Received 1 April 2013; revised 25 June 2013; accepted 29 June 2013. Date of publication 18 July 2013;
date of current version 20 September 2013.

Digital Object Identifier 10.1109/TETC.2013.2273888

Performance Guaranteed Routing Protocols for Asymmetric Sensor Networks

XIAO CHEN[1], ZANXUN DAI[1], WENZHONG LI[2], AND HONGCHI SHI[1]

[1]Department of Computer Science, Texas State University, San Marcos, TX 78666, USA
[2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
CORRESPONDING AUTHOR: X. CHEN (xc10@txstate.edu)

ABSTRACT In this paper, we propose performance guaranteed routing protocols in asymmetric sensor networks (ASNs) where two end nodes may not use the same path to communicate with each other. ASNs can be caused by hardware devices or environment. Different from most of the existing routing protocols in symmetric sensor networks, because of asymmetry, achieving desired routing performance in ASNs poses significant research challenges. To address these challenges, we first propose a general framework protocol called reverse path (RP) to deal with asymmetric links and then present two efficient routing algorithms LayHet and EgyHet built on RP to satisfy performance requirements. LayHet is a performance guaranteed layer-based routing protocol that embeds the shortest path information and saves energy by minimizing the number of broadcasts and the probability of forwarding. EgyHet is its energy-upgraded version that considers nodes' remaining energy. Simulation results comparing the proposed and existing protocols show that LayHet and EgyHet can reach the desired delivery rate earlier than the existing one and outperform it in terms of average hops, average packet replication overhead, and average control message overhead. In addition, as sensor energy reduces, the performance of LayHet and EgyHet eventually degrades more slowly than that of the existing one.

INDEX TERMS Asymmetric sensor networks, energy efficient, performance guarantee, reverse path, routing.

I. INTRODUCTION

Wireless sensor network (WSN), a typical application of Cyber-Physical Systems (CPS), is projected to have a significant impact on our daily lives, with usages in target tracking [2], intelligent homes [12], environment monitoring [17], disaster rescuing [20], self-touring systems [24], and home health care [26].

We address in this paper a FUNDAMENTAL problem in WSNs where two end nodes may not use the same path to communicate with each other. We refer to the WSNs that exhibit this kind of asymmetric communication as *asymmetric sensor networks (ASNs)*. ASNs can be the result of: Noise sources near a device affecting packet reception at that device [13]; Nodes powering down to conserve energy [16]; Devices transmitting with different powers explicitly causing unidirectional links [18]; And intractable factors such as barriers and environmental conditions affecting signal propagation [25].

Most of the existing routing algorithms are designed for symmetric communication networks where two nodes use the same path to communicate back and forth. Even those discussing heterogeneous networks assume that the links between nodes are symmetric [7], [10], [27], [28]. To the best of our knowledge, Ramasubramanian and Mosse [25] were the first to provide a framework solution for asymmetric links. But they did not address the issue of network performance guarantee.

The performance of routing algorithms in ASNs can be measured by various metrics, such as delivery rate, throughput, latency, overhead and so on. Although they have different importance to different applications, we find that the key and influential metric is the delivery rate in ASNs. For example, ASNs can function as the networking foundation for the notification systems [1] that are widely used in battlefields, financial institutions, emergency services, information technology, weather, government, education, sports, health care and so on. The delivery rate determines the effectiveness of these applications. It also affects other metrics like throughput, latency, overhead, and energy, because they are counted based on the delivered packets. Therefore, in this

paper, we will study new routing protocols that can guarantee the desired delivery rate with a high probability using minimum energy consumption for ASNs that have lossy links.

Achieving desired delivery rate in ASNs poses significant research challenges. First, the nodes' relationships are more complicated: Node A can directly transmit to B, but B may not be able to transmit to A directly. Second, because of the asymmetry, it is harder to get the feedback information such as delivery probability from a neighbor. Third, the path to send the message and the path to get the acknowledgement back may not be the same.

To address these challenges, inspired by [25], we first propose a general framework protocol called *RP (Reverse Path)* that finds reverse paths for asymmetric links to overcome the difficulty caused by them, and then design routing algorithms on the reverse paths to satisfy the performance requirements. Different from the distance-vector-based [21] reverse path protocol in [25], RP is a source-routing-based [9] protocol to allow easier troubleshooting and network performance management. Our first routing protocol built on RP that guarantees performance requirements for ASNs is ProHet [4], which is a probabilistic-based protocol for networks formed by heterogeneous nodes with different transmission ranges. It is a reactive algorithm suitable for large and more dynamic networks. In this paper, we will study efficient proactive routing protocols *LayHet* (a layer-based protocol) and its upgraded version *EgyHet* (an energy-efficient protocol) for less dynamic networks such as sensor networks. In its preparation part, LayHet identifies nodes' layer numbers which embed shortest path information for the sake of lossy links to guide routing in the right direction. In its routing part, to guarantee performance and save more energy, LayHet only allows a message holder to broadcast enough times so that at least one selected node will receive and forward the message. To further reduce energy consumption, we upgrade LayHet to EgyHet, where we consider the remaining energy of nodes when selecting forwarders. This paper is an extension of our two protocols LayHet and EgyHet published in conference proceedings [5], [6]. In this version, we make a substantial change in the network scope and related works, provide more detailed analysis and add more simulation results.

The rest of the paper is organized as follows: Section II references the related works; Section III presents the preliminary; Section IV introduces the reverse path protocol; Sections V and VI propose the LayHet and EgyHet protocols; Section VII is the analysis of the protocols; Section VIII evaluates the performance of the two protocols by simulations; And conclusion and future work are in Section IX.

II. RELATED WORKS

Routing protocols are typically designed for symmetric communication networks. Even in heterogeneous sensor networks where sensors have different transmission ranges, researchers assume that the links between nodes are symmetric [7], [10], [27], [28]. The asymmetric links make most of the existing protocols fail or operate with high overhead and low throughput.

