

Structured Communication in 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. We propose a model for communication in single-hop wireless sensor networks and define and evaluate the performance of a robust, energy balanced protocol for a powerful and general routing primitive - (N, p, k_1, k_2) routing. This routing primitive represents the transfer of N packets among p nodes, where each node transmits at most k_1 packets and receives at most k_2 packets. Permutation routing is an instance of this primitive and has been recently studied by other researchers in the context of single-hop wireless radio networks. Our proposed protocol is the first to exploit the availability of a low-power control channel in addition to the “regular” data channel to achieve robustness in terms of handling node and link failures - both permanent and transient. We use a dynamic, distributed TDMA scheme for coordination over the control channel, which leads to collision-free transmissions over the data channel. The objective is to minimize overall latency, reduce per-node and overall energy consumption, and maintain energy balance. Our protocol is robust because the coordination is decentralized, there is no single point of failure, and one node failure affects only the packets destined for (and transmitted from) that particular node. Simulation results for different scenarios are presented.

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 for a specific application such as target tracking, habitat monitoring, etc. In other words, the nature of tasks and interactions between tasks is already known at design time. The 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. Our primary

^{*} This work is supported by the DARPA Power Aware Computing and Communication Program under contract no. F33615-C-00-1633 monitored by Wright Patterson Air Force Base.

concern is with the design and optimization of application-specific *communication* (or routing) protocols in WSNs.

A number of application-specific routing protocols have been proposed over the past few years. Most of these have focused on specific application-level communication patterns, and customize the network protocol stack for efficiency. We believe that for large-scale sensor networks, it is unreasonable to expect the end user to adopt such a bottom-up approach and manually customize the routing, medium access, and/or physical layer protocols for each application. As WSNs evolve from simple, passive data collection networks to more sophisticated sensing and/or real-time actuation, the design complexity will correspondingly increase. The right balance has to be achieved between system efficiency and ease of design. One of the approaches to achieving such a balance is to design energy-efficient and robust protocols for a set of *structured communication primitives*, and export these primitives to the end user¹. “Structured communication” refers to a routing problem where the communication pattern is known in advance. Example patterns include one-to-all (broadcast), all-to-one (data gather), many-to-many, all-to-all, permutation, etc. This layer of abstraction provided by the library of communication primitives will allow the user to focus on high level design issues, instead of low-level 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. Feature extraction [2] and other complex image processing kernels are examples of sensor network applications that can benefit from using simple high-level system models and computation/communication primitives to hide the less relevant system details. The UW-API [3] is also a representative of the community’s interest in defining building blocks for collaborative structured communication and using them for applications such as target tracking.

The primary contribution of this paper is an energy-efficient, robust protocol for (N, p, k_1, k_2) routing in single-hop network topologies. 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 transferred by a node, and k_2 is the maximum number of packets that a node can expect to receive from others. We adopt a pragmatic approach towards measuring the efficiency of our protocol; based on the fact that 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. Permutation routing protocols such as those proposed in [4] are asymptotically near-optimal but have large overheads for small-size networks (see Sec. 2). Our protocols are not asymptotically optimal, but are efficient for single-hop network sizes that we expect will represent the overwhelming majority of real-world deployments.

(N, p, k_1, k_2) routing is a very general and powerful primitive whose special cases include almost all of the commonly encountered communication patterns

¹ The Message Passing Interface [1] standard for communication in parallel and distributed systems has a similar motivation.

- scatter, gather, all-to-all, etc. Our protocol tolerates permanent and/or transient node and link failures, dynamically adjusts the transmission schedule for maximum energy and latency savings, and is robust because all packets are transferred directly between the source and destination, without intermediaries. Therefore, node failures do not affect unrelated transmissions. Finally, our protocol is energy-balanced. For the scenario where $k_1 = k_2 = \frac{N}{p}$, each node spends almost the same amount of communication energy (see Sec. 5.3). For unequal k_1 and k_2 , the energy consumed is proportional to the number of packets transmitted and received by the node.

Our protocol is also the first, to our knowledge, that explicitly uses a low-power control channel for dynamic scheduling and fault tolerance. Such dual channel architectures are being explored in the sensor networking community [5, 6]. The driving force behind the exploration of multiple radio architectures is the concern with saving energy. A separate, low-power radio facilitates the off-loading of many control and coordination tasks that would otherwise need to be performed on the more heavy-duty data radio. The design of energy-efficient algorithms that acknowledge and exploit the existence of a low-power control channel promises to yield greater energy savings, in addition to optimizations over the “regular” data channel.

Hierarchical, clustered topologies have been advocated because of the quadratic relation between transmission range and transmission energy. By using short-range (and hence low power) radios and organizing the sensor network into clusters – with single-hop routing within clusters and multi-hop routing between clusters – significant energy savings can be realized. [7]. A hierarchical approach to data routing in such topologies is to first route packets to the suitable cluster that contains the destination node(s) and then redistribute packets within each cluster using efficient single-hop (N, p, k_1, k_2) routing.

