

# Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links\*

Yu Gu and Tian He

Department of Computer Science and Engineering  
University of Minnesota, Twin Cities  
{yugu,tianhe}@cs.umn.edu

## Abstract

In extremely low duty-cycle sensor networks, end-to-end communications cannot afford to maintain an always-awake communication backbone. Low duty-cycle, accompanied by the unreliable nature of wireless communication, makes it essential to design a new data forwarding scheme for such networks, so as to achieve network energy efficiency, reliability, and timeliness in an integrated fashion.

In this work, we introduce the concept of Dynamic Switch-based Forwarding (DSF) that optimizes the (i) expected data delivery ratio, (ii) expected communication delay, or (iii) expected energy consumption. DSF is designed for networks with possibly unreliable communication links and predetermined node communication schedules. Interestingly, we reveal that allowing *opportunistic looping* can actually reduce the end-to-end delay. To our knowledge, these are the most encouraging results to date in this new research direction. In this paper, DSF is evaluated with a theoretical analysis, extensive simulation, and physical testbed consisting of 20 MicaZ motes. Results reveal the remarkable advantage of DSF in extremely low duty-cycle sensor networks in comparison to three well-known solutions (ETX [3], PRR×D [19] and DESS [16]). We also demonstrate our solution defaults into ETX in always-awake networks and DESS in perfect-link networks.

## Categories and Subject Descriptors

C.2.1 [Computer Communication Networks]: Network Architecture and Design—*Wireless Communication*; C.2.2 [Computer Communication Networks]: Network Protocols

## General Terms

Algorithms, Performance, Design

## Keywords

DSF, lossy radio link, low duty-cycle, data forwarding

---

\*This work was supported in part by NSF grant CNS-0615063 and CNS-0626614. We would like to thank the anonymous reviewers and our paper shepherd Philip Levis for the invaluable comments that significantly improved this work.

## 1 Introduction

Many sensor network applications need to last a long time with a limited energy supply. To resolve the conflict between limited energy and application lifetime requirements, it is necessary to reduce node communication and sensing duty cycles. With the growing gap between application requirements and the slow progress in battery capacity [10], there are an increasing number of extremely low duty-cycle sensor networks designed and deployed. Together with lossy radio links, these new networks impose new challenges for data forwarding protocols.

In this work, we focus on *extremely low duty-cycle sensor networks with unreliable communication links*, in which energy management protocols [6, 15, 22, 24] schedule sensing and communication at each individual sensor device to enable a duty cycle of 1% or less. Essentially, during the operation of sensor applications, sensor nodes activate very briefly and stay in a dormant state for a very long period of time. Due to the devices' extremely limited energy budget, maintaining an always-awake communication backbone becomes infeasible. Consequently to forward a packet, a sender may experience *sleep latency* – the time spent waiting for the receiver to wake up.

In this paper we attempt to design a new data delivery method to optimize source-to-sink data delivery ratio, end-to-end (E2E) delay, or energy consumption under *unreliable* and *intermittent* connectivity within scheduled networks. The major intellectual contributions of this work are as follows:

- We propose a simple yet effective representation of sensor working schedules, which for the first time enables us to reveal the time-dependent data forwarding behavior within extremely low duty-cycle sensor networks.
- To the best of our knowledge, this is the first in-depth work to investigate the combined effect of *sleep latency* and *unreliable communication links*, which dramatically reduces the effectiveness of the existing solutions. A novel dynamic switch-based forwarding technique over time-dependent networks is proposed to achieve optimal expected delivery ratio (EDR), expected E2E delay (EED), or expected energy consumption (EEC), respectively.
- Interestingly, we demonstrate that the allowance of certain data forwarding loops can actually reduce end-to-end data delivery delay. In addition, we address practical issues such as time synchronization and changes in link quality over time.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 describes the need for a new data forwarding technique in extremely low duty-cycle sensor networks. Section 4 articulates the network model and related assumptions. Section 5 introduces the detailed design of DSF and discusses related issues. Section 6 describes our system im-

plementation and provides an evaluation on the TinyOS/Mote platform. Simulation results are presented in Section 7. Section 8 concludes the paper.

## 2 Related Work

The contribution of our work lies in the intersection of two important cutting-edge research topics. We demonstrate that the intriguing interaction between unreliable links and low-power duty-cycling necessitates a fundamentally new approach.

**Link-Quality-Based Forwarding:** Many recent works [11, 28, 29] reveal that wireless communication links, especially for the low-power sensor devices, are extremely unreliable and have a significant impact on data delivery.

De Couto et al. introduce the expected transmission count metric (ETX) to find high-throughput paths on multi-hop wireless networks [3]. Woo et al. show that cost-based routing using a minimum expected transmission metric obtains good performance in wireless sensor networks [23]. Seada et al. study the distance-hop trade-off for geographic routing in wireless sensor networks and show that the product of the packet reception rate (PRR) and the distance traversed toward the destination (D) is an optimal metric (PRR×D) for selecting a next-hop forwarder [19]. Lee et al. present SOFA, an on-demand solicitation-based forwarding protocol and show that SOFA outperforms the commonly used link estimation-based routing schemes implemented in TinyOS [12].

In these works, the authors assume the constant availability of connectivity with no sleep latency, which may not be true in extremely low duty-cycle sensor networks.

**Sleep-Latency-Based Forwarding:** In the research direction of low duty-cycle networks, Dousse et al. provide a solid analysis of bounds of the delay for sending data from a node to a sink in the networks with completely uncoordinated node working schedules [4]. Cao et al. present a Streamlined Forwarding (SF) strategy to reduce delay in end-to-end communication [1]. This strategy works well under a constrained communication pattern where only one sink is allowed. Yu et al. present algorithms to minimize the overall energy dissipation of the sensor nodes in the network subject to the latency constraint [27]. Lu et al. introduce various techniques for minimizing communication latency while providing energy-efficient periodic sleep cycles for nodes in wireless sensor networks [16]. More recently, Keshavarzian et al. introduce a multi-parent forwarding technique and propose a heuristic algorithm for assigning parents to the nodes in the network [9]. We note, however, that all these approaches in low duty-cycle networking assume perfect communication links.

We note that many MAC protocols, such as B-MAC [18], S-MAC [25], FPA [13] and SCP-MAC [26], effectively deal with the issues of lossy radio links through FEC/ARQ and reduce duty-cycle through the Low-Power-Listening (LPL) [18]. These intelligent layer 2 protocols use implicit network information, such as packet transmissions, in order to optimize their underlying schedules or energy use. In this paper, we consider the dual of this problem by using information from layer 2 at the network layer to make better link selections.

To the best of our knowledge, no prior work has thoroughly studied the impact of both *lossy radio links* and *sleep latency* at the network layer. In this work, we reveal that these two issues are intrinsically correlated and that a new forwarding protocol can benefit from considering both.

## 3 Motivation

Our work is motivated by the interesting intersection between sleep latency and unreliable communication links in wireless sensor networks.

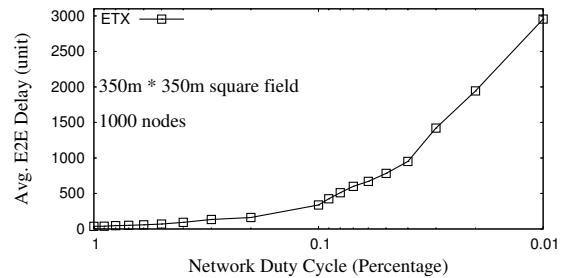


Fig. 1. E2E Delay vs. Network Duty Cycle

First, the state-of-the-art link-quality-based forwarding strategies such as ETX [3] and PRR×D [19] have demonstrated their superiority at improving network throughput and communication delay in traditional ad hoc and sensor networks. For both ETX and PRR×D, during a certain period of time each node usually has one fixed forwarding node for a destination. However, in extremely low duty-cycle scheduled sensor networks, metrics such as the expected transmission count (ETX) would suffer excessive delivery delays when waiting for the fixed receiver to wake up again if the ongoing packet transmission fails. Figure 1 shows the E2E delays from a randomly chosen source node to the sink node using ETX forwarding metrics under different network duty cycles in a randomly-generated network topology. The simulation was repeated 1000 times and the average value is reported in Figure 1 (in log-scale), which shows that as network duty cycle decreases, the E2E delay grows significantly. For example, at the duty cycle of 100%, the E2E delay of ETX is only 37.6 units of time. In contrast, when the duty cycle drops to 1%, the E2E delay increases to 2955.5 units of time, which is approximately an 80-fold performance degradation in end-to-end delay!

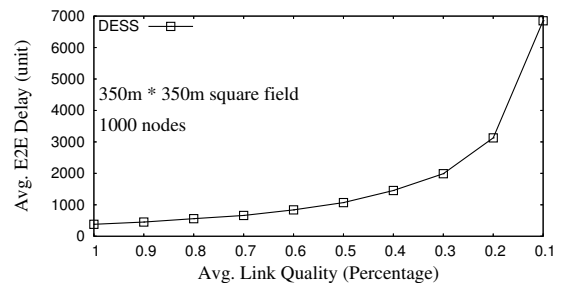


Fig. 2. E2E Delay vs. Average Link Quality

Second, sleep-latency-based forwarding [1, 16] ignores the reality that wireless radio quality is highly unreliable and that thus the optimality of their approaches holds only when the link quality in the network is perfect. Figure 2 shows the E2E delay from a randomly chosen source node to the sink node using delivery methods proposed in [16] under different average link quality in a random generated network topology. As shown in the figure, the E2E delay increases from 380.0 to 6851.4 units of time while the average link quality decreases from 100% to 10%, which is approximate a 20-fold performance degradation, even though global scheduling information is available.

The main observation from our initial studies is that both the link quality and the duty cycle of sensor nodes can significantly impact end-to-end communication. Although link-

quality-based forwarding [3, 19] and sleep-latency-based forwarding [1, 16] have demonstrated their effectiveness in their own contexts, they fail to deal with the combined effect exhibited in many real-world sensor network applications. This limitation motivates us to design a new data forwarding technique, which we discuss in the rest of the paper.

## 4 Models and Assumptions

Before presenting DSF in detail, we present the network model and assumptions used in this work. To simplify our description, we introduce DSF's design in a synchronized mode with discrete time. Later on, we explain why DSF works without time slots and only requires local synchronization. In other words, DSF works in CSMA networks where nodes are duty-cycled by upper-layer protocols such as sensing coverage [6] and power management [9].

### 4.1 Network Model

We assume a network with  $N$  sensor nodes. At a given point of time  $t$  a sensor node is in either an active or a dormant state. When a node is in the active state, it can sense and receive packets transmitted from neighboring nodes. When a node is in the dormant state, it turns off all function modules except a timer (for the purpose of waking itself up). In other words, a node can wake up to transmit a packet at any time, but can receive packets only when it is in its active state. Formally, we denote the network status at time  $t$  as  $G(t) = (V, E(t))$ , where  $V$  is a complete set of  $N$  nodes within the network, and  $E(t)$  is a set of directed edges at time  $t$ . An edge  $e(i, j)$  belongs to  $E(t)$  if and only if (1) node  $n_i$  is a neighboring node of  $n_j$ , and (2)  $n_j$  is active and hence able to receive data at time  $t$ . Essentially,  $G(t)$  represents the potential traffic flow within the network at time  $t$ . Obviously the connectivity of  $G(t)$  varies with time. In other words,  $G(t)$  is a time-dependent network.