To overcome the difficulty caused by the asymmetric links, some people use protocol-specific approaches. In proactive link-state protocols such as OLSR [8], nodes maintain a view of the network topology by communicating with their neighbors. With a complete view, the protocols can find routes in asymmetric networks. But partial views are used to reduce the cost. Garcia-Luna-Aceves and Bao propose a link-state protocol [3] with sufficient view to handle asymmetry. However, it incurs a high cost if views are larger. The proactive distance-vector protocols such as DSDV [21] maintain at each node a distance vector consisting of the length and the first-hop neighbor of the shortest path to other nodes. They assume symmetric links would fail in asymmetric networks. Prakash proposes a modified protocol where a node's distance vector is broadcast over multiple hops so that it can reach the nodes that have asymmetric links. However, the protocol increases the worst-case message size from $O(n)$ to $O(n^2)$ [23]. Reactive protocols such as AODV [22] and DSR [15] use route request packet (RREQ) and route reply (RREP) to discover routes. But in asymmetric networks, RREPs cannot be sent back along the original path. So AODV avoids asymmetric links in its path and DSR supports asymmetric links by sending RREPs on a separate route, which requires an additional route discovery.

In contrast to the above protocol-specific solutions for handling asymmetric links, the BRA protocol proposed by Ramasubramanian and Mosse [25] has the advantage of making the underlying asymmetry transparent to the above routing protocols by building reverse paths for asymmetric links. Prior to BRA, there were a few general-purpose frameworks to handle asymmetric links as well. The IETF working group on Unidirectional Link Routing (UDLR) proposes a protocol [11] that invokes tunneling and encapsulation to send multi-hop acknowledgments at the link layer and Nesargi and Prakash propose a similar tunneling-based protocol where control packets are tunneled through multi-hop reverse routes to the upstream nodes of unidirectional links [19]. However, the protocols do not specify what routes are used for the multi-hop tunnels.

Inspired by the idea in [25], we put forward a reverse path protocol in [4] for asymmetric links. Different from their distance-vector-based [21] reverse path algorithm, we explore source-routing-based [9] reverse path protocol to allow easier troubleshooting and network performance management. Furthermore, in [4], we propose a routing protocol called ProHet built on the reverse path protocol that can guarantee the performance requirements which are not discussed in [25]. ProHet is a reactive algorithm which is suitable for large and more dynamic networks. In this paper, we will focus on designing proactive algorithms for ASNs in relatively static environments such as sensor networks to realize efficient delivery-centric routing.

III. PRELIMINARY

The following are the definitions we use in the paper.

A. DEFINITION OF NODES' NEIGHBOR RELATIONSHIPS

An ASN can be represented by a directed graph $G = \{V, E\}$, where V is the set of sensors (also called nodes), and E is the set of links (also called edges) in the network. For example, if sensor B is in the transmission range of sensor A, then there is a directed link from A to B. We assume graph G generated from the sensor network is a strongly-connected directed graph. So the sensor network is also strongly-connected.

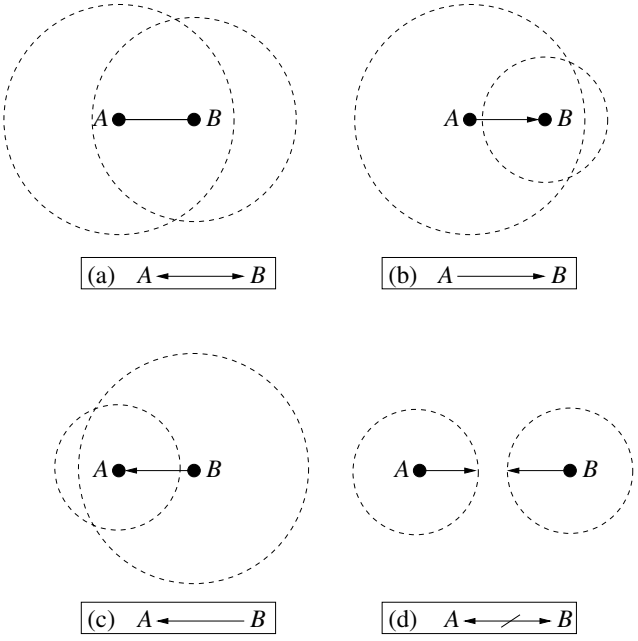


FIGURE 1. The neighbor relationships between two nodes *A* and *B*. (a) *A* and *B* are each other's in-out-neighbor; (b) *A* is the in-neighbor of *B* and *B* is the out-neighbor of *A*; (c) *B* is the in-neighbor of *A* and *A* is the out-neighbor of *B*; (d) *A* and *B* are non-neighbors.

In ASNs, we categorize neighbor relationships of sensors into four categories: (1) in-out-neighbor; (2) in-neighbor; (3) out-neighbor; and (4) non-neighbor. For two nodes A and B, as shown in Fig. 1, if $A \rightarrow B$ and $B \rightarrow A$, then A and B are in-out-neighbors of each other. If only $A \rightarrow B$ (or $B \rightarrow A$), then A (or B) is the in-neighbor of B (or A), and B (or A) is the out-neighbor of A (or B). If neither $A \rightarrow B$ nor $B \rightarrow A$, they are non-neighbors of each other. Also note that if A and B are each other's in-out-neighbors, they are each other's in-neighbor and out-neighbor.

B. DEFINITION OF PACKET LOSS RATE OF A LINK

We assume data is transmitted through lossy links. The packet loss rate of a link uv is defined as 1 minus the ratio of the number of packets N_d which are successfully received by node v and the total number of packets N_s sent by u. It can be expressed as:

$$p_v = 1 - N_d/N_s. \tag{1}$$

To obtain p_v, sender u must know N_d and N_s. N_s is straightforward because whenever u forwards a packet, it increments N_s by one and N_d is sent back as an acknowledgement from the destination in later Algorithm 5. After knowing N_d and N_s, u can calculate p_v locally and timely.

Next, we first present the reverse path protocol RP and then efficient routing protocols LayHet and EgyHet built on RP.

IV. REVERSE PATH PROTOCOL RP

Our idea of establishing the reverse path from a node to its in-neighbor is to let the node broadcast a "Find" message containing the source and destination IDs with an expiration length of 3 hops. We have shown in [4] that if we trace back three hops for each asymmetric link, the connectivity of the network can be up to 90%. Therefore, setting the maximum reverse routing path length to three is appropriate. The details of finding the reverse paths are described in Algorithm 1.

