A Learning-based Adaptive Routing Tree for Wireless Sensor Networks

Ying Zhang and Qingfeng Huang
Palo Alto Research Center (PARC) Inc.
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
Emails:{yzhang, qhuang}@parc.com

Abstract— **One of the most common communication patterns in sensor networks is routing data to a base station, while the base station can be either static or mobile. Even in static cases, a static spanning tree may not survive for a long time due to failures of sensor nodes. In this paper, we present an adaptive spanning tree routing mechanism, using real-time reinforcement learning strategies. We demonstrate via simulation that without additional control packets for tree maintenance, adaptive spanning trees can maintain the "best" connectivity to the base station, in spite of node failures or mobility of the base station. And by using a general constraint-based routing specification, one can apply the same strategy to achieve load balancing and to control network congestion effectively in real time.**

Index Terms— **constraint-based routing, real-time reinforcement learning, routing tree, wireless sensor networks**

I. Introduction

Large-scale ad-hoc networks of wireless sensors have become an active topic of research. Such networks share the following properties:

- *embedded routers* – each sensor node acts as a router in addition to sensing the environment;
- *dynamic networks* – nodes in the network may turn on or off during operation due to unexpected failure, battery life, or power management; attributes associated with those nodes (locations, sensor readings, load, etc.) may also vary over time;
- *resource constrained nodes* – each sensor node tends to have small memory and limited computational power;
- *dense connectivity* – the sensing range in general is much smaller than the radio range, and thus the density required for sensing coverage results in a dense network;
- *asymmetric links* – the communication links are not reversible in general.

Applications of sensor networks include environment monitoring, traffic control, building management, object tracking, etc.

This paper is based on "Adaptive Tree: A Learning-based Meta-Routing Strategy for Sensor Networks," by Y. Zhang and Q. Huang, which appeared in the Proceedings of the IEEE 3rd Consumer Communications and Networking Conference 2006, Las Vegas, USA, January 2006. © 2006 IEEE.

This work was funded in part by the Defense Advanced Research Project Agency contract #F33615-01-C-1904.

Routing in sensor networks has very different characteristics than that in traditional communication networks. First of all, address-based destination specification is replaced or augmented by a more general feature-based specification [1], such as geographic location [2] and information gain [3], or a fixed but maybe mobile sink. Secondly, routing metrics are not just shortest delays, but also energy usage [4] and information density [3]. Thirdly, in addition to peer-to-peer communication, multicast (one-to-many) and converge-cast (many-to-one) are major traffic patterns in sensor networks. Even for peer-to-peer communication, routing is more likely to be source or sink driven than table-based [5], and source/sink pairs often are dynamic (changing from time to time) or mobile (moving during routing).

We have proposed Message-initiated Constraint-Based Routing (MCBR) [6] for wireless ad-hoc sensor networks. MCBR is a framework of routing mechanisms composed of the explicit specification of constraint-based destinations, route constraints and QoS requirements for messages, and a set of QoS-aware meta-strategies. With the separation of routing specifications from routing strategies, general-purpose *meta* routing strategies can be applied. In contrast to most existing ad-hoc routing strategies with no QoS support, MCBR takes QoS specifications into account.

We have also proposed three *distributed* meta routing strategies based on real-time reinforcement learning [7]: real-time search [8], constrained flooding [9], and adaptive spanning tree [10]. All of these use the same reinforcement learning core, which estimates and updates the cost from the current node to the destination. This approach has a number of attractive properties: (1) explicit use of destination and QoS specifications for finding optimal routes; (2) automatic adaptation to different routes when network conditions change; (3) no need for extra maintenance packets; and (4) no infinite looping if a path to the destination exists. In contrast to most existing QoS routing, learning-based meta-strategies do not create and maintain explicit routes; instead, packets discover and improve the routes over time.

Adaptive trees, different from real-time search and constrained flooding routing strategies, do require a known sink at its initialization. A spanning tree is constructed at initialization, but automatically maintained during the routing process. Similar to the search-based meta strategy [8], implicit confirmation and retransmission are used, rather than sending periodical maintenance packets from the sink node as in many tree-based routing protocols. The basic ideas and protocol of adaptive trees have been presented previously [7], [10]. The new contributions of this paper include defining new routing metrics for energy-aware load balancing to increase lifetime, and for congestion-aware routing to reduce latency and increase reliability.

In this paper, we present an in-depth study of the adaptive tree protocol, as well as performance analysis and comparisons with other peer routing protocols. Simulation results show that the adaptive tree protocol is robust in dynamic and unreliable environments, and can be applied to achieve load balancing and to control network congestion effectively in real time.

The remainder of the paper is organized as follows. Section II surveys most recent protocols developed in mobile ad hoc and sensor network communities. Section III presents the main idea in the adaptive tree protocol. Section IV analyzes the protocol via simulation and compares performances with peer routing protocols. We show that (1) without changing the routing strategy, by using the right routing metrics, adaptive trees can be energy-aware or congestion-aware, to increase lifetime or to reduce latency, and (2) without additional control packets for tree maintenance, adaptive spanning trees can maintain the "best" connectivity to the base station, in spite of node failures or mobility of the base station. Section V concludes the paper.

