

# Energy-Efficient Communication in Multi-Channel Single-Hop Sensor Networks

Amol Bakshi and Viktor K. Prasanna  
Department of Electrical Engineering-Systems  
University of Southern California  
Los Angeles, CA 90089 USA  
{amol,prasanna}@usc.edu

## Abstract

*Sensor networks are usually designed and optimized for a specific functionality, and the nature of tasks on different nodes and the pattern of communication between nodes is known at design time. We are investigating a design approach for sensor networks that involves the creation of robust and energy-efficient protocols for a library of general communication primitives. Such a library will hide low-level networking details and provide building blocks for the end user who can then focus on high level design of the application. In this paper, we consider single hop sensor networks with multiple data channels, and define and evaluate the performance of a simple and energy-efficient protocol for a general and powerful primitive called  $(N, p, k_1, k_2)$  routing. This primitive denotes the transfer of  $N$  packets among  $p$  nodes with each node transmitting at most  $k_1$  packets and each node expecting to receive at most  $k_2$  packets. One-to-all (broadcast), all-to-one (gather), many-to-many, etc., are special cases of  $(N, p, k_1, k_2)$  routing. We use a separate low-power control channel for coordination among nodes. Simulation results show that our protocol leads to very high utilization of the available data bandwidth, and is also energy balanced. Fault-tolerant extensions of the protocol are also discussed.*

## 1. Introduction

Wireless networks of smart sensors have the potential to revolutionize data collection and analysis for a host of applications. Each smart sensor or ‘sensor node’ is composed of one or more environment sensors, a processing unit, storage, limited power supply, and a transceiver. Most wireless sensor networks (WSNs) are designed and optimized to understand and/or affect physical processes, in contrast to general purpose, application-independent data transport, i.e., they are designed for a specific application such as target tracking, habitat monitoring, etc. The main challenge

facing the designers is to customize the in-network computation and communication for maximum energy efficiency, which prolongs the lifetime of the network as a whole.

Embedded networked sensing is an area of active research in the community. Sensor networks are being modeled as parallel and distributed systems, databases, wireless ad-hoc networks, etc. Significant advances are being made in the hardware architectures of sensor nodes. A variety of different communication paradigms have also been formulated for specific routing patterns to support in-network query processing, data-centric routing and storage, system-wide network monitoring, etc [4, 7, 5]. Most of this work integrates application-specific optimizations into the network protocols. In the state-of-the-art, partitioning of computation tasks among nodes, node-level optimizations, and data routing is done in an ad-hoc, largely empirical manner. While this approach does not necessarily result in inferior designs, it does require the application developer to be aware of the details of the underlying node hardware, networking paradigms, and their performance.

We are exploring a new approach to application design for sensor networks, where the end user is presented with a library of pre-designed, parameterized implementations for computation and structured communication primitives, that hide most of the details about medium access, time synchronization, fault tolerance, etc. ‘Structured communication’ refers to a routing problem where the communication pattern is known in advance - for example, one-to-all, all-to-one, many-to-many, all-to-all, permutation, etc. The abstraction provided by the library of communication primitives will allow the user to focus on broader design issues, instead of hardware and networking details. Unlike the Internet, where point-to-point communication was the basic primitive, the collaborative in-network processing in sensor network applications (and its knowledge at design time) enables such a modular, composable approach.

Our earlier work has focused on design of routing primitives for multi-hop [1] and single-hop [2] single-channel sensor networks. *The primary contribution of this pa-*

per is a simple, robust and energy-efficient protocol for  $(N, p, k_1, k_2)$  routing in single-hop network topologies with multiple data channels.  $N$  denotes the number of packets to be transferred,  $p$  is the number of nodes,  $k_1$  is the maximum number of packets that can be sent by a node, and  $k_2$  is the maximum number of packets that a node can expect to receive from others.  $(N, p, k_1, k_2)$  routing is a very general and powerful primitive whose special cases include commonly encountered communication patterns - scatter, gather, all-to-all, etc. While this algorithm is applicable to ad-hoc and radio networks in general, the importance of energy balance in addition to total energy as a performance metric, and the role of application-specific communication protocols in the design of sensor network applications make this result especially suitable for WSNs.

In our protocol, packet transmissions are scheduled dynamically over the control channel with the goal of maximizing bandwidth utilization. Each packet transfer is preceded by a negotiation phase that makes the protocol robust because a failure of either the sender or receiver to respond as expected leads to cancelation of the packet transfer and the energy savings thereof. Also, all packets are transferred directly between the source and destination without intermediaries. By doing so, the impact of a node failure is limited only to those packets that are destined for or scheduled to be transmitted from that node. Typical *single-hop* WSNs will likely consist of a few tens or at most a few hundreds of nodes. Larger networks of thousands or tens of thousands of nodes will almost never have single-hop connectivity for various reasons. Simulation results show that for a  $k$ -channel network with a few tens of nodes, our protocol yields  $k$ -times (latency) speedup compared to the performance of the single channel case. Also, for the case  $k_1 = k_2$  (i.e., each node transmits and receives the same number of packets), the protocol is energy balanced - i.e., each node spends almost the same amount of energy.

