Cluster-Based Forwarding for Reliable End-to-End Delivery in Wireless Sensor Networks

Qing Cao[1], Tarek Abdelzaher[1], Tian He[2], Robin Kravets[1]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, 61801
[2]Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 55455
Email:{qcao2@cs.uiuc.edu, zaher@cs.uiuc.edu, tianhe@cs.umn.edu, rhk@cs.uiuc.edu}

***Abstract*—Providing efficient and reliable communication in wireless sensor networks is a challenging problem. To recover from corrupted packets, previous approaches have tried to use retransmissions and FEC mechanisms. The energy efficiency of these mechanisms, however, is very sensitive to unreliable links. In this paper, we present cluster-based forwarding, where each node forms a cluster such that any node in the next-hop's cluster can take forwarding responsibility. This architecture, designed specifically for wireless sensor networks, achieves better energy-efficiency by reducing retransmissions. Cluster-based forwarding is not a routing protocol. Rather, it is designed as an extension layer that can augment existing routing protocols. Using simulations, we demonstrate that cluster-based forwarding is effective in improving both end-to-end energy efficiency and latency of current routing protocols.**

I. Introduction

A fundamental problem in wireless communication is providing efficient and reliable end-to-end packet delivery [21], [25], [23], [19], [24], [14], [9]. Because wireless links tend to be unreliable due to factors such as interference, attenuation, and fading [11], [24], [27], [4], [28], previous protocols for reliable communication have tried to use two approaches to recover from corrupted packets, namely, packet retransmissions and forward error correction (FEC). Both approaches are sensitive to link quality. When links are weak, packet retransmissions are expensive since the energy spent on a failed node-to-node transmission is completely wasted. Similarly, FEC is expensive since it must be designed for the worst case if channel conditions change frequently. The problem of reliable energy-efficient communication deserves special attention in wireless sensor networks. Link quality in sensor networks is usually weaker compared to other wireless systems. Many links can have a loss probability well above 50% [11], [27]. Additionally, link quality is marked by significant variability due to changes in the environment. Meanwhile, the energy constraints of sensor networks are much more severe, because of the unattended nature of many sensor network applications [22], [12]. Therefore, it is imperative to design protocols in wireless sensor networks that implement efficient measures for minimizing energy loss.

This work takes inspiration from one class of promising techniques known as *cooperative communication* [13], which exploits the broadcast nature of wireless communication to improve energy efficiency. We refer to ExOR [2] and MRD [17] as two recent protocols in this area. However, designed for wireless networks (MANET), these protocols are not suitable for typical sensor network applications for three reasons. First, to reduce product cost, current sensor nodes are usually equipped with low-cost transceivers, such as CC2420 [5], which are quite different from the more powerful radio systems typical to other wireless networks. One restriction is the frame length. CC2420 has a transmit buffer of only 128 bytes, and the actual payload of a data packet is usually 30-50 bytes. Because of the size limitation, protocols in sensor networks cannot rely heavily on radio to transmit protocol state. This is in contrast, for example, to ExOR, where the packet header size alone ranges between 44 and 114 bytes, and is heavily used for state transfer. Hence, in general, MANET protocols cannot be directly ported to sensor networks.

Second, MANET protocols usually assume different communication patterns from sensor networks. For example, ExOR optimizes batches of packets, while MRD optimizes communication between the WLAN client and multiple access points (AP). Both communication patterns do not fit sensor networks, where data flows are usually low-rate and spontaneous.

Third, the communication stack of MANET is different from sensor networks. While MANET has widely assumed variants of 802.11 as its protocol stack, there is no agreement on (or standardization of) the individual protocols in sensor networks. Existing applications often develop their own routing implementations. Therefore, it may not be useful to propose yet another reliable end-to-end routing protocol, because of the difficulty in adapting existing applications to use any single protocol. Instead, it may be more effective to propose a modular approach that allows *extending* existing routing protocols.

In response to these challenges, we propose *cluster-based forwarding* (CBF), a general architectural extension to routing protocols that takes inspiration from cooperative communication, and is compatible with most existing routing protocols through carefully defined interfaces. We call the scheme "cluster-based" because, in this approach, groups of nodes cooperate with each other to forward packets. Clusters in CBF are more akin to *neighborhoods* than to clustering backbones as proposed in ad-hoc networks [1], [16]. Previous clustering methods for ad-hoc networks cannot be used in CBF, because selecting clusters is critical to the performance of CBF, and inappropriate clustering will introduce excessive overhead. Therefore, we design a customized approach in CBF based

on an analysis of energy cost.

The main contributions of CBF are summarized as follows. First, CBF is the first cluster-based forwarding service that is designed as an architectural extension to existing routing protocols in wireless sensor networks. To this end, CBF offers a performance improvement that requires minimal-to-no changes to both routing protocols and existing applications. This minimal-impact property of CBF enables its convergence with existing applications and protocols, while providing better end-to-end energy efficiency.

Second, CBF proposes to use "helpers", which reduces the number of retransmissions by adaptively migrating packet forwarding tasks from weak links to strong links, and by taking advantage of the occasionally successful transmissions over long (and likely lossy) links. To organize helpers around one node, we introduce two "helper patterns", the intermediate helper pattern and the distant helper pattern, in CBF's helper admission algorithm.

Third, the efficacy of CBF is demonstrated through performance comparisons of four different routing protocols, before and after applying CBF. These comparisons also validate our design goals of interfacing CBF with different communication stacks, with minimal-to-no changes.

This paper is organized as following. Section II surveys related work and discusses their differences with CBF. Section III presents an overview of the motivations for CBF. Section IV elaborates the detailed design choices of the CBF architecture. This architecture interfaces with both the network layer and the MAC layer of existing communication stacks, but remains independent of their internals, treating them as black boxes. Section V evaluates the performance of CBF, using four well-known routing protocols as baseline examples. Section VI concludes this paper.