II. Related Work

One may divide ad-hoc routing strategies into two categories: *structure-based* or *structure-less*. A structure-based routing strategy builds and maintains a routing structure, such as a spanning tree, a routing table, or one or multiple paths, while a structure-less routing strategy makes the decision of routing at every hop.

There are three elements for a routing protocol: destination specification, routing objectives and routing strategies. Most existing protocols use a *fixed* destination specification, routing objective and routing strategy. Most mobile ad hoc routing protocols are address-based, while GPSR [2] and GEAR [4] are geographical location-based. Directed Diffusion [1] first proposed a general attribute-based publish/subscribe scheme in ad-hoc sensor networks. In most cases, routing objectives are implicitly embedded in strategies. For example, AODV [11], DSR [12] and TORA [13] use shortest path, i.e., minimum number of hops, as the routing objective, while ABR [14] uses the degree of association stability, and SSR [15] uses signal stability or strength as one of the routing objectives in addition to the shortest path. GEAR uses energy level as a routing objective, in addition to the shortest path, while information-driven routing, such as CADR [3] and IDR [16], uses information gain to guide the routing process.

Most existing routing protocols are implicitly associated with their routing strategies. Traditional source-initiated protocols (AODV, DSR, TORA, ABS, SSR) are structure-based. There are also multipath strategies (e.g. [17]) that build and maintain multiple paths for extra robustness. For dynamic networks, in addition to initialization, periodic or repair-based maintenance is required to keep the structure up to date. Therefore, structure-based strategies are more suitable for relatively stable networks, since maintaining and repairing structures can be costly for highly dynamic networks.

Structure-less routing strategies, e.g. greedy-based, search-based or flooding-based, do not maintain the network structure explicitly. For example, GPSR uses a simple location-based greedy type of search, CADR deploys a greedy strategy using the information gain metric. The challenge of greedy-based strategies is to go around network holes. Recent work on randomized routing [18] chooses the next hop according to some distribution. Instead of building and maintaining a routing structure, some of structure-less strategies establish a "potential field". Data from upstream flows downstream according to the potential field. Similar to structure-based protocols, however, the maintenance of the potential field is necessary for dynamic networks. *Search-based methods* normally discover routes by selecting the next "best" hop at every node on the route. Routes may differ from packet to packet, even to the same destination node. GEAR, Q-routing [19], [20], ant-based routing [21] and NADV-based routing [22] are search-based. *Flooding-based methods*, on the other hand, allow each node to independently make the decision of rebroadcast, while making trade-offs between robust message delivery and total energy cost. Gradient Broadcast [23] and Gradient Routing [24] are flooding-based.

In the last several years, many routing protocols have been developed for wireless sensor networks, including Grid Routing [25], Directed Routing [26], Mint Routing [27], Backbone Routing [28], and the Constraint-based Routing Framework [6]–[10], [29]. Most of these routing protocols have been implemented on Berkeley motes [30], a widely used sensor network platform. Among these protocols, Backbone, Mint, and Grid are structure-based, Directed Routing and Constrained Flooding are flooding-based, and others, except the adaptive tree protocol, are search-based.

The Adaptive Tree Protocol is between structure-based and structure-less. It is structure-based since it builds and maintains a tree structure; it is structure-less since the parent of a node may change while forwarding a data packet. Unlike structure-based protocols, no control packets are necessary to maintain the tree structure.

III. Adaptive Tree Protocol

In this section, we will first introduce constraint-based routing and reinforcement-learning kernel, then present the adaptive tree protocol.

A. Routing Specification

Many protocols have been proposed for a variety of destination attributes and routing objectives in the last few years: for example, geographical (GPSR), energy-aware (GEAR), information-directed (IDR) routings. We advocate the framework of *Message-initiated Constraint-Based Routing* (MCBR): instead of developing protocols with special attributes and objectives, a meta-routing protocol is aimed at supporting a *class* of routing specifications. Even with the same meta-strategy, different messages can have different strategies when they have different specifications. This is along the line of Smart Packets for Active Networks [31]; however, in MCBR, packets do not carry code. Only the specification (and possibly an additional selection of a particular meta-strategy) is passed through the network. For networks with small data frames, one can even encode various specifications in nodes and let packets only carry a specification ID with parameters.

In MCBR, a routing specification consists of the destination specification, local route constraints, and a global routing objective. Constraints and objectives are specified in terms of attributes. Attributes can be anything from geographical locations to network bandwidths, from sensor values to internal clocks. The values of attributes can be constant, such as a node identifier or a unit cost, or can change from time to time. A *routing destination* is explicitly represented by a set of constraints on attributes. For example, in geographical routing, destinations are specified in terms of location constraints. Furthermore, in addition to destinations, *local route constraints*, if any, are specified explicitly. Examples of local route constraints are: avoiding a noisy area, avoiding congestion, and avoiding low-energy nodes, etc. One may also specify the regions or trajectories for information dissemination, e.g., the comb-needle model for queries and events [32] and traffic information over road structures. Finally, a *routing objective* is explicitly stated, such as a shortest path, maximizing energy levels over the route, maximizing connectivity over the route, or minimizing congestion.