Our system model is outlined in Section 2. Section 3 describes our  $(N, p, k_1, k_2)$  routing protocol for fault-free multi-channel networks. Section 4 discusses fault-tolerant extensions of the protocol. Section 5 discusses related work, and we conclude in Section 6.

## 2. Our System Model

The network consists of  $p$  nodes labeled  $S(1), S(2), \dots, S(p)$ . Each node has a unique identifier (ID) in the range  $[1, p]$ . The network is *homogeneous*, i.e. all nodes in the network are identical in terms of computing and communication capability. All nodes are initially time-synchronized with each other and remain time-synchronized for the duration of the routing. There are two communication channels: a data channel for trans-

fer of data packets and a low power control channel for coordination purposes. Every node is correspondingly equipped with a data radio and a control radio. At a given time, the control radio can be tuned to only one frequency, but the data radio can be tuned to one of  $f_d$  frequencies. In other words, there is one control channel and  $f_d$  data channels available in the network. All  $f_d + 1$  frequencies are distinct. Regardless of which frequency it is tuned to, a data or control radio can be in only one of three states at a given time: Transmit, Receive, or ShutDown. Energy consumption per packet is same for the Transmit and Receive states; and zero (negligible) for ShutDown. Instantaneous signaling is possible between the two radios on the same node. For example, the data radio can signal the control radio on successful receipt of a data packet. The control radio can likewise signal the data radio to shift the data radio from one state to another. The network has single-hop connectivity. If a data (control) radio in Receive state receives simultaneous transmissions from two or more data (control) radios in the Transmit state, a collision occurs and neither transmission succeeds.

**Costs:** Transmitting one packet over the data channel takes  $t_d$  time and  $e_d$  energy. Similarly, a packet transmission over the control channel requires  $t_c$  time and  $e_c$  energy. Packets sent over the control channel for coordination purposes are smaller than data packets.  $t_d/t_c = r_1$  and  $e_d/e_c = r_2$ , where  $r_1$  and  $r_2$  are positive integers. Based on state-of-the-art hardware and projections for the near future, it is reasonable to assume  $r_1$  to be in the range of 5-10, and  $r_2$  to be at least a few tens. We ignore energy and latency costs of computation.

## 3. $(N, p, k_1, k_2)$ routing in a fault-free network

### 3.1. Characteristics of Our Protocols

Our protocol for a fault-free single-hop network with one control channel and multiple data channels has the following properties: (i) there are never any collisions between control packets over the control channel, (ii) there are never any collisions between data packets over the data channel, (iii) data radios are ShutDown at all times except when a transmission or reception is in progress, (iv) a data packet is always transmitted directly from the source to the destination, without going through intermediate nodes, and (v) all coordination is decentralized (peer-to-peer).

A Request to Send (RTS) message, followed by the corresponding Clear to Send (CTS) constitutes one transaction over the control channel. Similarly, the transfer of one data packet from the source to the destination constitutes one transaction over the data channel. Over the control channel, time is divided into fixed-size control frames (CF) which

is the time required for one control transaction. Similarly, data frames (DFs) are defined as the time required to execute one data transaction.

### 3.2. Protocol Definition

**3.2.1. Stage I: Pre-processing** In  $(N, p, k_1, k_2)$  routing,  $k_1$  denotes the maximum number of packets to be transmitted by a node, and  $k_2$  denotes the maximum number of packets that each node expects to receive. For node  $S(i)$ , let  $k_1^i$  and  $k_2^i$  denote the exact number of packets to be transmitted and received. For simplicity of exposition, we assume in the remainder of this paper that each of the  $p$  nodes transmits exactly  $N/p$  packets and expects to receive exactly  $N/p$  packets, i.e.  $k_1^i = k_2^i = \frac{N}{p}, \forall i$ . In case  $k_1^i$  is different for each  $S(i)$ , a pre-processing phase is carried out over the control channel prior to the packet transfer. This phase consists of exactly  $p$  CFs. In the  $i^{\text{th}}$  CF,  $S(i)$  broadcasts the value of  $k_1^i$ . All  $p$  nodes remain awake for the  $p$  CFs. At the end of this phase, each node knows exactly how many packets every node has to transmit.

If each  $S(i)$  also knows how many packets are destined for itself, i.e., the value of  $k_2^i$ , it can shut off completely once it has received its  $k_2^i$  packets (and transmitted its  $k_1^i$  packets as described in Stage 2 below). Now, a node knows the destinations of the packets it has to transmit, but does not know how many packets are destined for itself, and no means of computing this number except by explicit communication with every other node. For cases where  $k_2^i$  is not the same for all  $S(i)$ , this information can be exchanged over the control channel in a series of  $p$  rounds, each round consisting of  $p$  CFs. In the  $j^{\text{th}}$  frame of the  $i^{\text{th}}$  round,  $S(j)$  broadcasts the number of packets in its buffer destined for  $S(i)$ . This requires a total of  $p^2$  slots, with each node remaining awake for  $2p-2$  slots. If  $N \gg p$ , and  $r_1$  and  $r_2$  are of the order of tens or a hundred, the overhead for the pre-processing phase is a small fraction of the total energy requirements for the actual routing process.