The remainder of this paper is organized as follows. In Section 2, we analyze the recently proposed energy-efficient [4] and fault-tolerant [8] protocols for permutation routing in radio networks, and differentiate our work. Section 3 defines our system model, and Section 4 presents our basic protocol using a low-power control channel and one data channel. This protocol is further enhanced in Section 5 to handle permanent and transient node and link failures. Simulation results are used to estimate the energy and latency savings achievable through our protocol in the face of faults. We conclude in Section 6.

2 Permutation Routing in a Single-hop Topology: State-of-the-Art

Energy-efficient protocols for permutation routing in single-hop radio networks were presented in [4] using Demand Assignment Multiple Access (DAMA). In [4], time is divided into fixed length slots, and the objective of the protocol is to avoid collisions by alternating reservation phases with the actual data transfers. There is no separate control channel, and reservation as well as data transfer is

Table 1. Simple Protocol in [4]

Nodes p	Packets n	%age Latency overhead	%age Energy overhead
50	1500	160	160
	2000	120	120
	2500	96	96
100	6000	163	163
	8000	122	122
	10000	98	98
200	20000	198	198
	30000	132	132
	40000	99	99

done using in-channel signaling. As a consequence, the same time and energy is required to transmit/receive a control packet as is required for data packets.

The “simple protocol” in [4] for $n \geq p^2$ requires $n + p^2 - 2p$ time slots and every station is awake for at most $2\frac{n}{p} + 2p - 4$ time slots. The more general, recursive protocol in [4] performs the task of permutation routing involving n items on a single-channel network in fewer than $2dn - 2(p - 1)$ time slots with no station awake for more than $\frac{4dn}{p}$ time slots, where $d = \lceil \frac{\log p}{\log \frac{p}{2}} \rceil$. Tables 1 and 2 show the latency and energy overheads of these two protocols for typical single-hop network sizes. Energy and time overheads are evaluated using the following rationale. The actual transfer of n packets of data requires n time slots. Under a uniform energy cost model – i.e., transmitting or receiving a packet requires one unit of energy – the actual transfer of n packets requires $2n$ units of energy. If T is the total number of time slots required for the routing, we define *latency overhead* as $\frac{T-n}{n}$. If node i in the network is awake for $e(i)$ slots, the total energy E required for the routing is $\sum_{i=1}^p e(i)$. We define *energy overhead* as $\frac{E-2n}{2n}$. As seen in the table, if the protocols are implemented in (fault-free) single-hop WSNs with a few hundred nodes, the energy and latency overheads are unacceptably high.

It is instructive to note that in state-of-the-art (contention-based) MAC protocols for wireless networks (IEEE 802.11) and wireless sensor networks [9], the source broadcasts an RTS (Request To Send) control message and the destination responds with a CTS (Clear To Send) control message. Other eavesdropping nodes adjust their schedules accordingly. The result is that actual data is never transferred unless (i) both the sender and receiver are guaranteed (with high probability) exclusive access to the channel, and (ii) other nodes in the vicinity are silenced for the duration of the transmission through the virtual carrier sense mechanism. The control messages are smaller (hence cheaper in time and energy) than the regular data packets. For instance, the S-MAC protocol [9] for sensor networks uses special 8-byte packets for RTS/CTS on the Berkeley Motes, where regular data packets can be upto 250 bytes [10]. Of course, contention based pro-

Table 2. General Recursive Protocol in [4]

Nodes p	Packets n	%age Latency overhead	%age Energy overhead
50	1500	293	300
	2000	295	300
	2500	96	100
100	6000	296	300
	8000	297	300
	10000	98	100
200	20000	298	300
	30000	298	300
	40000	99	100

protocols cannot be compared with DAMA (TDMA) schemes. Even then, *the basic idea of a cheap negotiation phase preceding each data transfer is attractive* and can be exploited in TDMA-like synchronization mechanisms, as we show in later sections. If this negotiation is performed over a low-power radio (as against in-channel), the benefits in terms of energy saving are even greater.

The protocol in [4] is not designed to handle any type of communication failures among the participating nodes. A fault-tolerant protocol for the same problem is outlined in [8], which assumes the following:

- nodes can fail either due to damage sustained during deployment, or by the eventual depletion of energy reserves, and
- after the first phase of the routing protocol has identified a node as ‘fault-free’, it remains fault-free till the protocol completes.