We represent the states of each node  $n_i$  with a working schedule  $\Gamma_i = (\omega_i, \tau)$ .

- $\omega_i$  is an infinite binary string, in which 1 denotes the active state and 0 denotes the dormant state. Clearly, the duty cycle of a node is the percentage of 1's in the binary string. Since the working schedules of the sensor nodes are normally periodic (for sensing purposes), the infinite binary string  $\omega_i$  can be described using a regular expression.
- The state transitions between active and inactive states are time-driven. We use  $\tau$  to denote the time span a bit in the binary string  $\omega_i$ . For example, the total time-span of the binary string (1001) with  $\tau$  of 2 seconds is 8 seconds.

We note that the simple 2-tuple  $(\omega_i, \tau)$  is generic enough to represent arbitrary sensor nodes working schedules. Theoretically, when  $\tau \rightarrow 0$ ,  $\omega_i$  can precisely characterize any on/off behavior of node  $n_i$ . For clarity of presentation, we begin our design with a simplified assumption that it takes time  $\tau$  to transmit one packet and receive acknowledgment from a receiver. The assumption on the round-trip transmission time bound  $\tau$  holds well when traffic/congestion is low, which is the case in extremely low duty-cycle sensor networks. In addition, B-MAC [18] has already used link-level implicit acknowledgement to support fixed round-trip transmission time. We address the case that time  $\tau$  is long enough to transmit multiple packets in Section 5.5.

### 4.2 Time-Expanded Network

To visualize the data delivery process in a time-dependent network  $G(t) = (V, E(t))$ , we replicate  $G(t)$  with regular graphs

$G = (V, E)$  along with the time dimension. We call this is a *time-expanded network*. In this section, for a given sensor network topology and node working schedules, we describe how we can build a corresponding time-expanded network. The resulting time-expanded network can help us better understand the data delivery method introduced in the rest of the paper.

Given a network  $G(t) = (V, E(t))$  with  $n$  nodes and node working schedules  $\Gamma_i = (\omega_i, \tau)$ , where  $i \in V$ , we use the following rules to construct its corresponding time-expanded network.

- For any node  $i \in V$  at time  $t$ , we build a distinct node  $N_{it}$ .
- For each newly built node  $N_{it}$ , if node  $j$  is a neighboring node of the node  $i$  and  $p$  is the position of first active bit in  $\omega_j$  after time  $t$ , we build a directed edge from  $N_{it}$  to  $N_{jp}$  with a length of  $(p-t)\tau$ .
- At the destination node  $d$ , we connect all its time-expanded nodes to a null node with edge lengths of zero.

To illustrate the above network mapping rules, we provide a walk-through of time-expanded network construction from a time dependent graph. Figure 4 shows how to construct a time-expanded network from the linear time-dependent network shown in Figure 3. As shown in Figure 3, for node 1 at time 1, the only node that is within its communication range is node 2, and its first active state after time 1 appears at time 3, so in Figure 4, we build a directed edge from node 1 at time 1 to node 2 at time 3 with an edge length of  $2\tau$ . Similarly, we construct other edges in Figure 4. Finally, for the destination node 4, we connect all its time-expanded nodes from time 1 to time 6 to a null node with edge lengths of zero.

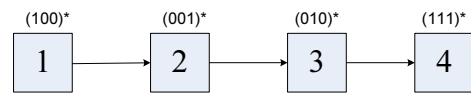


Fig. 3. A Linear Network

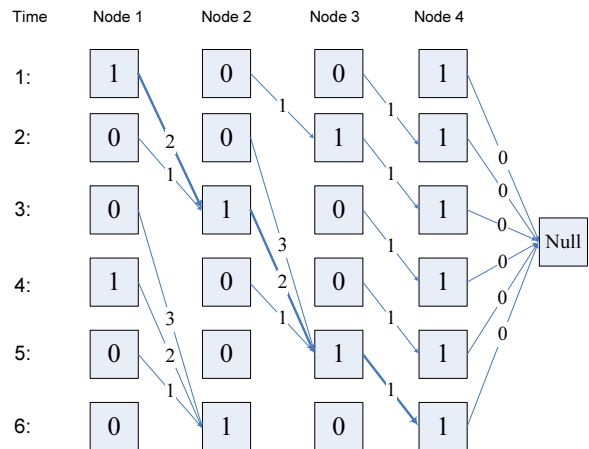


Fig. 4. Time-Expanded Network

### 4.3 E2E Delay in Time-Expanded Network

Obviously, if all the nodes are in active states, end-to-end (E2E) delay in the above network model equals  $H\tau$ , where  $H$  is the minimum number of hops between a source and a destination. However, if nodes in a network have certain working schedules  $\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_N\}$ , transmission at each hop could be delayed by waiting for the intermediate receivers to wake up.

To further illustrate the data delivery process in the extremely low duty-cycle network, Figure 4 demonstrates the process of delivering a packet from node 1 to node 4 when the packet is

ready to be sent at time 1. For the sake of simplicity, in this example we assume all links are perfect with no packet loss and will discuss cases when the link quality is not perfect in detail in the following sections. At the first two time intervals, node 2 (the only neighbor of node 1) is in the dormant state, and thus no packets could be transmitted. At the third time interval, node 2 becomes active, which allows node 1 to transmit a packet to it, so the packet is delivered from node 1 to node 2. At the second hop, node 2 waits one time interval for node 3 to wake up, and the packet arrives at node 3 at time 5. Finally, since node 4 is active all the time, without any additional waiting, node 3 delivers the packet to node 4 at time 6. The total end-to-end delay therefore is 5 units of time (i.e., arrives at the destination at time 6 and is ready to be sent at time 1 at the source).

## 5 Main Design

As shown in Section 4.3, when the link quality is perfect, the end-to-end delay is the sum of two types of delays: (1) the total transmission delay, which is the product of number of hops and  $\tau$ , and (2) The sleep latency, which is the time spent on waiting for the receivers to wake up at each hop. However, the unreliable radio links between low-power sensor devices suggests that the packet transmission between a sender and a receiver would not always be 100% successful. As a result, the waiting time at each hop is highly impacted not only by the node working schedule but also by the link quality, which inspires us to design a dynamic switch-based data forwarding protocol.

Since every operation within an extremely low duty-cycle sensor network is time-dependent, for the sake of clarity we use the terms *node* and *time-expand node* interchangeably in the rest of the paper. We have organized this design section into five components. Section 5.1 describes the basic design of Dynamic Switch-based Forwarding (DSF). Section 5.2 analyzes the expected delivery ratio, E2E delay, and energy consumption, assuming the forwarding action is known *a priori*. Section 5.3 optimizes the forwarding action to achieve maximum delivery ratio, minimal delay, and energy efficiency, respectively. Section 5.4 describes the distributed implementation of DSF. Section 5.5 presents our observations and extensions of DSF.

### 5.1 The Basic Design of DSF

Differently than traditional data forwarding techniques such as ETX and PRR $\times$ D, we allow *multiple potential forwarding nodes* at each hop. For a given sink, each node maintains a sequence of forwarding nodes sorted in the order of the wake-up time associated with them. To start sending a packet, a node looks up the time associated with the first node in the sequence, wakes up at that time interval, and tries to send the packet. If the transmission is successful, forwarding is done. Otherwise, the node fetches the next wake-up time from the sequence and tries to send the packet again. This retransmission process over a single hop continues until the sending node confirms that the packet has been successfully received by one of forwarding nodes or the sending node reaches the end of the sequence and drops the packet.

Formally, we define the sequence of forwarding nodes at a node  $e$  as:  $S_n^e$

**Definition 1 (Forwarding Sequence  $S_n^e$ ).**  $S_n^e$  is a sequence of  $n$  nodes that can forward packets from node  $e$  to the sink. This sequence is sorted based on the wake-up time of the nodes. Formally,  $S_n^e = (s_1^e, s_2^e, \dots, s_n^e)$ . Let  $t(s_i^e)$  be the wake-up time of node  $s_i^e$ , Forwarding sequence  $S_n^e$  satisfies  $t(e) < t(s_1^e) < t(s_2^e) < \dots < t(s_n^e)$ .

Figure 5 demonstrates the packet transmission process between one sender and  $n$  nodes in its forwarding sequence. In Figure 5, node  $A$  has a packet to be sent and its forwarding sequence is  $S_n^A = (B_1, B_2, \dots, B_n)$ . First, node  $A$  wakes up at time  $t_1$  and tries to transmit the packet to the node  $B_1$ . If the data delivery is successful, node  $A$  ends the current packet forwarding session. However, if the transmission fails, the node  $A$  wakes up again at time  $t_2$  and tries to send the packet to the node  $B_2$ . This retransmission process continues with node  $A$  repeatedly trying to send the packet to the node in the sequence  $S_n^A$ . If the transmission fails at the last node  $B_n$ , node  $A$  drops the packet.

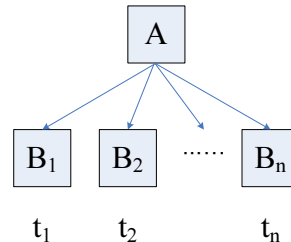


Fig. 5. Example of Dynamic Switching

From the above example, we can see that the major advantage of dynamic switching is the use of a forwarding sequence to reduce the time spent on transmitting a packet successfully at each hop rather than waiting for a particular forwarding node to wake up again after failure, as in such solutions as ETX, PRR $\times$ D and DESS.

### 5.2 The Modeling of EDR, EED, and EEC

Given a known forwarding sequence  $S_n^e$  at a node  $e$ , we can model the expected delivery ratio, the expected E2E delay and the expected energy consumption for the node. Here, for the sake of clarity, we describe a scenario with a single sink node, that can be extended easily for scenarios with multiple sink nodes.

Formally, these three metrics are defined as:

**Definition 2 (Expected Delivery Ratio  $EDR_e(S_n^e)$ ).** The expected delivery ratio at node  $e$  for a given forwarding sequence  $S_n^e$ , denoted by  $EDR_e(S_n^e)$ , is the expected packet delivery ratio from node  $e$  to the sink node (over multi-hop path).

**Definition 3 (Expected E2E Delay  $EED_e(S_n^e)$ ).** The expected E2E Delay at node  $e$  for a given forwarding sequence  $S_n^e$ , denoted by  $EED_e(S_n^e)$ , is the expected data delivery delay for the packets sent by node  $e$  and received by the sink node (over multi-hop path).