#### 3.2.2. Stage II: Packet Transfer

**Case I: One Control Channel, One Data Channel** Let the CFs be numbered in increasing order, with the first CF being  $CF[1]$ . Similarly, let the DFs be labeled such that a DF that immediately succeeds  $CF[i]$  is labeled  $DF[i]$ .

$CF[i]$  is said to be *owned* by  $S(i)$  if no other node is supposed to send an *RTS* in  $CF[i]$ .  $CF[i]$  is said to be *successful* if  $S(i)$  sends an *RTS* addressed to some  $S(j)$  and  $S(j)$  sends a *CTS* in acknowledgment.  $DF[i]$  is said to be *owned* by nodes  $S(i)$  and  $S(j)$  if  $CF[i]$  is successful.

Data is transferred in  $DF[i]$  if and only if  $CF[i]$  is successful. Node  $S(i)$  owns  $CF[\frac{N(i-1)}{p} + 1]$  through  $CF[\frac{N \cdot i}{p}]$ . Every node  $S(j)$  maintains a counter  $c_j$  initialized to 0 that keeps track of how many packets have been received. One

$S(i)$  owns the CF, say  $CF[x]$  and broadcasts *RTS*s. All non-owner nodes which have not yet received their  $N/p$  packets ( $c_j < N/p$ ) monitor the control channel in the Receive mode during this CF.  $S(i)$  switches its data radio to Transmit mode for  $DF[x]$  and the destination  $S(j)$  switches its data radio to the Receive mode for  $DF[x]$ .  $S(i)$  transmits the packet to  $S(j)$  in  $DF[x]$ . Data radios of all  $S(k)$ , where  $k \neq i$  and  $k \neq j$ , stay in ShutDown mode in  $DF[x]$ .

This basic protocol is robust because all packets are transferred directly between the source and destination and there is no single point of failure. The fact that packets are not routed through intermediate nodes means that a single node failure affects only the packet transmissions destined for that node and the packets that were supposed to be transmitted by that node. *Failures are therefore isolated and their effect on unrelated packet transmissions is minimized.*

#### Case II: One Control Channel, Multiple Data Channels

For the single channel case, the first  $N/p$  CFs were owned by  $S(1)$ , the next  $N/p$  by  $S(2)$  and so on. In the ideal case (no faults), the routing finished in exactly  $N$  CF's, each followed by a DF. The latency was not affected by the distribution of the  $N$  packets among the  $p$  nodes. In the multi-channel case, however, the distribution of packets has a significant impact on the utilization of the data bandwidth and the total latency of the routing. For example, let all the  $N/p$  packets at  $S(i)$  ( $\forall i$ ) be destined for  $S(i+1) \pmod{p}$ <sup>1</sup>. In the very first CF,  $S(1)$  sends an *RTS* to  $S(2)$ , which replies with a *CTS*. All  $f_d$  data channels are available initially, because no data transfer has started.  $S(1)$  transmits a data packet over one of the available channels to  $S(2)$ . Now, the data radios on  $S(2)$  is in the Receive state and is busy till the transfer is complete. All packets on  $S(1)$  are destined for  $S(2)$ , so  $S(1)$  has nothing to transmit over the other  $f_d - 1$  channels. If the first  $N/p$  CFs were to be assigned to  $S(1)$  as in the single-channel case, the protocol will perform the same as the single-channel protocol, and  $f_d - 1$  channels will remain unutilized. A more dynamic coordination is required over the control channel if the available data bandwidth is to be exploited to a greater extent. One such coordination mechanism is presented as part of the protocol below.

In addition to the *RTS* and *CTS* packets used for data transfer, we use a *PASS* control packet. A *PASS* packet is different from *RTS/CTS* in that it uses broadcast addressing, i.e., it is not addressed to a specific destination node.

Each node maintains the following information which represents the *state* of the network:

1. A *channel allocation vector* (CAV) that consists of  $f_d$  tuples, where tuple  $i$  corresponds to data channel  $i$  ( $1 \leq i \leq f_d$ ). Each tuple records the sender, destination, and end time

<sup>1</sup>  $(\pmod{p})$  is valid if nodes are numbered from 0 through  $p - 1$  and not from 1 through  $p$ . For conciseness of representation, we make the former assumption here.

of a data packet transmission (if any) over the corresponding data channel. Since all coordination is performed over the single control channel, and all nodes are monitoring the control channel, each node has complete information over the data transfers in progress over every data channel.