Theorem 3.2. in [8] states that permutation routing of n packets in a single-hop network with p nodes and k channels can be solved in $\frac{2n}{k} + (\frac{p}{k})^2 + \frac{p}{k} + \frac{3p}{2} + 2k^2 - k$ slots. Also, the upper bound on energy consumption for all p nodes is $4nf_i + \frac{2n}{k} \frac{3p^2}{k} + pk^2 + \frac{p^2}{2k} + \frac{p^2}{2} + 4kp$ slots, where f_i is the number of faulty stations in a group of $\frac{p}{k}$ stations. The total number of faulty stations is therefore $f_i k$. Table 3 shows the latency and energy overhead of this protocol for typical single-hop network sizes.

In addition to the high overheads (which could be greatly reduced by running the same protocol on a dual channel communication architecture), the fault model is too optimistic for many sensor network deployments. Communication links between nodes can be prone to intermittent failures (transient, high error rates) due to environment dynamics. Therefore, a node that is identified as faulty at one stage of the protocol cannot be completely discounted, especially if the non-responsiveness was transient. Also, nodes could fail at any time, not just at the beginning of a routing phase. Our protocol is based on a general fault model that anticipates these possibilities.

Table 3. Fault-tolerant Permutation Routing Protocol in [8]

Nodes p	Faults $k.f_i$	Packets n	%age Latency overhead	%age Energy overhead
50	6	1500	48	807
		2000	36	755
		2500	29	724
100	10	6000	45	1197
		8000	33	1148
		10000	27	1118
200	16	20000	52	1831
		30000	34	1754
		40000	26	1715

We use RTS/CTS-like signaling on a time synchronized control channel, i.e., there are no collisions while sending the RTS/CTS messages. Compared to contention-based RTS/CTS signaling, this mechanism is more energy efficient². Requiring a (contention-free) reservation phase before every single transmission does incur an overhead compared to protocols that schedule all packet transmissions initially and require no coordination during the data transfer phase (e.g., [4]). However, the messages exchanged in the reservation phase allow the protocol to be highly robust (see Sec. 5). Also, if these messages are exchanged over a low-power radio, the cost is further reduced.

In the following sections, we define a model for communication in single-hop WSNs that uses separate data and control channels. We then define and evaluate our efficient and robust protocol for permutation routing.

3 Our System Model

1. 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.
2. All nodes are initially time-synchronized with each other and remain time-synchronized for the duration of the routing.
3. There are two communication channels: a data channel for transfer 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. Note that *channel* is sometimes used to denote different frequencies on the same transceiver unit, and not separate transceivers. In our system architecture,

² As an example, the multiaccess protocol in the WINS system [11] is motivated by the advantages of synchronicity over random access in improving energy performance of a wireless sensor network.

the control channel is always a separate transceiver unit. There can be multiple frequencies available to the data radio, and these will constitute different *data channels*. In this paper, we consider a single data channel. Therefore, radio and channel can be used interchangeably.

4. A data or control radio can be in one of three states: Transmit, Receive, or ShutDown. We assume that energy consumption per packet is same for the Transmit and Receive states; and zero (negligible) for ShutDown.
5. 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 mode to another.
6. The network has single-hop connectivity. Any data (control) radio can directly communicate with any other data (control) radio in the network.
7. 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_{dp} time and e_{dp} energy. Similarly, a packet transmission over the control channel requires t_{rs} time and e_{rs} energy. We assume that control packets sent over the control channel are smaller than data packets sent over the data channel. $t_{dp}/t_{rs} = r_1$ and $e_{dp}/e_{rs} = r_2$, 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 also consider two types of faults in the network:

- **Transient faults:** Communication links both into and out of k ($0 \leq k \leq p$) nodes are subjected to transient outages. Let TF be the set of these k nodes. p_{tf} is the probability that a node $S(i) \in TF$ will be unavailable at a given time slot. p_{tf} is calculated independently for each of the nodes in TF , and also for each time slot.
- **Permanent faults:** Any of the p nodes can fail with a certain probability p_{pf} in any time slot and remain unresponsive to all future requests. Failure probability is uniformly distributed across all nodes in the network.

Values of k , p_{pf} , and p_{tf} will depend on the particular sensor network deployment. The choice of permanent or transient fault model for algorithm design is also up to the end user. Algorithms designed for transient faults will work correctly for permanent faults, although with a larger overhead (see Sec. 5).

Since coordination takes place on the control channel, the above fault models really apply to the (non-) availability of the control radios. We assume that failure of the control channel between a pair of nodes implies a failure of their data channel. Also, any fault that occurs is assumed to be *non-malicious*, i.e., the node simply stops communicating with other nodes for the duration of the fault.

For sake of simplifying the analysis, 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*. Otherwise, the running time of the protocol could be non-deterministic and unbounded. We analyze (and simulate) the protocol for N reservation phases, each possibly followed by a data phase, where N is the total number of packets to be routed. In real sensor networks, data samples will be relevant only for a short duration because they represent a physical phenomenon to be detected in real-time. Enough spatio-temporal redundancy can be expected in a robust sensor network deployment for occasional packet loss to be tolerated at the application level. Even if this may not always be the case, we believe it is a good policy to drop a packet if the destination is not responding than to keep transmitting to a possibly dead sensor node and hold up useful data packets meant for other live destinations.