Algorithm 1 RP: Finding Reverse Paths for Asymmetric Links

1: Initialization
   a) Every node in the network broadcasts a "Hello" message.
   b) If two nodes A and B can receive each other's "Hello" message and the corresponding "Ack" of the "Hello" message, then each adds the other to its in-out-neighbor list.
   c) If A receives B's "Hello" message, but not the "Ack" to its own "Hello" message, then A knows that B is its in-neighbor and adds it to its in-neighbor list. Then, A will perform the next step to find a reverse routing path to B.
2: Node A tries to find a reverse routing path to each of its in-neighbors by broadcasting a "Find" message containing the source ID ("A"), the destination ID (the ID of the in-neighbor to which it wants to find a reverse path (e.g. "B")), and an expiration length of 3 hops.
3: If some node C receives a "Find" message,
   1) if it is the destination listed in the message, it will
      a) add the source node to its out-neighbor list;
      b) send the identified reverse routing path to the source node by a "Path" message containing the reverse route.
   2) if it is not the destination node and the expiration length is greater than 0, it will rebroadcast the message after the following modifications:
      a) decrement the expiration length by one;
      b) append its own ID to the message.
   3) in all other cases, it will drop the message.

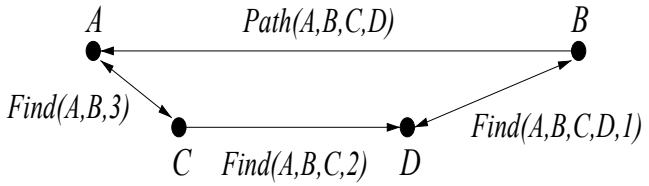


FIGURE 2. **An example of finding a reverse path.**

We use the example in Fig. 2 to explain the algorithm 1. There are four sensors A, B, C and D in the graph. Take sensors A and B as an example. If sensor A receives B's "Hello" message, but not the "Ack" to its own "Hello" message, then A knows that B is its in-neighbor and adds it to its in-neighbor list. Then, sensor A wants to find a reverse path to B. Sensor A broadcasts a message "Find(A,B,3)" containing the source ID, the destination ID and an expiration length of 3 hops. This message is received by sensor C, since C is not the destination node, it decrements the expiration length by 1, appends its own ID to the message and broadcasts the changed message "Find(A,B,C,2)". Then sensor D gets the message. Since it is not the destination either, it does the same thing as C and broadcasts the changed message "Find(A,B,C,D,1)". When the destination sensor B gets the message, it adds the source node A to its out-neighbor list and sends the identified reverse routing path to A by the "Path(A,B,C,D)" message.

Next, we present two routing protocols LayHet and EgyHet built on RP for ASNs.

V. LAYHET ROUTING PROTOCOL

In this section, we propose the LayHet protocol that is built on RP. The protocol has two parts: The preparation part which includes assigning layer numbers to the nodes and adjusting layer numbers periodically; And the routing part which includes the sender broadcasting H times and the receivers forwarding messages with probabilities estimated from link states with neighbors. The details are as follows:

A. PREPARATION

1) DECIDING INITIAL LAYER NUMBERS

This part is done by Algorithm 2 where each node finds its layer number which represents its shortest hop count to the sink. A node u broadcasts an exploration packet EP containing a hop-count 0 and its ID to the sink. On the way, the hop-count is incremented and the path is recorded. After the sink receives EP, it waits for a while for more copies of EP to arrive as a result of broadcast. Then it picks the EP with the smallest hop-count. The sink increments the smallest hop-count by 1. That becomes the hop-count c from u to the sink. Then the sink sends back an Ack of EP containing c to u via all the forwarding nodes on the path. Because of the asymmetric links, the reverse paths may be used. When u receives c, it knows that its layer number to the sink is c. A good point of the algorithm 2 is that each node may have multiple chances to adjust its initial layer number: Once by the Ack from the sink addressing itself, others by the Acks from the sink addressing other nodes if it is the relay node on the paths. Multiple adjustments are necessary because of the lossy links. The closer the node is to the sink, the more accurate its layer number can be because it is more likely to be a relay node and thus has more chances to adjust its layer number. The accuracy of the layer numbers for lower layer nodes is more important than that for the higher layer nodes since lower layer nodes are more likely to relay messages for others.

Algorithm 2 DILN: Deciding Initial Layer Numbers

1: Node u broadcasts an exploration packet EP containing a hop-count $c = 0$ and the source ID.
2: **if** a node v receives EP **then**
3: **if** it is the sink node **then**
4: it waits for a while for more copies of EP to arrive. Then it picks an EP with the smallest hop count. It increments the hop count by 1 and generates an acknowledgement EP_{ack} containing the value of the current hop count c and the path involving all the forwarding nodes on the path back to the source u. The later arrived copies of EP are dropped.
5: When an intermediate node m on the path receives EP_{ack}, it adjusts its own layer number according to hop count c and its location on the path.
6: **if** m's previous node t is its in-out-neighbor **then**
7: it sends EP_{ack} directly to t;
8: **else if** m has a reverse path to t **then**
9: m sends EP_{ack} to t via the reverse path of the asymmetric link $t \rightarrow m$;
10: **else**
11: m simply drops EP_{ack}
12: **end if**
13: **else**
14: it increments the hop count by 1, appends its ID to EP and rebroadcasts EP
15: **end if**
16: **end if**
17: After u receives EP_{ack}, it knows its layer number to the sink is c.

2) ADJUSTING LAYER NUMBERS

Due to the lossy links, some nodes may not be put into the right layers initially. So the layer numbers of nodes need to be adjusted. To reduce the overhead, the adjustment of layer numbers can be embedded in Algorithm 5 in Section V-B. When a node u communicates with its out-neighbor v to find out the packet loss rate of link uv, the out-neighbor also sends back its layer number. If u's layer number is at least 2 more than v's layer number, u will adjust its layer number to v's layer number plus 1.

Algorithm 3 BRD-H: BRoaDcasting H Times

1: Except at the beginning when the packet loss rates are generated randomly, source node u finds out the packet loss rates $p_1, p_2, \ldots, p_K$ with its K lower layer out-neighbors using Algorithm 5.
2: Node u calculates the number of times H it should broadcast using Formula (4) in Section VII.
3: Node u broadcasts the message plus its link packet loss rates $p_1, p_2, \ldots, p_K$ H times.

Algorithm 4 FWD-M: ForWarDing Messages

1: **repeat**
2: If a node v receives a message from a higher layer neighbor u along with the packet loss rates of u's links, it uses Formula (5) in Section VII to decide its probability Γ to forward the message.
3: If it forwards, it becomes the new source and applies the algorithm 3.
4: If it does not forward, it will simply drop the message.
5: **until** the message reaches the sink.