2. A *packet count vector* (PCV) that consists of  $p$  tuples, where tuple  $i$  corresponds to node  $S(i)$ . Each tuple  $i$  contains a record of the number of packet transmitted by  $S(i)$  and the number of packets received by  $S(i)$  till that time. Again, since each node monitors the control channel and all data transfers are explicitly flagged by an RTS/CTS exchange, each node has the information needed to maintain the packet count vector.

Since the size of the CAV and PCV data structures is proportional to  $f_d$  and  $p$  respective, and both these constants are relatively small (unlike  $N$  which can be large), the storage requirements at each node are not excessive.

Initially, all data channels are idle and all data radios are in ShutDown mode. In our protocol, the queue of outgoing packets present at each node is sorted in increasing order of destination ID prior to the start of the protocol. At the start of the protocol,  $S(1)$  owns  $CF[1]$ .

As in the earlier protocol, as soon as a node has finished transmitting  $N/p$  packets and has received  $N/p$  packets, it withdraws from the protocol and remains ShutDown. Since each node maintains a PCV, no special control packet is required to signal this. The following therefore applies only to nodes that are still participating in the protocol.

At each CF, the owner  $S(i)$  inspects its CAV. If all channels are busy,  $S(i)$  waits till at least one channel becomes available.  $S(i)$  then traverses its outgoing (sorted) packet queue to determine if there exists at least one packet whose destination  $S(j)$  is not in the process of transmitting or receiving a data packet. If such a packet is found,  $S(i)$  performs an RTS/CTS exchange with  $S(j)$  and starts a packet transfer over any of the available data channels. If no such packet is found,  $S(i)$  broadcasts a *PASS* message.

*How is the ownership of CFs transferred among nodes?* Our protocol uses a completely decentralized mechanism to select the owner of the next CF, based on a variant of the round-robin policy. The very first owner is  $S(1)$ . Let  $S(i)$  denote the owner of some CF where it has transmitted either an RTS or a *PASS* packet. The owner  $S(j)$  of the next CF is chosen by considering nodes  $S(i+1), S(i+2), \dots$  (mod  $p$ ) in this order till a node  $S(j)$  is found which (i) has not withdrawn from the protocol, and (ii) is not busy during the next CF. The information in the CAV and PCV is sufficient for each node (including  $S(j)$ ) to derive the identity of the owner of the next CF.

*Why is a PASS packet required?* Only the owner of an outgoing packet knows its destination. Therefore, a scenario might arise where the owner of a CF does not have even

one packet whose intended destination is available. In such a case, a *PASS* packet is broadcast and the owner of the next CF is selected using the decentralized mechanism described in the previous paragraph. In a fault-free network, the *PASS* packet is not strictly necessary because a lack of RTS transmission by the owner of the CF can be interpreted as a lack of packets that can be transmitted successfully. However, we chose to transmit a *PASS* packet as a liveness indicator to be used in future extensions of this protocol for a faulty network.

Fig. 1 illustrates the working of this protocol for network with two data channels. The numbers above the CFs represent the owner of the corresponding CF.  $S(1)$  owns the first CF.  $S(3)$  owns the next CF because  $S(2)$  is busy receiving a packet from  $S(1)$ . Once these two transfers are in progress, the earliest time at least one channel becomes available is when the transfer over D1 finishes. At that time,  $S(3)$  and  $S(4)$  are still busy over D2, hence  $S(5)$  becomes the owner of the next CF. Only one occurrence of a *PASS* packet transmission occurs as shown. In this case,  $S(2)$  owned the last CF, and  $S(3)$  becomes the owner of the next CF because it is the first node in the sequence  $S(3), S(4), \dots$  that is free when D1 becomes available. However, the only packet  $S(3)$  has remaining is destined for  $S(2)$ . Since  $S(2)$ 's data radio is busy,  $S(3)$  passes ownership to  $S(4)$  by broadcasting a *PASS* packet.

If the permutation is represented by a graph where the vertices represent the nodes and edge  $(i, j)$  represents a packet to be transferred from node  $i$  to node  $j$ , our protocol continually attempts to find an  $f_d$ -matching of the graph in a dynamic decentralized manner.

**3.2.3. Simulation Results** The speedup in latency achieved by our multi-channel routing protocol is illustrated in Fig. 2. We simulated a single-hop network with 50 nodes and increased the packets per node from 25 through 50 in steps of 5, i.e., the total number of packets in the network increased from 1250 (25 packets per node) through 2500 (50 packets per node). We chose to simulate upto 5 data channels in the network. A uniform random number generator was used to distribute the packets among nodes in the network subject to the constraint that no node can be initialized with more than  $N/p$  packets. All experiments were repeated 40 times and the results were averaged. As seen from the plot, our protocol yields almost  $f_d$ -times speedup for a network with  $f_d$  data channels. The results also show that the incremental gain in speedup reduces as the number of channels is increased beyond 3 or 4.