**Definition 4 (Expected Energy Consumption  $EEC_e(S_n^e)$ ).** The expected energy consumption at node  $e$  for a given forwarding sequence  $S_n^e$ , denoted by  $EEC_e(S_n^e)$ , is the expected energy consumption to deliver a packet from node  $e$  to the sink node (over multi-hop path). We note that since receiving (idle) energy is fixed for a given working schedule, we include only senders' transmission energy in EEC.

Our model for computing EDR, EED, and EEC values is distributed and can be executed at individual sensor nodes independently. At the sink node ( $b$ ), obviously, its forwarding sequence is empty, the  $EDR_b(0)$  value is 100% (i.e., no packet loss), while  $EED_b(0)$  and  $EEC_b(0)$  values are both zeros (i.e., no delay and no energy consumption). Consequently, we can obtain following initial equations:

$$EDR_b(0) = 1, EED_b(0) = 0, EEC_b(0) = 0 \quad (1)$$

Let the bi-directional link quality  $p_{ei}$  denotes the success ratio of a round-trip transmission (DATA and ACK) between node  $e$  and the  $i^{th}$  forwarder in  $S_n^e$ . The link quality  $p_{ei}$  can be influenced by multiple factors such as transmission power and the distance between a sender and a receiver. We note that in extremely low duty-cycle sensor networks, traffic congestion is rare and hence has little effect on link quality.

The overall probability  $P^e(i)$  that a packet transmission by node  $e$  is successful at the  $i^{th}$  forwarder (after  $i-1$  failures) can be represented as:

$$P^e(i) = \left[ \prod_{j=1}^{i-1} (1 - p_{ej}) \right] p_{ei} \quad (2)$$

**Expected Delivery Ratio (EDR):** Obviously, EDR value for node  $e$  is the sum of the product of the probability that the transmission is successful at a particular forwarder and its corresponding EDR value for all nodes in  $S_n^e$ . Assuming node  $e$  has  $n$  nodes in its forwarding sequence and letting  $EDR_i$  be the EDR value for the  $i^{th}$  forwarder ( $s_i^e$ ) in  $S_n^e$ , we have the following recursive equation for  $EDR_e(S_n^e)$ .

$$EDR_e(S_n^e) = \sum_{i=1}^n P^e(i) EDR_i \quad (3)$$

**Expected E2E Delay (EED):** The EED value of node  $e$  represents the expected delay for the packets sent by node  $e$  that reach the sink node  $b$ . Consequently, the probability that the packet transmission is successful at a certain forwarder is under the condition that the packet is delivered by one of the forwarders in  $S_n^e$ . Therefore, the conditional probability is  $P^e(i)' = \frac{P^e(i) EDR_i}{EDR_e(S_n^e)}$ . Letting  $EED_i$  be the EED value for the  $i^{th}$  forwarder in node  $e$ 's forwarding sequence and  $d_i$  be the delay for node  $e$  to wait node  $s_i^e$  in  $S_n^e$  to wake up, then  $EED_b(S_n^e)$  can be represented as:

$$EED_e(S_n^e) = \sum_{i=1}^n P^e(i)' (d_i + EED_i) \quad (4)$$

**Expected Energy Consumption(EEC):** Similarly, let  $EEC_i$  be the EEC value for the  $i^{th}$  forwarder in  $S_n^e$  and that the expected energy consumption for successful packet transmission at node  $s_i^e$  is the sum of  $EEC_i$  and  $i$  units of energy consumption (note that energy wasted in  $i-1$  failed transmission should be included as well). The probability that the retransmission of a packet reaches the  $i^{th}$  forwarder  $P^e(i)'$  at node  $e$  is conditional on the data delivery ratio  $EDR_e(S_n^e)$ . Therefore,  $P^e(i)' = \frac{P^e(i) EDR_i}{EDR_e(S_n^e)}$ , and we can formulate the  $EEC_e(S_n^e)$  as:

$$EEC_e(S_n^e) = \sum_{i=1}^n P^e(i)' (i + EEC_i) \quad (5)$$

The recursive calculation of EDR, EED and EEC can be implemented at individual nodes distributively. The main idea is to radiate known initial conditions ( $EDR_b(0) = 1$ ,  $EED_b(0) = 0$ ,  $EEC_b(0) = 0$ ) from the sink node, so that the process of calculating EDR, EED and EEC values propagates outward from the sink nodes to the rest of the network.

**Model Validation:** Figures 6, 7, and 8 show a pair-wise comparison of the calculated expectation values and the simulated E2E delay and energy consumption according to the radio model in [30], which considers the oscillation of radio links, for a randomly generated network graph in which each node has an average of six neighbors. In all three figures, the nodes are sorted with respect to their corresponding expectation values and we simulate the packet delivery process to the sink node

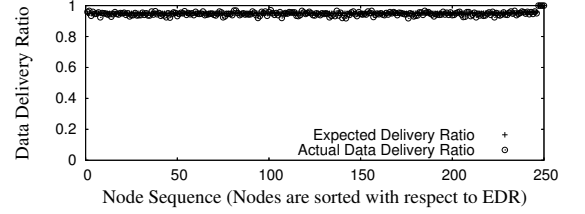


Fig. 6. Expected Delivery Ratio vs. Actual Delivery Ratio

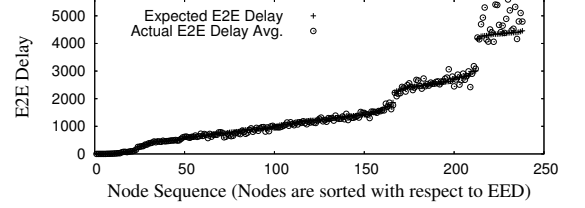


Fig. 7. Expected E2E Delay vs. Actual E2E Delay

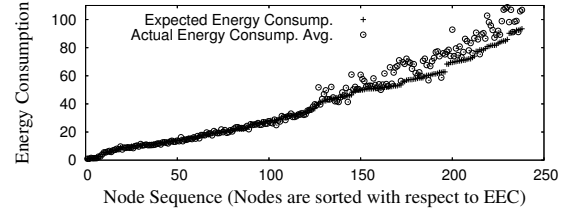


Fig. 8. Expected Transmission Energy Consumption vs. Actual Transmission Energy Consumption

1,000 times and report the average value. For the data delivery ratio, the average difference between expectation and simulation data is 0.96%, with a standard deviation of 0.74%. For the E2E delay, we observe an average 5.5% difference between expectation and simulation results, with a standard deviation of 4.8%. For the transmission energy consumption, the difference is an average of 6.1%, with a standard deviation of 4.7%. We also studied the deviation between expectation values and simulation results for many other randomly generated network graphs and obtained similar results. Figures 6, 7, and 8 confirm that our model for computing expected delivery ratio, E2E delay, and energy consumption accurately captures the behavior of the packet delivery process in DSF.

### 5.3 Optimizing the Forwarding Sequence

In the previous section, we described the model for calculating EDR, EED, and EEC for a *given* forwarding sequence. In this section, we will discuss how we can obtain a forwarding sequence that is *optimal* in terms of the maximum expected data delivery ratio, minimum expected E2E delay, or minimum expected transmission energy consumption at individual sensor nodes, respectively.

In practical network settings, especially in low duty-cycle sensor networks, a sender should not endlessly retransmit a packet because it would consume significant energy at the sending nodes. Therefore, we set the maximum time bound for a sender to retransmit a particular packet as  $T$ . Consequently, at node  $e$ , with known neighboring nodes and their corresponding working schedule  $\Gamma$ , we can have a full sequence of potential forwarding nodes that wake up before  $T$ .

Formally, let  $s_i^e$  be a next-hop node with wake-up time  $t(s_i^e)$ . Node  $e$ 's full sequence  $S_m^e$  under the bound  $T$  is:

$$S_m^e = (s_1^e, s_2^e, \dots, s_m^e) \text{ where } s \in S_m^e \iff t(e) \leq t(s) \leq t(e) + T.$$

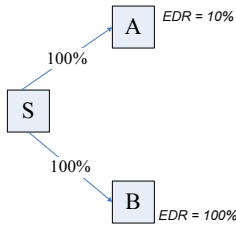
While noting that it is possible that multiple next-hop nodes may wake up at the same time, for now we first describe a simplified scenario where no next-hop nodes wake up at the same

time interval. The revised solution for accommodating multiple wake-up nodes is presented in Section 5.5.2.

### 5.3.1 Optimizing Expected Delivery Ratio (EDR)

Because the length of the potential forwarding sequence of a node is finitely subject to the maximum retransmission time interval  $T$ , under the reality of unreliable link quality among pairs of wireless sensor devices, packets sent by a source node may not all arrive the destination sink node. Therefore, when reliable transmission has the highest priority for a sensor network application, the optimization of the expected data delivery ratio (EDR) is critical.

Intuitively, in order to maximize the expected data delivery ratio at node  $e$ , we should try to send packets as long as one of the next-hop nodes is awake. The reasoning behind this is plausible, as since we want to maximize the expected data delivery ratio, we should take every opportunity to move the packet out of the sender. However, this intuition does not lead us to an optimal expected data delivery ratio, and Figure 9 presents a counterexample. In Figure 9, suppose the full forwarding sequence of the node  $S$  is  $S_2^S = (A, B)$ . If we choose both node  $A$  and  $B$  to form  $S$ 's forwarding sequence  $S_2^S$ , according to the Equation 3,  $EDR_e(S_2^S) = 10\%$ . In contrast, if we choose only node  $B$  to be included in  $S_2^S$ , the corresponding  $EDR_e(S_1^S) = 100\%$ . Therefore, in order to optimize the expected data delivery ratio at a node  $e$ , we shall select a subsequence from the full sequence  $S_m^e$ . By definition, a subsequence can be obtained by removing some of the elements from the original sequence without disturbing the relative positions of the remaining elements. As an example,  $(B, E, D, G)$  is a subsequence of  $(A, B, E, C, F, D, H, G)$ .



**Fig. 9. Example for Selecting a Subset of Nodes in Potential Forwarding Sequence**

To select an optimal subsequence  $S_{opt}^e$  from the full sequence  $S_m^e$ , we adopt a dynamic programming approach. Clearly, the last node  $s_m^e$  in  $S_m^e$  must be included in  $S_{opt}^e$ , since  $s_m^e$  provides the last chance for node  $e$  to retransmit before the packet is dropped. Starting from this optimal substructure, we can attempt to include nodes (one by one) from  $S_m^e$  **backwards** into  $S_{opt}^e$ . If the inclusion of a node from  $S_m^e$  into  $S_{opt}^e$  increases  $EDR_e(S_{opt}^e)$ , we then add this node into  $S_{opt}^e$  permanently. Otherwise, we discard the node and try to add the next node. The above forwarding sequence selection process continues until we reach the node  $s_1^e$  in the full sequence  $S_m^e$ . The optimality of this dynamic programming algorithm is based on the fact that the optimal  $EDR_e(S_{opt}^e)$  can be constructed efficiently from its optimal substructures. *The decisions made to include or exclude a later node in the forwarding sequence does not affect the optimality of decisions made to include or exclude earlier nodes and vice versa.* For each backward augmentation of the forwarding sequence, we guarantee the maximum data delivery ratio of the sequence between the newly augmented node and the last node. This forwarding sequence, then, serves as an optimal substructure for augmenting additional forwarders until the process reaches the first node in the sequence.