II. Related Work

Providing energy-efficient and reliable end-to-end packet delivery for wireless sensor networks is a challenging task due to the unreliable nature of wireless links [28], [27], [24]. Several approaches have been proposed for reliable communication in sensor networks. Alec Woo [24] chooses reliable routes based on link connectivity statistics obtained dynamically from an EWMA estimator. RMST [21] tracks packet fragments so that receiver initiated requests can be satisfied when individual pieces of an application payload get lost. Robust data delivery [10] simultaneously sends packets along multiple paths at the expense of increased communication overhead. FEC-based approaches [14] try to encode error recovery information into packets to compensate for the effect of lossy links.

While the previous approaches are quite useful, the energy efficiency of these protocols could be considerably affected by unreliable links. When weak links are chosen, these protocols may introduce excessive energy overhead by retransmitting the packets. To solve this problem, recently, researchers have studied improving energy-efficiency using a class of techniques called cooperative communication. We refer to [13] as a survey of this topic, and refer to ExOR [2], MRD [17] and ROMER [26] as recent protocols. However, as shown in the previous section, these protocols are different from our work in terms of the assumptions, goals, and approaches. Because of the unique challenges of sensor networks, they cannot be directly ported and used.

In sensor networks, the only work we know that takes inspiration from cooperative communication is SPaC [7]. The key idea of this protocol is that nodes combine multiple corrupted packets into correct packets. SPaC allows one node to receive two or more corrupted versions of a packet from its upstream nodes through overhearing. The focus of SPaC is thus on reliability. By allowing multiple copies of the same data packet, SPaC also increases the demand for storage space and computation overhead of sensor nodes. Our work is different from SPaC in that given the memory constraints on sensor network nodes, CBF does not store any corrupted packets during communication. The original use of helpers in CBF is also different, which makes hop-wise delivery more likely to succeed.

III. Overview of Cluster-Based Forwarding

In this section, we first present our experimental results on link quality. These results lead to two observations that motivate CBF, and each observation leads to a helper pattern. We also present practical design concerns of CBF that need to be addressed.

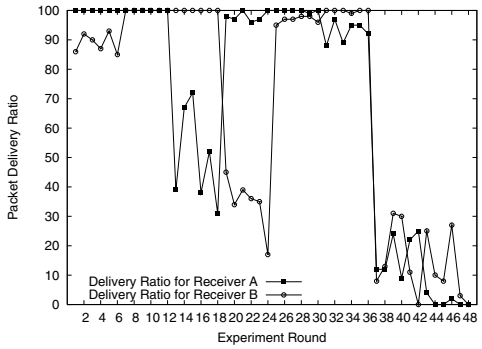

Fig. 1. Comparison of Packet Delivery Probability at Different Distances

A. Link Quality in Reality

Recent investigations [11], [27], [24], [28] indicate three distinct data reception regions for wireless communication: fully connected, transitional and disconnected. In the fully connected region, nodes transmit packets reliably, approaching 100% delivery probability in the absence of congestion; in the transitional region, link quality varies considerably: some links may exhibit perfect quality while others the opposite; in the disconnected region, no links or only weak links exist. Simulation models for these three regions have also been presented (e.g., the simulation model in [28]).

To help quantify these three regions, we carry out an experiment using MicaZ sensor nodes. Figure 1 shows our experimental results of delivery probability between MicaZ

nodes at different distances. This experiment uses one sender and two receivers. The distance between them changes from $5ft$ to $40ft$, in steps of $5ft$. At each distance, the sender sends six rounds of packets, with 100 packets in each round. The packet delivery probability is plotted for comparison. Observe that the link quality varies considerably with distance from the sender to the receiver, where the observed fully connected region is 0-12 feet, the transitional region 12-36 feet, and the disconnected 36 feet or more.

B. Motivating Observations through Examples

We now study the implications of the three regions. Consider an example scenario as shown in Figure 2, where sender A transmits a packet to receiver B. Assume that two other nodes, C and D, are located within the fully connected region of node B. The routing layer chooses B instead of C or D for the following reasons. From the routing layer's perspective, C is located somewhere between A and B, and is not as good as B as the next relay. On the other hand, while D is indeed a better node than B in terms of its distance to the destination, it is not selected because the link between A and D is too weak[1].

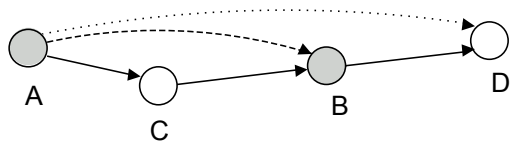

Fig. 2. Hop-wise Transmission Example

While the routing layer neglected C and D, we have the following two observations concerning their roles to improve packet delivery. First, if the quality of the link between C and B is better than the link between A and B, and if C has received the packet from A while B has not, it is better to shift the transmission task from AB to CB. Intuitively, such a shift can reduce the expected number of retransmissions needed because retransmitting from C is more likely to be successful. Note that, while C may alternatively be chosen as an intermediate hop between A and B by the routing layer, such a conservative choice will double the local traffic by adding an extra hop on the path of *all* packets. If the successful transmission probability between A and B is not negligible, energy can be saved by sending to B directly. Only if such a transmission fails that the packet will be forwarded by C. Hence, the extra hop is introduced only for *some* packets. In fact, an advantage of CBF is that it does allow using more hops in routing while providing a reliable conservative alternative—in case of failure—that is no worse than using the more conservative route by default.

Second, if D receives the packet from A, because D is better than B, then, regardless of whether B has received the packet or not, D can continue the forwarding task, and skip B.

[1]We use the words *sender* and *receiver* to refer to the single-hop sender and receiver, and use the words *source* and *destination* to refer to the origin and final sink of the multiple-hop packet delivery task.