We also measured the energy spent by nodes in coordination over the control channel. This coordination energy includes both the participation of a node in RTS/CTS exchanges, and in monitoring the channel and updating its state information when the RTS/CTS does not involve the

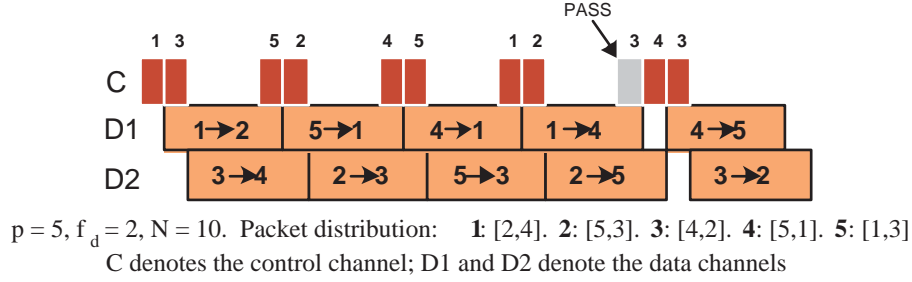


Figure 1: The Multi-Channel Protocol: Illustration

node. As shown in Fig. 3, (i) all nodes spend almost the same amount of energy for coordination, and interestingly, (ii) the coordination energy is independent of the number of data channels. Note that result (i) is for the random packet distribution we simulated, where for a large value of  $N$  ( $=2500$ ), every node in the system remains awake for a large fraction of the total number of cycles required to complete the routing. Result (ii) can be explained by the fact that for a given packet distribution and a fault-free network, there is a unique, completely ordered sequence of RTS, CTS, and PASS packet transmissions over the control channel. If the number of channels is increased, the routing is finished in fewer cycles but the number and order of control packets remains the same, thereby leading to the same energy consumption for coordination.

Fig. 4 shows the number of cycles each node remains active. Dotted lines above each plot indicate the total cycles required to complete the routing. As seen in the graph, almost all nodes remain active throughout the duration of the routing (for the random packet distribution simulated in this particular experiment). This observation can be partly explained by the round-robin nature of the transfer of CF ownership among nodes, which results in most of the nodes having pending transmissions till the very end of the routing. If the nodes have unequal number of packets to transmit and/or receive, the behavior will be different.

In Fig. 5, we plot the number of cycles by which a node receives all  $N/p$  packets destined for itself. As can be observed in the plot, in the single channel case nodes with lower ID receive their packets far earlier than nodes with higher IDs. This is the natural result of each node having sorted its outgoing packet queue and choosing the lowest numbered destination available. When the number of channels increases, this trend is not observed so strongly.

#### 4. $(N, p, k_1, k_2)$ Routing in a Faulty Network

**The Fault Model:** We define a *communication failure (fault)* as the inability of a node to both receive packets from one or more nodes in the network and transmit packets to one or more nodes in the network. A node suffer-

ing from communication failure is *unavailable* to all of the other nodes for the duration of the failure. Such a fault affects *only* the ability of a node to communicate with other nodes. A faulty node still retains its computation capabilities, and the system state (if any) maintained in the node memory also remains unaffected.

We define the *permanent fault model* as follows. Any of the  $p$  nodes of the network can have a communication failure with a certain probability  $p_{pf}$  in any time slot and will then remain unresponsive to all future requests. Failure probability is uniformly distributed across all nodes in the network. Any fault that occurs is assumed to be *non-malicious* - i.e., the failed node simply stops responding. The values of  $p_{pf}$  will depend on the particular sensor network deployment. We assume that if  $a$  sends a request to  $b$  and does not receive a response (i.e., regards  $b$  as failed), the packet addressed to  $b$  is removed from  $a$ 's buffer.

#### Protocol Definition

##### Case I: One Control Channel, One Data Channel

If the nodes can fail permanently with a non-zero probability  $p_{pf}$ , one of the following three scenarios can occur in some  $CF[x]$ .

1.  $S(i)$  sends RTS,  $S(j)$  sends CTS: This means the sender and receiver are both alive, and data is transferred in  $DF[x]$ .
2.  $S(i)$  sends RTS,  $S(j)$  does not respond: This means that  $S(j)$  has failed. Since the RTS contains the ID of  $S(j)$ , all listeners (including  $S(i)$ ) know that  $S(j)$  has failed. As the fault is assumed to be permanent, all nodes can assume that remaining transactions to  $S(j)$  can be safely canceled. However, only the node that owns a packet knows the identity of the destination. Hence, the CFs where some  $S(k)$  transmits an RTS to  $S(j)$  (knowing that  $S(j)$  will not respond) cannot be proactively eliminated because other nodes do not know exactly when such CFs will occur. The real savings in this case will occur if  $S(j)$  fails in some  $CF[x]$ , where  $x < \frac{N \cdot j}{p} + 1$ . In that case, all nodes have information about the exact  $N/p$  CFs that are owned by  $S(j)$ , and all nodes know that  $S(j)$  has failed. As soon as  $DF[N \cdot j/p]$  terminates  $S(j+1)$  can start its transmis-

