. (ii) We formulate the problem of delay optimal routing without data fusion as a quadratic integer programming (QIP). Then we provide a lower bound and an efficient heuristic, bounding the optimal solution to the QIP from above and below. (iii) Based on the centralized heuristic, we propose a distributed, dynamic routing algorithm, GreenWave with Congestion and Flow control (GWCF), which is shown to improve both end-to-end delay and throughput performance of a sensor network in the no-data-fusion case.

The rest of the paper is organized as follows. In the next section, the assumptions and notations used in this paper are given, followed by the motivation and overview of GreenWave routing in Section III. In Section IV, we consider routing with in-network data fusion, and study the performance of GreenWave in that case through simulations. Section V is devoted to the analysis of the no-fusion case, in which we provide a QIP formulation of the problem, a lower bound algorithm, and a heuristic solution. Also in this section, we propose a distributed, dynamic routing protocol based on the centralized heuristic algorithm, and analyze its performance. Finally, Section VI concludes the paper.

II. MODEL AND ASSUMPTIONS

We study wireless ad hoc networks in which all nodes have the same transmission range (or radius). The transmission range of a node determines the set of nodes it can communicate directly, which are also called its *neighbors*.

Consider a wireless sensor or community network with n sensor nodes (mesh routers) and m sink nodes (gateways). Typically n is much larger than m , and in some cases m may be just 1. Each sensor node tries to convey its data to *any one of the sink nodes*. We assume that n is known to all nodes in the network, and each node has a unique ID. Let S be the set of all sensor nodes in the network and let S_i denote the sensor i , for $i = 1..n$. Similarly R denotes the set of all sink nodes in the network and R_j the sink node j , for $j = 1..m$. Moreover let $V = S \cup R$ be the set of all nodes in the network. For any node $v \in V$, the set of its neighbors is denoted by $\Delta(v)$. A *2-neighbor* of node v is defined as a node which is at most two hops away from v , i.e. u is a 2-neighbor of v if and only if u and v are neighbors of each other or they have at least one neighbor in common. We denote the set of 2-neighbors of v by $\Delta_2(v)$. The neighborhood size of node v is denoted by $\delta(v) = |\Delta(v)|$ and the largest neighborhood size in the network by δ . Similarly, the 2-neighborhood size of node v is $\delta_2(v) = |\Delta_2(v)|$ and the largest 2-neighborhood size throughout the network is denoted by δ_2 .

We assume that the sensor nodes are stationary, which is the common case for wireless sensor and community networks, and that the media access control is done by an STDMA-based protocol, which could be one of the many reviewed in the previous section. Rather than adopting a specific MAC protocol, we assume a generic STDMA protocol abstracted as follows: All nodes have the same frame size, i.e. number of time slots in each frame, denoted by Λ . Each node $u \in V$ has a time slot t_u and keeps a local table (local frame), F_u , of its neighbors' time slots relative to its own. If $v \in V$ is a neighbor of u , the entry $F_u(v)$ is the time slot in u 's frame that coincides with t_v . This is depicted in figure 1. Any distributed STDMA protocol can be mapped onto this abstract structure, even in the lack of global time synchronization, such as in [19].

On the other hand, most of the ideas presented in this paper can be easily applied to reservation-based medium access protocols in general. However, in order to provide more concrete results such as performance comparisons based on simulations, we present our ideas based on a generic STDMA MAC protocol as described.

In this paper we consider the problem of finding delay optimal routing paths between sensor and sink node pairs, with the objective of minimizing the total end-to-end delay. The proposed approach could be used for both pull (sensor to sink) and push (sink to sensor)

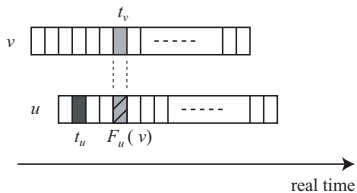


Fig. 1. The time slot of a node relative to another node's time frame. The values t_u and $F_u(v)$ are according to node u 's local clock, and t_v is according to v 's.

modes of communication [7]. However, for the clarity of presentation, we will only consider constructing routes from sensors to sink nodes. The application of proposed ideas to the other case is quite straightforward.

III. GREENWAVE ROUTING - MOTIVATION

End-to-end delay guarantees is one of the most important Quality of Service (QoS) parameters in real-time applications of community networks and some wireless sensor network applications, such as surveillance and tracking. As it is generally harder to provide delay guarantees with contention-based schemes in wireless ad hoc networks, STDMA protocols emerge as a promising alternative, given the stationary placement of sensor devices, and the high potential for spatial channel reuse. To the best of our knowledge this paper proposes the first routing protocols that improve the QoS level of STDMA MAC protocols by effectively decreasing the end-to-end delay in wireless sensor and mesh networks.

The collision-free transmissions in time-multiplexed channel access schemes come with the price of increased latency, especially in multi-hop communication environments such as sensor networks. From the source to its destination, a packet suffers some delay at each intermediate node due to the fact that a node has to wait for its time slot before it can relay the packet to the next hop. However if the routing path is carefully chosen for a source-destination pair, the delay can be minimized. Figure 2 demonstrates two extreme cases for a packet routed between u and v via two intermediate nodes. On the top, the relaying nodes x and y have consecutive time slots such that it takes only three time slots for a packet to reach from u to v , whereas at the bottom, an unfortunate selection of relaying nodes increases the end-to-end delay to 19 time slots. This observation is the basic motivation of this work.

The movement of packets in a network could be regarded as the vehicle traffic in a city. Then each

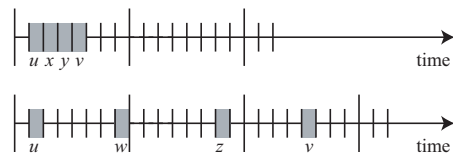


Fig. 2. Demonstration of how significantly route selection affects the end-to-end delay in time slotted communication. Shaded slots represent the relaying of data by the labelled node.

relaying (intermediate) node along the packet's route might represent an intersection. The packet waits at relaying nodes for the allocated time slot, just as a vehicle waits at intersections for the green light. The *green-wave* idea in traffic regulation is widely used in large cities to allow continuous flow of traffic along the main routes, and provides the inspiration for the name of our protocols.

Our approach resembles to those of Time-Driven Priority (TDP) and Isochronet switching architectures for traditional high speed networks. In TDP [30], the channel is also time-multiplexed but the nodes can use *any slot* in their frame to route different QoS packets. Thus in effect, the frame at one node is divided into a set of slots with different colors and the realtime packets of one class are guaranteed to be forwarded in its assigned color. This provides a pipelining architecture where realtime traffic gets exactly one hop closer to the destination in each time step. Isochronets [31], on the other hand, consider given routing trees in which data can move uninterruptedly as long as there is no competing traffic. They employ RDMA (route division multiple access) for scheduling between routes when they have data coming in simultaneously.

IV. ROUTING WITH DATA FUSION

We first focus on a network scenario using in-network data fusion, i.e. any incoming data to a node are aggregated in some way and then relayed to the next hop. For instance, consider a scenario where each sensor simply reports some physical sensor reading such as temperature or pressure. Depending on the specific application, it may be sufficient for a relaying node, after receiving readings from multiple sensors, to transmit some set of statistics of the received values –such as $\langle \text{average, min, max} \rangle$ – instead of transmitting each individual reading. In-network data fusion is a method of significant energy savings and should be employed wherever the application scenario allows. With in-network data fusion,

there is no queuing delay since a node may receive multiple packets from different sources in a single time frame, but it suffices to form and send a single packet after processing all received packets so far in the frame. Hence the amount of congestion at relaying nodes, or the number of packets received in a single frame, has no effect on the relaying delays. In this section we formulate and solve the problem of minimizing the end-to-end delay in such sensor network applications.

As the first step, we construct an auxiliary *directed* graph $G(V, E)$, where $V = S \cup R$ is the vertex set corresponding to the set of sensor and sink nodes in the network, and E is the edge set corresponding to the links between pairs of nodes that are within each other's transmission range. We use the same notations for vertices in the graph and nodes in the network to ease the presentation, and it will be clear from the context which one is implied. Since we are interested in finding routes from the sensor nodes to the sink nodes in the network, we adapt the terms *source vertices* and *destination vertices* in the graph corresponding to sensor nodes and sink nodes in the network, respectively. Hence each $S_i \in S$ is called a source vertex, and each $R_j \in R$ is called a destination vertex in graph G . We assume that G is strongly connected, which is equivalent to the basic assumption that the network is connected.