4 An Energy-Efficient Protocol Using a Low-Power Control Channel

Collisions over the data channel lead to significant energy overhead due to retransmission. Since the control channel is low-power, collisions of control packets might not cause much of an energy overhead, but are still undesirable due to the nondeterministic latency of conflict resolution. Instead of contention-based mechanisms, we use *TDMA-like coordination over the control channel* and asynchronous transfers over the data channel. For energy efficiency, it is also important that a node's radios are active only when they are the sender or recipient of an ongoing transmission, and asleep at all other times.

Our basic protocol ($p_{pf} = p_{tf} = 0$) has the following properties:

- there are never any collisions between control packets over the control channel
- there are never any collisions between data packets over the data channel
- data radios are ShutDown at all times except when a ‘useful’ transmission or reception is in progress
- control messages are exchanged in aTDMA fashion over the control channel
- data packets are transmitted asynchronously over the data channel
- all data packets are transmitted directly from the source to the destination, without using intermediate nodes
- all coordination is decentralized (peer-to-peer)

Description:

Time is divided into frames each consisting of exactly one reservation phase (RP) and at most one data transfer phase (DP). Let the RPs be numbered in increasing order, with the first reservation phase being $RP[1]$. Similarly, let the DPs be labeled such that a DP that immediately succeeds $RP[i]$ is labeled $DP[i]$. $RP[i]$ is said to be *owned* by $S(i)$ if no other node is supposed to send an *RTS* in $RP[i]$. $RP[i]$ is said to be *successful* if $S(i)$ sends an *RTS* addressed to some

$S(j)$ and $S(j)$ sends a *CTS* in acknowledgment. $DP[i]$ is said to be *owned* by nodes $S(i)$ and $S(j)$ if $RP[i]$ is successful. Data is transferred in $DP[i]$ iff $RP[i]$ is successful. Node $S(i)$ owns $RP[\lfloor \frac{N(i-1)}{p} + 1 \rfloor]$ through $RP[\lfloor \frac{N.i}{p} \rfloor]$.

Every node $S(j)$ maintains a counter c_j initialized to 0. The algorithm followed by each node is given in Fig. 1. One $S(i)$ owns the RP and broadcasts *RTS*es. 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 RP. $S(i)$ switches its data radio to Transmit mode for $DP[x]$ and the destination $S(j)$ switches its data radio to the Receive mode for $DP[x]$. $S(i)$ transmits the packet to $S(j)$ in $DP[x]$. Data radios of all $S(k)$, where $k \neq i$ and $k \neq j$, stay in ShutDown mode in $DP[x]$.

Pre-processing: A pre-processing phase is carried out only over the control channel, if k_1 is different for different nodes. This phase consists of exactly p slots, where each slot is equivalent to an *RP* in terms of time and energy cost. In the i^{th} RP, $S(i)$ broadcasts the value of k_1^i . All p nodes remain awake for the p RPs. At the end of this phase, each node knows the owner of each of the N RPs and *DP*s to follow. It might also be desirable for each node to know how many packets are destined for itself, i.e., for each $S(i)$ to know the value of k_2^i . When this value

is known, $S(i)$ can shut off completely once it has received k_2^i packets (except in the *RPs* it owns). 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 is not the same for all nodes, this information can be exchanged over the control channel in a series of p rounds, each round consisting of p slots³. In the j^{th} slot of the i^{th} round, $S(j)$ broadcasts the number of packets in its buffer destined for $S(i)$. This phase requires 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.

```

If I am owner of current RP
    remove next packet from transmit queue
    send RTS to the destination
    wait for CTS from destination
    transmit packet in the following DP
else
    if I have received less than N/p packets
        listen to RTS sent by owner of RP
    if I am the destination of the RTS
        reply with a CTS
        receive packet in the following DP
        increment count of received packets
    else
        sleep in the following DP
else
    do nothing (ShutDown)
    
```

Fig. 1. The Basic Protocol

³ This approach is similar in some respects to the reservation protocol in [4].

The pre-processing phase of p^2 slots, where each node computes the value of its k_2 might not be useful in real-world scenarios when either the transient or permanent fault model is applicable and values of p_{pf} , p_{tf} , r_1 and r_2 are non-negligible. The primary purpose of each node knowing its k_2 is to allow it to shut down after it has received k_2 packets, knowing that there will be no more packets destined for itself. However, consider a scenario where some $S(i)$ fails with N/p packets in its buffer, and those packets are addressed to the other $p-1$ nodes. These $p-1$ nodes will never receive their k_2 packets and will remain awake for the entire N slots, thereby negating any usefulness of knowing their k_2 . The utility of the advance collaborative computation of k_2 's for each $S(i)$ is only in ideal, fault-free scenarios which are unrealistic in typical sensor networks. In the protocol description and analysis in this paper, we assume that each node knows its k_2 and the value of k_2 is the same for all nodes.