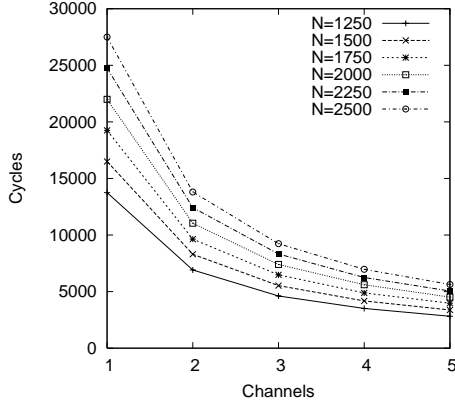


Figure 2: Speedup for Multiple Channels

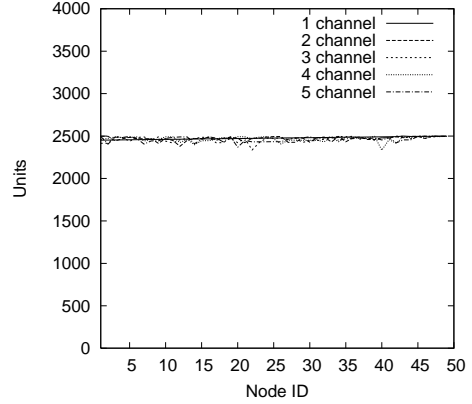


Figure 3: Distribution of Coordination Energy

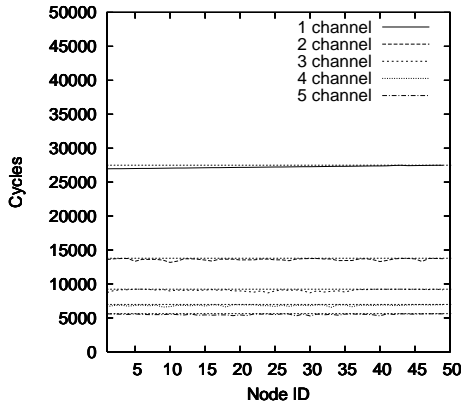


Figure 4: Node Participation (Overall)

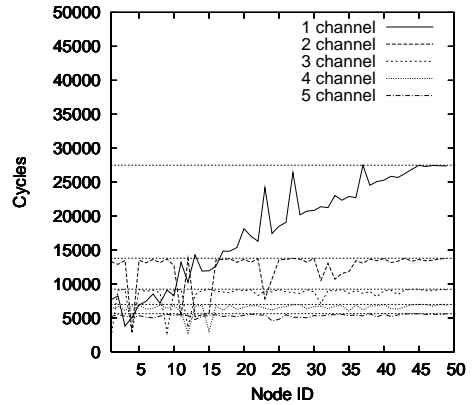


Figure 5: Node Participation (Received)

sions. This policy saves  $(t_d + t_c)N/p$  latency and at most  $2.N.(e_d + e_c)/p$  energy per permanent fault.

3.  $S(i)$  does not send *RTS*: Since the *CFs* are assigned to specific nodes based on their IDs, all listeners in  $CF[x]$  know the ID of the owner  $S(i)$  of  $CF[x]$ . A lack of *RTS* allows all listeners to collaboratively and implicitly cancel  $DF[x]$ , and also all remaining *CFs* and *DFs* owned by  $S(i)$ , proceeding directly to  $CF[\frac{N(i+1)}{p} + 1]$ . The best case is when  $S(i)$  fails in the very first *CF* owned by  $S(i)$ , leading to a cancellation of the remaining  $\frac{N}{p} - 1$  *CFs* and  $N/p$  *DFs*. The latency and energy savings are comparable to those for the previous scenario.

In the ideal case (no node failures), a node can stop participating in future *CFs* and *DFs* if it has received  $N/p$  packets and transmitted  $N/p$  packets. In case of failures, every failed *CF* create one more ‘receiver’ (in the worst case) whose count of received packets will never reach  $N/p$ , and who will therefore participate for all  $N$  *CFs*, waiting for a packet that will not be delivered. This is the primary reason for the energy overhead of the reservation phase for the permanent fault model.

#### Case II: One Control Channel, Multiple Data Channels

Our multi-channel protocol can also be easily extended

to handle permanent faults. Note that a form of implicit exclusion of nodes from future participation is already being carried out. When a node has received and transmitted its allotted number of packets, the fact is reflected in the state information maintained at each of the participating nodes, which implicitly remove that node from future consideration. This removal means that the node is no longer considered for future *CF* ownership.