A network can be represented as a graph $\langle V, E\rangle$, where V is the set of nodes and E is the set of connections. For an asymmetric network, $(v, w) \in E$ does not imply $(w, v) \in E$. Given a *destination constraint* C_m^d of message m, a node v is a *destination node* for m iff C_m^d is satisfied at v. For example, address-based routing, i.e., sending a message to a node with an address a_d, can be represented using the destination constraint $a = a_d$, where a is the address attribute. Geographical routing, e.g., sending a message to a circular region centered at (x_0, y_0) with radius c, can be represented using the destination constraint $(x - x_0)^2 + (y - y_0)^2 \leq c$, where x and y are location attributes.

Given a *local route constraint* C_m^r of message m, the *active* network of $\langle V, E\rangle$ for m is a subnet $\langle V_m, E_m\rangle$, such that $v \in V_m$ iff C_m^r is satisfied at v and $(v, w) \in E_m$ iff $v, w \in V_m$ and $(v, w) \in E$. For example, a message that should avoid congested nodes while routing to its destination has a local route constraint $l \leq l_m$, where l is the message load attribute (e.g., number of messages in the node's queue) and l_m is the load limit. One can also use geographical constraints (e.g., directional routing) to reduce collision and save energy for a flooding-based strategy. In general, local route constraints redefine the network connectivity on a message-by-message basis.

MCBR explicitly specifies routing objectives. A *local objective function* o is defined on a set of attributes: $o : A_1 \times A_2 \times \ldots \times A_n \to R^+$, where A_i is the domain of attribute i and R^+ is the set of *positive* real numbers. The *value* of o at a node v, denoted $o(v)$, is $o(a_1, a_2, \ldots, a_n)$, where a_i is the current attribute value of attribute i at node v. A local objective function can be a constant such as the unit transmission cost, which induces the shortest path if the objective is minimized. For another example, an energy-aware objective can be defined as $ku+c$, where u is the amount of used energy in the node, and k and c are constants. With this objective, energy-aware routing can be achieved. Similarly, one may use $k/n + c$ as a local objective, where n is the number of neighbors. With this objective, connectivity-aware routing can be achieved. Multi-objectives can be obtained by combining individual objectives, e.g., in a weighted sum.

A local objective can be aggregated over the routing path to form a global routing objective. A *global additive objective function* O is defined as $O(p) = \sum_{i=0}^{n} o(v_i)$, where o is a local objective function and p consists of a sequence of nodes $v_0, \ldots, v_n$.

There are in general two classes of routing. One is *anycast*, namely finding an optimal path from the source to *one* of the destination nodes; it is *unicast* if its destination is unique. The other is *multicast*, namely finding an optimal tree from the source to *all* the destination nodes.

An MCBR *specification* for a message m is a tuple $\langle v_m^0, C_m^d, C_m^r, O_m\rangle$. The *goal* of routing is to deliver the message from v_m^0 to one (anycast) or all (multicast) of the destination nodes V_m^d satisfying C_m^d via a sequence or a tree of intermediate nodes $p : v_m^1, \ldots, v_m^{n-1}$ such that C_m^r is satisfied at v_m^i and $\min_p O_m(p)$. Two messages are considered to have the same *type* if they have the same destination and local route constraint as well as the same routing objective.

One should notice that global route constraints are not defined in MCBR. It is well-known that finding an optimal path with an additive objective while satisfying an additive constraint is NP-hard. Unicast MCBR with an additive objective is essentially a *weighted shortest path problem*. Our goal is to make MCBR a simple (in terms of computation) yet still powerful (in terms of representation) mechanism for ad-hoc sensor networks.

B. Reinforcement Learning Kernel

MCBR separates routing specification from routing strategies. We have proposed learning-based meta-strategies for MCBR [7]. Real-time reinforcement learning [33] has been studied and applied mostly in agent-based path planning [34]. We apply this powerful technique to develop *distributed* meta routing strategies for sensor networks.

Given a routing specification of a message, including the destination and routing objective, one can define a cost function on each node, called *Q-value*, indicating the minimum cost-to-go from this node to the destination. Initial Q-values of nodes can be obtained during network initialization if the sink node is known, or estimated when the node receives a packet of that type at the first time . Furthermore, a node also stores its neighbors' Q-values, *NQ-values*, which are estimated initially according to the neighbors' attributes and updated when packets are received from neighbors.

The learning-based meta routing strategies typically consist of an *initialization* phase, a *forwarding* phase, and a *confirmation* phase. Learning happens in all phases. For each packet sent out from a node, the current Q-value of the node for the type of message is attached. All the nodes are set to be in promiscuous listening mode. Whenever a node overhears a packet of type m from node n with Q-value $Q(n)$, whether it is the designated receiver or not, it updates the corresponding NQ-value by

$$NQ_m(n) \leftarrow rNQ_m(n) + (1-r)Q(n) \tag{1}$$

and re-estimates its own Q-value using the equation

$$Q_m \leftarrow (1-\alpha)Q_m + \alpha(o_m + \min_n NQ_m(n)) \tag{2}$$

where r is an update rate and α is a learning rate, o_m is the current value of the local objective function, and n is a neighbor of this node.