Let  $S_{opt}^e(k)$  denote the optimal forwarding subsequence in terms of maximizing EDR metric from the sequence  $S_k^e = (s_{m-k+1}^e, s_{m-k+2}^e, \dots, s_m^e)$ . Obviously,  $S_{opt}^e(m)$  is the optimal subsequence we want to obtain.

We have the following initial optimal substructure:

$$\begin{aligned} S_{opt}^e(0) &= () \\ S_{opt}^e(1) &= (s_m^e) \end{aligned} \quad (6)$$

Building upon the previous optimal substructure, when we attempt to include the next node  $s_j^e$  in  $S_m^e$  into  $S_{opt}^e(k-1)$  backwards, there are two possible outcomes:

- According to the model of  $EDR_e(S^e)$ , if the appending of node  $s_j^e$  to  $S_{opt}^e(k-1)$  increases the expected delivery ratio, we insert node  $s_j^e$  in front of the existing sequence  $S_{opt}^e(k-1)$  to obtain  $S_{opt}^e(k)$ .
- If the inclusion of node  $s_j^e$  into sequence  $S_{opt}^e(k-1)$  does not increase the data delivery ratio, the optimal forwarding sequence remains unchanged.

Formally, let  $s_j^e \oplus S$  denote inserting node  $s_j^e$  to the front of the sequence  $S$ , the corresponding recursive equation for  $S_{opt}^e(k)$  can be represented as:

$$S_{opt}^e(k) = \begin{cases} S_{opt}^e(k-1) & EDR_e(S_{opt}^e(k-1)) > EDR_e(s_j^e \oplus S_{opt}^e(k-1)) \\ s_j^e \oplus S_{opt}^e(k-1) & \text{Otherwise} \end{cases} \quad (7)$$

The complexity for this dynamic programming algorithm is  $O(mT)$ , where  $m$  is the density of next-hop nodes and  $T$  is the maximum per-hop delay allowed.

### 5.3.2 Optimizing Expected E2E Delay (EED)

In many sensor network applications, such as military surveillance [7], target tracking [20] and infrastructure monitoring [21], the delay for the source-to-sink communication is critical to the performance of the system.

We note that if there is no bound on the expected delivery ratio (EDR) for the forwarding sequence, the optimal forwarding sequence in terms of minimizing delay can be trivially achieved by including only a single node  $j$  which has the minimum  $(d_j + EED_j)$  value among all nodes in  $S_m^e$  (Equation 4). However, with such a quick-and-dirty solution, especially when the link quality between node  $e$  and node  $j$  is low, node  $e$  may suffer from an extremely low packet delivery ratio to the sink node and consequently may cause the whole network to be unavailable. Therefore, it is important to minimize the EED metric for the node  $e$  under the constraint that the EDR metric of the forwarding sequence is greater than a certain bound  $R$ . The bound  $R$  must be less or equal to the optimal EDR value that could be achieved at the node  $e$ .

Similarly to maximizing EDR, we also adopt a dynamic programming approach to select a subset of nodes in  $S_m^e$  backwards to optimize EED. But in contrast, the last node in  $S_m^e$  is no longer guaranteed to be the optimal initial optimal substructure, since the inclusion of the node may increase the expected E2E delay. Instead, to optimize EED, we need to try every node in the full sequence  $S_m^e$  as the last node in the optimal subsequence. For example, if we suppose  $S_m^e = (B, E, D, G)$ , we need to obtain optimal subsequences from  $(B, E, D, G)$ ,  $(B, E, D)$ ,  $(B, E)$ , and  $(B)$  with  $G, D, E, B$  chosen, respectively.

Suppose node  $s_{last}^e$  is selected as the last node and  $S_{opt}^e(last, k)$  represents the optimal forwarding subsequence in terms of EED chosen from the sequence  $S_k^e(last) = (s_{last-k+1}^e, s_{last-k+2}^e, \dots, s_{last}^e)$ , where  $k \leq last$  and  $last \in \{1, 2, \dots, m\}$ .

For each last node, we have the following initial optimal substructure for  $S_{opt}^e(last, k)$  in terms of minimal EED:

$$\begin{aligned} S_{opt}^e(last, 0) &= () \\ S_{opt}^e(last, 1) &= (s_{last}^e) \end{aligned} \quad (8)$$

Similar to the recursive equations for maximizing EDR, for each node  $s_j^e$  in  $S_k^e(last)$ , the optimized forwarding sequence for EED is:

$$S_{opt}^e(last, k) = \begin{cases} S_{opt}^e(last, k-1) & EED_e(S_{opt}^e(last, k-1)) < EED_e(s_j^e \oplus S_{opt}^e(last, k-1)) \\ s_j^e \oplus S_{opt}^e(last, k-1) & \text{Otherwise} \end{cases} \quad (9)$$

After having all  $S_{opt}^e(last, last)$  where  $last \in \{1, 2, \dots, m\}$ , we chose the forwarding sequence with the minimal EED value, under the constraint that  $EDR \geq R$ . The complexity for optimizing EED is  $O(Tm^2)$ .

### 5.3.3 Reducing Expected Energy Consumption (EEC)

For applications such as scientific exploration, the difficulty of entering the sensing field and the corresponding high cost of system deployment calls for the longevity of the system, making energy conservation the highest priority for the system design. Similarly to the optimization of EED, if we do not have a bound on the expected delivery ratio, the optimal forwarding sequence for the minimal EEC would include only one node with the smallest EEC value in  $S_m^e$  and may also experience an extremely low source-to-sink data delivery ratio. Therefore, in this section we reduce EEC under the constraint that EDR of the forwarding sequence is above threshold  $R$ .

Unlike optimizing EED, in Equation 5, where  $i$  represents the index of forwarding node in the forwarding sequence, the  $i$  value changes for each already selected forwarding node as we backwardly add early nodes. In other words, the decisions made to include or exclude an **early** node in the forwarding sequence **does affect** the expected energy of **later** nodes. Lacking an optimal substructure, we can only choose either an exhaustive search (in the case that a forwarding sequence is small) or a greedy heuristic algorithm. We found that the greedy case for EEC is actually very effective. The main idea of the greedy algorithm is that starting with an empty optimal forwarding sequence, we continuously add the unselected node in  $S_m^e$  that results in a minimal increase in EEC into the optimal forwarding sequence until the EDR of the optimal forwarding sequence reaches  $R$ . Empirical results indicate that the greedy algorithm obtains optimal results 85% of the time and the suboptimal results are within 5% of the optimal values.

### 5.3.4 The Impact of EDR Constraints on Optimality

We note that the EDR bound  $R$  imposes a non-convex constraint on the EED and EEC optimization problems. To optimize the forwarding sequence efficiently, the optimization processes described in Sections 5.3.2 and 5.3.3 first identify an optimal forwarding sequence under unconstrained search space. If the resulting sequence satisfies the EDR bound  $R$ , it is also an optimal solution to the original constrained problem. However, it is also possible that the resulting sequence violates the constraint especially when the EDR bound  $R$  is very high. In this case, we select the optimal EDR forwarding sequence from  $S_i^e$ , where  $i$  is the minimal value leads to  $EDR_e(S_i^e) \geq R$  to satisfy the constraints (instead of achieving optimal EED or EEC).

Obviously, if the percentage of constraint violation is high, our solution is not effective. To evaluate this issue, we studied the impact of a high EDR bound on the optimality of our

solution. Figure 10 shows the percentage of optimality under different EDR bounds. Clearly, our solution is very effective in identifying optimal solutions. For example, even with a 99% delivery ratio, 98.4% solutions are optimal.

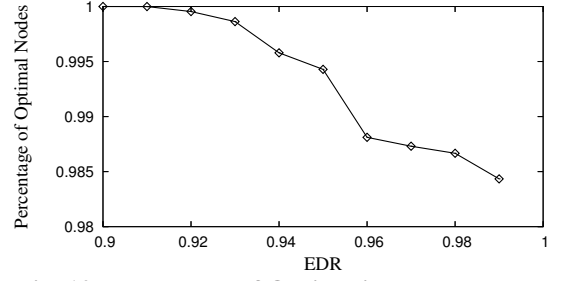


Fig. 10. Percentage of Optimality vs. EDR

### 5.3.5 Special Cases: ETX and DESS

We note that when nodes in the network are always active with no sleeping schedules, our EDR, EED, and EEC metrics and corresponding forwarding sequences default into those of the ETX solution. In addition, when all radio links among neighboring nodes are perfect, EDR, EED, and EEC default into those in the DESS solution. To a certain degree, we argue that EDR, EED, and EEC metrics are more generic data forwarding metrics, considering both link quality and sleep latency. In other words, ETX and DESS are two special cases of a more generic DSF solution. To validate this empirically, we will show such a convergence in the evaluation section later.

## 5.4 Distributed Implementation of DSF

In Sections 5.2 and 5.3, we discussed the model for computing EDR, EED, and EEC, and the algorithms for optimizing the forwarding sequence at a node in terms of one of those metrics. In this section, we will describe the detailed protocol implementation at each individual sensor device to compute its expectation values and decide the forwarding sequence upon the optimizing metric.

As mentioned in Section 5.2, both the EED and EEC values of the sink node are zero while the EDR value is one, since no further in-network communication is necessary once a packet has arrived at the sink node. Therefore, at the initial time of sensor network deployment, the sink node has already had its expectation values known and is the only node that has its optimized metric value ready. The distributed algorithm for each node other than the sink node is shown in Algorithm 1. Similarly to the distributed Bellman-Ford algorithm (but using only sinks as destinations), the initialization phase starts with the sink node broadcasting its EDR, EED, and EEC values. The nodes receive the broadcasted message [Line 1], start to calculate their own expectation values according to the model and optimization process for a particular metric as discussed in the previous two sections [Lines 2-4], and then broadcast their own expectation values if the change exceeds a certain value. This process [Lines 1-7] continues at a node until it receives no information about updated expectation values from all its neighbors. At this time, its optimizing metric also converges to its minimum at every node in the network. The convergence speed of Algorithm 1 is very fast. For example, all nodes converge within 2 seconds in our test-bed, as shown in the evaluation section.

According to Algorithm 1, we can see that the protocol implementation can easily be applied to the scenario when there are multiple sink nodes and the resulting expectation values at each node are the expectations to its nearest sink node in terms of the optimizing metric.