The observations above help reduce the total number of transmissions. We call both C and D *helper nodes*, and each of them represents a *helper pattern*. In the first pattern, the helper, C, is an intermediate node that works by shifting the forwarding task from a weak link, AB, to a strong link, CB. We call this pattern an *intermediate* helper pattern. In the second pattern, the helper, D, is a distant node that is exploited opportunistically if it receives A's packet. We call this pattern a *distant* helper pattern.

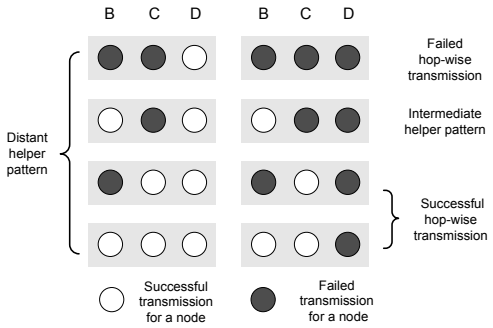

Fig. 3. Hop-wise Transmission Outcomes

Now return to Figure 2. Observe that each of B, C, and D gives rise to two outcomes for the probabilistic hop-wise packet delivery between A and B, success and failure. We have a total of eight combinations of outcomes as shown in Figure 3. Here, black and white circles represent failure and success to receive a packet, respectively, by each of B, C, and D. For a packet sent from A, of all the eight combinations, four can improve communication using the distant helper pattern, and one using the intermediate helper pattern. CBF is designed to improve energy efficiency by taking advantage of these five combinations.

C. Design Concerns

While the mechanism of CBF is intuitive, there are several concerns that have to be addressed. The primary concern is overhead. While both helper patterns reduce the number of retransmissions, they also introduce overhead. In the previous example, if helper C wants to take over the forwarding task, it has to inform A to stop retransmissions. Similarly, if D wants to become a new sender, taking the responsibility of B, it has to inform A and B to stop forwarding the current packet. If not addressed carefully, this overhead may exceed the savings brought about by CBF.

CBF proposes two algorithms to ensure that energy savings dominate. The first algorithm, called the *helper admission algorithm*, takes a cost-analysis approach when forming clusters. The second algorithm, called the *forwarder resolution algorithm*, removes duplicates at very low overhead. The details of these algorithms are presented in the next section.

IV. Cluster-Based Forwarding Architecture

This section presents the CBF architecture. We assume long-term, low-rate traffic patterns, where congestion is rare. The link quality in such networks is relatively stabilized. We intend to consider the effect of congestion in our future work.

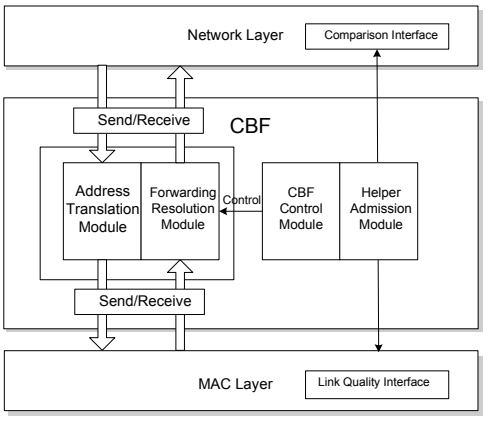

Fig. 4. CBF Architecture

A. Architecture Overview

The CBF architecture is shown in Figure 4. The CBF layer is designed as a middle layer between the network layer and the MAC layer. The CBF layer consists of four modules, the CBF control module, the helper admission module, the address translation module, and the forwarding resolution module.

Algorithm 1 CBF Control Algorithm

```
PHASE I: CBF layer initialization
PHASE II: CBF helper admission
PHASE III:
for Each packet do
  repeat
    {PHASE III.a Sender-side algorithm:}
    Receive a packet from the network layer
    Translate the next-hop node address of this packet to
    the next cluster address
    Send this packet to the MAC layer
    {PHASE III.b Receiver-side algorithm running on mul-
    tiple nodes}
    Receive a packet from the MAC layer
    Forwarding resolution to select a unique receiver
    Send this packet to the network layer on the selected
    receiver only
  until The packet reaches destination
end for
```

The CBF control module maintains state information and coordinates the actions between other modules. The control algorithm is shown in Algorithm 1. This algorithm consists of three phases. The first phase, initialization, sets the initial program state, such as whether or not CBF is enabled. The CBF control module then calls other modules in the CBF layer to finish the other two phases.

The helper admission control module is responsible for the second phase, CBF helper admission. This module selects a subset of neighbors as helpers for a receiver, which help it receive packets from the sender.

The address translation module and the forwarding resolution module are responsible for the third phase, which consists of two parts. On the sender side, the address translation module receives packets from the network layer, and translates their addresses into cluster addresses. On the receiver side, because multiple helpers, as well as the receiver, may receive the same packet, the forwarding resolution module removes redundant packets and selects one unique receiver to forward the packet along the multi-hop delivery path. Both the address translation module and the forwarding resolution module use cluster information maintained by the control module. Their relationship is shown in Figure 4.

B. Helper Admission

In the helper admission module, each node selects a subset of neighbors as its helpers using the helper admission algorithm. Our description of this algorithm consists of two parts. The first part describes its interfaces with other layers of the communication stack. The second part describes its implementation details.

*1) **Algorithm Interfaces**:* The admission algorithm uses two interfaces: the link quality interface provided by the MAC layer, and the comparison interface provided by the network layer.

The link quality interface provides packet delivery success probability information between the current node and its neighbors. Representative approaches to estimate link quality are based on the snooping of packet sequence numbers [24], or the use of the Link Quality Indicator (LQI) parameter [5]. In this paper, we take an approach similar to [24], where each node gathers link quality information to each neighbor by exchanging sequence-number-stamped packets, and broadcasts gathered information back to its neighbors. The quality of the wireless link between nodes A and B is represented by a vector (p, q), where p is the packet delivery success probability from A to B, and q is the probability in the opposite direction. We also use (p, q) to denote a link where there is no confusion.