Using Q-value estimates, *real-time search* [8] passes the packet to the "best" neighbor according to the current estimates, *constrained flooding* [9] decides if and when to rebroadcast the packet according to the estimates, and *adaptive spanning tree* [10] forwards the packet to its parent, with parents possibly changing over time pointing to a neighbor with the best Q-value estimate.

C. Adaptive Spanning Tree

Different from the other two meta-strategies, the adaptive tree protocol requires a known sink at the initialization. An adaptive spanning tree can be built using the reinforcement learning core described in the previous section. The initialization phase builds an initial spanning tree rooted at the sink. The initial spanning tree may not be optimal, and in the mobile sink case, connections may change from time to time. Each node other than the sink node has a pointer to its parent, which is the neighbor with smallest NQ-value. The forwarding phase passes the received packet to its parent. A node's parent may change if the neighbor with the smallest NQ-value changes. Each packet will be forwarded once, including the sink node that broadcasts with Q-value 0 each time receiving a packet. Implicit packet confirmation is used: if the packet is not heard from the forwarded node within a certain time period, the NQ-value of that node is updated, and the parent pointer is reset to the neighbor with the smallest NQ-value. A fixed maximum number of retransmissions at any node may be applied when the implicit confirmation fails.

```
Initialization phase:
    for all $v$ do $NQ_m(v) \leftarrow \inf$ end
    received ($m$, Q) at $w$ from node $u$ do
1.      $NQ_m(u) \leftarrow Q$;
2.      $Q_m \leftarrow (1-\alpha)Q_m + \alpha(o_m + \min_v NQ_m(v))$;
3.      $p'_m \leftarrow$ argmin$_n NQ_m(n)$; (random tie break)
4.      if $p_m \neq p'_m$ then broadcast($m$, $Q_m$); $p_m \leftarrow p'_m$; end
    end
Forwarding phase:
    received ($m$, Q) at $w$ from node $u$ do
1.      if satisfied($\mathcal{C}_m^d$) then
2.          $Q_m \leftarrow 0$; broadcast($m$, 0); return;
3.      end
4.      $NQ_m(u) \leftarrow rNQ_m + (1-r)Q$;
5.      $Q_m \leftarrow (1-\alpha)Q_m + \alpha(o_m + \min_v NQ_m(v))$;
6.      if $|(NQ_m(p_m) - \min_n NQ_m(n)| > \delta$
7.          $p_m \leftarrow$ argmin$_n NQ_m(n)$; (random tie break)
8.      end
9.      if designated($m$) then send($m$, $Q_m$) to $p_m$; end
    end
Confirmation phase:
    timeout ($m$ to $v$)
1.      $NQ_m(v) \leftarrow NQ_m(v) + c$;
2.      $p_m \leftarrow$ argmin$_n NQ_m(n)$; (random tie break)
3.      if (resend) then send($m$, $Q_m$) to $p_m$; end
    end
```

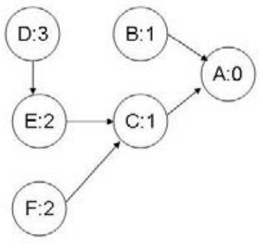
Figure 1. Adaptive spanning-tree meta-strategy.

The pseudo code is illustrated in Figure 1. NQ-value updates are designed as follows. In line 4 of the forwarding phase, we use link loss rates for calculating r, i.e., $r \leftarrow L/(R+L)$ where L is the number of failed transmissions and R is the number of successful transmissions. If link loss rates are high, NQ-value of that neighbor will not be changed much; if link loss rates are close to zero, NQ-value is updated with the new received value. In line 1 of the confirmation phase, each failed confirmation increases NQ-value of that neighbor by c. For example, $c \leftarrow e^{L/(R+1)}$. In this case c increases with the number of failed transmissions exponentially. In our experiments, we set δ and α be 1, and the maximum number of retransmissions be 1 or 2. Similar to the real-time search strategies [8], changing these parameters will change the overall performance of various routing metrics.

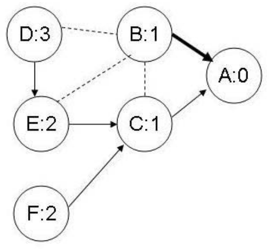
D. Properties of Adaptive Tree Protocol

Unlike the most existing routing trees, the adaptive routing tree can adjust to a better structure during routing *without extra* control packets; Figure 2 illustrates such a scenario. Furthermore, asymmetric or broken links will be removed automatically; Figure 3 illustrates the situation.

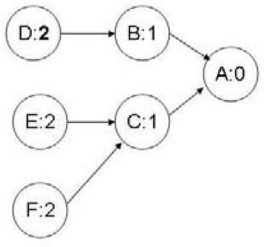
In both cases, `node-ID:Q-value` is illustrated in each circle and the arrow points to its current parent.