For each link (u, v) in the wireless network, there are two weighted (and directed) edges, (u, v) and (v, u) , in the auxiliary graph. The weight (cost) of the directed edge (u, v) is denoted by $w(u, v)$ and is assigned as

$$w(u, v) = (F_u(v) - t_u) \bmod \Lambda.$$

Intuitively, weight $w(u, v)$ is the amount of delay observed if node v relays a packet received from node u . By the intrinsic property of STDMA protocols, two neighbor nodes can not have overlapping timeslots, i.e. $t_u \neq F_u(v)$, for any $(u, v) \in E$. Hence we have

$$1 \leq w(u, v) \leq \Lambda - 1$$

Figures 3 and 4 demonstrate a simple example network and the corresponding auxiliary graph. A network with 6 nodes is shown in figure 3. The set of sensor nodes and sink nodes are not identified as they are irrelevant for the construction of the auxiliary graph. The local time frame and allocated time slot of each node is also shown, respecting their alignments in real time. Figure 4 represents the auxiliary graph corresponding to this network. Consider, for example, nodes a and b in figure 3. Suppose that b receives a packet from a which it needs to relay to some other node. After a sends this packet

at its timeslot t_a , node b can not relay it immediately after receiving it; it has to wait for its allocated time slot t_b . Note that t_b coincides with the time slot in a 's frame which is three slots after t_a . Hence the packet will suffer a delay of 3 time slots, which also includes the transmission time of the packet. Therefore the weight of the directed edge (a, b) is set to 3 in the auxiliary graph. If, on the other hand, a forwards a packet of b , then the delay would be $-3 \bmod \Lambda = 7$, since $\Lambda = 10$ for this example. Also note that $w(u, v) + w(v, u) = \Lambda$ for any two vertices u and v in the auxiliary graph.

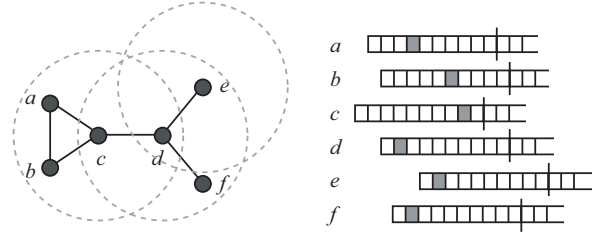


Fig. 3. An example network with 6 nodes. Transmission ranges are shown only for nodes c , d and e , but all links are demonstrated by the lines between neighbor nodes. On the right is the local time frames of each node shown with their relative alignment.

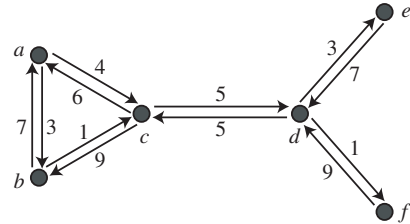


Fig. 4. The auxiliary directed graph G corresponding to the network in Figure 3.

Our objective is to minimize the end-to-end delay from each sensor node to *any* sink node. More precisely, for two nodes u and v , let $\psi(u, v)$ denote the shortest path length between vertices u and v in graph G , where path length is defined as the sum of the weights of all edges on that path. Then for each source vertex S_i , the objective is first to find its *closest* destination vertex R_j , which we define as the vertex satisfying

$$\psi(S_i, R_j) = \min_{k=1..m} \psi(S_i, R_k).$$

Then for each such (S_i, R_j) pair, we want to find the shortest path from S_i to R_j .

Given the auxiliary graph G , we observe that the problem is a generalization of the single-destination shortest paths problem such that there are *multiple copies of the destination*. Here we use the phrase “single-destination shortest paths”, noting that single-source and single-destination shortest paths problems are equivalent since one can be converted to the other by reversing all edge directions in the graph. A generalized version of the Bellman-Ford algorithm [6] may be used to construct shortest-path trees, each rooted at a destination vertex. In this approach each sink node (destination vertex) can be treated as a copy of the single destination. Then the only modification needed is to allow multiple destinations in Bellman-Ford algorithm by initializing the d values of all destination vertices to 0, where d is the local shortest-path estimate.

For the distributed implementation of the algorithm, each node broadcasts its current distance estimate in its contention-free time slot. Consider node u ; during the Λ times slots between any two occurrences of its time slot t_u , each neighbor of u gets a chance to access the channel and communicate its distance estimate. Hence, the algorithm executes in a synchronous manner, such that during each time period of Λ time slots, all nodes broadcast their distance estimates once and all links in the network are relaxed, corresponding to one iteration of the centralized algorithm. Algorithm 1 presents the local execution of the algorithm at node u at some instant t , assuming that u has a virtual clock $C_u(t)$, which is simply a counter incremented at each time slot.

Algorithm 1 GreenWave(node u , time t)

- 1: **if** $C_u(t) \bmod \Lambda = t_u$ **then**
 - 2: broadcast $d[u]$;
 - 3: **else**
 - 4: **if** received $d[v]$ from neighbor v , such that
 $d[v] + w(u, v) < d[u]$ **then**
 - 5: $d[u] \leftarrow d[v] + w(u, v)$;
 - 6: $\pi[u] \leftarrow v$;
-

Each node executes Algorithm 1 locally in every time slot after initializing its local variables. It would suffice for each node to execute the algorithm for $\tau = n\Lambda$ time slots, given that they start execution at the same time. Even if there is a drift between the times the nodes start the algorithm, each node may execute the algorithm for an increased amount of time to offset the different starting times at individual nodes. The value of τ could easily be adjusted by achieving an upper bound on the maximum difference between the times each node starts

the algorithm. We can safely assume that $\tau = O(n)$, and the message complexity of the algorithm is then $O(n)$ for each node, since each node sends only one message for each of the τ time frames.

After node u is done with the GreenWave algorithm by executing it for τ time slots, it forwards all incoming packets to $\pi[u]$.

A. Performance Analysis

In this section we analyze the effect of GreenWave routing on the end-to-end delay of sensor-to-sink paths. To the best of our knowledge, there is no routing protocol specifically designed to work on top of STDMA MAC protocols, hence to quantify the gain by routing data over *green-wave* paths, we compare GreenWave routing to the shortest-hop routing, which is a basic protocol that routes packets over minimum-hop paths. Many routing protocols in the literature are based on shortest-hop routing. Moreover, we also compare the abstract STDMA protocol to the contention-based 802.11 protocol. Our objective is to demonstrate that GreenWave routing decreases the end-to-end delay in a network such that under certain conditions, a contention-free protocol may surpass a contention-based protocol regarding the end-to-end delays even at unsaturated traffic conditions.

We use OPNET Modeler (11.5) to simulate the GreenWave routing protocol operating on top of an STDMA protocol as described in Section II. We implement the randomized slot allocation technique described in [19] for the STDMA protocol. The results for the IEEE 802.11 protocol is obtained using its standard model included in the wireless module of OPNET Modeler. We also implement a shortest-hop routing algorithm which works with both 802.11 and STDMA MAC protocols.

The frame size is a crucial parameter of STDMA protocols, significantly effecting the protocol performance. Note that contention-free time slot allocation in a multi-hop ad hoc network is equivalent to the distance-2 vertex coloring in a graph. Because of the hidden terminal problem, nodes in a network that are within two-hop distance can not be assigned the same time slot, just as 2-distant vertices in a graph can not be assigned the same color. Hence the smallest possible frame size can not be determined efficiently, since the coloring problem is NP-complete. However, a simple upper bound on the minimum frame size is the maximum 2-neighborhood size, δ_2 , in the network, meaning that any frame size larger than δ_2 is feasible. In obtaining the results of this section, we calculate δ_2 in the given graph, and set this value as the frame size in the STDMA protocol.

Another parameter needed for the STDMA protocols is the *slot time*, or the duration of each time slot, which should be long enough for a packet to be transmitted over the channel. We use 1 Mbps as the channel rate, so that it is compatible with the 802.11 protocol. Though the channel rate is typically smaller for most sensor network applications, our general inference from the results on STDMA protocols would not be affected for a different channel rate. On the other hand, whenever we make a comparison between STDMA and 802.11 protocols, the use of 1 Mbps as the channel rate is in the favor of 802.11, as lower data rates are not supported by the standard. Unless otherwise stated, we use a packet size of 1 Kbits for the simulations in this section, hence the slot time of 1 ms. is appropriate for the STDMA protocol.

Performance metric — Given a sensor/mesh network, we compute the end-to-end delay of the path from each sensor to its closest sink. We then average the delay over all such paths to get the performance of the protocol on the given network. Throughout this section, each data point is an average value over 20 random networks, which are generated by a random geometric graph model: nodes are scattered in a unit torus, uniformly at random, and a pair of nodes that are at most r units apart from each other are connected (set as neighbors in the network), where $0 < r < 1$ is the transmission radius. We also ensure that each randomly generated network is connected. For all networks studied in this section, we set the number of sink nodes as three.

To ease the presentation of results, we adapt the notations *SH* for shortest-hop routing, *GW* for GreenWave routing.