While the link quality interface is used for admitting both intermediate helpers and distant helpers, the comparison interface is specifically used for admitting distant helpers. Note that this interface is completely optional: CBF can also be functional without using distant helpers. However, we observe that most network layers can export such an interface with minimal changes to their code. Therefore, we consider it worthwhile to provide this interface in exchange for improved CBF performance. Formally, the comparison interface is defined as the following function.

Comparison function:
Compare(NODE N1, NODE N2, NODE DESC)
Parameters: NODE N1, NODE N2, NODE DESC
Return value: NODE N1 or N2

In this function, N1 and N2 are from the set of the current node and its neighbors. The function compares N1 and N2 in terms of their advances towards the destination DESC, and selects the better one as the returned node. We observe that this interface is inherently supported by many routing protocols. We now give a few examples. In the geographic forwarding routing protocol, this function is implicitly implemented

by comparing physical distances to the destination. Another example is DSR, where the interface can be implemented by using the path information that is stored in DSR packet headers. Different nodes can be compared based on their hop distances to the destination, if they lie on the forwarding path of the current packet. More generally, we conceive the following implementation for the comparison interface. When a node, referred to as *the current node*, invokes such an interface, it works as follows. First, if either N1 or N2 is the next-hop node for the destination DESC, the function returns this next-hop, N1 or N2. Second, if the current node is either N1 or N2, and the next-hop node is not involved, the function returns this current node. Finally, if neither the next-hop node nor the current node is involved in the comparison, the function randomly returns N1 or N2. The semantics of this implementation is that the next-hop node is the best, the current node the second, and all the other nodes are less favorable. This simple implementation is sufficient for the helper admission module to use. Because this implementation allows at most one distant helper (essentially, the next-hop node of the receiver), more fine-grained comparisons, as is the case with geographic forwarding, are preferred to further improve the performance of CBF.

*2) **Algorithm Design**:* We now describe the design of the helper admission algorithm. For both intermediate helpers and distant helpers, the intuition of this algorithm is that a neighbor node should become a helper if it can introduce more savings than costs. To find such neighbors, the algorithm takes a cost analysis approach. Because an intermediate helper and a distant helper follow different cost analysis procedures, the algorithm first uses the comparison interface to classify potential helpers. In this step, the receiver classifies each neighbor as either a potential intermediate helper C (if the receiver itself is better), or a distant helper D (if this neighbor node is better).

In the following derivations, we assume uniformly deployed sensor networks, where the number of nodes receiving a broadcast packet is roughly equal. Hence, the cost of a transmission consists of the sending cost of the sender, and the receiving cost of a fixed number of receivers. We define this sum as one cost unit. We also assume the most commonly used ACK model, which uses acknowledgements to ensure reliable transmissions[2].

We now derive the energy cost for sending a packet over a link with quality (p, q). Since the round-trip combined packet delivery probability is pq, the expected number of transmission rounds is $1/pq$. Therefore, the expected transmission cost at the sender side is $1/pq$. On the receiver side, B sends out acknowledgements for every data packet it receives. Assuming that the packet size ratio between an acknowledgement packet and a data packet is λ. Therefore, the expected cost of acknowledgements is $\lambda \times 1/q$. The total energy cost, normalized to the cost of transmitting a data packet once, is $1/pq + \lambda/q$.

Next, we consider an intermediate helper, C. Every time it receives a data packet that the receiver B has lost, it sends a (very short) Request To Send (RTS) packet to B. If B agrees, it replies with a Clear To Send (CTS) packet. After receiving CTS, C replies with the lost data packet, thereby shifting the delivery task from AB to CB. C only sends RTS once. If it does not receive the CTS packet, it silently drops the data packet. The reason for this design choice is to avoid multiple copies of the data packet, and will further be explained in the forwarder resolution module.

We now analyze helper C's savings and costs. To differentiate between links, we denote the link between A and B as $(p_{\mathrm{AB}}, q_{\mathrm{AB}})$, and the link between C and B as $(p_{\mathrm{CB}}, q_{\mathrm{CB}})$.

Helper C can save energy only if it receives a CTS from B. The expected cost of sending the data packet from C to B is $1/p_{\mathrm{CB}}q_{\mathrm{CB}} + \lambda/q_{\mathrm{CB}}$. Because the event that C receives the CTS packet from B occurs with a probability of $p_{\mathrm{CB}}q_{\mathrm{CB}}$, the expected savings introduced by C, denoted by $\mathbb{S}$, are:

$$\mathbb{S} = p_{\mathrm{CB}}q_{\mathrm{CB}}[\frac{1}{p_{\mathrm{AB}}q_{\mathrm{AB}}} + \frac{\lambda}{q_{\mathrm{AB}}} - (\frac{1}{p_{\mathrm{CB}}q_{\mathrm{CB}}} + \frac{\lambda}{q_{\mathrm{CB}}})] \quad (1)$$

The costs of helper C come from two sources. First, the RTS/CTS exchanges between C and B. Second, if B receives the packet from C, it has to inform A to stop retransmitting the data packet. The first cost $\mathbb{C}_1$ can be written as follows.

$$\mathbb{C}_1 = \lambda + q_{\mathrm{CB}}\lambda \quad (2)$$

Because B informs A only after it receives the data packet from C, this cost is associated with a probability of $p_{\mathrm{CB}}q_{\mathrm{CB}}$. Assuming that B informs A uses an acknowledgement-based model, this cost $\mathbb{C}_2$ can be written as follows.

$$\mathbb{C}_2 = p_{\mathrm{CB}}q_{\mathrm{CB}}[\frac{\lambda}{p_{\mathrm{AB}}q_{\mathrm{AB}}} + \frac{\lambda}{p_{\mathrm{AB}}}] \quad (3)$$