Heterogeneity: We assume that clocks on all nodes are perfectly synchronized. Control radios of all nodes participating in $RP[x+1]$ go to sleep for t_{dp} time from the beginning of $DP[x]$. Depending on the error rate of the channel, the actual transmission time could be more or less than t_{dp} . Now, the control radio cannot sense if the data channel is busy or not. Therefore, some signaling mechanism is required to coordinate between the control radios and the data radios. *We describe a simple extension to the basic protocol to handle heterogeneity.* The concept is essentially the same as the backoff mechanism in MAC protocols, except that all nodes backoff for a predetermined time period as against the random backoff used to resolve contention in most MAC protocols.

Suppose a data transfer is in progress in $DP[x]$, $S(i)$ and $S(j)$ being the source and destination respectively. Exactly two data radios will be active; $S(i)$'s in Transmit and $S(j)$'s in Receive. We require each of these radios to inform the control radio (through an interrupt mechanism) about the successful completion of the packet transfer – which could occur in more or less than t_{dp} .

In the enhanced protocol, the duration of RP is extended by a small time t_{wait} – added to the beginning of the RP. Let $S(k)$ be the owner of $RP[x+1]$. At the beginning of $RP[x+1]$, $S(i)$'s control radio will know if the packet transfer has completed in t_{dp} time or not. If it has completed, there will be no transmission in the initial t_{wait} time. After t_{wait} , $S(k)$ will broadcast an *RTS* and the protocol will proceed as usual. If the transfer has not completed, $S(i)$'s control radio will immediately broadcast a *WAIT* packet over the channel without waiting for t_{wait} . All control radios that are monitoring the channel will go to sleep for a predetermined time t_{ht} to allow $DP[x]$ to complete. The same procedure is followed when the control radios once again switch from ShutDown to Receive mode after t_{ht} . We assume that t_{dp} and t_{ht} are determined *a priori* (maybe through empirical means) and are provided to the protocol. Depending on the accuracy of estimates for t_{dp} and t_{ht} , the number of wait periods will be different.

Robustness: The 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

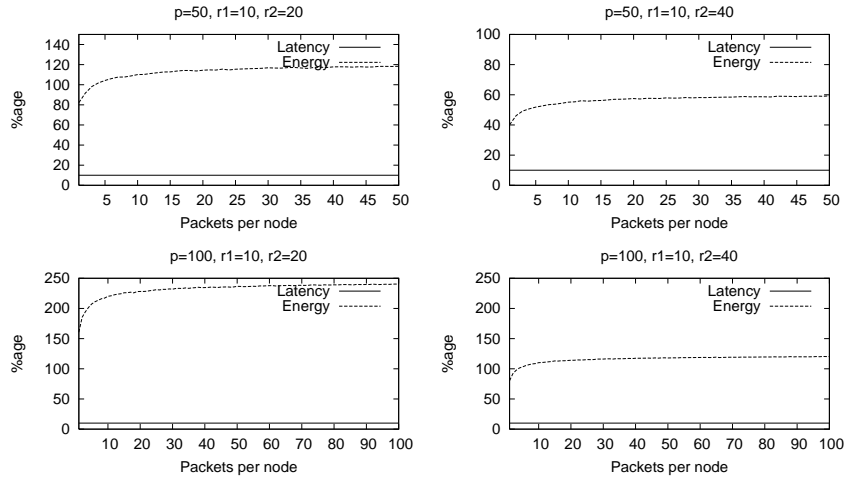


Fig. 2. Latency and Energy Overheads: Ideal Case

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.*

Performance: Total latency of our routing is $N.(t_{dp} + t_{rs})$. Since $r_1 = t_{dp}/t_{rs}$, the latency overhead of the reservation phase is $\frac{1}{1+r_1}$. For $r_1 = 20$, the reservation phase requires approximately 5% of the total latency.

Every node stays awake for exactly $2.N/p$ DPs - consuming $2.N.e_{dp}/p$ energy. This is the minimum energy required for data transmission in the permutation routing. *The protocol is therefore optimal in terms of e_{dp} (and t_{dp}).* Every node participates in the N/p RPs that it owns. Also, a node stops participating in RPs (not owned by itself) as soon as it has received N/p packets. Because a node does not know in which RP it will receive an *RTS*, every node stays awake for an average of $N(p+1)/2p$ RPs. The *average energy* spent by each node in the RP is therefore $e_{rs}.N(p+1)/2p$.