Figure 5 shows the average end-to-end delay and single-hop delay values for GreenWave routing and the shortest-hop routing on an STDMA protocol. The network size is varied from 500 to 1000 nodes, keeping the transmission radius constant at 0.1 units. As the network size increases with a fixed transmission radius the network gets denser, necessitating a larger frame size. In Figure 5, as the network grows from 500 to 1000 nodes, the average frame size increases from 68 to 135 and both the end-to-end delay and the average single-hop delay almost doubles when shortest-hop routing is used. On the other hand, the delay figures almost remain constant with GreenWave routing, albeit the increase in frame size. As the network gets denser, the chances of GreenWave routing to find *fast paths* increases, hence the delay does not increase significantly despite larger frame sizes.

In Figure 6, the contention-free STDMA protocol is

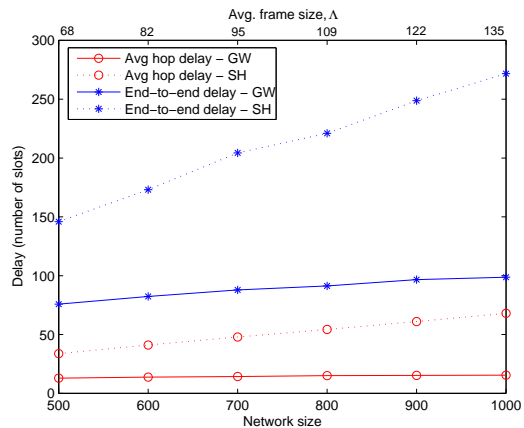


Fig. 5. Average single-hop and end-to-end delay versus network size for an STDMA protocol using both GreenWave (GW) and shortest-hop (SH) routing.

compared to the contention-based 802.11 protocol under different conditions of traffic load. The network size is fixed at 1000 nodes and the frame size for the STDMA protocol, Λ , equals 135 on average, which is set to δ_2 for each random network. The inter-arrival times for the sensing events are assumed to be exponentially distributed with mean values varied from 0.1 seconds to 0.5 seconds. The traffic load has no effect on the STDMA delays, since there is no queuing involved due to in-network data-fusion. On the other hand, higher traffic load causes higher contention and increased channel access latency in 802.11. Hence, as the event inter-arrival times increase, the end-to-end delay decreases for 802.11. We observe that even under relatively moderate traffic load of 3 packets per second, the STDMA protocol end-end delay falls below that of 802.11 when GW routing is used. On the other hand, when SH routing is used for both protocols, STDMA performs much worse compared to the 802.11 protocol. The aim of this figure is not to provide an accurate comparison of contention-free and contention-based schemes, but to demonstrate the significant effect of coupling a contention-free MAC protocol with an appropriate routing protocol.

V. ROUTING WITHOUT DATA FUSION

In some scenarios, in-network data fusion may not be available such that the relaying node forwards each single packet it receives as is. Then the delay observed on a packet is greatly affected by how many other packets the relaying node receives in that time frame. In other words, the flow of different commodities may have strong effects on each other's end-to-end delays.

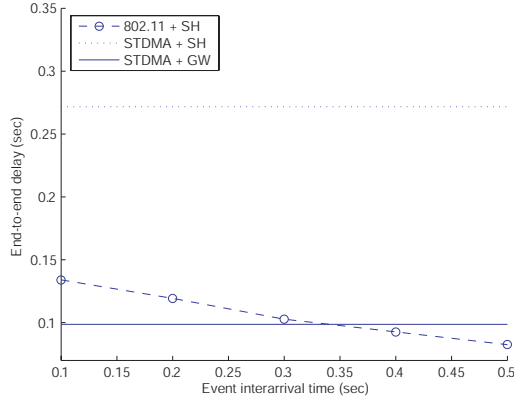


Fig. 6. Average end-to-end delay versus traffic load for contention-free (STDMA) and contention-based (802.11) MAC protocols using both GW and SH routing, on a network of 1000 nodes.

Let us first demonstrate this effect by a simple example.

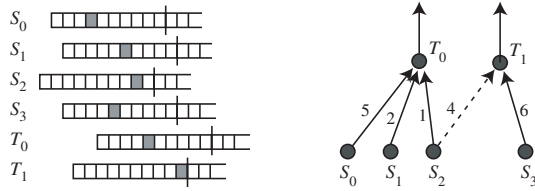


Fig. 7. A small sample network. Local time slots of nodes are shown on the left. The delay on each link (u, v) is calculated by $w(u, v) = (F_u(v) - t_u) \bmod \Lambda$, and the resulting routing tree is shown on the right. The dashed arrow only shows that S_2 and T_1 are neighbors.

Consider the simple example in figure 7, where there are four sensor nodes and two relaying nodes. To illustrate the idea, assume for now that the four sensors are the only data producers and the relaying nodes just forward their data. The local allocated time slots for each node are also shown on the left. If the link weights are assigned simply by the formula $w(u, v) = (F_u(v) - t_u) \bmod \Lambda$, then three nodes report their data through T_0 and only one node to T_1 as a result of the algorithm. However, if in-network data fusion is not applicable and if all three nodes S_0, S_1, S_2 have packets to transmit in the same time frame, then T_0 will have three packets to relay, where it can only send one at a time. Therefore the effective delay on these packets will be higher than the ones shown on the right in figure 7. If we assume that each node transmits a packet in every time frame, and that T_0 transmits the packets one by one in the order

they were received, then in the steady state the delays for S_0, S_1, S_2 will all go up by 2Λ as shown on the left in figure 8. On the other hand, S_2 is also a neighbor of T_1 and thus T_1 could relay its packets as well. The graph on the right hand side of figure 8 shows the effective delays if S_2 joins the tree of T_1 instead of T_0 . Note that the total delay -total link cost- on the right is less than the one on the left, hence is preferable. In this section, we try to model this modified problem by taking into account the interaction among different flows in the network.

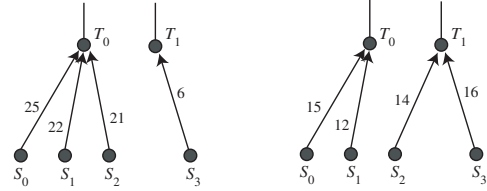


Fig. 8. The two possible solutions to the example of figure 7, this time taking into account the interaction among different flows for calculating delays. The routing tree on the right yields a lower total delay and hence is preferable.

A. Problem Formulation

Suppose that we are given a *routing graph* which is formed by the union of n paths, from each source vertex to some destination vertex. Each path in the routing graph defines a flow originating from a unique source vertex. We define the *congestion* on vertex v as the number of different paths going through v , and denote it by C_v . Hence the corresponding node v relays the packets of C_v different nodes, which we call the *descendants* of v . Note that v also “relays” its own traffic, hence a total of $C_v + 1$ distinct flows are leaving out of v . Assume that each node senses an event (hence transmits) in each time frame with probability p . The fixed probability p may be considered as an averaged quantity that reflects a possibly bursty traffic generation in the long term. This approach has been extensively applied before to make the theoretical modeling tractable [8] [9] [10] [11]. p also serves as a tuning parameter in our model to represent network scenarios with different event frequencies – hence different traffic loads.

Consider an edge (u, v) on the routing graph. Assume that some flow k is active on (u, v) at some instant. We want to find the *effective delay* on flow k incurred by the link (u, v) by taking into account (i) the time slots of u and v , and (ii) other flows that are relayed by v . With probability $(1 - p)^{C_v}$, none of the other descendants of v transmits in this frame, and the delay incurred by the

link (u, v) is then just $w(u, v)$. Similarly, with probability $C_v p(1-p)^{C_v-1}$, only one of the other descendants of v transmits in this frame, and in this case the delay for u goes up to $w(u, v) + \Lambda$. Note that the packet of flow k may be the first one to be relayed by v , but in the steady state, if two nodes are sending packets and v is relaying them alternately, then both of them suffer Λ amount of extra delay. Considering all possible cases, we can calculate the expected value of the delay when v relays a packet from u , for some given flow. Let $w'(u, v)$ be the effective delay on edge (u, v) . Then –using $E[X] = \sum xP\{X = x\}$ – we can compute the expected value of $w'(u, v)$ as

$$\begin{aligned} E[w'(u, v)] &= \sum_{i=0}^{C_v} \left[(w(u, v) + i\Lambda) \binom{C_v}{i} p^i (1-p)^{C_v-i} \right] \\ &= w(u, v) \cdot \sum_{i=0}^{C_v} \left[\binom{C_v}{i} p^i (1-p)^{C_v-i} \right] \\ &\quad + \Lambda \cdot \sum_{i=0}^{C_v} \left[i \binom{C_v}{i} p^i (1-p)^{C_v-i} \right] \\ &= w(u, v) + pC_v\Lambda. \end{aligned}$$

The last step follows from the fact that the first summation is the sum of all probabilities for a random variable \mathbf{X} that follows a binomial distribution, $b(i; C_v, p)$, and the second summation is the expected value of \mathbf{X} , which equals pC_v .