(a)

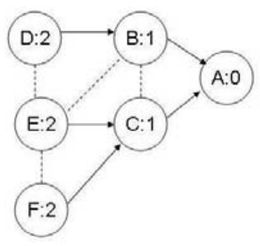


(b)

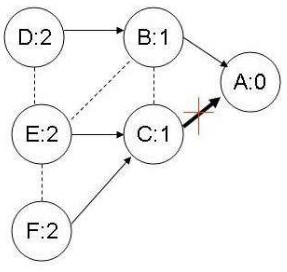


(c)

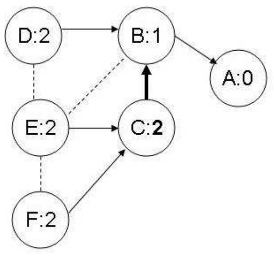
Figure 2. Adaptive to a better connection: (a) initial tree (b) node B sends a packet to A, and nodes C, D, and E hear, in addition to A (c) node D changes its parent to B with lower Q-value.



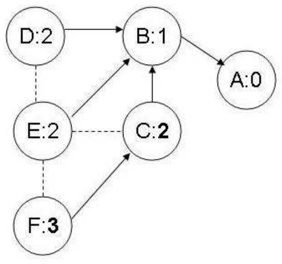
(a)



(b)



(c)



(d)

Figure 3. Adaptive to an asymmetric or broken connection: (a) current tree (b) C sends a packet to A with no confirmation (c) C updates its parent link to B with new Q-value (d) E updates its parent and F updates its Q-value when they hear a transmission from C.

A fundamental question is whether this process of adaption induce infinite loops. The following theorem states that such a situation is not possible, if the network is relatively stable, i.e., the optimal structure do not change during the process.

Theorem 1: Given a network $\langle V, E \rangle$, assuming there is a path from any node to the root and the actual Q-value is stable, a packet will be delivered in $\mathcal{O}(nd)$, where n is the number of nodes in the network and d is the depth of the tree. Furthermore, it approaches to an optimal path if the initial Q-value estimates are *admissible*, i.e., no larger than the actual Q-values.

Proof. The formal proof is similar to that of [8], [34]. ■

Theorem 2: If a node v with Q-value Q_v lost its connection to its parent, and node w connects to the subtree rooted at v via v' with the smallest Q-value Q_w. Node v' will change its parent in $n_v(Q_w - Q_v)/c$ steps where c is the minimal local cost and n_v is the number of nodes in subtree v.

Proof. At each step, there is at least one node with its value increased by c. The worst case is to visit each node $(Q_w - Q_v)/c$ times before pointing to w. ■

E. Routing Specifications for Adaptive Trees

One of the most important advantages of MCBR is that one can separate meta-routing strategies from routing objectives, so that different objectives can be achieved for different type of messages using the same meta-strategy.

We have experimented two types of routing specifications, in addition to the shortest hop-counts.

1) Energy-aware: Instead of assuming constant cost for each hop, we can specify energy-aware cost functions, $c(e)$, where e is an attribute indicating the current energy level, such that $c(e_1) \leq c(e_2)$ if $e_1 \geq e_2$. e.g., $c(e) = e_{max}/(e+1)$ where e_{max} is a constant indicating a maximum energy level. It is easy to see that for this cost function, if e is high, then c is low, and if e is low, c

becomes high. With this cost function, load balancing can be achieved automatically.

2) Congestion-aware: Let l be the current transmission queue length of a node. The larger the l, the more congested. Let $c(l)$ be a cost function, such that $c(l_1) \leq c(l_2)$ if $l_1 \leq l_2$. We can use a simple cost function such as $c(l) = l + 1$. With this cost function, routing will automatically select pathes with smaller queue length or less congested nodes. This will results low latency and/or higher reliability for large traffic.

IV. Simulation Results

In this section, we analyze the performance of the adaptive tree protocol using a simulator that is particularly designed for sensor networks [35].

A. Radio and MAC Models

Our protocol study uses the radio propagation model and the MAC layer communication model provided by PROWLER [36], a probabilistic sensor network simulator. PROWLER [36], written in Matlab, is an event-driven simulator that can be set to operate in either deterministic mode (to produce replicable results while testing an algorithm) or in probabilistic mode that simulates the nondeterministic nature of the communication channel. PROWLER consists of a *radio propagation model* and a *MAC-layer model*.

The simple radio model in PROWLER attempts to simulate the probabilistic nature in wireless sensor communication observed by many [37] [27]. The propagation model determines the strength of a transmitted signal at a particular point of the space for all transmitters in the system. Based on this information the signal reception conditions for the receivers can be evaluated and collisions can be detected. The transmission model is given by:

$$P_{rec,ideal}(d) \leftarrow P_{transmit}\frac{1}{1+d^{\gamma}}, \quad where\ 2 \leq \gamma \leq 4 \quad (3)$$

$$P_{rec}(i,j) \leftarrow P_{rec,ideal}(d_{i,j})(1+\alpha(i,j))(1+\beta(t)) \quad (4)$$

where $P_{transmit}$ is the signal strength at the transmitter and $P_{rec,ideal}(d)$ is the *ideal* received signal strength at distance d, α and β are random variables with normal distributions $N(0,\sigma_\alpha)$ and $N(0,\sigma_\beta)$, respectively. A network is asymmetric if $\sigma_\alpha > 0$ or $\sigma_\beta > 0$. In (4), α is static depending on locations i and j only, and β is dynamic which changes over time. A node j can receive a packet from node i if $P_{rec}(i,j) > \Delta$ where $\Delta > 0$ is the threshold. There is a collision if two transmissions overlap in time and both could be received successfully. Furthermore, an additional parameter p_{error} models the probability of a transmission error caused for any other reason. The default radio model in PROWLER has $\gamma = 2$, $\sigma_\alpha = 0.45$, $\sigma_\beta = 0.02$, $\Delta = 0.1$ and $p_{error} = 0.05$. Figure 4 shows a snapshot of the radio reception curves in this model.