---

**Algorithm 1** Complete protocol implementation at a node  $e$ 

---

**Input:** Received expectation values from a neighboring node**Input:** The current neighbor table  $NT$  of the node  $e$ **Output:** The current optimal forwarding sequence

- 1: Update the received expectation values in  $NT$
  - 2: **for** Each active state in node  $e$ 's working schedule  $\omega_S$  **do**
  - 3:   Run either EDR, EED or EEC optimization process to obtain the optimal forwarding sequence in terms of the optimizing metric
  - 4: **end for**
  - 5: **if** Any node  $e$ 's optimizing metric changes noticeably **then**
  - 6:   Broadcast updated expectation values
  - 7: **end if**
- 

## 5.5 Design Issues and Optimization

This section completes the description of our design by examining several design issues concerning DSF under CSMA networks, forwarding sequence optimization with simultaneous wake-up neighboring nodes, opportunistic looping, batch transmission and changing link qualities.

### 5.5.1 DSF under CSMA Networks

For the sake of clarity, we here introduce DSF in a synchronized mode. Clearly, the operations of DSF depend on neither time slots nor global time synchronization. The sufficient condition for DSF is that every node knows the *wake-up time* and the *link quality* of neighboring forwarders. To understand neighbors' wake-up times, local synchronization is needed, which can be achieved using a MAC-layer time-stamping technique [17], which achieves  $2.24\mu\text{s}$  accuracy with an overhead of a few bytes of packets exchange among neighboring nodes for every 5 minutes. Since the expected  $\tau$  value is set around  $2000\mu\text{s}$  to  $20,000\mu\text{s}$  (according to the data rates of different radio chips), an accuracy of  $2.24\mu\text{s}$  is by far sufficient.

Unlike TDMA networks, DSF does not require a node to start a transmission at the beginning of a time slot  $\tau$ . As long as a node knows the wake-up time of its neighboring nodes, it can decide its optimal forwarding sequence. In terms of performance, CSMA is more favorable in low duty-cycle networks where network traffic is very low, since a node in TDMA networks has to wait for its turn to transmit and becomes inefficient in such scenarios. Although DSF works in both TDMA and CSMA networks, CSMA is a better choice.

### 5.5.2 Simultaneous Wake-Up

In Section 5.3, for the sake of simplicity we optimized the forwarding sequence under the assumption that at each node, no multiple neighboring nodes would wake up at the same time interval. In this section, we complete the forwarding sequence optimization process to deal with multiple neighbors waking up at the same time.

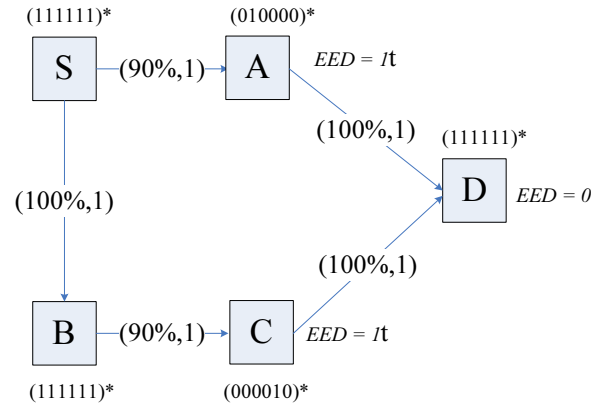
In contrast to Section 5.3, when we attempt to include a node  $j$  from  $S_m^e$  into  $S_{opt}^e(k-1)$ , instead of only inserting the node  $j$  to the front of  $S_{opt}^e(k-1)$ , there are two possible actions:

- If the wake-up time associated with node  $j$  is different from the wake-up time associated with the first node in sequence  $S_{opt}^e(k-1)$ , we just append node  $j$  to the front of  $S_{opt}^e(k-1)$  as described in Section 5.3.
- If the wake-up time associated with node  $j$  is the same as the wake-up time associated with the first node in sequence  $S_{opt}^e(k-1)$ , we **replace** the first node in  $S_{opt}^e(k-1)$  with the node  $j$ .

After properly including the node  $j$  into sequence  $S_{opt}^e(k-1)$ , we can follow the same procedure presented in Section 5.3, comparing the optimizing metric for new sequence with  $S_{opt}^e(k-1)$  and deciding if node  $j$  should be within  $S_{opt}^e(k)$  or not.

### 5.5.3 Opportunistic Looping

We note that the forwarding sequence optimization for EED (Section 5.3) does not require that a potential forwarder has a smaller EED value than the sender (i.e., a forwarder could be chosen even if it has a large EED value as long as the participation of this forwarder can reduce the sender's EED value). This relaxation allows a packet to travel through temporary loops in the path to its destination to potentially reduce the end-to-end delay! We term this interesting, albeit counterintuitive, phenomena *opportunistic looping*. Figure 11 shows an example of opportunistic looping, in which node  $S$  sends a packet to node  $D$ . Each node is assigned a working schedule and each edge is assigned a tuple  $(p, d)$ , in which  $p$  is round-trip success ratio and  $d$  is waiting time. We also tag each node with an EED value, calculated by Equation 4.



**Fig. 11. Opportunistic Looping**

As shown in Figure 11, node  $S$  can deliver a packet to node  $A$  with a 90% success ratio. If successful, the packet arrives at node  $A$  and then arrives at node  $D$  with total delay of two. In case of a delivery failure between  $S$  and  $A$ , node  $S$  has two options. In the first option, node  $S$  does nothing, but waits for five units of time before it tries  $A$  again. The other option is to forward this packet to node  $B$  in one unit of time, and then node  $B$  tries to deliver the packet to node  $D$  through node  $C$ . In the later case, even if node  $B$  fails to deliver the packet to node  $C$ , node  $B$  can loop this packet back to  $S$  before node  $A$  becomes available again. Obviously, the loop  $S \rightarrow B \rightarrow S$  is opportunistic in nature. In other words, compared with idle waiting, this loop can potentially reduce delay if transmission from node  $B$  to node  $C$  is successful.

In the results of our simulation study, we find that the behavior of selecting forwarders with larger EED values is not rare. Figure 12 shows the forwarders' EED values versus the senders' EED values in a random 2000-node network. There we can see that a fairly large number of nodes have their forwarders with larger EED values. Based on simulation results from 100 randomly generated networks, we observe that nearly one-tenth of the nodes have forwarders with larger EED values.

We also investigated opportunistic looping from the perspective of the E2E path. The CDF curve in Figure 13 shows the probability that a packet traveling through multiple loops decays exponentially. For example, 84% of packets are delivered

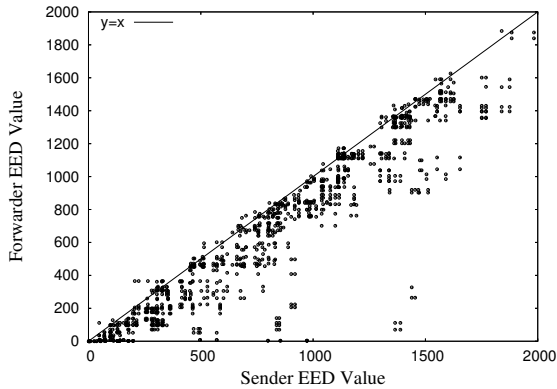


Fig. 12. Forwarder EED Values vs. Sender EED Values

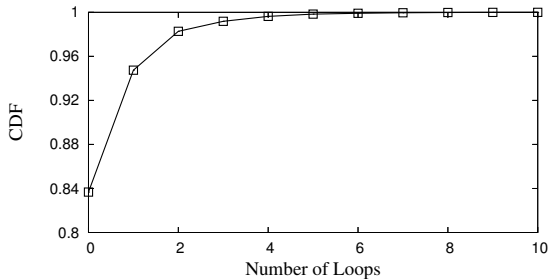


Fig. 13. Number of Loops in E2E Paths

without loops, while only 0.4% of packets are delivered through 4 hops. Due to space constraints, we omit the detailed analysis on this *loop decay* phenomena.

Opportunistic looping can be optionally disabled by requiring that forwarders have a smaller EED value during the forwarding selection process. Figure 14 compares the selection process with and without opportunistic looping. Clearly, opportunistic looping can noticeably reduce the end-to-end delay. For example, with an average of 4 neighboring nodes, the E2E delay for with and without loops is 3109 and 3745, respectively.

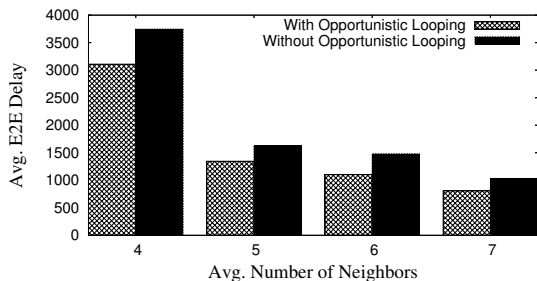


Fig. 14. Delay with and without Opportunistic Looping

On the other hand, it should be emphasized that although opportunistic looping reduces end-to-end delay, it introduces additional communication overhead. Figure 15 compares the average EEC with and without opportunistic looping. By comparing Figures 14 and 15, a system designer can decide whether to use opportunistic looping based on the tradeoffs between the delay requirement and the energy budget.

#### 5.5.4 Batch Transmission

While describing the network model in Section 4, we assume that during time  $\tau$ , at most one packet can be transmitted between a sender and a receiver. This is true if sensors are equipped with slow radio chips. For sensor nodes with fast radio (e.g., MicaZ with 250kbps CC2420 radio [2]), however,

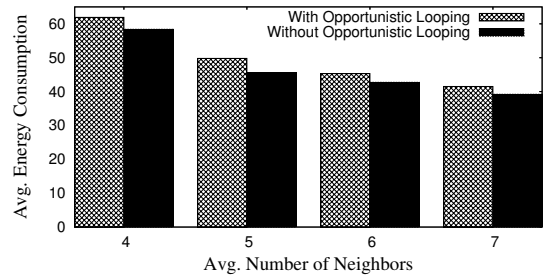


Fig. 15. Energy with and without Opportunistic Looping

it is possible to transmit multiple packets with time  $\tau$ . In addition, with a relatively large  $\tau$  value (e.g., 100 milliseconds), the accuracy requirement on time synchronization (in coverage scheduling) can be reduced, since such state-of-the-art solutions as FTSP [17] can easily achieve sub-millisecond accuracy.

Besides providing relief in the time synchronization constraint, the relatively large  $\tau$  also helps reduce the E2E delay of the source-to-sink communication. Instead of attempting to transmit only one packet during one  $\tau$ , as discussed in Section 5, the sending node now could take advantage of a longer duration of  $\tau$  and repeatedly transmit the packet until the packet has been successfully received or the duration of  $\tau$  ends. We need only a simple modification of the basic design to support batch transmission. Specifically, we rewrite  $p_{ei}$ , the bi-directional link quality between two nodes in Section 5.2, as  $p'_{ei} = 1 - (1 - p_{ei})^m$ , where  $m$  is the maximum number of re-transmissions allowed during one  $\tau$ . Essentially, the new  $p'_{ei}$  value represents the probability that the receiver received the packet by  $m$  transmissions.

We also note that although the increasing  $\tau$  improves the packet delivery rate over that of one link, the increased  $\tau$  also adds more delay to the network. Therefore, a suitable  $m$  and consequently  $\tau$  could be derived to further minimize the source-to-sink delay.