B. ROUTING

In this section, we present the routing part of LayHet to achieve the desired delivery rate using local information. Our routing protocol contains three phases: Broadcasting H times, Forwarding messages, and Updating packet loss rate periodically which are presented in Algorithms 3, 4 and 5, respectively. The desired delivery rate is preset to Δ. Before the routing in the network begins, the packet loss rates of the links between a node u and its K lower layer out-neighbors are generated randomly because the network does not have any routing history. Later, the packet loss rates are updated by Algorithm 5 periodically so that the next routing can be guided by more accurate information in the network. After u knows the packet loss rates of the links, it broadcasts the message it wants to send to the sink H times so that at least one of its K lower layer out-neighbors can receive the message in order to achieve the desired delivery rate Δ. Next, in the algorithm 4, a receiving node v will forward the message at a probability of Γ to avoid flooding the network with unnecessary messages to save energy. The calculation of H and Γ is presented in Section VII.

We use the wireless sensor network in Fig. 3 to explain the LayHet protocol. In the figure, each black dot represents a sensor, which is responsible for collecting data. The right most node is the sink, which is responsible for processing data after collection. All the other nodes will send the collected data to the sink. In the initialization of the layer numbers, each node applies Algorithm 2 to determine its layer number to the sink. After the initialization, the nodes are put into different layers relative to the sink. Because of the lossy links in the network, some nodes may not be put in the right layers initially. But the node layer numbers can be adjusted by Algorithm 5 later.

Algorithm 5 UPR-P: Updating Packet Loss Rate Periodically

1: Each node u will update the packet loss rate of each of its links with its out-neighbors every T time period.
2: Suppose node u sends out N_s messages to node v during T time period. At the end of T, node u sends a message to v asking "How many messages out of N_s have you received?".
3: After v receives the inquiry, it replies directly or through the reverse path with the answer "N_d". Also it attaches to the message its layer number for u to adjust its layer number.
4: After u receives the answer, it updates the packet loss rate of link uv to $1 - \frac{N_d}{N_s}$. Also if u's layer number is at least 2 more than v's layer number, u adjusts its layer number to v's layer number $+1$.

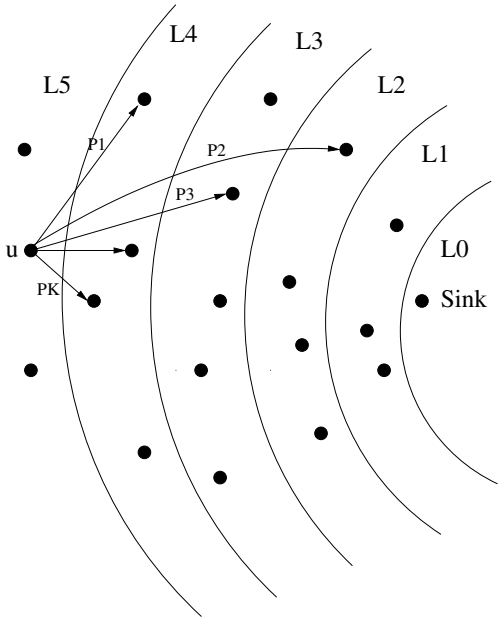


FIGURE 3. A data forwarding scenario.

After the preparation, routing is carried out. Suppose a source node u in Layer L_5 wants to send a message to the sink. It has K out-neighbors in the lower one-hop, two-hop and three-hop layers. A node may have a one-hop out-neighbor in the lower two or three-hop layers because we consider opportunistic communications exploiting the nature of broadcast. Based on the packet loss rates of the links to these out-neighbors, u broadcasts the message H times calculated by Formula (4) in Section VII. This guarantees that at least one of these neighbors will receive the message with a high probability. Then each of these receivers will decide the probability Γ to forward the message by Formula (5) in Section VII. The purpose of the forwarding probability is to reduce the number of replicated packets in the network. If a node chooses to forward, it becomes the new source and reapplies the routing protocol. Then hop by hop, the message

will be relayed to the sink. Every T period of time, a sender will update the packet loss rate of each of its links to its out-neighbors so that the calculations of H and Γ can be more accurate next time.

Algorithm 6 CAL-H: CALculating H Value and Selecting Forwarders

1: Order node u's K lower layer out-neighbors in non-increasing order according to their remaining energy levels. Here we use a node's remaining energy level to represent the node. Suppose the sequence $E = \{E_1, E_2, \ldots, E_K\}$.
2: $S = \{E_1\}, i = 0$.
3: Calculate H based on sequence S using Formula (4) in Section VII.
4: **repeat**
5: $H_{pre} = H, i = i + 1$.
6: **if** $i == K + 1$ **then**
7: return H and S
8: **end if**
9: $S = S \cup \{E_i\}$.
10: Calculate H based on sequence S using Formula (4) in Section VII.
11: **until** $H == H\_pre$
12: return H and $S = S - \{E_i\}$.

VI. EGYHET ROUTING PROTOCOL

LayHet can be more energy-efficient if the remaining energy of nodes is considered. We upgrade LayHet to EgyHet by making the following changes: Algorithm 3 calls Algorithm 6 to select only a subset of K lower layer out-neighbors according to their remaining energy as forwarders to satisfy the desired delivery rate. In Algorithm 6, S represents the selected forwarding nodes with the highest remaining energy. First, the K lower layer out-neighbors of node u are ordered in non-increasing order based on their remaining energy levels. Then starting from the node with the highest remaining energy, we add node one by one to S. After adding a new node, we use Formula (4) in Section VII to calculate H. The H value may be reduced with the increase of the node number in S. The algorithm stops if the newly added node does not reduce H any more or if all of the K nodes are added. After the H value is known, u broadcasts the message containing the packet it wants to send to the sink, the selected forwarding nodes in S and its link loss rates to the forwarding nodes H times so that at least one of the lower layer out-neighbors can receive the message with a high probability in order to achieve the desired delivery rate Δ. In Algorithm 4, only the selected nodes will decide their probabilities to forward. The unselected ones will simply drop the message.

VII. ANALYSIS

In this section, we first give the upper-bound of the desired delivery rate that a network can achieve, then show that our algorithms can guarantee the performance if H and Γ are properly calculated.

A. UPPER-BOUND OF DESIRED DELIVERY RATE

Obviously, if the delivery rate Δ is set too high, the delivery probabilities of nodes in the network are too low, or the network is too sparse, the desired delivery rate cannot be achieved. So we need to find the upper-bound of Δ to make it achievable.