The MAC layer communication is modeled by a simplified event channel that simulates the Berkeley motes' [30] CSMA MAC protocol. When the application emits the *Send_Packet* command, after a random *Waiting_Time* interval the MAC layer checks if the channel is idle. If not, it continues the idle checking until the channel is found idle. The time between idle checks is a random interval characterized by *Backoff_Time*. When the channel is idle the transmission begins, and after *Transmission_Time* the application receives the *Packet_Sent* event. After the reception of a packet on the receiver's side, the application receives a *Packet_Received* or *Collided_Packet_Received* event, depending on the success of the transmission.

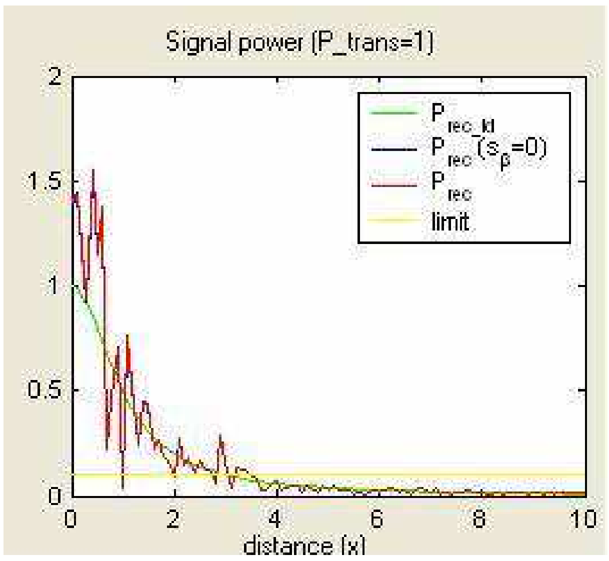


Figure 4. Snapshot of radio reception curves for the default model.

B. Routing Application Models

Our simulation tests were done in RMASE [38], an application built on PROWLER. RMASE provides network generation and performance evaluations for routing algorithms.

RMASE provides a convenient graphical interface to define the most common network topologies. The basic topology is a rectangular x-y grid. The *grid size*, *spacing*, *density*, *shift* and *offset* parameters allow one to specify a variety of topologies, from rectangular and triangular grids to totally random placements. One can also specify one or more network holes, with a specific location and size or randomly placed. One can also import node locations from a file, used in real experiments.

The routing scenario model in RMASE define the properties and behaviors of the *source* and *destination*, containing a set of source and destination nodes for a routing application, respectively. RMASE also defines a set of performance metrics that will be discussed in the following section.

RMASE supports a layered architecture, including at least the MAC layer, a routing layer, and the application layer, with the MAC layer at the bottom and the application layer at the top. It is the algorithm designer's choice to put individual functions at different layers so that common functions can be shared by different algorithms.

In our tests, the network is a 7×7 sensor grid with small random offsets. The maximum radio range is about $3d$, where d is the standard distance between two neighbor nodes in the grid. Figure 5 shows an instance of the connectivity of such a network. The default radio model in PROWLER is used in most experiments. The radio data rate is 40Kbps and each packet has 960 bits. The application sends out one packet per second from the sources.

The results are based on the average of 10 random runs.

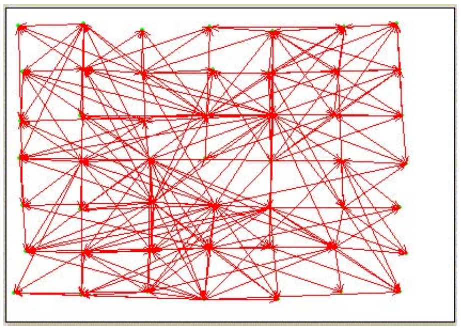


Figure 5. Instance of radio connectivity.

C. Performance Metrics

We have developed a set of performance metrics for comparing different routing algorithms in RMASE, including latency, delivery ratio, and energy efficiency.

- *Latency*: average delays of packets from sources to the destination.
- *Delivery ratio*: the total number of packets received at the sinks vs. the total number of packets sent from all the sources.
- *Energy efficiency*: be the ratio between the total number of packets received at the sink(s) vs. the total number of transmissions in the network.

All these metrics are calculated using their cumulative average values, i.e., at time t, the performance value is the average from 0 to t.