As defined earlier in Sec. 2, if routing one packet takes one unit of time, and T is the total time taken to route n packets, we define latency overhead as $\frac{T-n}{n}$. Similarly, if E is the total energy required to route n packets and transmitting (or receiving) one packet takes one unit of energy, we define energy overhead as $\frac{E-2n}{2n}$.

We simulated our protocol for a 100-node network ($p = 100$) and varied the packets per node (N/p) from 1 through 100 - i.e., the total number of packets in the network vary from 100 through 10,000. Values of r_1 and r_2 will depend on the radio electronics - typical values of r_1 will be a few tens (see Sec. 2), and the value of r_2 could be upto a hundred (see transmit/receive energies of different radios in [12]).

The top two graphs in Fig. 2 show latency and energy overheads of our protocol for a 50 node network for two different values of r_2 . The bottom two graphs show the same for a 100 node network. As expected, *latency overhead is constant, regardless of the size of the network or the number of packets*. A greater r_2 results in a lesser energy overhead, because nodes spend less energy in the RPs. Note that a 0% overhead represents the ideal, prescient algorithm whose performance is an absolute upper bound on the performance of any (N, p, k_1, k_2) protocol. The 0% overhead is unachievable in practice (in our model) and used here only as an absolute benchmark. For a 50-node network with $r_2 = 40$, our energy overhead is an almost 10-fold improvement over those in [8] (see Table 3) and for a 100-node network, the improvement is about 4-fold for $r_2 = 20$ and 11-fold for $r_2 = 40$.

Energy overhead does not appear to vary significantly beyond a certain (low) value of N/p because the factor of r_2 difference between e_{dp} and e_{rs} makes the incremental energy overhead less significant as the total energy consumption increases. We discuss the energy balance of our protocol in Sec. 5.3.

5 Permutation Routing in a Faulty Network

5.1 Handling Permanent Faults

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

- $S(i)$ sends RTS , $S(j)$ sends CTS : This means the sender and receiver are both alive, and data is transferred in $DP[x]$.
- $S(i)$ 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 RPs 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 RPs will occur. The real savings in this case will occur if $S(j)$ fails in some $RP[x]$, where $x < \frac{N \cdot j}{p} + 1$. In that case, all nodes have information about the exact N/p RPs that are owned by $S(j)$, and all nodes know that $S(j)$ has failed. As soon as $DP[N \cdot j/p]$ terminates $S(j+1)$ can start its transmissions. This policy saves $(t_{dp} + t_{rs})N/p$ latency and at most $2 \cdot N \cdot (e_{dp} + e_{rs})/p$ energy per permanent fault.
- $S(i)$ does not send RTS : Since the RPs are assigned to specific nodes based on their IDs, all listeners in $RP[x]$ know the ID of the owner $S(i)$ of $RP[x]$. A lack of RTS allows all listeners to collaboratively and implicitly cancel $DP[x]$, and also all remaining RPs and DPs owned by $S(i)$, proceeding directly to $RP[\frac{N(i+1)}{p} + 1]$. The best case is when $S(i)$ fails in the very first RP owned by $S(i)$, leading to a cancellation of the remaining $\frac{N}{p} - 1$ RPs and

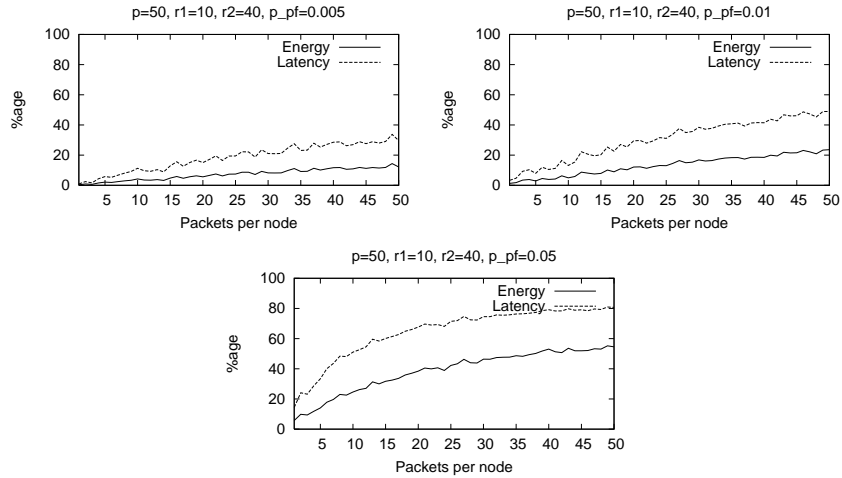


Fig. 3. Latency and Energy Savings: Permanent Faults

N/p DPs. The latency and energy savings are comparable to those for the previous scenario.