Assume node u has a total of K lower layer out-neighbors to select from whose delivery probabilities are $p_1, p_2, \ldots, p_K$. To achieve the desired delivery rate, the following relation must be satisfied:

$$1 - (1 - p_1)(1 - p_2) \cdots (1 - p_K) \geq 1 - (1 - p_{min})^K$$
$$\geq 1 - (1 - p_{min})^{out\text{-}d_{min}} \geq \Delta \quad (2)$$

In the formula, p_{min} is the minimum delivery probability of nodes in the whole network. The value $out\text{-}d_{min}$ represents the minimum K in the whole network. So $K \geq out\text{-}d_{min}$. The values of p_{min} and $out\text{-}d_{min}$ can be obtained after a network has been set up and several rounds of packet delivery have been conducted. So Δ is upper-bounded by

$$\Delta \leq 1 - (1 - p_{min})^{out\text{-}d_{min}} \quad (3)$$

The inequality indicates that the achievable deliver rate Δ depends on and is bounded by the nodes' delivery probabilities and the network density. In this paper, when we set the desired delivery rate Δ, we make sure that Condition (3) is satisfied.

B. PERFORMANCE GUARANTEE

Next, we provide an analysis of our routing algorithms LayHet and EgyHet to show that if H and Γ are properly selected, there is a high chance that the routing can achieve the desired delivery rate Δ and reduce the number of replicated messages in the network. Since EgyHet considers more network parameters than LayHet, the analysis will follow EgyHet. The analysis for LayHet is exactly the same by involving all the K lower layer out-neighbors without considering their remaining energy. To show the analysis more clearly, we do it reversely, that is: Given the desired delivery rate Δ, decide the number of broadcasts H and the forwarding probability Γ to meet the desired delivery rate and reduce the number of replicated messages. We assume a node u knows the packet loss rates $p_1, p_2, \ldots, p_K$ of all of its links to K lower layer out-neighbors and the remaining energy of these out-neighbors through initial random setting or periodic message exchanges in Algorithm 5.

Assume node u at layer i (see Fig. 3) has a total of K lower layer out-neighbors. It wants to broadcast a message to k ($k \leq K$) selected lower layer forwarding nodes based on their remaining energy. In the LayHet protocol, $k = K$. In EgyHet, k can be less than K if H remains unchanged if adding one more node. Because of lossy links, it may need to broadcast multiple times to make sure that at least one

node in the forwarding set can receive the message. Normally the message needs to travel i hops to reach the sink. Assume the transmission in each layer is the same. To guarantee the overall delivery rate Δ, in each layer, we should guarantee the success rate of transmission to be at least $\Delta^{\frac{1}{i}}$. We assume the loss rate of the link from u to its j-th forwarding node is p_j. A transmission is successful if at least one of the forwarding nodes receives the message. That is, the probability

$$\begin{aligned} &Pr\{\textit{at least one selected node receives the message after } H \textit{ transmissions}\} \\ &= 1 - (\prod_{j=1}^{k} p_j)^H \geq \Delta^{\frac{1}{i}} \\ \text{Then, } \quad H &\geq \frac{ln(1-\Delta^{\frac{1}{i}})}{\sum_{j=1}^{k} ln(p_j)} \end{aligned} \tag{4}$$

We set H to the minimum integer that can satisfy Formula (4). After node u broadcasts the message H times, the message is transmitted to one or more forwarding nodes with high probability. To reduce redundancy and save energy, not all the selected nodes receiving the message will forward the message. The receivers only forward the message with probability Γ and drop the message with probability $1-\Gamma$. Given that a message has been received by a few forwarding nodes, we should make sure that at least one node will forward the message. That probability

$$\begin{aligned} &Pr\{\textit{at least one selected node will forward the message}\} \\ &= 1 - Pr\{\textit{no node will forward the message}\} \\ &= 1 - \prod_{j=1}^{k} (Pr\{\textit{the jth node does not receive the message}\} + Pr\{\textit{the jth node receives the message}\} \cdot Pr\{\textit{the jth node doesn}'\textit{t forward the message}\}) \\ &= 1 - \prod_{j=1}^{k} (p_j^H + (1-p_j^H)(1-\Gamma)) \\ &= 1 - \prod_{j=1}^{k} (1 - \Gamma + p_j^H \Gamma) \geq \Delta^{\frac{1}{i}} \end{aligned}$$

Then, $1-\Delta^{\frac{1}{i}} \geq \prod_{j=1}^{k}(1-\Gamma+p_j^H\Gamma) \geq (1-\Gamma+p_{min}^H\Gamma)^k$, in which p_{min} is the minimum value of p_j, $(1 \leq j \leq k)$.

Solving this inequality yields

$$\Gamma \geq \frac{1-(1-\Delta^{\frac{1}{i}})^{\frac{1}{k}}}{1-p_{min}^H} \tag{5}$$

Parameter Γ can be set to the minimum value that satisfies Formula (5). Now that H and Γ are calculated according to requirements, there is a high chance that the routing can achieve the desired delivery rate Δ and reduce the number of replicated messages in the network.

VIII. SIMULATIONS

In this section, we evaluate the performance of our protocols LayHet and EgyHet using a self-written simulator in Java language. We compare them with our previous work ProHet in [4] because to the best of our knowledge, ProHet is the only one that handles asymmetric links with performance guarantee. The BRA protocol in [25] deals with asymmetric links but does not consider delivery rate and is shown not to guarantee delivery rate by simulation [4]. Therefore, we compare our algorithms with ProHet.

A. COMPARISON OF PROTOCOLS FOR ASNs

We use the following metrics to evaluate the performance of the protocols:

- Delivery rate: the ratio of the number of packets successfully delivered to the sink to the total number of packets generated.
- Average hops: the average hops of a packet successfully sent from a source to a sink.
- Average packet replication overhead: the average number of packet replications used to successfully deliver a packet.
- Average control message overhead: the average number of control messages which include all of the communication messages (except the main packet) to identify neighbors, find reverse paths and update nodes' delivery probabilities needed to successfully deliver a packet.