For convenience of presentation, we call the first term in (1), $w(u, v)$, the *static cost* and the second term, $pC_v\Lambda$, the *congestion cost*. Instead of assigning this cumulative cost as a whole to the edge, we attribute the static cost to the edge, (u, v) , and the congestion cost to vertex v . Hence the cost of vertex v is $pC_v\Lambda$. This approach will allow us to model the problem more neatly and help us deriving a lower bound.

We now present a quadratic integer programming (QIP) formulation of the problem based on a variation of the multicommodity flow problem. We define a commodity originating from each sensor node, destined to any one of the sink nodes. Therefore there is a total of n commodities (flows). We then try to minimize the total cost (weight) of all commodities. Let X_{uv}^k be a decision variable such that

$$X_{uv}^k = \begin{cases} 1 & \text{if } (u, v) \in E \text{ is used for commodity } k \\ 0 & \text{otherwise} \end{cases}$$

For any routing scheme, i.e. a collection of paths from each sensor to a sink node, the following conditions must

hold.

$$\sum_{v \in V} X_{uv}^u = 1 \quad , \quad \forall u \in S \quad (2)$$

$$\sum_{u \in S} \sum_{v \in R} X_{uv}^k = 1 \quad , \quad \forall k \in S \quad (3)$$

$$\sum_{v \in V} X_{uv}^k = \sum_{v \in S} X_{vu}^k \quad , \quad \forall k \in S, u \in S, u \neq k \quad (4)$$

$$X_{uv}^k \in \{0, 1\} \quad , \quad \forall (u, v) \in E, k \in S \quad (5)$$

Here the first constraint states that each commodity u must initiate from source vertex $u \in S$. The second constraint forces each commodity to reach a destination vertex $v \in R$. The third constraint is the flow conservation equation and must be satisfied for all vertices except the source and destination vertices of the flow. Since no flow can go through a destination vertex as an intermediate vertex, we restrict the incoming flow of a vertex, i.e. the right hand side of the constraint, to the flow coming from the vertices in S . Also note that whenever we use the term X_{uv}^k , there is an implicit constraint on u and v such that $(u, v) \in E$.

(1) Using the decision variables X_{uv}^k , the congestion cost of a vertex can be expressed as

$$C_v = \sum_{k \in S} \sum_{u \in S} X_{uv}^k.$$

The cost of a path P_k is the sum of the costs of all edges and vertices along the path, which is given by

$$Z_k = \sum_{(u,v) \in E} \left[X_{uv}^k (w(u, v) + p\Lambda C_v) \right]. \quad (6)$$

The objective is to minimize the total cost of all paths, each corresponding to a commodity originating from a source vertex, i.e. to minimize $\sum_{k \in S} Z_k$. In its open form, we have the optimization problem

$$\text{minimize } \sum_{k \in S} \sum_{(u,v) \in E} \left[X_{uv}^k \left(w(u, v) + p\Lambda \sum_{l \in S} \sum_{u \in S} X_{uv}^l \right) \right] \quad (7)$$

subject to constraints (2), (3), (4) and (5). This is a QIP problem, with a quadratic objective function and linear constraints. The solution of this minimization problem provides an optimal routing strategy that can be used to measure the performance of a practical routing algorithm. Clearly however, this QIP can be efficiently solved for only small network sizes. Hence we provide a lower bound in the next section, followed by our heuristic routing algorithm with congestion control and the numerical results to assess its performance.

Note that omitting the congestion costs, we obtain the following linear objective function that gives the solution for the data-fusion case.

$$\text{minimize } \sum_{k \in S} \sum_{(u,v) \in E} X_{uv}^k w(u,v) \quad (8)$$

The difference between the fusion and no-fusion cases is demonstrated in Fig. 10, which presents the solutions obtained in CPLEX optimizer [12] for both objective functions (7) and (8) on the simple network of Fig. 9. In Fig. 10(a), S_1 relays the flow from four nodes including itself and R_1 receives data from all five sensors, while R_2 is unutilized and S_2 does not relay any data besides its own traffic. However, in figure 10(b), the flows reaching a relaying node from different sources may be split into separate destinations, distributing the relaying burden among nodes. For instance, S_1 relays the data from S_4 to R_1 , while it relays S_3 's data to R_2 .

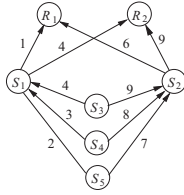


Fig. 9. Sample graph with weights assigned by $w(u,v) = (F_u(v) - t_u) \bmod \Lambda$, where $\Lambda = 10$.

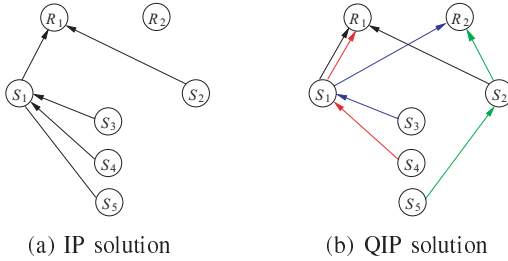


Fig. 10. The routing paths obtained for the graph of Figure 9 by (a) the objective function (8) -for the case with data fusion- and (b) the objective function (7) with $p = 0.5$ -for the case with no data fusion.

B. Lower Bound

We now try to derive a lower bound for the solution of the QIP minimization problem. Our approach is based on considering the static costs and the congestion costs independently. The congestion cost of path P_k can be extracted from (6) as

$$p\Lambda \sum_{(u,v) \in E} (X_{uv}^k \mathcal{C}_v) = p\Lambda \sum_{(u,v) \in P_k} \mathcal{C}_v.$$

The last summation term is the *path congestion*, which is the total congestion value of all nodes along the path P_k . If L_C is a lower bound on the sum of the path congestion values over all paths, then the total (network-wide) congestion cost is at least $p\Lambda L_C$. Moreover, let L_W be a lower bound for the sum of the static costs over all paths, which can be obtained as shown in Section IV. Then the total cost for all paths, the objective function value, can not be lower than $L_W + p\Lambda L_C$.

We now try to derive L_C , the lower bound on the total congestion along all paths. First we observe that each commodity has to terminate at one of m destination vertices. Hence there is always a total of n flows coming into the set of destination vertices. Let n_i denote the number of sensors reporting to sink i , hence the *congestion* on sink i is n_i . Then the *path congestion* value of each path coming into i is at least n_i , and the total congestion on these n_i paths is at least n_i^2 . Summing over all sink nodes, we get $\sum_{i=1}^m n_i^2$, which is minimized when $n_i = n/m$. Hence a simple lower bound on the overall congestion value is $L_C = n^2/m$. This bound holds regardless of the network topology, and is tight when the network graph is complete. However we can efficiently improve it considering the given topology of a network.

Given the graph G corresponding to a network topology, we first find the shortest path lengths from all sources to their closest destinations, using the hop count as the distance metric. Let $S^{(i)}$ denote the set of vertices with shortest hop distance i to any one of the destination vertices, and let $n^{(i)} = |S^{(i)}|$. Hence $S^{(i)} = \{u \in S \mid \psi_u = i\}$, where ψ_u is the shortest hop distance from u to the closest destination vertex. Also let ψ be the largest of these shortest hop distances, i.e. $\psi = \max_{u \in S} \psi_u$. Initially, we apply the same strategy on the destination vertices as we did for the general lower bound. Then we remove all destination vertices along with their incident edges from G , and set all vertices in $S^{(1)}$ as the new destination vertices. Since all paths corresponding to the $n - n^{(1)}$ remaining nodes (those that are at least two hops away from any sink node) have to terminate at one of these new destination vertices, we can apply the same reasoning above to obtain that the additional congestion introduced at this level is at least $(n - n^{(1)})^2/n^{(1)}$. We then remove all vertices in $S^{(1)}$ from the graph and apply the same idea for nodes in $S^{(2)}$ and so on until all vertices in the graph are removed. Adding the cost values at each level, we obtain the overall congestion

bound

$$L_C = \sum_{i=0}^{\psi} \frac{\left(\sum_{j=i+1}^{\psi} n^{(j)}\right)^2}{n^{(i)}}. \quad (9)$$

The lower bound on the total cost, i.e. on the objective function value of the QIP, is then $L_W + p\Lambda L_C$.

C. A Heuristic Solution

Recall that the end-to-end delay on a communication path has two main factors (i) *static cost* due to the time-multiplexed channel access and (ii) *congestion cost* due to the intercrossing traffic at relaying nodes. The Green-Wave routing presented in Section IV aimed to minimize the static costs along the routing paths. Minimizing the congestion cost requires (a) distributing/balancing the load among nearby nodes and (b) minimizing the path lengths, i.e. the number of hops from sensors to the sinks. We also observe in Section V-A, as demonstrated by Figure 10, that an optimal solution to the routing problem in no-fusion case should consist of a collection of paths from sensors to the sinks rather than a tree-based solution. Our objective is to devise a heuristic algorithm that extends the idea of GW routing so that it takes these observations into account.