#### 5.5.5 Accommodating Link Quality Change

Clearly link quality could change over a long period of time [14], and therefore the DSF design must be able to accommodate such changes. As described in Section 5.4, the implementation of DSF allows each node to re-evaluate its own EDR, EED, and EEC values with updated information from neighboring nodes. The continuous updates might not be a suitable solution if energy consumption is the paramount concern. To reduce the energy consumption and avoid oscillation in forwarding decisions, we permit information exchange only when changes in link quality exceed a certain threshold. This would lead to temporary suboptimal metrics, since it is possible that the outdated link quality information could be used for optimization.

## 6 Implementation and Evaluation

We have implemented a complete version of the DSF forwarding scheme on the TinyOS/Mote platform in nesC [5] with 20 MicaZ motes. To compare performance, we also implemented ETX [3] on the motes. The major components of DSF implementation include neighbor discovery, link quality measurement, the forwarding sequence optimization algorithms discussed in Section 5.3, and data forwarding with an optimized forwarding sequence.

- **Neighbor Discovery:** The neighbor discovery component at each individual node manages both the broadcasting of the working schedule and the maintenance of the neighbor table. To announce the existence of a node, the neighbor discovery component broadcasts the node ID and working

schedule information with a configurable parameter that decides the number of retransmissions. In current prototype implementation, we set the number of retransmissions 10. While receiving a broadcasted working schedule announcement, the neighbor discovery component checks whether the source of the packet has been in its neighbor table. If the source does not exist in the neighbor table, the neighbor discovery component appends the source and corresponding working schedule into its neighbor table. Otherwise, the neighbor discovery component would just ignore the received broadcasting packet and does nothing. In short, the neighbor discovery component attempts to keep track of the schedules of all neighbors.

- Link Quality Measurement:** To measure the pairwise link quality between a node and its neighbors, the link quality measurement component at each individual node sends a number of packets to each of its neighbors and utilizes the link layer acknowledgement from B-MAC [18] to calculate the pairwise link quality. Depending on the desired accuracy of measurement, the link quality measurement component provides a configurable parameter to set the number of message transmissions between pairs of nodes. In order to minimize the impact of interference, in our current implementation we serialize the link quality measurement process among nodes in the network; in other words, nodes in the network measure their link qualities in sequence. When the data forwarding component forwards packets, the link quality measurement component updates link quality information accordingly and triggers forwarding sequence optimization if necessary.
- Forwarding Sequence Optimization:** Currently the heart of DSF design, the forwarding sequence optimization component implements EDR and EED optimizations faithfully according to the description in Section 5.3. The two optimizations are necessary to create optimal forwarding sequences of EED for comparison with ETX in the following subsection. In the future, we also plan to complete the forwarding sequence optimization component with inclusion of EEC implementation.
- Data Forwarding:** The data forwarding component is shared by both DSF and ETX. Whenever a node has a packet ready to send, according to the specified forwarding scheme, the data forwarding component at a node attempts a single packet transmission to the designated forwarding node when it is in the active state.

We use FTSP [17] for the purpose of time synchronization among motes and Deluge [8] for the purpose of wireless reprogramming. The compiled image occupies 27,398 bytes of code memory and 1,137 bytes of data memory.

During the experiment, we *randomly* placed 20 MicaZ motes along the hallway of our office building and tuned the transmission power to ensure the multi-hop communication between the source node and the sink node. In this experiment, immediately after deployment, all nodes are in the initialization phase, with all nodes keep awake. Each node randomly generates a periodic 1% working schedule, represented as a regular binary string as described in Section 4 with switching rate  $\tau$  sets to be 20ms (a relatively large  $\tau$  value here to reduce the impact of the time synchronization module). After generating working schedules, nodes start to broadcast their own working schedules and measure the pairwise link quality for their neighboring nodes. With known working schedules of neighboring nodes

and corresponding link qualities, nodes in the network start to carry out the Algorithm 1, calculate the specified metrics and decide the forwarding sequence for DSF. ETX shares the identical information as DSF and builds its own forwarding metrics accordingly. After Algorithm 1 and ETX converge at a node, the node begins to execute its working schedule with a timer-driven FA logic, turns off the radio at dormant bits, and enables the radio at the active bits. At the data forwarding phase, the node furthest away from the sink is selected as the source node and sends the packets using DSF and ETX alternatively so as to minimize the impact of temporal link qualities.

This testbed experiment was repeated multiple times with different node placement and working schedules. The results show the similar trend that resulted in all the experiments, and we report one collected dataset from the experiments in the following subsection.

### 6.1 Performance Comparison

In this section, we describe and compare the empirical E2E delay and energy consumption for DSF and ETX. In the experiment, the source node sends 100 packets to the sink node with DSF of optimal EED and ETX forwarding scheme, respectively.

Figure 16 shows the E2E data delivery delay for DSF and ETX. The packets in the figure are sorted according to their E2E delay, making it clear that ETX experiences heavy penalties when its single-hop transmission has failed, since it has to wait for the fixed forwarding node to wake up again. In contrast, when DSF encounters a single-hop transmission failure, its capability to dynamically switch the forwarding node significantly reduces the E2E delay. For instance, among 100 sent packets, the maximal E2E data delivery delays for DSF and ETX are 4317ms and 15426ms respectively, while the average delays are 849ms and 3942ms.

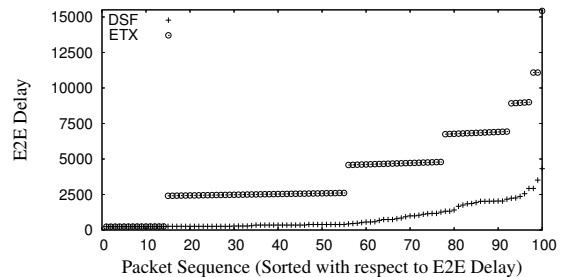


Fig. 16. E2E Data Delivery Delay

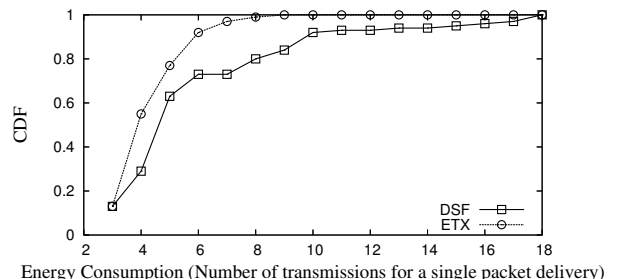


Fig. 17. Energy Consumption

In addition to the E2E delay, we are also interested in the energy consumption of the two comparing protocols. Figure 17 demonstrates the energy consumption (number of transmissions for a single packet delivery) for DSF and ETX. From the figure, we can see that ETX incurs a smaller number of transmissions than DSF. For example, all of the packet deliveries for ETX finished with a maximum of 9 transmissions, while about 84% of

the packets for DSF arrived at the sink node within 9 transmissions. However, the DSF shows a better delay-energy efficiency than ETX. With the same 9 transmissions, the delay for DSF and ETX is 1785ms and 15426ms, respectively.

## 6.2 System Insights

In this section, we investigate the internal state of each sensor node and reveal the corresponding statistics for DSF.

Figure 18 demonstrates the greater diversity of forwarder link qualities for DSF over those for ETX. While almost all ETX forwarders have link qualities above 50%, the distribution of forwarder link qualities for DSF is roughly uniform and ranges from 3% to 97%. Such diversity in forwarder link qualities for DSF, along with its smaller E2E delay, leads us to conclude that unreliable links are also helpful in reducing E2E delays in low duty-cycle sensor networks.

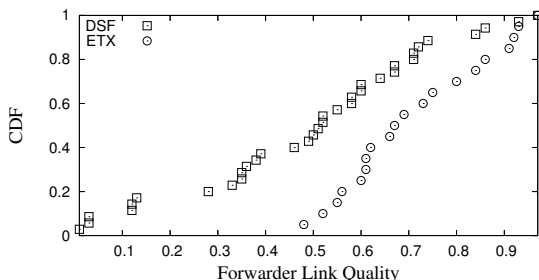


Fig. 18. Diversity in Forwarder Link Qualities

Figure 19 shows the relationship between the number of nodes in the forwarding sequence and the number of available neighboring nodes for each sensor node in the experiment. The node sequence is ordered by the node's distance to the sink node. From this figure, we can see that most nodes have more than one node in their forwarding sequence. We also observe that generally, as the node's distance to the sink node increases, the number of forwarding nodes in the forwarding sequence also increases, since in order to maintain a certain data delivery ratio, the more distant nodes normally need to select more of their neighboring nodes. For example, the average number of forwarding nodes for the first 10 nodes is 1.8 nodes, while the value for the last 10 nodes is 3.8 nodes.

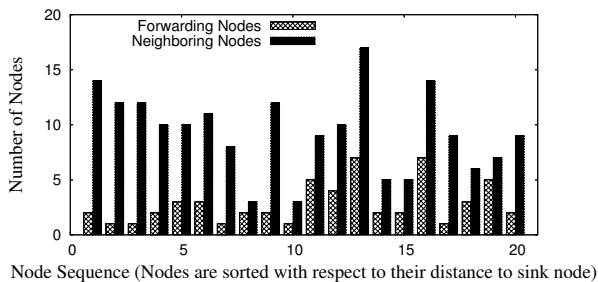


Fig. 19. Number of Forwarding Nodes vs. Number of Neighboring Nodes

In addition to studying the distribution of the forwarding nodes, we also investigated how fast each node converges to its optimal forwarding sequence. To track the convergence speed of the DSF, we recorded the number of times that each node executed its forwarding sequence optimization procedure, as shown in Figure 20. There we can see that the forwarding sequence optimization process at all nodes converges within 18 executions of the optimization procedure. Furthermore, the number of executions of the optimization procedure at individual nodes is pro-

portional to the number of neighboring nodes. This observation is also consistent with our complexity analysis for forwarding sequence optimization procedures.

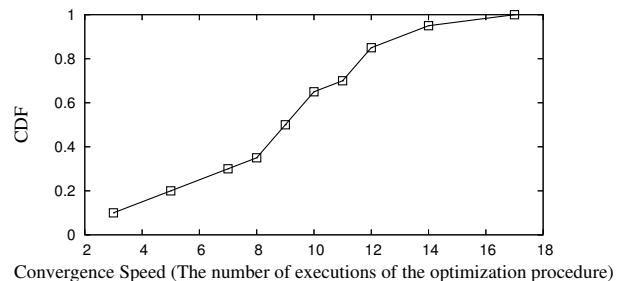


Fig. 20. DSF Convergence Speed

## 7 Large-Scale Simulation

The results of the following system evaluation indicates that our proposed approaches can be efficiently implemented on resource-constrained sensor nodes and demonstrates their effectiveness in improving source-to-sink wireless communication between sensing nodes and sink. However, this evaluation was restricted to a limited design space. In order to understand the performance of the proposed scheme under numerous network settings, in this section, we provide simulation results with 250 nodes. We compared the performance of DSF with following state-of-the-art solutions:

- ETX [3] by Douglas S. J. De Couto et al. in Mobicom'03.
- PRR×D [19] by Karim Seada et al. in SenSys'04.
- DESS [16] by Gang Lu et al. in INFOCOM'05.