To ensure that costs are smaller than savings, we have:

$$\mathbb{S} > \mathbb{C}_1 + \mathbb{C}_2 \quad (4)$$

For a distant helper, denoted as D, the situation is slightly different. Because a distant helper is a better next-hop than the current node, it does not need to deliver the data packet to B again. Instead, it informs B that it will send out the packet directly. Therefore, a distant helper introduces more costs, but at the same time, more savings as well. The savings and costs can be written as follows, following the same analysis procedure for intermediate helpers.

$$\mathbb{S} = p_{\mathrm{DB}}q_{\mathrm{DB}}[\frac{1}{p_{\mathrm{AB}}q_{\mathrm{AB}}} + \frac{\lambda}{q_{\mathrm{AB}}}] \quad (5)$$

$$\mathbb{C}_1 = \lambda + q_{\mathrm{DB}}\lambda \quad (6)$$

[2]A related reliable transmission model is the NACK model, which uses negative acknowledgements. The NACK model is appropriate for data streams, where follow-up packets can be used to detect previous packet losses. The energy cost over a link (p, q) for the NACK model is $1/p + (1-p)\lambda/pq$, which is lower than that of the ACK model. However, because of the stream requirement of the NACK model, we assume the ACK model in this paper.

$$\mathbb{C}_2 = p_{\rm DB} q_{\rm DB} [\frac{\lambda}{p_{\rm DB} q_{\rm DB}} + \frac{\lambda}{q_{\rm DB}}] \quad (7)$$

$$\mathbb{C}_3 = p_{\rm DB} q_{\rm DB} [\frac{\lambda}{p_{\rm AB} q_{\rm AB}} + \frac{\lambda}{p_{\rm AB}}] \quad (8)$$

And to admit D, it must satisfy:

$$\mathbb{S} > \mathbb{C}_1 + \mathbb{C}_2 + \mathbb{C}_3 \quad (9)$$

Results 4 and 9 show the requirements for one neighbor node to be accepted as a helper. The admission algorithm simply applies these requirements to each neighbor to obtain a cluster.

To have an intuitive understanding of the above requirements, we give an example. Suppose that in a sensor network, the length of a data packet is 30 bytes, the length of a control packet 3 bytes, and the link between A and B has a quality of $(0.5, 0.5)$. Inequality (4) and Inequality (9) reduce to quadratic functions, and indicate that an intermediate helper C must have a link to B with $p_{\rm CB} > 0.58$ (assuming $p = q$), while a distant helper D only requires a link to B with $p_{\rm DB} > 0.27$ (assuming $p = q$), to improve performance.

C. Address Translation

The address translation module translates next-hop node addresses into next-hop cluster addresses. Each receiver forms its own cluster. Therefore, a natural design choice is to use the next-hop node address itself as the cluster address. In the CBF architecture, an optional design choice is that for each hop, the sender specifies whether the next-hop transmission will use CBF or not. Therefore, an optional field (one bit) is used in the packet to denote whether CBF is enabled for the next hop.

D. Forwarder Resolution

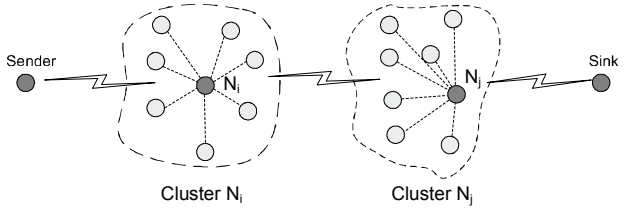

Fig. 5. Cluster Forwarding Scenario

The forwarder resolution module of CBF ensures that no duplicated packets are delivered. One source of potential duplicates is when multiple helpers receive the same lost data packet. Figure 5 shows such an example. We now describe how such duplicates are removed.

To avoid radio interference between packets, the forwarder resolution module uses time slots to coordinate between helpers. In general, the receiver assigns early time slots to distant helpers because they can potentially bring about more savings.

The time slot assignment between distant helpers works as follows. The receiver compares such helpers using the comparison interface, and assigns earlier time slots to those with more advances to the destination. Between intermediate helpers, the approach is different. Because intermediate helpers send the lost data packet to the receiver anyway, the time slot assignment is based on comparing link quality between an intermediate helper and the receiver. Those helpers with better combined round-trip packet delivery probability, pq, get earlier time slots.

Note that, although we use time slot assignments in the resolution procedure, we do not need a time synchronization service. The reason is that only those helpers that receive a lost data packet from the sender will wait for their time slots. Due to the broadcast nature of the wireless medium, the data packets themselves can serve as implicit synchronization points, much like the approach used in RBS [8].

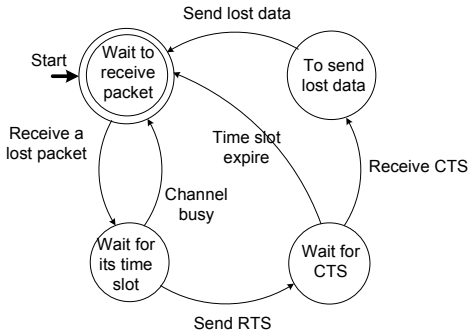

Fig. 6. Intermediate Helper State Transitions

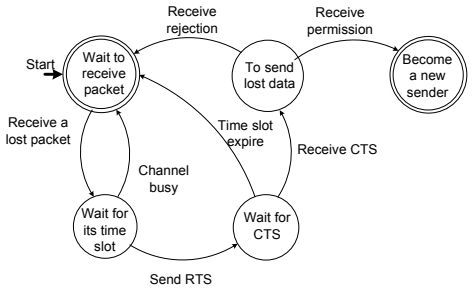

Fig. 7. Distant Helper State Transitions

We next describe how a helper works during the forwarder resolution. The state transition graphs of an intermediate helper and a distant helper are shown in Figure 6 and Figure 7. We now explain these state transitions. Once one helper receives a data packet from the sender, it waits for its pre-determined time slot to send RTS. When its time slot comes, the helper first listens to check whether the channel is clear. If it is, it follows with an RTS packet to the receiver. If the receiver replies with a CTS, this helper gets a permission to proceed. It either follows with a lost data packet, if it is an intermediate helper, or follows with a request to become a new sender, if it is a distant helper.