In our experiments, nodes were randomly deployed in a 500 m $\times$ 500 m area. To diversify the transmission ranges of the nodes, we used the idea in [25] to let a node have one of the three transmission ranges: the *minimum* (40 m), the *normal* (50 m), and the *maximum* (60 m). A transmission range was selected randomly out of the three for a node in simulation. We also considered link loss and randomly set the link loss rates to be between 0% and 20%. To implement message sending and receiving, a virtual concept of *time slots* was used. In each time slot, we randomly chose a sensor to generate a new message and let it send the message to the sink. Each node used a buffer to cache packets from other nodes. Assume all packets in the buffer could be transmitted to the next hop node within one time slot. The simulation time was set to 1000 time slots. During the experiments, we randomly generated 30 different deployments of asymmetric sensor networks, set the desired delivery rates to be 99% (very high) and 80% (medium), and varied the number of nodes from 200 to 475 with a step of 25 and averaged the simulation results.

The simulation results are shown in Figs. 4 and 6. Since LayHet and EgyHet have similar results, we add Figs. 5 and 7 to show their differences better. From the results, we can see that all of the three algorithms can guarantee the desired delivery rate after the network density reaches a certain level. This is because with the increase of network density, the connection between nodes increases, so a message can get more chances to be delivered to the sink. Also LayHet and EgyHet

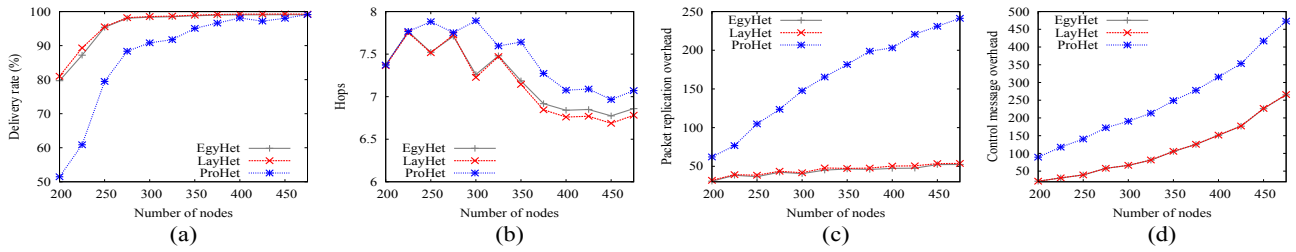


FIGURE 4. Comparison of ProHet, LayHet and EgyHet with assurable delivery rate 99%. (a) Average delivery rate. (b) Average hops. (c) Average packet replication overhead. (d) Average control message overhead.

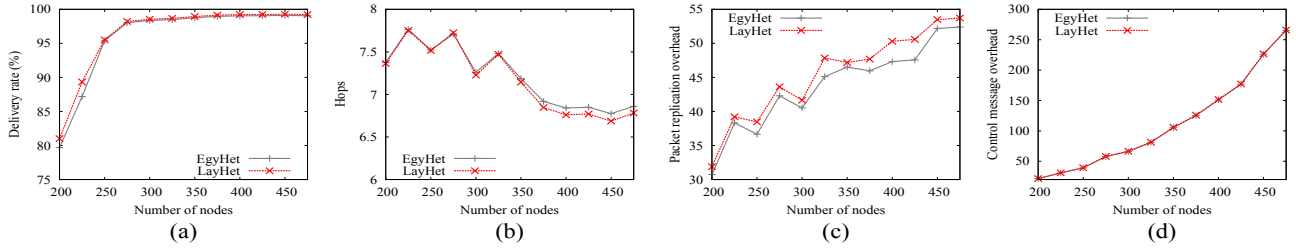


FIGURE 5. Comparison of LayHet and EgyHet with assurable delivery rate 99%. (a) Average delivery rate. (b) Average hops. (c) Average packet replication overhead. (d) Average control message overhead.

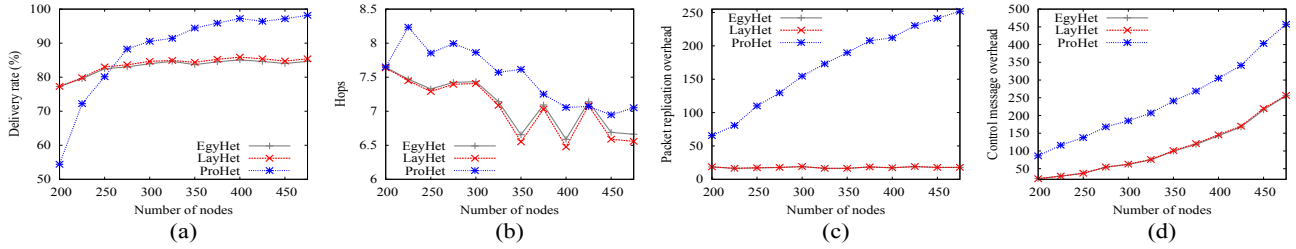


FIGURE 6. Comparison of ProHet, LayHet and EgyHet with assurable delivery rate 80%. (a) Average delivery rate. (b) Average hops. (c) Average packet replication overhead. (d) Average control message overhead.

can reach the desired delivery rate earlier than ProHet. After the network becomes dense enough, for example, with more than 250 nodes in the 80% desired delivery rate setting, ProHet's delivery rate will continue increasing but LayHet and EgyHet's delivery rates will keep at the 80% delivery rate level. This is because LayHet and EgyHet are designed to satisfy the desired delivery rate but do not go over that for the purpose of saving energy. In ProHet, on the other hand, more nodes will receive the message in denser topologies by the nature of opportunistic routing, so its delivery rate will eventually reach 100%. Comparing the delivery rates of LayHet and EgyHet, LayHet is better due to the fact that EgyHet uses a subset of LayHet's forwarders in each forwarding. For the number of hops to send a message from a source to the sink, LayHet and EgyHet are better than ProHet because the layer numbers in them embed the shortest path information. But the average hops of ProHet are also close to the ideal results. EgyHet has a little higher hop number than LayHet again because of its using a subset of LayHet's forwarders in each forwarding. The major improvement of LayHet and EgyHet over ProHet lies in the packet replication overhead and the control message overhead. The packet replication overhead of LayHet and EgyHet is substantially less than that of ProHet and the reduction in control message overhead in LayHet and EgyHet is also large even with their initial overhead to set up layer numbers counted. This indicates that the proactive protocols LayHet and EgyHet can save a lot of overhead in each hop by identifying node layers at the beginning whereas the reactive protocol ProHet has more overhead in each hop trying to discover the route. The packet replication overhead and control message overhead of EgyHet are smaller than those of LayHet (though not very obvious in Figs. 5(d) and 7(d)) for the same subset reason and thus proves the improvement in EgyHet.