The modified protocol to handle permanent faults is as follows. If a node  $i$  which has not yet received and transmitted its packets fails to transmit an *RTS* or a *PASS* packet in a *CF* of its ownership, or fails to respond to an *RTS* transmitted by in *CF* owned by another node, it is considered as failed. All listener nodes know the identity of this node because: (i) an *RTS* packet contains the identity of its (non-responding) destination, and (ii) each participant has enough information to calculate the owner of a given *CF*. When such a failed node  $i$  is identified, the participants simply flag it for exclusion from future consideration. Also, the *CF* in which the failure was detected is immediately followed by another *CF*, without any activity being initiated over the data channel.

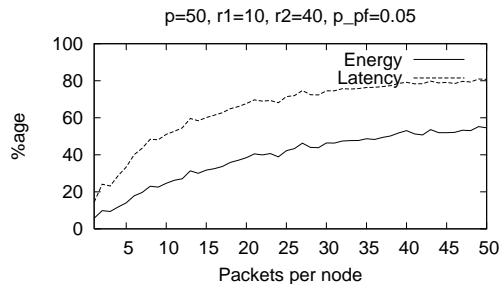
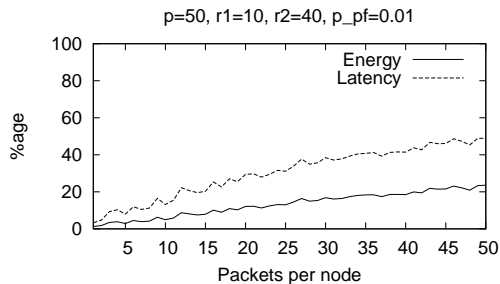


Figure 6: Latency and Energy Savings: Single Channel (from [2])

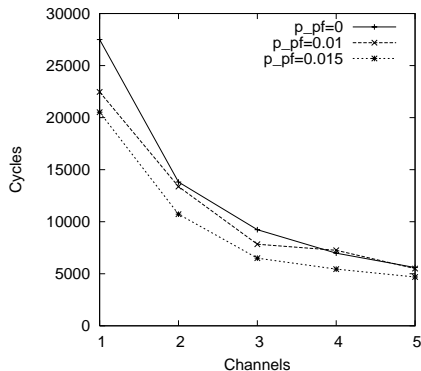


Figure 7: Effect on Routing Latency

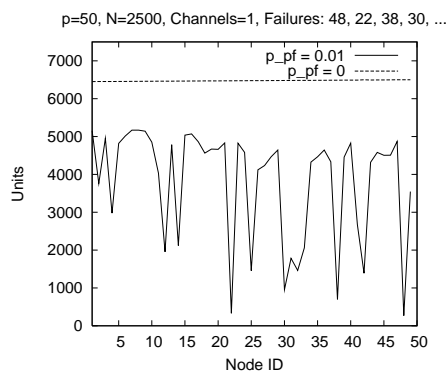


Figure 8: Distribution of Energy Consumption

## Simulation Results

The single-channel protocol was simulated for a 50-node network for two different values of  $p_{pf}$  shown in Fig. 6. For each case, we measured the energy and latency savings obtained, compared to an approach that does not dynamically compress the schedule in response to the knowledge of permanent failure of one or more nodes. As seen from the graphs, energy and latency savings become very significant as the failure probabilities increase. Note that the results in Fig. 6 were presented earlier, as indicated, in [2]. More exhaustive simulation results for energy and latency savings for both permanent faults and transient faults for single-channel single-hop WSNs appeared in [2]. These results are presented here because the latency and energy savings in the multi-channel case are similar to the single-channel case, because both the single- and multi-channel fault-tolerant protocols are based on the same principle of removing nodes from further consideration if they cease to respond to RTSes or cease to transmit an RTS/PASS packet in the CF of their ownership.

Figure 7 shows the latency savings obtained by running the modified protocol in the multi-channel case. The ideal case (no faults) is shown by the  $p_{pf} = 0$  case, which again illustrates the fact that having  $k$  data channels leads to an almost  $k$ -times speedup in terms of total latency of routing. For two different values of  $p_{pf}$ , we simulated the same network of 50 nodes and 50 packets per node. Due to detection of faults through the RTS/CTS exchange, and elimina-

tion of the DFs where no useful transmission can occur, the latency savings are obtained.

Figure 8 shows the distribution of energy dissipation across the nodes in the network, in a fault-free ( $p_{pf} = 0$ ) and a faulty network with  $p_{pf} = 0.01$ . This energy represents the overall energy spent in packet transmission as well as coordination. The graph shows that our protocol is energy balanced in the fault-free case. In the particular protocol run shown in the figure, nodes 48, 22, 28, and 30 were among the earliest to fail. The whole sequence of failures is not listed. As is reflected in the plot, these nodes have some of the lowest energy dissipation figures, since they stopped participating early in the routing. The overall energy consumption is also low because when a node fails, other nodes delete packets destined for that node from their transmit queues, and thereby save energy.

## 5. Related Work

Protocols for permutation routing in single-hop wireless networks were proposed in [6]. Their system consists of  $p$  wireless nodes with limited energy supplies. Each node is equipped with a GPS (or similar mechanism) that allows all local clocks to be perfectly synchronized with a global clock. Time is divided into fixed length slots. There is no separate control channel, and reservation as well as data transfer is done using the data channel. As a consequence,

the same time and energy is required to transmit/receive control packets as is required for data packets.