On the other hand, at the beginning of the time slot, if the current helper detects that the channel is not clear, it is likely that a previous helper has got a permission from the receiver, and is sending the lost data packet. The forwarder resolution module takes a conservative approach, where the

current helper silently drops the data packet, to avoid any possible interference between helpers.

To save overhead, CBF could use overhearing to reduce unnecessary transmissions, if nodes work in listening mode by default, as is usually the case for the radio circuit of current sensor networks. The principle of overhearing is that because the wireless medium is shared, each node can overhear data packets sent by its neighbor. Both the sender and the helpers can use overhearing to adjust their actions. At the sender side, if the sender overhears a data packet being sent out again by the receiver, it knows that this data packet must have been received successfully, and safely removes it from its buffer. At the helper side, if one helper overhears that another helper has started to send a lost data packet to the receiver, this helper assumes that the lost data packet can be recovered and therefore, no longer needs to send its RTS when its time slot comes.

Finally, the receiver maintains the uniqueness of each data packet, by following two rules. First, if no distant helpers exist, it only admits a lost data packet from intermediate helpers once. Second, if there are distant helpers, it only gives one distant helper the permission to serve as a new sender. For all the RTS packets received from helpers after either of these two actions occurs, the receiver stops sending out CTS packets, so that all the remaining helpers do not get permissions. This also explains why RTS packets are sent only once by a helper: additional RTS packets may lead to undesired traffic that not only consumes extra bandwidth and energy, but also confuses the receiver.

V. Performance Evaluation

In this section, we present performance evaluation results. We use four routing protocols as baselines. We implement CBF as extensions for each of them, and compare the performance of these routing protocols before and after applying CBF.

A. Comparison Baselines

We first introduce the comparison baselines. We choose four node-based routing protocols, shown as following.

- Hop-based Spanning Tress [3], [18] (termed SPT-HOP)
- ETX-based Spanning Trees [6] (termed SPT-ETX)
- Geographic Forwarding [15] (termed GF-HOP)
- Geographic Forwarding Extension [20] (termed GF-ETX)

The first baseline, hop-based spanning tree, uses flooding to find paths. A shorter path (i.e., one with fewer hops) is considered better. While such aggressive path-length optimization has become deprecated (as it tends to choose longer, unreliable links), the addition of the CBF extension allows opportunistic use of longer links while providing a reliable alternative as backup.

ETX-based spanning tree is an adaptation of the ETX-based DSR from [6], which takes into account the effect of link quality on routing performance. In [6], the authors showed that this new routing protocol can achieve better performance, by associating an ETX-based cost metric to each link. Both hop-based spanning tree routing and ETX-based spanning tree routing take advantage of global information to make routing decisions.

TABLE I
Simulation Settings

Radio			
Modulation	FSK	Encoding	Manchester
Output Power	-7 dBm	Frame	50 bytes
Transmission Medium			
Path Loss Exponent	3	PL_{D_0}	52 dBm
Noise Floor	-105 dBm	D_0	1m
Deployment Configuration			
Area Height	300 m	Area Width	300 m
Node Number	2500	Range	10-25m
Protocols			
SPT	SPT-ETX	GF	GF-ETX
Performance Metrics			
End-to-End Energy Cost		End-to-End Delay	

Geographic forwarding is another protocol that has been widely used in sensor networks. Many variants of this protocol exist today [15], [20], [9]. Unlike the previous two protocols, geographic routing does not rely on global topology information to make routing decisions. Instead, this class of protocols makes routing decisions in a localized neighborhood. While this property makes it more vulnerable to topology holes, geographic forwarding usually has lower overhead, compared to spanning-tree-based protocols.

We choose two geographic forwarding protocols, the basic version and an extension presented in [20]. In this extension, nodes use both geographic information and link quality to make routing choices. For each neighbor T, the sender S will calculate the metric $DistanceAdvanced(ST) \times LinkQuality(ST)$, and choose the neighbor node that maximizes this metric as the next hop relay, to achieve improved performance.

B. Simulator Details

Our simulator is implemented as follows. The radio model is implemented according to [28], with several adjustable parameters. We set these parameters strictly according to the hardware specifications of current sensor networks (i.e., we use CC2420 radio hardware as the reference setting, as shown in Table I.). These parameters accurately reflect the performance of MicaZ nodes in that they have the same modulation method (FSK), encoding method, frame length and path loss exponent.

During each simulation, the simulator deploys 2500 nodes randomly in a $300m \times 300m$ field. A sink is positioned at the center of the field, and each node sends a packet to the sink over multiple hops, using different routing protocols. We choose two evaluation metrics, the end-to-end delay and energy cost. The delay metric measures how long it takes for the packet to arrive at the sink, normalized to the time for transmitting one data packet once. The cost metric is defined as the ratio of the total number of packets, in bits, sent by all nodes (including all control signals and retransmissions), by the total number of non-redundant packets received by the sink. This metric essentially measures *goodput*, and reflects both communication overhead and energy efficiency.