We first construct an auxiliary graph G as in Section IV, with the weights based on nodes' contention-free time slots. We then find the shortest paths from each source vertex to its closest destination vertex on the auxiliary graph G . Let d_u be the minimum distance of source vertex u to a destination vertex on G . We also find the shortest *hop distance* from each source vertex to a destination vertex, denoted by h_u for vertex u . Next, we construct a spanning subgraph G' of G in the following manner: For each edge (u, v) in G , if $h_v < h_u$, we connect u to v in G' , otherwise G' has no edge from u to v . The resulting spanning subgraph is a directed acyclic graph. (Assume for the sake of contradiction that G' has a cycle $v_1, v_2, \dots, v_k, v_1$. Then we have $h_{v_1} > h_{v_2} > \dots > h_{v_1} > h_{v_1}$; a contradiction. Hence G' is acyclic.) Our heuristic performs on this reduced graph.

A directed acyclic graph has at least one vertex with no incoming edges, which we call a *leaf vertex*. Note that each leaf vertex on G' has a single flow to carry, which originates at the vertex itself. At the first iteration of the algorithm, we pick a leaf vertex, say u , and select the next hop, say v , for the flow originating at u . (The procedure for selecting the next hop will be clear later.) Then we associate this additional flow to v by incrementing its flow counter f_v by one, after which we

delete u and its outgoing edges from G' . In a sense, we are contracting u into v by assigning its flow to v , which has now two flows to route, those from u and v itself. The flow counter of each node is initialized to 1 and serves to keep track of the number of flows at each vertex. At each step of its execution, the heuristic algorithm removes a vertex u from G' and determines the next hops for f_u flows. Eventually G' shrinks into a set of R destination vertices, when the end-to-end routes for all flows are determined.

Note that the loop invariant that G' is directed acyclic is maintained at each iteration, hence there is always a leaf vertex to proceed with the algorithm until all source vertices are removed. At each iteration of the heuristic algorithm, a leaf vertex with maximum d value is selected, breaking ties arbitrarily. Let u be such a vertex. For each of the f_u flows on u , the algorithm iterates over all outgoing edges (u, v) of u , computing $d_{uv} = d_v + w(u, v) + p\Lambda f_v$ for each edge, and picks the one that yields the minimum d_{uv} value as the next hop for the flow under consideration. The last term in the computation of d_{uv} serves as our heuristic tool to distribute the load and minimize congestion. Let y be the next hop vertex selected by u for the flow originated at vertex x . Then the association of flow x is transferred from u to y , and the flow counter of y , f_y , is incremented by one. After this process is repeated sequentially for all f_u flows, the vertex u is deleted from G' along with its outgoing edges, completing one iteration of the algorithm. The next hop vertices selected for each flow are recorded during the execution of the algorithm, which are then used to construct end-to-end routes from sensors to sinks.

We now illustrate the execution of the heuristic algorithm on a simple network, for which the auxiliary graphs G and G' is shown in Fig. 11. Assume that we are given $\Lambda = 10$ and $p = 0.5$ for this example. From G , we compute d_i , the shortest distances from each sensor i to a sink node, as given in Table I.

TABLE I

THE SHORTEST DISTANCES FROM EACH SENSOR TO A SINK NODE FOR GRAPH G IN FIG. 11

node i	S_1	S_2	S_3	S_4	S_5	S_6	S_7
d_i	7	4	4	11	6	7	10

We then construct the directed acyclic graph G' based on the shortest hop distances, as shown in Fig. 11. As explained, the heuristic works on graph G' , computing the next hop for a commodity at each step. By deleting

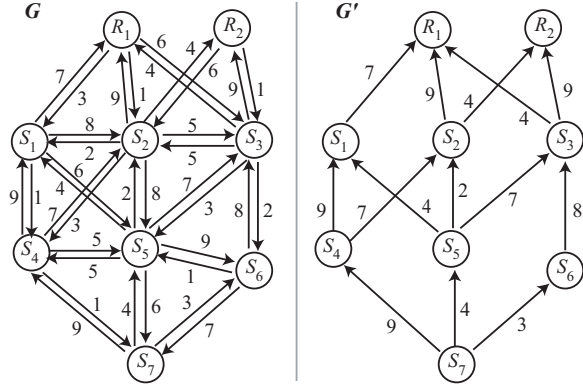


Fig. 11. Example auxiliary graphs G and G' used in the heuristic algorithm.

(contracting) the processed vertices, it eventually reduces the graph G' into a set of destination vertices. Fig. 12 presents the snapshot of G' after some selected steps of the algorithm.

In the first step, S_7 is selected as it is the only leaf vertex, after which the algorithm iterates over all outgoing edges of S_7 and computes

$$\begin{aligned} d_{S_7 S_4} &= 11 + 9 + 0.5 \cdot 10 \cdot 1 = 25, \\ d_{S_7 S_5} &= 6 + 4 + 0.5 \cdot 10 = 15, \\ d_{S_7 S_6} &= 7 + 3 + 0.5 \cdot 10 = 15. \end{aligned}$$

The minimum among those (breaking the ties arbitrarily) is chosen as S_5 in this example; hence, S_7 is contracted into S_5 and f_{S_5} is incremented to 2. In steps two and three, the leaf vertices S_4 and then S_6 are processed in the same way and contracted into vertices S_2 and S_3 , respectively.

In step four, the only leaf vertex S_5 (now shown as “ S_5, S_7 ” in the figure) is selected for processing. The algorithm processes each of the two flows S_5 and S_7 in order, before removing the vertex from G' . For the flow that originates at S_5 , it computes

$$\begin{aligned} d_{S_5 S_1} &= 7 + 4 + 0.5 \cdot 10 = 16, \\ d_{S_5 S_2} &= 4 + 2 + 0.5 \cdot 10 \cdot 2 = 16, \\ d_{S_5 S_3} &= 4 + 7 + 0.5 \cdot 10 \cdot 2 = 21. \end{aligned}$$

Assuming that S_1 is the chosen vertex among the two minimum values, the association of flow S_5 is transferred to S_1 and f_{S_1} is set to 2. Then in the fifth step, the algorithm processes the second flow at the current vertex S_5 , which is the flow that originates at S_7 . Note that we follow the vertex labels of the original graph G' in the text, so the vertex shown as “ S_5, S_7 ” in the figure is

referred to as “ S_5 ” in the following.

$$\begin{aligned} d_{S_5 S_1} &= 7 + 4 + 0.5 \cdot 10 \cdot 2 = 21, \\ d_{S_5 S_2} &= 4 + 2 + 0.5 \cdot 10 \cdot 2 = 16, \\ d_{S_5 S_3} &= 4 + 7 + 0.5 \cdot 10 \cdot 2 = 21. \end{aligned}$$

Therefore, the flow from S_7 is transferred to S_2 , after which f_{S_2} is incremented to 3, and vertex S_5 is removed along with its outgoing edges. The snapshot of G' after step five is shown in Fig. 12.

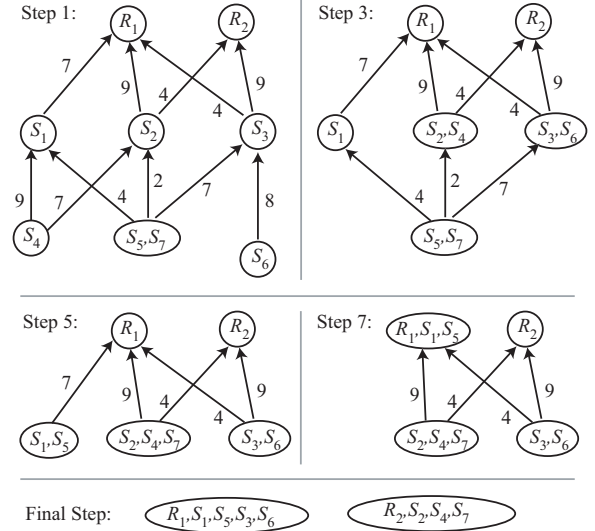


Fig. 12. Execution of the heuristic algorithm on the example network of Fig. 11, shown as a series of snapshots of graph G' taken after some selected steps of the algorithm.

The algorithm proceeds in a similar fashion until the graph G' reduces into the two-vertex graph shown in Fig. 12. Recording the transfer of flows between vertices during the algorithm execution, we can construct the final routing paths, as illustrated in Fig. 13.