We simulated this protocol for a 50-node network for three different values of p_{pf} shown in Fig. 3. For each case, *we measure the energy and latency savings obtained by our dynamic rescheduling*, 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. Even for a very low p_{pf} of 0.5%, upto 15% energy and upto 30% latency can be saved through dynamic rescheduling and elimination of wasteful transmissions and receptions.

5.2 Handling Transient Faults

Transient faults preclude most of the collaborative dynamic rescheduling that is possible in the case of permanent faults, because non-responsiveness of a node in some RP/DP cannot be used to infer permanent node failure. Consider the same three possibilities during some $RP[x]$ as in the previous protocol.

- $S(i)$ sends RTS , $S(j)$ sends CTS : This means the sender and receiver are both alive, and data is transferred in $DP[x]$.
- $S(i)$ sends RTS , $S(j)$ does not respond: Since the fault could be transient, the only optimization possible in this scenario is for all listeners to collaboratively eliminate $DP[x]$, i.e., $RP[x+1]$ immediately succeeds $RP[x]$. The RPs and DPs owned by the failed $S(j)$ cannot be eliminated (as was possible for a permanent fault) because $S(j)$ can recover at any time slot. A latency saving of t_{dp} and energy saving of $2.e_{dp}$ can be achieved per transient fault.

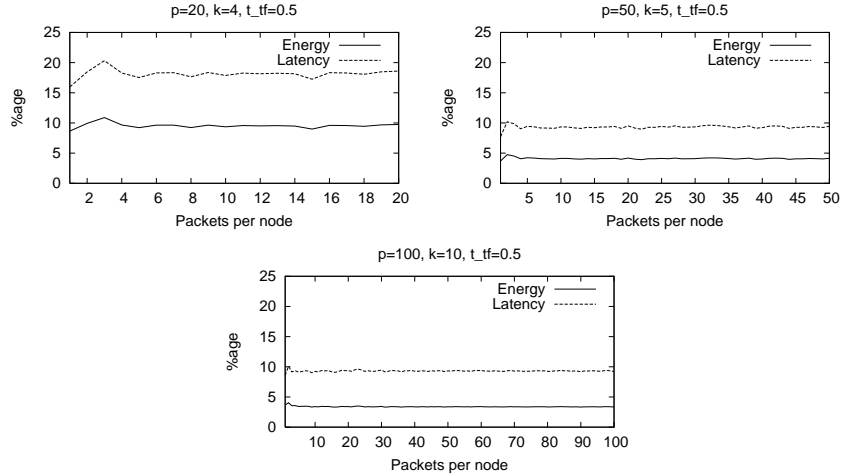


Fig. 4. Latency and Energy Savings: Transient Faults

- $S(i)$ does not send RTS: The policy to be followed in this case is the same as that for the previous scenario. $DP[x]$ is eliminated and $RP[x+1]$ follows $RP[x]$. Latency and energy savings are same as for the previous scenario.

Fig. 4 shows simulation results for the energy and latency savings that are achievable in face of transient failures. Three different network sizes were chosen – the values of p , k , and p_{tf} are provided with the corresponding graphs.

In the ideal case (no node failures), a node can stop participating in future RPs and DPs if it has received N/p packets and transmitted N/p packets. In case of failures, every failed RP 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 RPs, waiting for a packet that will not be delivered. This is the primary reason for the energy overhead of the reservation phase for both the transient and permanent fault models.

5.3 Energy Balance

In addition to total latency and total energy, energy balance is a critical measure of algorithm performance in wireless sensor networks. If the distribution of workload (especially communication) among nodes is unequal, there will be a rapid depletion of the energy reserves of the overused nodes and possibly a network partition. Since the time to network partition is a measure of the lifetime of a sensor network, equal distribution of workload is an important measure of the efficiency of algorithms designed for WSNs.

Permutation routing protocols in [4, 8] are not designed for energy balance. As shown in the first graph of Fig. 5, our protocol is *energy balanced*, i.e., all