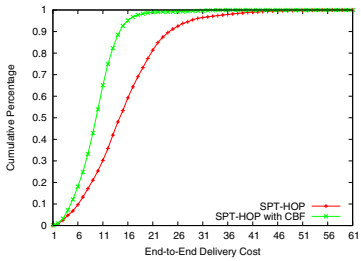

Fig. 8. SPT-HOP Cost

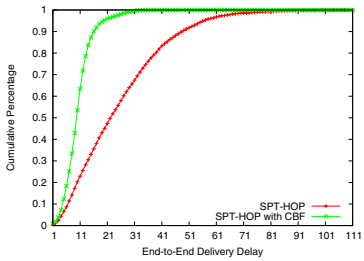

Fig. 9. SPT-HOP Delay

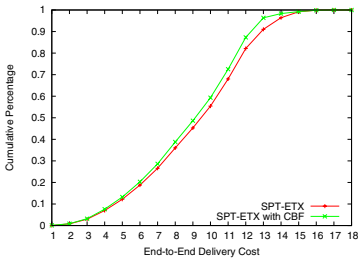

Fig. 10. SPT-ETX Cost

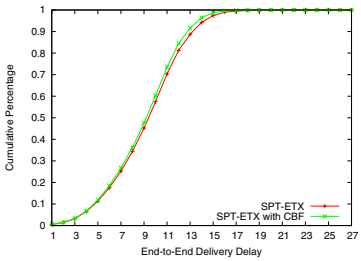

Fig. 11. SPT-ETX Delay

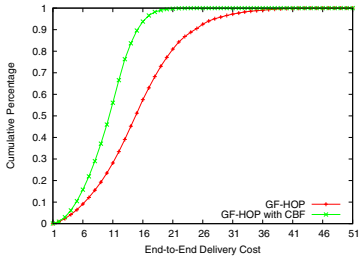

Fig. 12. GF-HOP Cost

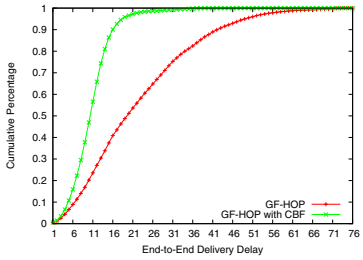

Fig. 13. GF-HOP Delay

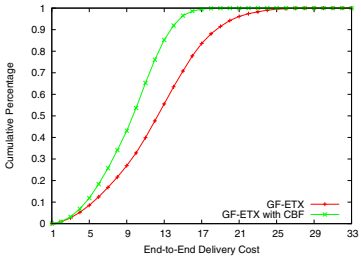

Fig. 14. GF-ETX Cost

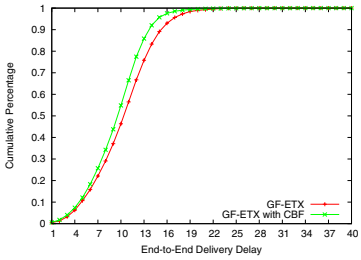

Fig. 15. GF-ETX Delay

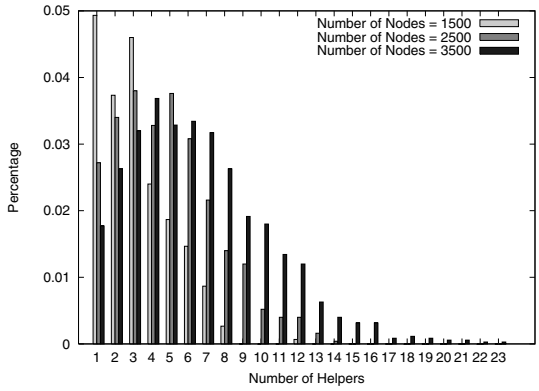

Fig. 16. Helpers and Density

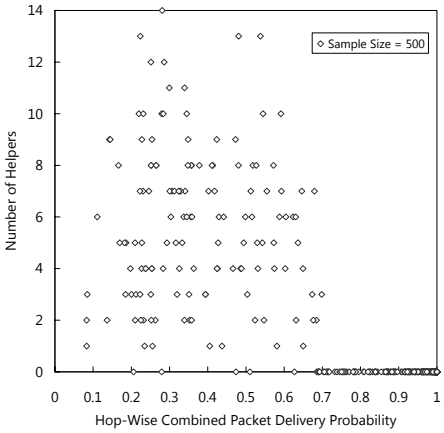

Fig. 17. Helpers and Link Quality

C. Evaluation Results

We present the experimental results in Figure 8 to Figure 15. All results are based on a total of ten rounds of simulations.

The figures are organized as follows. Figure 8 and 9 are results for the SPT-HOP-based routing protocol. Figure 10 and 11 are results for the SPT-ETX-based routing protocol. Figure 12 and Figure 13 are results for the GF-HOP-based routing protocol. Figure 14 and Figure 15 are results for the GF-ETX-based routing protocol.

We have three observations concerning these results. First, CBF reduces the end-to-end cost and delay for each routing protocol. Because of the cost-analysis procedure in the helper admission module of CBF, the reduction in cost is expected. The reduction in end-to-end delay is more interesting. An examination of simulation traces reveals that this is attributed to a considerable decrease in the number of retransmissions: without CBF, the sender will time-out and resend a lost packet. Such time-outs considerably increase end-to-end delay. Therefore, while CBF increases the possibility of successful hop-wise packet delivery, i.e., fewer time-outs, it reduces the end-to-end delivery delay as well.

Second, we observe that while the four routing protocols without CBF have considerably different performance, after applying CBF, their performance is comparable. The reason is that CBF exploits spatial diversity and link quality variations. While a particular routing protocol, such as SPT-HOP and GF-HOP, may be inefficient by choosing weak links, CBF fixes their inefficiencies by aggressively using helpers to transmit over such weak links, and achieves better end-to-end performance. In fact, the resulting performance for these inefficient protocols after applying CBF is comparable to that of routing protocols that have built-in link-quality-based measures, such as SPT-ETX and GF-ETX.

A third observation is that the performance improvements after applying CBF on SPT-ETX (Figure 10 and Figure 11) are minimal. We analyzed the simulation traces and found out that in general, SPT-ETX has already selected pretty good paths using its built-in link-quality-based measures. Therefore,

its space for performance improvement by exploiting link quality is limited. In fact, CBF does not considerably change the behavior of SPT-ETX in the experiments. On the other hand, because CBF does not degrade protocol performance, a property that is guaranteed by its cost-analysis procedure, we could still observe a slight performance improvement brought by CBF.