D. Numerical Results

In this section we try to assess the performance of our heuristic algorithm. Given an instance of the problem, let OPT denote the optimal solution to this QIP. Since we can find the optimal solutions for small networks by using a QIP solver, we first compare against the optimal solution in Table II, presenting the results obtained for a network with 10 sensor nodes and single sink node, by varying the probability p . The OPT values are obtained using the optimization software CPLEX [12]. Since any valid routing scheme (that connects each sensor node to some sink node) is a heuristic algorithm and the result of the heuristic provides an upper bound on OPT, our original GW routing algorithm can be used to set an

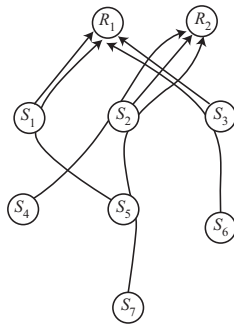


Fig. 13. The result of the heuristic algorithm on the example network of Fig.11. The arrows show the routing paths originating from each data source, also indicating the relay nodes if any.

TABLE II

COMPARISON OF THE OPTIMAL QIP SOLUTION THE RESULTS OF THE HEURISTICS AND LOWER BOUND (LB).

p	LB	OPT	HR1	HR0
0	42.76	42.76	42.76	42.76
0.1	79.44	88.42	94.64	97.06
0.3	152.80	171.42	190.12	205.66
0.5	226.16	253.06	287.64	314.26
0.7	299.52	334.54	384.70	422.86
1.0	409.56	456.70	536.54	585.76

upper bound on the solution, and provide more insight on the performance of our heuristic. In presenting the results of this section, we refer to our original GW routing algorithm as *Heuristic0*, or HR0, and the new heuristic proposed in the previous section as *Heuristic1*, or HR1.

For each result, we compute the objective function value of the QIP and divide by n to obtain the average end-to-end delay of a path in number of time slots. Each data point in the table or figures of this section represents a value averaged over 20 random networks.

For larger networks, we compare the solutions returned by the lower bound, HR0 and HR1. In Figure 14, the end-to-end delay figures for each heuristic and the lower bound are shown for different network sizes. The transmission probability p is fixed at 0.1. It is clear from the figure that HR1, the heuristic taking congestion into account, performs much better than HR0, the heuristic based on pure GreenWave routing, hence the gap between the lower and upper bounds for the optimal solution is significantly reduced. Figure 15 demonstrates the trade-off between congestion minimization and routing over *fast* paths, when p is very small. We observe

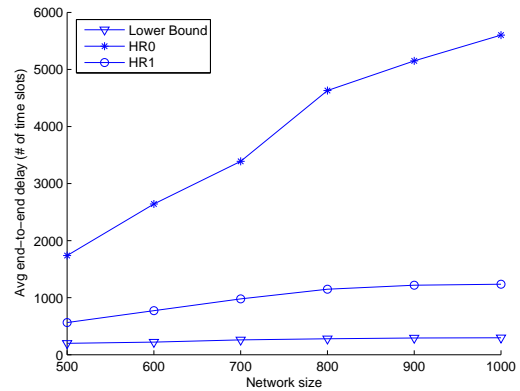


Fig. 14. Comparison of the solutions by the heuristic and the Green-Wave algorithm to the lower bound on larger networks. Transmission radius r and the probability p are both fixed at 0.1 for all network sizes.

that pure GreenWave routing provides lower end-to-end delays when the traffic load is below some threshold. In such a case, the system behaves similar to a network with in-network data fusion since traffic load is very low and there is little or no queueing at the relaying nodes. The new heuristic HR1 works on a reduced graph, hence it may fail to find some green-wave paths and the gain due to congestion elimination can not make up for it when p is very small. However, HR1 becomes increasingly more preferable as p increases, as also demonstrated in Figure 14.

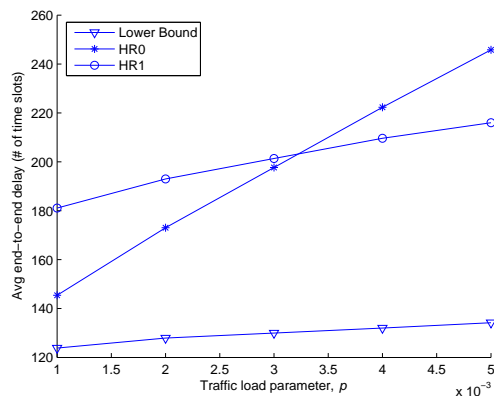


Fig. 15. Comparison of the heuristics and the lower bound for very small values of p , for a fixed network size of 500 nodes. Under very light traffic conditions, the pure GreenWave routing (HR0) is preferable similar to the data-fusion case. Since HR1 works on a reduced graph, it fails to find some green-wave paths, and the gain due to congestion elimination can not make up for it when p is very small.

TABLE III
A SMALL SAMPLE OF ROUTING TABLE AT NODE u

Node ID	Distance	Backlog
v	$d_v + w(u, v)$	1
x	$d_x + w(u, x)$	1
y	$d_y + w(u, y)$	0

E. GreenWave Routing with Congestion and Flow Control (GWCF)

In this section, we propose a distributed and dynamic routing protocol for the no-fusion case, based on the insights from our heuristic algorithm in the previous section. *GreenWave Routing with Congestion and Flow Control* uses the packet-switching technique rather than building end-to-end circuits.

TDMA-based channel access protocols impose a bounded service level at relaying nodes determined by the frame size. Since each node is allowed to transmit once in every Λ consecutive time slots, a higher incoming data rate would cause its buffers to fill up. Hence it is desirable to have a blocking or flow control mechanism to keep the traffic load in the network at a sustainable level. The simplest solution of dropping packets at the relaying nodes with a buffer overflow is not acceptable for most sensor network applications as it would yield unreliable sensor readings with many missing data, especially for regions remote from the data sinks.

We first propose a novel congestion and flow control technique which is at the heart of our GWCF routing protocol, and can be employed by any routing protocol residing above an contention-free MAC layer.

Implicit Flow Control (IFC) - The collision-free transmission of messages in TDMA-based MAC protocols may be used for the extra benefit of inferring the backlog size at a neighbor node at no additional cost. Suppose that node u transmits a packet pk_u in its time slot σ_u to be relayed by its neighbor v . If node v has no backlogged packet prior to the injection of the packet from u , then v relays pk_u in slot σ_v ; and overhearing this transmission, u can acknowledge that its packet has been relayed and also infer that v had a backlog size of zero at the reception time of pk_u . If on the other hand, u overhears the transmission of k other packets from v before v relays pk_u , then u infers a backlog of size k at node v . IFC regulates the injection and movement of data traffic in the network by requiring that any node which has transmitted a packet to be relayed by its neighbor v does not send/relay any more packets to node v until it

overhears the relaying of its packet by v . This ensures that the backlog size of each node is bounded and also that all nodes find fair chances to transmit their data. Moreover, holding the packets back at the relaying nodes propagates through the routing trees of GWCC, causing a push-back effect and prompting the sensors to slow down the event reporting rate in congested areas. IFC is a simple yet novel implementation of the *back-pressure policy* studied in different ways in various contexts (see e.g. [13]).

In algorithm GWCF, all nodes discover their d and h values at network deployment, where d_u is the weighted shortest path distance and h_u is the minimum path length from sensor u to a sink node (just as in the heuristic algorithm in Section V-C). Then each node u builds a table which consists of a subset of its neighbors; $\mathcal{N}_u = \{v \mid h_v < h_u\}$. For each neighbor v in its table, u stores the shortest distance to a sink node through v , denoted by $d_{uv} = d_v + w(u, v)$. Building these tables locally at each node is analogous to a distributed construction of the directed acyclic subgraph in our heuristic algorithm.

Also associated with node $v \in \mathcal{N}_u$ is a bit value indicating whether v has a backlogged packet received from u . Let B_{uv} be a binary variable such that

$$B_{uv} = \begin{cases} 1 & \text{if } v \text{ has a backlogged packet received from } u \\ 0 & \text{otherwise} \end{cases}$$

Each node locally builds its table and initializes all backlog values to 0. When a packet arrives to the routing layer of a node - either locally from the higher layer or as a routing request from a neighbor node - u chooses as the next hop the node $v \in \mathcal{N}_u$ which has the minimum d_{uv} value and $B_{uv} = 0$. If such a node exists, then u transmits/relays the data to v , and sets B_{uv} to 1. Otherwise, u refrains from sending a packet in the current frame, and the packet is hold in the queue. On the other hand, for each $v \in \mathcal{N}_u$ such that $B_{uv} = 1$, u listens the channel in v 's time slot. If u overhears the transmission of a packet sent earlier from u to v , then u resets B_{uv} to 0.