B. EFFECT ON DELIVERY RATE WHEN SENSOR ENERGY IS REDUCED

In this simulation, we want to see how the delivery rates of protocols are affected if sensor energy reduces.

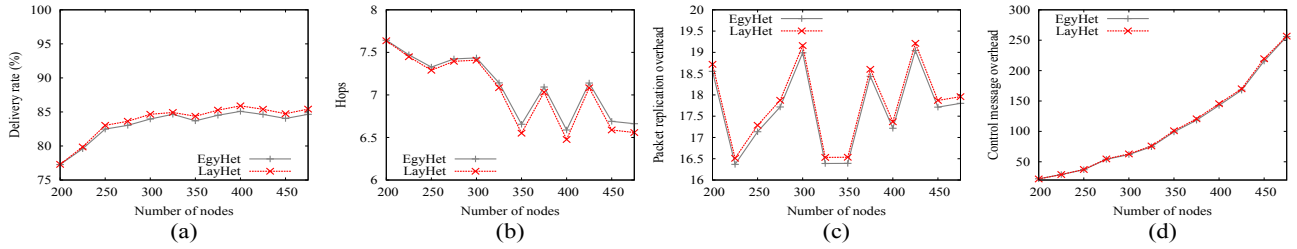


FIGURE 7. Comparison of LayHet and EgyHet with assurable delivery rate 80%. (a) Average delivery rate. (b) Average hops. (c) Average packet replication overhead. (d) Average control message overhead.

1) ENERGY MODEL

We assume that each node u has a finite and unreplenishable initial energy e_u, which is a non-negative integer value. For the energy consumption of sending and receiving a message by a node, we adopt the first order radio model [14] where for k-bit data over distance l, the transmission energy $E_T(k, l)$ and the receiving energy $E_R(k)$ are calculated as follows:

$$E_T(k, l) = E_{elec} \times k + \epsilon_{amp} \times k \times l^2 \quad (6)$$

$$E_R(k) = E_{elec} \times k \quad (7)$$

where $E_{elec} = 50$ nJ/bit and $\epsilon_{amp} = 100$ pJ/bit/m^2. When the distances among nodes are in the order of one hundred meters, the term with ϵ_{amp} is much larger than the term with E_{elec}. Therefore, we assume that for each node, sending one unit-sized message costs one unit of energy while receiving one message costs zero energy.

2) PARAMETER SETTING

We again assume that, in our simulations, a sensor has one of the three transmission ranges 40 m, 50 m or 60 m and a link loss rate between 0% and 20% initially. We randomly generated 30 different deployments of ASNs, set the desired delivery rate to be 99% and randomly generated the sink and the sources. We tried from 10 to 550 messages with a step of 50. In the routing process, we set the ratios of the length of a control message and that of a regular message to be 1:25 and 1:50. We set the initial energy level for each node to be 2000 when the ratio is 1:25 and 4000 for each node when the ratio is 1:50. Whenever a sensor sent a packet, some energy would be deducted from its energy level using the above energy model. The simulation results are shown in Fig. 8(a)(b).

From the figures, we can see that, with the sending of the messages, the remaining energy of sensors decreases and after a certain point, the network can not satisfy the desired delivery rate any more. Regardless of message length ratios and nodes' remaining energy levels, the delivery rate of EgyHet is better than those of LayHet and ProHet, which justifies the energy consideration in EgyHet. In Fig. 8(a), when the number of messages is between 90 and 200 and in Fig. 8(b), when the number of messages is between 90 and 250, ProHet's delivery rate is better than that of LayHet. This is because LayHet tries to use the shortest path between a source and the sink.

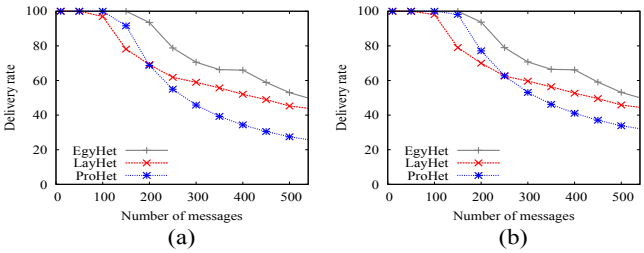


FIGURE 8. Comparison of ProHet, LayHet and EgyHet's delivery rates as sensors run out of energy. (a) Delivery rate with control message and regular message length ratio 1:25. (b) Delivery rate with control message and regular message length ratio 1:50.

The failure of the sensors on or close to the shortest path will make the routing more difficult. On the other hand, ProHet can use detours so that its delivery rate during this section does not decrease as much as LayHet's. But ProHet depletes nodes' energy faster. So eventually its delivery rate falls faster.

IX. CONCLUSION

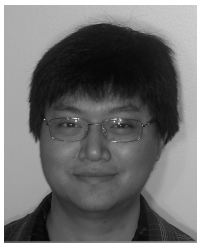
In this paper, we designed performance guaranteed routing protocols in asymmetric sensor networks where two end nodes may not use the same path to communicate with each other. To address the difficulty caused by the asymmetric links, we first proposed a general framework protocol RP that finds reverse paths for asymmetric links. Then we presented two efficient routing algorithms LayHet and EgyHet built on RP to satisfy performance requirements. Simulation results showed that LayHet and EgyHet can reach the desired delivery rate earlier than the existing protocol and outperformed it in terms of average hops, average packet replication overhead and average control message overhead. Furthermore, LayHet and EgyHet's performance degrades more slowly than the existing one as sensors run out of their energy. In this paper, we focused on designing efficient routing protocols on the top of the reverse path protocol RP. The study of the reverse path protocol itself and the comparison with the one proposed by Ramasubramanian and Mosse can be an independent topic, which we will leave for the future work. We believe asymmetric links are very common in many wireless networks. Besides the spatial reasons mentioned in this paper, they can be the result of the time dependency of nodes' connections such as in the case of delay tolerant networks, vehicular networks and

mobile social networks. In the future, we will study efficient routing algorithms in these wireless asymmetric networks.