It is generally known that by adding more nodes to a sensor network, i.e., by increasing its density, its communication performance can be improved, because nodes have more next-hop choices and may find better routes to destinations. Therefore, it is interesting to evaluate the impact of density on CBF. In the following experiment, we change the number of nodes deployed. In addition to the previous setting where 2500 nodes are deployed, we also evaluate two more settings with 1500 and 3500 nodes deployed in each setting, and record the number of helpers for each node. The number of helpers for those nodes are shown in Figure 16, and we observe that as the node density increases, the number of helpers increases as well. This experiment validates our intuition that adding more nodes may help communication, because more nodes are likely to have more helpers.

The last experiment answers the following question: exactly which nodes have helpers? More specifically, we try to correlate the number of helpers of one node to its next-hop link quality. We use the combined round-trip packet delivery probability, i.e, pq, to measure the quality of a link (p, q), and present the results in Figure 17. We select a sample of randomly chosen 500 nodes, and for each node, plot both the number of its helpers and its next-hop link quality. Observe that most of the helpers belong to those nodes with weak next-hop links (i.e., a lower combined packet delivery probability). On the other hand, those nodes with stronger next-hop links, positioned on the right side of the figure, mostly have no helpers at all. This figure is also consistent with how CBF achieves performance improvements, namely, using helpers to recover from the otherwise failed transmissions over weak links.

VI. Conclusions and Future work

In this paper, we proposed CBF, a cooperative approach to improve end-to-end delivery performance for wireless sensor networks, by exploiting the use of helper clusters. We demonstrate that CBF reduces end-to-end cost and delay for generic routing protocols. Designed as an architectural extension, CBF can be easily plugged into existing communication stacks to improve performance. We believe this design choice makes CBF particularly attractive to sensor network system designers. Furthermore, by using the technique of cooperative communication, CBF represents a growing direction of wireless communication protocols.

While CBF is designed for low-data-rate sensor networks where congestion is rare, we will study how to extend CBF to congested sensor networks in our future work. CBF may need to be modified because when congestion exists, the link quality may change dramatically from time to time, and helpers need to be updated in time to compensate for such changes. Such design changes await further study.

VII. Acknowledgements

This work is supported in part by NSF grants CNS 06-15318, CNS 06-26342, CNS 05-54759, CNS 05-53420, CNS 06-26614, and CNS 06-15063. The work is also supported by a Vodafone Fellowship.

References

[1] S. Basagni. Distributed clustering for ad hoc networks. In *ISPAN*, 1999.
[2] S. Biswas and R. Morris. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. In *ACM SIGCOMM*, August 2005.
[3] J. Broch et al. Dsr : The dynamic source routing protocol for multihop wireless ad hoc networks. In *Ad Hoc Networks, Addison Wesley*, 2001.
[4] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical Model of Lossy Links in Wireless Sensor Networks. In *IPSN'05*, 2005.
[5] Chipcon. *CC2420 Data Sheet*. http://www.chipcon.com/.
[6] D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-Throughput Path Metric for Multi-Hop Wireless Routing. In *ACM MOBICOM*, 2003.
[7] H. DuboisFerriere, D. Estrin, and M. Vetterli. Packet combining in sensor networks. In *ACM Sensys*, 2005.
[8] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *OSDI*, 2002.
[9] D. Ferrara et al. Macro: An integrated mac/routing protocol for geographic forwarding in wireless sensor networks. In *Infocom*, 2005.
[10] D. Ganesan et al. Highly resilient, energy efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 1(2), 2002.
[11] D. Ganeshan et al. Complex Behavior at Large Scale: An Experimental Study of Low-Power Wireless Sensor Networks. In *Proc. of the Intl. Workshop on Distributed Event-Based Systems*, 2002.
[12] T. He et al. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. In *TOSN*, 2006.
[13] T. E. Hunter and A. Hedayat. Cooperative communication in wireless networks. In *IEEE Communications Magazine*, 2004.
[14] J. Jeong and C. T. Ee. Forward error correction for sensor networks. In *UCB Technical Report*, 2003.
[15] B. Karp. *Geographic Routing for Wireless Networks*. PhD thesis, Harvard University, 2000.
[16] V. Kawadia and P. R. Kumar. Power control and clustering in ad hoc networks. In *Infocom*, 2003.
[17] A. Miu et al. Improving loss resilience with multi-radio diversity in wireless networks. In *ACM Mobicom*, 2005.
[18] C. E. Perkins and E. M. Royer. Ad-hoc on demand distance vector routing. In *WMCSA*, 1999.
[19] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akylidiz. Esrt: Event-to-sink reliable transport in wireless sensor networks. In *Mobihoc*, 2003.
[20] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari. Energy Efficient Forwarding Strategies for Geographic Routing. In *Sensys*, 2004.
[21] F. Stann and J. Heidemann. RMST: Reliable Data Transport in Sensor Networks. In *SNPA '03*, 2003.
[22] G. Tolle et al. A macroscope in the redwoods. In *ACM Sensys*, 2005.
[23] C. Wan, S. Eisenman, and A. T. Campbell. CODA: COngestion Detection and Avoidance in Sensor Networks. In *Sensys 03*, 2003.
[24] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Sensys*, 2003.
[25] N. Xu et al. A Wireless Sensor Network for Structural Monitoring. In *SenSys 2004*, 2004.
[26] Y. Yuan et al. Resilient opportunistic mesh routing for wireless mesh networks. In *IEEE WiMesh*, 2005.
[27] J. Zhao and R. Govindan. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *Sensys 03*, 2003.
[28] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *SECON*, 2004.