Fig. 16 demonstrates the execution of GWCF at a node, u , that has two nodes, x and y , in its routing table. The time slot allocation for these three nodes is shown at the top of the figure. It is assumed that the shortest-path distances of nodes x and y are both 5 slots; hence, we have $d_{ux} = 5 + 3 = 8$ and $d_{uy} = 5 + 6 = 11$. The figure shows the routing table and the routing decision made at u in three consecutive time frames. Also shown in the figure are the transmission queues for both x and y , each queue entry being the node id of a packet source.

It is assumed that u has one packet to send at all three slots, t , $t + \Lambda$, and $t + 2\Lambda$, while x and y neither generate data nor receive packets from nodes other than u during these three frames.

Initially (at time t), the backlog indicators at node u are both set to 0. Thus, the packet at u is relayed to x since d_{ux} is the minimum among the two nodes. A red (or dashed) line in the figure indicates a data transmission. At time $t + \Lambda$, the current backlog values at u and the updated transmission queues at x and y are given in the second row of Fig. 16. Since u did not yet receive the implicit notification from x , B_{ux} remains 1 and the second packet at u is forwarded to y at time $t + \Lambda$. For the sake of illustration, it is assumed in the second time frame that x has $B_{xv} = 1$ for all v in its routing table; hence, x is blocked in its time slot in this frame (at $t + \Lambda + \sigma_x$). This is indicated in the figure by u still residing in x 's queue at $t + 2\Lambda$. Therefore, u has now all backlog indicators set to 1 in its table, and is blocked from accessing the channel in the third frame, demonstrating the back-pressure caused by x on u .

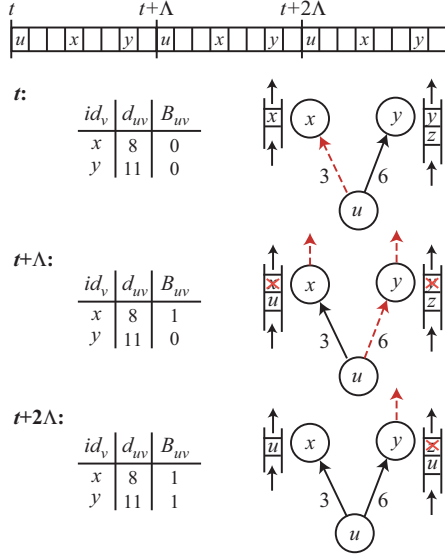


Fig. 16. Demonstration of GWCF by a simple example. Node u 's routing table and the corresponding routing decision (red or dashed line) are given for three consecutive time frames. Transmission queues of the two relay nodes are also shown, where each entry is the node id of a packet source.

Recall that the collection of local routing tables in GWCF is analogous to a distributed version of the directed acyclic graph G' in the heuristic algorithm. Then, a node may only receive packets from its *direct predecessors* in G' . Moreover, *IFC* imposes that each

one of them may have at most one packet waiting at any neighbor's queue. Therefore, the transmission queue size at each node is bounded by the number of its direct predecessors in the auxiliary directed graph.

With GWCF protocol, node u needs to keep its receiver on only in the time slots owned by the nodes in \mathcal{N}_u , and the slots owned by the nodes in $\{v \mid u \in \mathcal{N}_v\}$. The node may turn off its radio receiver in the remaining slots for energy savings.

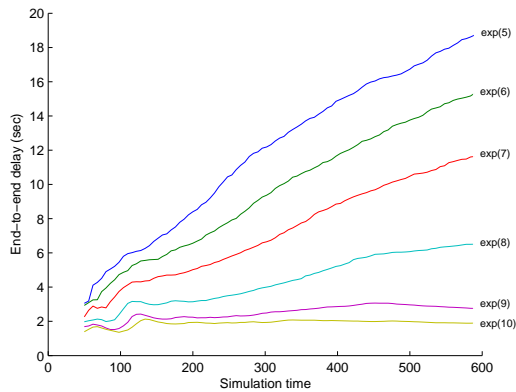
GWCF can also be made robust and reliable easily by the use of IFC technique. If a message of u is not acknowledged by a relaying node v (implicitly by IFC), then either the packet is lost or node v has failed. As a remedy to the first case, u transmits the packet after a timeout or when u overhears the transmission of two packets from v with the same source. In the second case, u does not overhear any transmissions from v (after a succession of retransmissions), and it marks v as failed in its table until it overhears a message from v .

F. GWCF Simulation Results

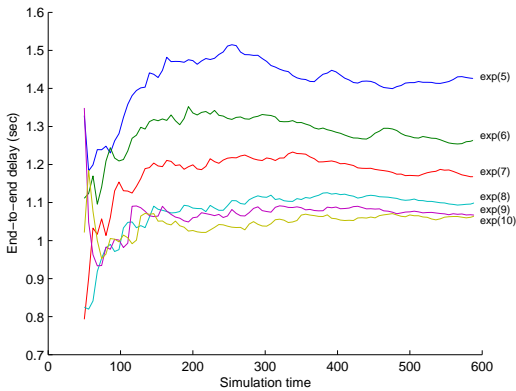
We implement the algorithm GWCF in OPNET Modeler, using the same settings as in Section IV-A unless otherwise stated. We study two different application scenarios; (i) a set of event sources are placed uniformly at random in the network field, each of which independently generate events according to a given traffic distribution, (ii) each sensor node independently generates a sensing event according to a common distribution. In this section, our primary aim is to present the improvements provided by the congestion and flow control aspects of GWCF over the basic GW routing (hence also over shortest-path routing) when data is not fused along routing paths. Since the results follow similar trends for both application scenarios, here we present only the results for the case with independent traffic generation at sensor nodes. Also, in the results of this section, we consider exponentially distributed interarrival times for packet generation at the sensor nodes, though other distributions such as constant traffic produce results with a similar trend.

The results for the shortest hop (SH) routing are very close to those of the basic GW routing since GW routing provides improvement only on the *static delay* component and the delay due to congestion costs are usually much more dominant in the no-fusion case unless the traffic load is very low. Hence we present only GW routing in the plots for comparisons, also representing the SH routing algorithm.

Figures 17 through 19 together demonstrate the improvements by GWCF on both end-to-end delay and network throughput. Note that one can simply reduce the end-to-end delays by reducing the traffic load (by filtering out data at the source or dropping packets in the network), however this generally results in reduced throughput. We observe that GWCF routing reduces and bounds the end-to-end delay while enabling more traffic to be pushed from sensors to the sinks. This is due to the congestion-aware routing scheme that utilizes the underlying STDMA MAC protocol.



(a) GW / SH



(b) GWCF

Fig. 17. The average end-to-end delay values as the simulation progresses for (a) GW / SH (b) GWCF routing on a network of 500 sensor nodes and 3 sink nodes. The mean packet interarrival time is varied from 5 sec to 10 sec for each sensor node. With pure GW routing, the end-to-end delay grows unboundedly except for low traffic conditions. On the other hand, GWCF keeps the end-to-end delay bounded at much lower levels, regardless of the traffic load. Note the difference in y-axis scale of the two figures.

In Figure 17, the average end-to-end delay is shown

for both GW and GWCF routing as the simulation progresses. The mean packet interarrival time is varied from 5 sec. to 10 sec. so that we could observe the condition when GW routing starts to operate at a sustainable level by keeping end-to-end delays bounded (Figure 17(a)). On the other hand, we observe in Figure 17(b) that the end-to-end delays with GWCF routing always remain bounded regardless of the traffic load. Please note the difference in the y-axis scale of Figures 17(a) and 17(b).

Figure 18 presents end-to-end delay results for higher data arrival rates. The packet interarrival times are exponentially distributed with a mean varied from 0.1 sec to 2.5 sec. Besides the average end-to-end delay value, we mark the minimum and maximum delay values among sink nodes by vertical lines. We observe that there may be large differences in the end-to-end delay values observed at different sink nodes with pure shortest-path based routing, whereas GWCF reduces this gap by balancing the load among the paths leading to each sink node.

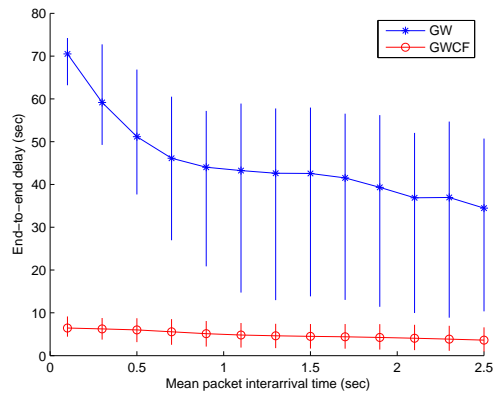


Fig. 18. The average end-to-end delay values for GW and GWCF routing, with higher data arrival rates. The mean packet interarrival time is varied from 0.1 sec to 2.5 sec for each sensor node. The end points of the vertical lines represent the minimum and maximum values among the 3 sink nodes, where the main plot is an average over all sink nodes.