REFERENCES

[1] *Notification System* [Online]. Available: http://en.wikipedia.org/wiki/Notification_system
[2] I. Alyildiz, Y. Sankarasubramaniam W. Su, and E. Cayicrci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–116, Aug. 2002.
[3] L. C. Bao and J. J. Garcia-Luna-Aceves, "Link-state routing in networks with unidirectiona links," in *Proc. IEEE ICCCN*, Oct. 2005, pp. 358–363.
[4] X. Chen, Z. X. Dai, W. Z. Li, Y. F. Hu, J. Wu, H. C. Shi, and S. L. Lu, "Prohet: A probabilistic routing protocol with assured delivery rate in wireless heterogeneous sensor networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1524–1531, Apr. 2013.
[5] X. Chen, Z. X. Dai, W. Z. Li, and H. C. Shi, "A layer-based routing protocol for heterogeneous wireless sensor networks," in *Proc. IEEE ICC*, Jun. 2012, pp. 228–232.
[6] X. Chen, Z. X. Dai, and H. C. Shi, "Egyhet: An energy-saving routing protocol for wireless heterogeneous sensor networks," in *Proc. ICNC*, Jan. 2013, pp. 778–782.
[7] X. Chen, W. Y. Qu, H. L. Ma, and K. Q. Li, "A geography–based heterogeneous hierarchy routing protocol for wireless sensor networks," in *Proc. 10th IEEE HPCC*, Sep. 2008, pp. 767–774.
[8] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *Proc. IEEE INMIC*, Mar. 2001, pp. 62–68.
[9] R. C. Dixon and D. A. Pitt, "Addressing, bridging, and source routing (LAN interconnection)," *IEEE Netw.*, vol. 2, no. 1, pp. 25–32, Jan. 1988.
[10] X. Du, M. Guizani, X. Yang, and H. H. Chen, "Two tier secure routing protocol for heterogeneous sensor networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 9, pp. 3395–3401, Sep. 2007.
[11] E. Duros, W. Dabbous, H. Izumiyama, N. Fujii, and Y. Zhang, *A Link Layer Tunneling Mechanism for Unidirectional Links*. New York, NY, USA: RFC Editor, 2001.
[12] I. A. Essa, "Ubiquitous sensing for smart and aware environments," *IEEE Personal Commun.*, vol. 7, no. 5, pp. 47–49, Oct. 2000.
[13] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "An empirical study of epidemic algorithms in large scale multihop wireless networks," Intel Corp., Santa Clara, CA, USA, Tech. Rep. IRB-TR-02-003, Mar. 2002.
[14] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. HICSS*, Jan. 2000.
[15] D. B. Johnson and D. A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*. Norwell, MA, USA: Kluwer, 1996.
[16] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *Proc. 10th Int. Conf. ASPLOS*, Oct. 2002, pp. 96–107.
[17] A. M. Mainwaring, D. E. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM Int. Workshop WSNA*, 2002, pp. 88–97.
[18] M. K. Marina and S. R. Das, "Routing performance in the presence of unidirectional links in multihop wireless networks," in *Proc. IEEE Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 2002, pp. 85–97.
[19] S. Nesargi and R. Prakash, "A tunneling approach to routing with unidirectional links in mobile ad hoc networks," in *Proc. 9th ICCCN*, Oct. 2000, pp. 522–527.
[20] D. A. Patterson, "Rescuing our families, our neighbors, and ourselves," *ACM Commun.*, vol. 48, no. 11, pp. 29–31, Nov. 2005.
[21] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. Conf. Commun. Archit., Protocols Appl.*, Oct. 1994, pp. 234–244.
[22] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad hoc on-demand distance vector (AODV) routing," Ph.D. dissertation, Sun Microsyst. Lab., Adv. Develop. Group, Menlo Park, CA, USA, Jun. 2002.
[23] R. Prakash, "Unidirectional links prove costly in wireless ad hoc networks," in *Proc. 3rd Int. Workshop Discrete Algorithms Methods Mobile Comput. Commun.*, Aug. 1999, pp. 15–22.
[24] J. M. Rabaey, M. J. Ammer, J. L. da Silva, Jr., D. Patel, and S. Roundy, "PicoRadio supports ad hoc ultra-low power wireless networking," *IEEE Comput.*, vol. 33, no. 7, pp. 42–48, Jul. 2000.
[25] V. Ramasubramanian and D. Mosse, "BRA: A bidirectional routing abstraction for asymmetric mobile ad hoc networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 116–129, Feb. 2008.
[26] A. Sixsmith and N. Johnson, "A smart sensor to detect the falls of the elderly," *IEEE Pervas. Comput.*, vol. 3, no. 2, pp. 42–47, Apr./Jun. 2004.
[27] Q. Zhang and W. G. Chang, "A power efficiency routing protocol for heterogeneous sensor networks," in *Proc. 4th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Oct. 2008, pp. 1–4.
[28] W. Y. Zhang, X. J. Du, J. Wu, S. D. Soysa, and Y. Liu, "Near-minimum-energy routing in heterogeneous wireless sensor networks," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2010, pp. 1–5.

XIAO CHEN is an Associate Professor of computer science with Texas State University, San Marcos, TX, USA. She received the Ph.D. degree in computer engineering from Florida Atlantic University, Boca Raton, FL, USA. Her current research interests include sensor and *ad hoc* wireless networks. She served as an Associate Editor, a Program Committee Member, a Session Chair, and a Reviewer of numerous international journals and conferences.



ZANXUN DAI received the master's degree of computer science with Texas State University-San Marcos, San Marcos, TX, USA, and the bachelor's degree in software engineering from the Harbin Institute of Technology, Harbin, China. His current research interests include sensor networks and *ad hoc* wireless networks.

WENZHONG LI received the B.S. and Ph.D. degrees from Nanjing University, Nanjing, China, both in computer science. He is currently an Associate Professor of computer science with Nanjing University. His current research interests include wireless networks, pervasive computing, and social networks. He published over 30 papers at international conferences and journals. He was the winner of the Best Paper Award of ICC in 2009. He is a member of ACM.

HONGCHI SHI is a Professor and the Chair of computer science with Texas State University, San Marcos, TX, USA. Prior to joining Texas State University, he was an Assistant/Associate/Full Professor of computer science and electrical and computer engineering with the University of Missouri, Columbia, MO, USA. He received the Ph.D. degree in computer and information sciences from the University of Florida, Gainesville, FL, USA. His current research interests include parallel and distributed computing, wireless sensor networks, neural networks, and image processing. He has served on many organizing and/or technical program committees of international conferences. He is a member of ACM.