It is hard to compare the lifetime metric for different routing algorithms, although it is one of the most important metrics for battery-powered unattended sensor networks. One definition of the network lifetime has been the time to network separation [4]. This definition has the disadvantage that it cannot distinguish between separation of half the network and separation of a single node, and between a single node that is peripheral and a single node that is the base station. Another definition can be the time until the network loss rate is above a given threshold. Loss rate measures the relevance to the application: a single connected peripheral node leads to a small loss rate, while a single disconnected base station leads to a large loss rate (if it is the source or the sink). Similarly, separation of half the network leads to a large loss rate. However, both of these definitions require to run the simulation until the property of life-time is satisfied. We define lifetime predication as follows:

- *Lifetime prediction*: Let $\bar{e} = \sum e_i/n$ be the average energy left in the network where n is the total number of nodes in the network and σ_e be the standard deviation of energy distribution. We call $L = \bar{e} - 3\sigma_e$ the lifetime prediction metric, which is used to approximate the lifetime metric. From this metric, one can see that if two algorithms A and B use the same amount of energy, but if B is energy-aware and uses energy more evenly across the network than A does, B is predicted to have a longer lifetime.

D. Effects of Routing Metrics

We first exam the effects of two routing metrics described in Section III-E, comparing them to the shortest hop-counts or shortest path objective.

1) Energy-aware: Using the cost function in Section III-E.1, we see that the cost doubles if more than half of the energy is consumed. To see this effect clearly, we set the maximum energy at each node to be 100, each transmission consumes a unit power, receiving or idling power is ignored. The source is at a node close to $(0,0)$ and the sink is at a node close to $(6,6)$. The source sent a total of 50 packets.

Assuming all links are reliable and if the adaptive tree is optimal initially, there would be a single path for the shortest path objective (Figure 6) but there are multiple paths for a load balancing objective (Figure 7).

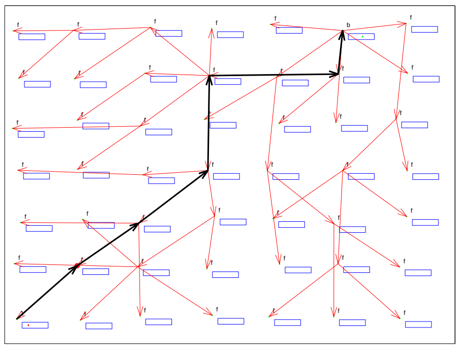


Figure 6. A single routing path by the shortest path objective.

Figure 8 shows the lifetime prediction metric comparing load balancing objective and the shortest path objective. The load balancing objective has larger values indicating longer lifetime.

2) Congestion-aware: In this test, we again let the sink node close to $(6,6)$, and let each node have 50% chance to be a source node which will send a packet randomly within 10 seconds. We compare latency and delivery ratios between the congestion-aware objective in Section III-E.2 and the shortest path objective. Figure 9 and 10 show that the congestion-aware objective has much shorter latency and slightly better delivery ratios.

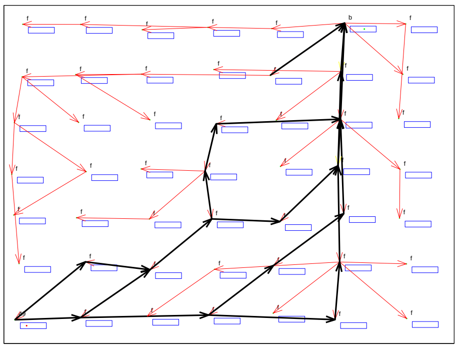


Figure 7. Multiple paths by load balance objective.

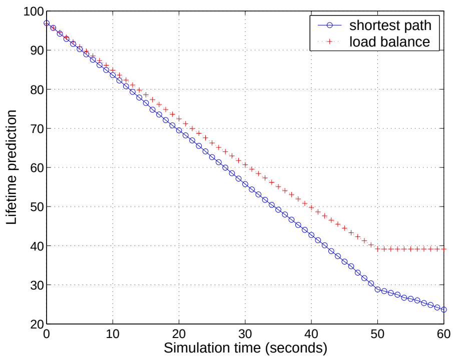


Figure 8. Lifetime prediction metric: load balance vs. shortest path.

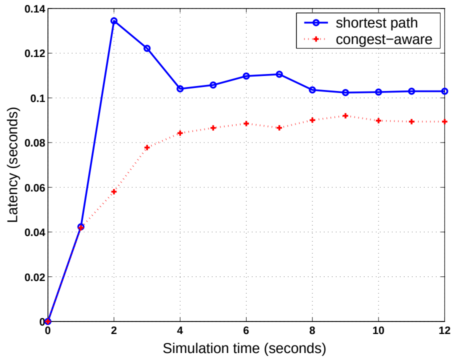


Figure 9. Latency: congestion-aware vs. shortest path.

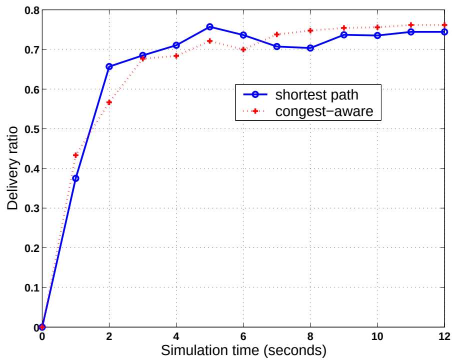


Figure 10. Delivery ratio: congestion-aware vs. shortest path.

E. Comparison with Peer Protocols

In this section, we evaluate the routing performance of the adaptive spanning tree in two settings: (1) node failures and (2) mobile sinks, and compare performances of the adaptive spanning tree with other peer routing protocols.