Figure 19 plots the total amount of traffic received at the sink nodes for both protocols, and demonstrates that the gain in end-to-end delays does not come as a result of lower throughput in GWCF routing. Though the use of flow control by the IFC technique adjusts the amount of data in the network to a reduced level, the congestion control allows more data to be carried within the network by distributing the load over non-optimal routes.

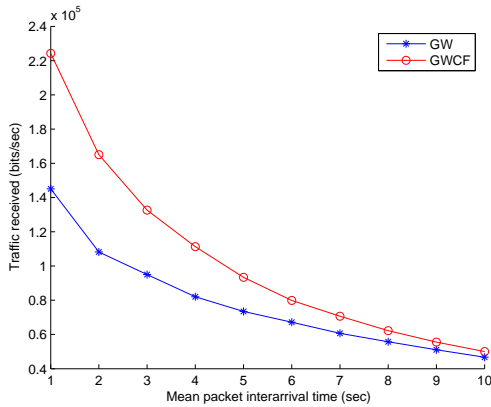


Fig. 19. Total amount of traffic received at the sink nodes per second as a function of the traffic load at the sensor nodes for both GW and GWCF routing on a network of 500 sensor nodes and 3 sink nodes.

VI. CONCLUSION

In this work, we claim that contention-free or more generally reservation-based MAC protocols should be coupled with the routing protocol in multi-hop wireless networks for significant performance improvements and energy savings. Hence we presented efficient routing schemes for wireless sensor and mesh networks specifically designed to work with STDMA MAC protocols. Two cases were considered; routing with in-network data fusion and without data fusion. The case with fusion makes the problem easier since there is no queueing delay, hence the intersection of paths (congestion) does not affect the end-to-end delays. We presented *GreenWave routing*, a distributed and scalable protocol that forms end-to-end routes from sensors to sinks, based on an underlying TDMA-based MAC protocol. By computational results on random networks, we showed that GreenWave routing significantly decreases the communication delay, and thus increases the throughput of the network under saturation conditions. Moreover, we presented an important result that a contention-free STDMA protocol may outperform the contention-based 802.11 protocol, when accompanied by our GreenWave routing algorithms. Recall that the STDMA protocols have already the desirable properties for an energy-constrained sensor network, since there is no energy waste due to collisions and retransmissions. Hence having their delay and throughput performance even just as comparable to the 802.11 protocol makes them much more attractive for use in sensor networks.

On the other hand, if in-network data fusion is not

available, then the problem becomes harder since we should take into account the interaction among the flows from different sources. We formulated this problem as a QIP, which is expressed as a combined minimization of the *static costs*, as defined for the fusion case, and the *congestion costs*, which depends on the interaction among different flows (routes). Since the QIP can not be efficiently solved for large networks, we provided a lower bound and a new heuristic algorithm to bound the optimal solution from above and below. With the insights from the QIP and its heuristic solution, we proposed a distributed routing protocol which controls the flow of data from sensors to sinks to keep buffers at a limited level and provide low end-to-end delays for routed traffic while increasing the network throughput.

REFERENCES

- [1] R. Bruno, M. Conti, and E. Gregori, *Mesh Networks: Commodity Multihop Ad Hoc Networks*, IEEE Communications Magazine, Vol. 43, No. 3, March 2005.
- [2] W. Allen, A. Martin, A. Rangarajan, *Designing and Deploying a Rural Ad-Hoc Community Mesh Network Testbed*, The IEEE Conference on Local Computer Networks, November 2005. Page(s) 740743.
- [3] I. F. Akyildiz and X. Wang, *A Survey on Wireless Mesh Networks*, IEEE Communications Magazine, vol. 43, no. 9, s23-s30, September 2005.
- [4] <http://pdos.csail.mit.edu/roofnet/>
- [5] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE standards 802.11, January 1997.
- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms, Second Edition*, MIT Press, September, 2001, pp. 588–592.
- [7] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, *A Survey on Sensor Networks*, IEEE Communications Magazine, August 2002.
- [8] Y. Wang, J. J. Garcia-Luna-Aceves, *Modeling of Collision Avoidance Protocols in Single-Channel Multihop Wireless Networks*, Wireless Networks, Volume 10, Issue 5, pp. 495–506, September 2004.
- [9] F. Cali, M. Conti, E. Gregori, *Dynamic Tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit*, IEEE/ACM Transactions on Networking, Vol. 8, No. 6, pp. 785–799, December 2000.
- [10] H. Takagi and L. Kleinrock, *Optimal Transmission Range for Randomly Distributed Packet Radio Terminals*, IEEE Transactions on Communications Vol. 32 No. 3, pp. 246-257, March 1984.
- [11] L. Wu and P. Varshney, *Performance Analysis of CSMA and BTMA Protocols in Multihop Networks: (i). Single Channel Case*, Information Sciences, Vol. 120 pp. 159177, 1999.
- [12] CPLEX Mathematical Programming Optimization Software, ILOG Inc, <http://www.ilog.com/products/cplex>.
- [13] L. Tassiulas, *Adaptive back-pressure congestion control based on local information*, IEEE Transactions on Automatic Control, Vol. 40, No. 2, pp. 236250, February 1995.

- [14] W. Ye, J. Heidemann, D. Estrin, *Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks*, IEEE/ACM Transactions on Networking 12, 3 (Jun. 2004), 493-506.
- [15] T. Van Dam, K. Langendoen, *An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks*, The First ACM Conference on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, CA, November 2003.
- [16] J. Polastre, J. Hill, D. Culler, *Versatile Low Power Media Access for Wireless Sensor Networks*, SenSys'04, Baltimore, Maryland, November 35, 2004.
- [17] *MICA2: Wireless Measurement System*, Crossbow Inc., <http://www.xbow.com/>
- [18] I. Rhee, A. Warrier, M. Aia, and J. Min, *ZMAC: a Hybrid MAC for Wireless Sensor Networks*, SenSys'05, San Diego, California, November 24, 2005.
- [19] C. Busch, M. Magdon-Ismail, F. Sivrikaya and B. Yener, *Contention-Free MAC protocols for Wireless Sensor Networks*, In proceedings of the 18th International Conference on Distributed Computing (DISC 2004), Amsterdam, The Netherlands, pp. 245–259, October 2004.
- [20] V. Rajendran, K. Obraczka, J. J. Garcia-Luna-Aceves, *Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks*, SenSys'03, Los Angeles, California, November 57, 2003.
- [21] L. Bao, J. J. Garcia-Luna-Aceves, *A New Approach to Channel Access Scheduling for Ad Hoc Networks*, In Proc. ACM Mobicom, Rome, Italy, July 2001.
- [22] M. Ringwald, K. Römer, *BitMAC: A Deterministic, Collision-Free, and Robust MAC Protocol for Sensor Networks*, 2nd European Workshop on Wireless Sensor Networks (EWSN 2005), Istanbul, Turkey, January 2005.
- [23] T. Herman and S. Tixeuil, *A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks*, ALGOSENSORS 2004, pp. 45–58.
- [24] R. Kalidindi, L. Ray, R. Kannan, S. Iyengar, *Distributed Energy Aware MAC Layer Protocol For Wireless Sensor Networks*, in Proc. International Conference on Wireless Networks, pp. 282-286, 2003.
- [25] K. A. Arisha, M. A. Youssef, and M. F. Younis, *Energy-Aware TDMA-Based MAC for Sensor Networks*, IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT 2002), New York City, New York, May 2002.
- [26] K. Sohrabi, G. Pottie, *Performance Of A Novel Self-Organization Protocol For Wireless Ad-Hoc Sensor Networks*, IEEE Vehicular Technology Conference, Amsterdam, Netherlands, September 1999.
- [27] M. Arumugam, S. S. Kulkarni, *Self-stabilizing Deterministic TDMA for Sensor Networks*, Proceedings of the second International Conference on Distributed Computing and Internet Technology (ICDCIT), LNCS 3816, pp. 69-81, Bhubaneswar, India, December 22-24, 2005
- [28] I. Rhee, A. Warrier, J. Min, and L. Xu, *DRAND:: Distributed Randomized TDMA Scheduling for Wireless Ad-hoc Networks*, In Proceedings of the Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (Florence, Italy, May 22 - 25, 2006), MobiHoc '06. ACM Press, New York, NY, 190-201.
- [29] J. Grönkvist, *Novel Assignment Strategies for Spatial Reuse TDMA in Wireless Ad hoc Networks*, Wireless Networks, 12, 2 (Mar. 2006), 255-265.
- [30] C. S. Li, Y. Ofek, and M. Yung, *"Time-driven priority" flow control for real-time heterogeneous internetworking*, in Proc. IEEE INFOCOM96, 1996, pp. 189197.
- [31] Y. Yemini, and D. Florissi, *Isochronets: a high-speed network switching architecture*, in Proceedings of INFOCOM, IEEE, San Francisco, California, USA, April 1993.