The protocols in [6] are structured as a series of reservation stages, each followed by a packet transfer stage. Their reservation protocol runs as follows. Assume that transmissions to some node  $S(j)$  are to be scheduled without conflicts. Let  $n_i$  denotes the number of packets at  $S(i)$  ( $i \neq j$ ) destined for  $S(j)$ . In the first time slot,  $S(1)$  transmits  $n_1$  to  $S(2)$ . In the second slot,  $S(2)$  transmits  $n_1 + n_2$  to  $S(3)$  and so on. In the packet transfer stage, node  $S(i)$  wakes up at time slot  $n_1 + n_2 + \dots + n_{i-1} + 1$  and transmits its  $n_i$  packets to  $S(j)$ . For  $n > p^2$ , the reservation phase is run for  $\sqrt{p}$  groups of  $\sqrt{p}$  nodes each instead of individual nodes. Each node now declares the number of items it possesses that are destined for any of the stations in a given group  $G(j)$ . In the packet transfer stage, the  $\sqrt{p}$  stations in  $G(j)$  receive the packets such that each station finally has exactly  $n/p$  packets. A further redistribution of packets within the group is then performed. This procedure is repeated for each group. This approach is then generalized by partitioning the nodes into  $p^{\frac{1}{d}}$  groups of  $p^{1-\frac{1}{d}}$  stations each - where  $d$  is an integer from 1 to  $\lceil \log p \rceil$ . Packets are first routed among these  $p^{\frac{1}{d}}$  groups, and then each group is subdivided further and the protocol proceeds recursively.

In a network with  $k$  data channels [6] stations are divided into  $k$  groups of  $p/k$  stations each. First, each group  $G(i)$  is allotted its own channel  $C(i)$ . The  $n/k$  items in each  $G(i)$  are locally routed such that each station stores items destined for exactly one group  $G(j)$ . In the second phase, all stations that store items destined for some group  $G(i)$  are allotted channel  $C(i)$  and those items are routed to the stations in  $G(i)$ . Finally, items are locally routed within  $G(i)$ . The first stage essentially partitions the set of packets into  $k$  subsets such that packets in each subset can be locally (independently) routed in the second stage. However, the activity on each channel is not uniform; i.e., the channel utilization will depend on the distribution of packets among nodes; and so will the energy balance of the routing.

The protocols in [6] do not tolerate failures. A fault-tolerant protocol for permutation routing is outlined in [3] with the following fault model: (i) nodes can fail either due to damage sustained during deployment, or by the eventual depletion of energy reserves, (ii) no new faults occur during the execution of the permutation routing protocol, (iii) each faulty station is dead and does not send malicious messages, and (iv) the number of faulty stations is a very small percentage of all the stations. This fault model is too optimistic for typical sensor network deployments since the assumption that nodes do not fail during the routing is unrealistic for many real-world scenarios.

## 6. Concluding Remarks

We described a robust, energy-efficient protocol for  $(N, p, k_1, k_2)$  routing on single-hop sensor networks with multiple data channels. Our system model includes a special low-power radio over which coordination and scheduling of packet transfers is carried out. Our protocol always transfers packets directly from source to destination, without using intermediaries, and the effect of node failure is localized. Simulation results show that our protocol is energy balanced - assuming all nodes receive and transmit the same number of packets. If distribution of packets is non-uniform, it will be proportionately reflected in the energy balance. Energy efficiency will of course depend upon the value of  $r_2$  - the ratio between the energy required to transmit a packet over the data channel to that required to transmit a packet over the control channel. A simple extension to the multi-channel routing protocol also makes it tolerant to permanent communication failures.

## References

- [1] A. Bakshi and V. K. Prasanna. Energy balanced communication in wireless sensor networks. In *Proc. 5th Intl. Conf. on Mobile and Wireless Communication Networks*, October 2003.
- [2] A. Bakshi and V. K. Prasanna. Structured communication in single hop sensor networks. In *1st European Workshop on Wireless Sensor Networks*, January 2004.
- [3] A. Datta. Fault-tolerant and energy-efficient permutation routing protocol for wireless networks. In *Proc. 17th International Parallel and Distributed Processing Symposium*, April 2003.
- [4] N. Jain, D. K. Madathil, and D. P. Agrawal. Energy aware multi-path routing for uniform resource utilization in sensor networks. In *2nd Workshop on Information Processing in Sensor Networks*, April 2003.
- [5] A. Manjeshwar and D. P. Agrawal. APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 2002.
- [6] K. Nakano, S. Olariu, and A. Zomaya. Energy-efficient permutation routing protocols in radio networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):544–557, Jun. 2001.
- [7] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *International Conference on Communications (ICC), Workshop on Sensor Network Protocols and Applications*, May 2003.