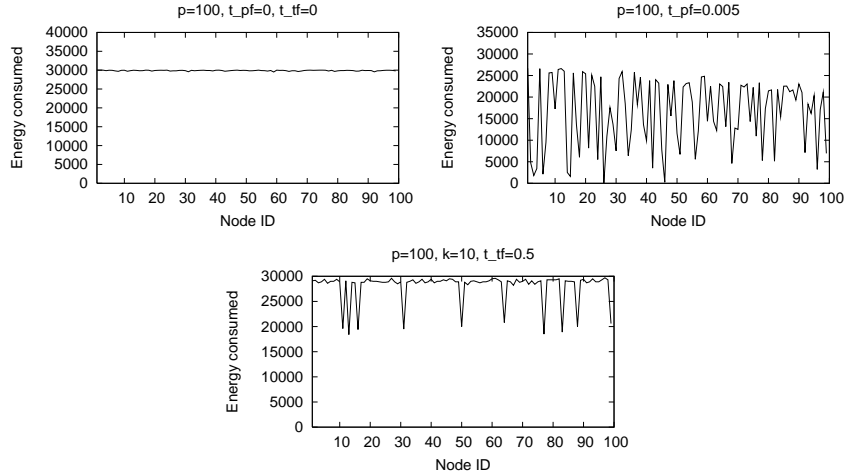


Fig. 5. Energy Balance

nodes spent approximately the same amount of energy. The second and third graph in Fig. 5 show the distribution of energy consumption in the presence of permanent and transient faults respectively. For the permanent fault model, the energy distribution reflects the energy savings that result from a detection of the faults and elimination of future transmissions to and from the faulty nodes. In the transient fault model (third graph in Fig. 5), the 10 points of low energy consumption correspond to the k ($=10$) faulty nodes in the network. The energy balance of the protocol can still be observed for the non-faulty nodes.

6 Concluding Remarks

The protocol described in this paper is designed specifically for one control channel and one data channel. We are investigating extensions to the protocol which will allow efficient and robust (N, p, k_1, k_2) routing with one control channel but multiple data channels.

By defining ‘clean’, high-level models for (classes of) sensor networks, a large body of research on parallel and distributed systems can be leveraged - including structured communication on regular and irregular topologies [13, 14]. Although a majority of work in embedded networked sensing is from the traditional networking perspective, there is a growing body of research that is approaching sensor networking from a parallel and distributed systems point of view [15–17]. Our work in this paper is a contribution towards a top-down approach to sensor network application design based on successive layered abstractions.

References

1. MPI Forum, MPI: A message-passing interface standard. *International Journal of Supercomputer Applications and High performance Computing* **8** (1994)
2. Krishnamachari, B., Iyengar, S.S.: Efficient and fault-tolerant feature extraction in sensor networks. In: 2nd Workshop on Information Processing in Sensor Networks. (2003)
3. Ramanathan, P., Wang, K.C., Saluja, K.K., Clouqueur, T.: Communication support for location-centric collaborative signal processing in sensor networks. In: DIMACS Workshop on Pervasive Networks. (2001)
4. Nakano, K., Olariu, S., Zomaya, A.: Energy-efficient permutation routing protocols in radio networks. *IEEE Transactions on Parallel and Distributed Systems* **12** (2001) 544–557
5. Guo, C., Zhong, L.C., Rabaey, J.M.: Low power distributed MAC for ad hoc sensor radio networks. In: Proceedings of IEEE GlobeCom. (2001)
6. Power Aware Sensing, Tracking and Analysis (PASTA), <http://pasta.east.isi.edu/>
7. Bandopadhyay, S., Coyle, E.: An energy efficient hierarchical clustering algorithm for wireless sensor networks. In: Proceedings of IEEE InfoCom. (2003)
8. Datta, A.: Fault-tolerant and energy-efficient permutation routing protocol for wireless networks. In: Proc. 17th International Parallel and Distributed Processing Symposium. (2003)
9. Ye, W., Heidemann, J., Estrin, D.: An energy-efficient MAC protocol for wireless sensor networks. Technical Report ISI-TR-543, USC/ISI (2001)
10. Ye, W., Heidemann, J., Estrin, D.: A flexible and reliable radio communication stack on motes. Technical Report ISI-TR-565, Information Sciences Institute, University of Southern California (2002)
11. Pottie, G., Clare, L.: Wireless integrated network sensors: Towards low-cost and robust self-organizing security networks. In: Proceedings of SPIE, Vol. 3577. (1999) 86–95
12. Estrin, D., Sayeed, A., Srivastava, M.: Wireless sensor networks (tutorial). In: Eighth ACM Intl. Conf. on Mobile Computing and Networking (MOBICOM). (2002)
13. Leighton, T.: Methods for message routing in parallel machines. In: Proceedings of the 24th annual ACM symposium on Theory of computing, ACM Press (1992) 77–96
14. Barth, D., Fraigniaud, P.: Approximation algorithms for structured communication problems. In: Proc. of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures. (1997)
15. Bhuvaneshwaran, R.S., Bordim, J.L., Cui, J., Nakano, K.: Fundamental protocols for wireless sensor networks. In: International Parallel and Distributed Processing Symposium (IPDPS) Workshop on Advances in Parallel and Distributed Computational Models. (2001)
16. Singh, M., Prasanna, V.K., Rolim, J., Raghavendra, C.S.: Collaborative and distributed computation in mesh-like wireless sensor arrays. In: Personal Wireless Communications. (2003)
17. M. C. Pinotti and C. S. Raghavendra (Co-Chairs), 3rd International Workshop on Wireless, Mobile and Ad Hoc Networks (WMAN), IPDPS 2003.