### 7.1 Simulation Setup

In the simulation, we deployed 250 sensor nodes randomly in a 150m×150m square field. A sink was positioned in the center of the deployment field, and each sensor node sent its packet to the sink over multiple hops. The radio model was implemented according to [30], which considers the oscillation nature of the radio links and has several adjustable parameters. Except as otherwise specified, we set these parameters strictly according to the CC2420 radio hardware specification [2]. These parameters accurately reflect the performance of MicaZ motes in that they have the same modulation method, encoding method, frame length and path loss exponent.

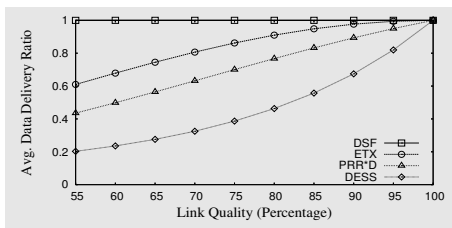
In all experiments, we set the sender retransmission time bound  $T$  equals  $200\tau$ , which is also the length of the node working schedule. Each experiment was repeated 30 times with different random seeds, node deployments, and node working schedules. Data collected at each node was obtained by averaging 1000 source-to-sink communications. The 95% confidence intervals are within 1~10% of the means.

### 7.2 Performance Evaluation

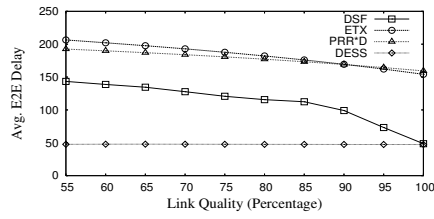
This section compares the data delivery ratio, E2E delay and energy consumption per delivered packet of source-to-sink communications among DSF, ETX, PRR×D, and DESS under different link qualities and duty cycles.

For the simulation of different link qualities, we first used CC2420 radio specifications to obtain the neighbor table for each sensor node, then set the pairwise link quality according to the simulation configurations.

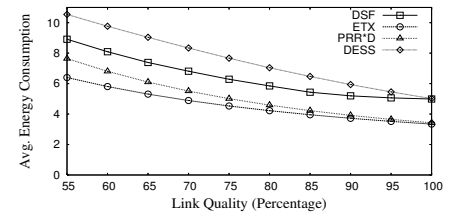
In following three subsections, evaluation figures for optimizing metrics are shaded to highlight their performances.



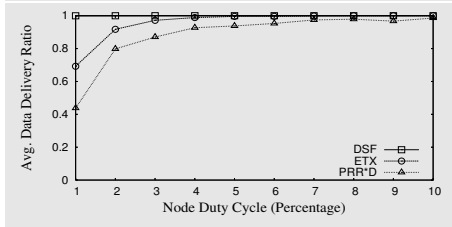
(a) Delivery Ratio vs. Link Quality



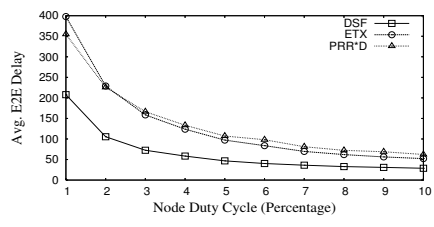
(b) E2E Delay vs. Link Quality



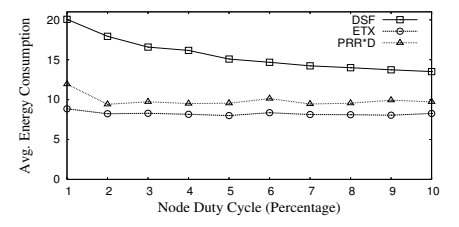
(c) Energy vs. Link Quality



(d) Delivery Ratio vs. Duty Cycle

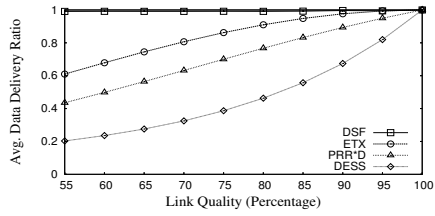


(e) E2E Delay vs. Duty Cycle

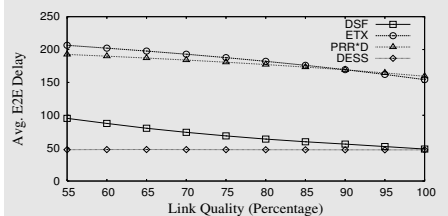


(f) Energy vs. Duty Cycle

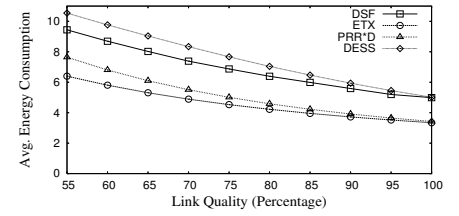
**Fig. 21. Optimizing Expected Delivery Ratio (EDR)**



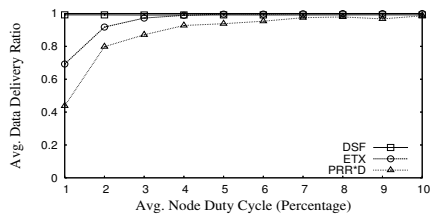
(a) Delivery Ratio vs. Link Quality



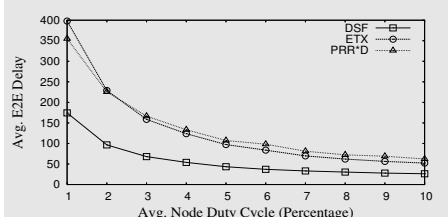
(b) E2E Delay vs. Link Quality



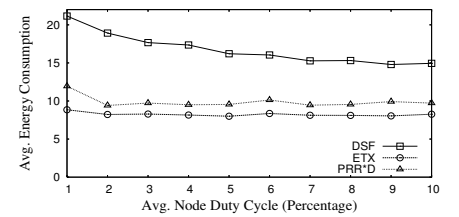
(c) Energy vs. Link Quality



(d) Delivery Ratio vs. Duty Cycle

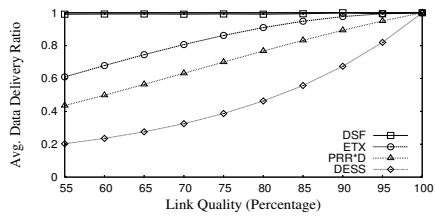


(e) E2E Delay vs. Duty Cycle

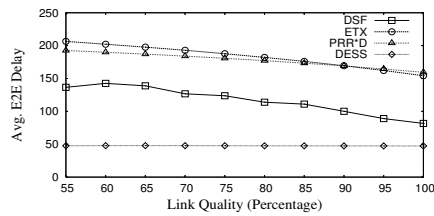


(f) Energy vs. Duty Cycle

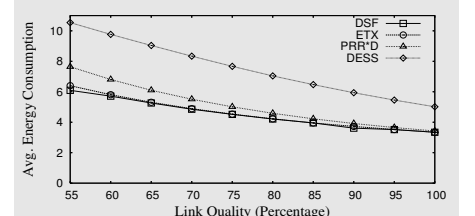
**Fig. 22. Optimizing Expected E2E Delay (EED)**



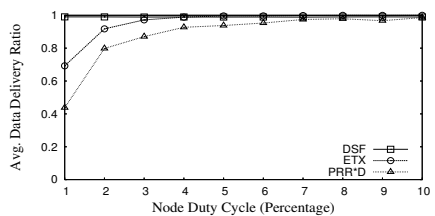
(a) Delivery Ratio vs. Link Quality



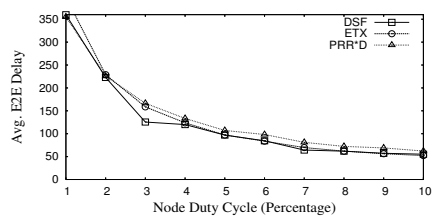
(b) E2E Delay vs. Link Quality



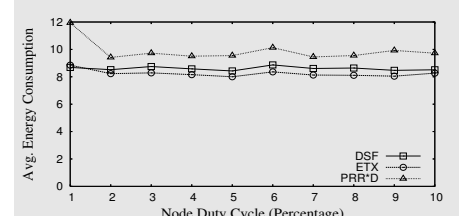
(c) Energy vs. Link Quality



(d) Delivery Ratio vs. Duty Cycle



(e) E2E Delay vs. Duty Cycle



(f) Energy vs. Duty Cycle

**Fig. 23. Reducing Expected Energy Consumption (EEC)**

### 7.2.1 Optimizing Expected Delivery Ratio

In this section, we examine the performance difference among DSF with optimal EDR, ETX, PRR×D, and DESS under different link qualities and duty cycles.

**Varying Link Qualities:** Figure 21(a) shows the data delivery ratio among the four compared schemes. The figure clearly shows that under the low link qualities, ETX, PRR×D and DESS can deliver only a very small portion of packets, while DSF with optimal EDR is able to deliver most of the packets to the sink node. For example, when the link quality is 55%, DSF delivers 99.9% of packets, while ETX, PRR×D, and DESS deliver only 61.0%, 43.5% and 20.3% of packets, respectively. Therefore, when the data delivery ratio is the primary design goal of a sensor network application, DSF would be a good choice for the system.

Figure 21(b) and Figure 21(c) show the corresponding E2E delay and energy consumption for four schemes. From Figure 21(b), we observe that DESS has the smallest and most constant E2E delay at all link qualities because at each hop, DESS would attempt to transmit its packet to the forwarder only once on the shortest delay path during one round of the node working schedule. Therefore, all the packets for DESS that reach the sink node are those for which every single-hop transmission is successful with one single attempt, and that consequently represent the minimal possible delivery delay, which is a constant value. At the same time, however, DESS experiences the largest packet loss among the four compared schemes. DSF, on the contrary, has the largest data delivery ratio though a smaller E2E delay than ETX and PRR×D. However, DSF's high data delivery ratio also incurs energy penalties.

From Figure 21(c), we can see that DSF has a slightly higher energy consumption per delivered packet than ETX and PRR×D since it attempts more transmissions and delivers more packets than these schemes. DESS ignores the link quality completely, has a very low data delivery ratio, and wastes much energy on transmitting packets that do not arrive at the sink node, therefore having the largest energy consumption per delivered packet. For instance, at a link quality of 55%, the per-delivered packet energy consumption for DSF, ETX, PRR×D, and DESS is 8.91, 6.40, 7.64 and 10.54, respectively.

**Varying Duty Cycles:** Figure 21(d) reports the data delivery ratio under different node duty cycles. It shows that under all node duty cycles, DSF with optimal EDR has a higher data delivery ratio than ETX and PRR×D. As the node duty cycle increases, the data delivery ratio for all schemes increases as well. For example, the delivery ratio for DSF, ETX, and PRR×D increases from 99.9%, 69.3%, and 43.8% to 100%, 99.9%, and 98.6%, respectively, when duty cycle increases from 1% to 10%. Figure 21(e) shows that the corresponding E2E delay for DSF is smaller than the other two baseline schemes, even with a higher data delivery ratio. Figure 21(f) shows again that the high data delivery ratio of DSF results in higher energy consumption.

### 7.2.2 Optimizing Expected E2E Delay

In this section, we examine the performance difference among DSF with optimal EED, ETX, PRR×D, and DESS under different link qualities and duty cycles. For optimal EED at each node, we set the data delivery ratio bound as 99%.

**Varying Link Qualities:** Figure 22(b) shows the end-to-end delay for four forwarding schemes under different link qualities. At link qualities less than 100%, the E2E delay is larger for DSF than for DESS, for the reason mentioned in the previous

subsection. Meanwhile, the E2E delay for DSF is much smaller than for ETX and PRR×D. For example, at a link quality of 90%, the E2E delay for DSF, ETX, and PRR×D is 56.2, 169.4, and 178.3, respectively. When the link quality reaches 100%, the results for DSF with optimal EED converges with those of DESS. In Figure 22(c), we can see that the energy consumption for DSF is still higher than that for ETX and PRR×D. However, we also observe that DSF is more delay-energy efficient than the other schemes. For example, when the link quality is 80%, the per-energy delay for DSF, ETX, PRR×D, and DESS is 10.07, 47.41, 50.36, and 14.61, respectively.

**Varying Duty Cycles:** Figure 22(e) shows the end-to-end communication delay under different node duty cycles. There we can see that DSF has a smaller delay than the baseline schemes under all duty cycles while retaining a high data delivery ratio (Figure 22(d)). The overall energy consumption for DSF is still higher than that for the other schemes. However, as mentioned before, the per-energy delay for DSF is much smaller than for ETX and PRR×D. For example, at a duty cycle of 5%, the per-energy delay for DSF, ETX, and PRR×D is 3.82, 14.08, and 16.33, respectively.

### 7.2.3 Reducing Expected Energy Consumption

This section presents the performance differences among DSF with optimal EEC, ETX, PRR×D, and DESS under different link qualities and duty cycles. For an optimal EEC at each node, we set the data delivery ratio bound as 99%.

**Varying Link Qualities:** In Figure 23(c), energy consumption for DSF approaches the ETX at all link qualities while maintaining high data delivery ratio. For example, when link quality is 70%, the energy consumption for DSF, ETX, PRR×D, and DESS is 4.21, 4.21, 4.59, and 7.04, respectively. When link quality approaches 100%, DSF converges to the ETX in terms of energy consumption. In addition, with equivalent energy consumption, the E2E delay for DSF is smaller than for ETX and PRR×D. At a link quality of 80%, the E2E delay for DSF, ETX, and PRR×D is 113.95, 182.13, and 197.18, respectively. Interestingly, we notice that under optimal EEC, DSF does not converge to the DESS when link quality reaches 100%, because when optimizing EEC, DSF would seek the delivery path with the minimum number of transmissions instead of the minimum E2E Delay.

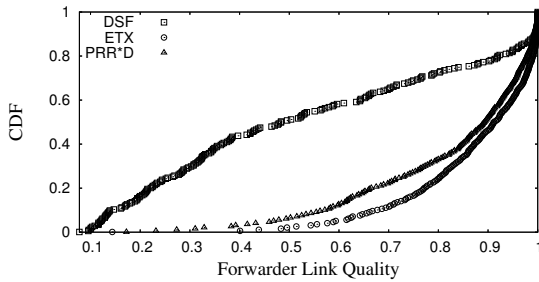
**Varying Duty Cycles:** Figure 23(f) shows the energy consumption under different node duty cycles. From the figure, we observe that the energy consumption for DSF approaches that of ETX and is better than that of PRR×D. With a higher data delivery ratio (Figure 23(d)) and comparable energy consumption, the end-to-end delay for DSF is still smaller than for the baseline schemes.

## 7.3 Insights

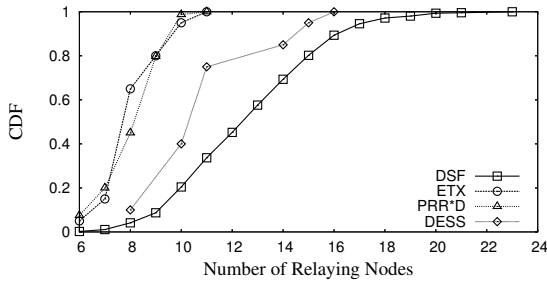
In the previous section, we saw the significant improvement of the source-to-sink communication for DSF over ETX, PRR×D, and DESS. In this section, we reveal the underlying reasons why DSF provides better performance than those state-of-the-art solutions.

### 7.3.1 Diversity in Link Quality

Both ETX and PRR×D generally prefer reliable links and try to avoid highly unstable links. While this intuitive approach holds well in traditional wireless networks, we saw that as node duty cycle decreases, the delay of such schemes becomes excessive since the time spent on waiting for the forwarder to wake up again is no longer tolerable. Figure 24 shows the CDF curve of



**Fig. 24. Diversity in Forwarder Link Qualities (Average # of Neighbors=6)**



**Fig. 25. Diversity in Delivery Paths (# of Neighbors=6)**

the forwarder's link qualities for 200 randomly sampled senders from DSF, ETX, and PRR×D. From the figure, we can see that the distribution of DSF link quality is roughly uniform, with no obvious range being favored, while ETX and PRR×D select much more reliable links. This observation strengthens our understanding that unreliable links are as useful as highly reliable links for minimizing the source-to-sink communication delay in low duty-cycle networks.

### 7.3.2 Diversity in Delivery Paths

In the previous subsection, we demonstrated that picking low-quality links is beneficial in low duty-cycle sensor networks for reducing the source-to-sink communication delay. In this section, we show the greater diversity of delivery paths for DSF over those for ETX, PRR×D, and DESS. In the simulation setup, 150 nodes are deployed in a  $160m \times 160m$  field. Forty source nodes on the edge of the field send their packets to the sink node located in the center of the field. In Figure 25, we show the number of nodes that relay the packets sent by the source nodes during 100-packet delivery processes for DSF, ETX, PRR×D, and DESS. Clearly, DSF explores a much larger neighbor space than the other three schemes in these 100 packet transmission processes. For example, the maximum number of relaying nodes for DSF, ETX, PRR×D, and DESS is 23, 11, 11, and 16, respectively. This again demonstrates DSF's adaptability to the presence of unreliable radio links and the low duty-cycle of sensor nodes.

## 8 Conclusion

In this work, we propose a dynamic switch-based forwarding (DSF) scheme for extremely low duty-cycle sensor networks, which addresses the combined effect of unreliable radio links and sleep latency in data forwarding. We derive a distributed model for data delivery ratio (EDR), E2E delay (EED), and energy consumption (EEC) at individual nodes and optimize the forwarding action in terms of these three metrics. To evaluate the performance of DSF, we have fully implemented the DSF in a network of 20 MicaZ motes and performed extensive simulation with various network configurations. The results demonstrate that DSF significantly improves source-to-sink communication over several state-of-the-art solutions in low duty-cycle sensor networks with unreliable radio links.

## 9 References

- [1] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic. Towards Optimal Sleep Scheduling in Sensor Networks for Rare Event Detection. In *IPSN'05*, 2005.
- [2] Chipcon. *CC2420 Product Information and Data Sheet*. Available at <http://www.chipcon.com/>.
- [3] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High Throughput Path Metric for Multi-Hop Wireless Routing. In *MOBICOM'03*, 2003.
- [4] O. Dousse, P. Mannersalo, and P. Thiran. Latency of wireless sensor networks with uncoordinated power saving mechanisms. In *MobiHoc '04*, 2004.
- [5] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *PLDI'03*, 2003.
- [6] Y. Gu, J. Hwang, T. He, and D. H.-C. Du. uSense: A Unified Asymmetric Sensing Coverage Architecture for Wireless Sensor Networks. In *ICDCS '07*, 2007.
- [7] T. He, S. Krishnamurthy, L. Luo, T. Yan, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM Transaction on Sensor Networks*, 2006.
- [8] J. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *SensSys'04*, 2004.
- [9] A. Keshavarzian, H. Lee, and L. Venkatraman. Wakeup Scheduling in Wireless Sensor Networks. In *MobiHoc '06*, 2006.
- [10] H. Kiehne. *Battery Technology Handbook*. Marcel Dekker, 2003.
- [11] Y. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *NSDI'05*, 2005.
- [12] S. Lee, K. J. Kwak, and A. T. Campbell. Solicitation-Based Forwarding for Sensor Networks. In *SECON'06*, 2006.
- [13] Y. Li, W. Ye, and J. Heidemann. Energy and Latency Control in Low Duty Cycle MAC Protocols. In *WCNC'05*, 2005.
- [14] S. Lin, J. Zhang, G. Zhou, L. Gu, T. He, and J. A. Stankovic. ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks. In *SensSys '06*, 2006.
- [15] J. Liu, F. Zhao, P. Cheung, and L. Guibas. Apply Geometric Duality to Energy-efficient Non-local Phenomenon Awareness using Sensor Networks. *IEEE Wireless Communications*, 11(6), 2004.
- [16] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay Efficient Sleep Scheduling in Wireless Sensor Networks. In *INFOCOM'05*, 2005.
- [17] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The Flooding Time Synchronization Protocol. In *SensSys'04*, 2004.
- [18] J. Polastre and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *SensSys'04*, November 2004.
- [19] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari. Energy-efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks. In *SensSys '04*, 2004.
- [20] N. Shrivastava, R. M. U. Madhow, and S. Suri. Target Tracking with Binary Proximity Sensors: Fundamental Limits, Minimal Descriptions, and Algorithms. In *SensSys '06*, 2006.
- [21] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline. PIPENET: A Wireless Sensor Network for Pipeline Monitoring. In *IPSN'07*, 2007.
- [22] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In *SensSys'03*, 2003.
- [23] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *SensSys 2003*, 2003.
- [24] T. Yan, T. He, and J. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *SensSys 2003*, November 2003.
- [25] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *INFOCOM*, 2002.
- [26] W. Ye, F. Silva, and J. Heidemann. Ultra-low Duty Cycle MAC with Scheduled Channel Polling. In *SensSys '06*, 2006.
- [27] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *IEEE INFOCOM*, 2004.
- [28] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *SensSys '03*, 2003.
- [29] G. Zhou, T. He, and J. A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *MobiSys'04*, June 2004.
- [30] M. Zuniga and B. Krishnamachari. Analyzing the Transitional Region in Low Power Wireless Links. In *IEEE SECON'04*, 2004.