The two peer routing algorithms used for comparing performances with the adaptive tree protocol are:

- *Backbone tree* (**Backbone** [28]) – this algorithm uses directed diffusion [1] to create a backbone tree in the initialization phase and passes packets to parents during routing. To handle the problem of asymmetric links, it establishes a symmetric link neighborhood in the initialization. To save energy, there is no periodic maintenance packets.
- *Grid Routing* (**Grid**) [25] – this algorithm builds a spanning-tree with “grid” information to avoid long and unreliable links; There are periodic tree maintenance packets at every 15 - 25 seconds.

A total of 50 packets are sent for both tests.

1) Node failures: To test the robustness of the adaptive tree protocol under un-predicable link failures, we set link failure probabilities p_{error} to be 0, 0.05 and 0.1, respectively. The source is at a node close to $(0, 0)$ and the sink is at a node close to $(6, 6)$. Figure 11 shows the delivery ratios of the three algorithms. From the graph, we see that **Backbone** works very well when there are no fault links. The symmetric links guarantee the 100% delivery of packets at this rate (1 packet per second). However, performance of **Backbone** degrades quickly with the increasing of link failures. **Grid** does not work well overall due to the collision between data packets and regular maintenance packets from the sink node. Reducing intervals of maintenance packets will improve the performance of **Grid** in this case. The adaptive tree protocol works better than the other two protocols, with the highest delivery ratios in all three cases.

2) Mobile sinks: To test the robustness of the adaptive tree protocol with moving sinks, we set the sink velocities be 0.05d/s and 0.1d/s in both x and y directions, respectively. The source is at $(0, 0)$, and the sink is at $(1, 1)$ initially. Figure 12, 13 and 14 show delivery ratios, packet latencies, and energy efficiency, respectively, for both velocities. From these figures, we see that: (1) similar to faulty links, the adaptive tree protocol has high delivery ratios in mobile sink scenarios. (2) The latency of packets increases in general due to the fact that the mobile sink is moving away from the source, resulting longer paths. For **Backbone**, latency does not change after 5 seconds since no more packets received after 5 seconds. (3) **Backbone** also has the highest energy efficiency in this case, since all the packets received are within one hop and no forwarding for packets more than one hop. **Grid** has the least efficiency due to extra maintenance packets.

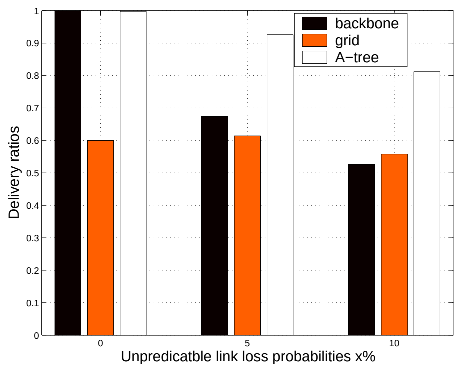


Figure 11. Delivery ratios at different failure rates.

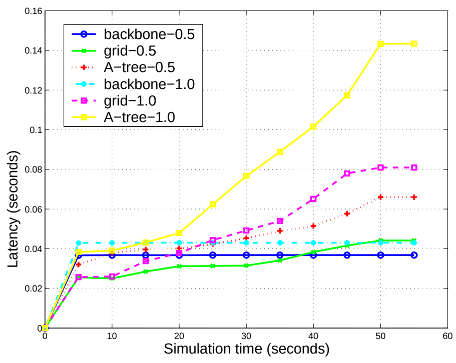


Figure 13. Latency with mobile sink.

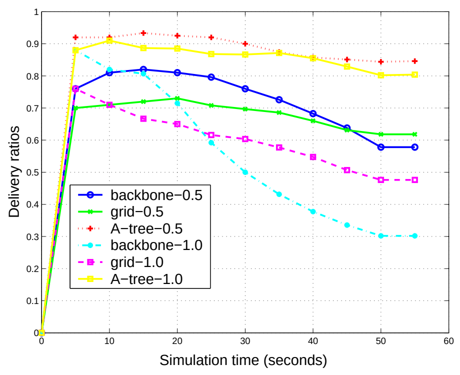


Figure 12. Delivery ratios with mobile sink.

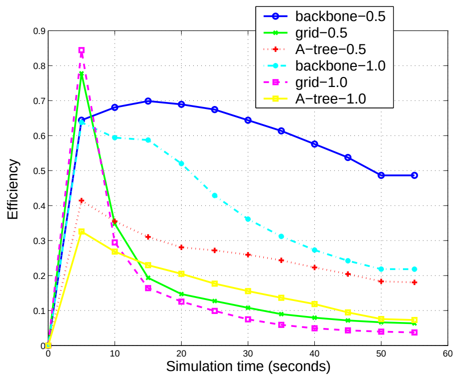


Figure 14. Energy efficiency with mobile sink.

V. Conclusion

We have presented in this paper the adaptive tree protocol, a type of reinforcement-based meta-routing strategy for the constraint-based routing. We have studied the properties of such protocol and shown in our experiments that the adaptive tree protocol is robust for un-predicable link failures and mobile sinks. We have also demonstrated the use of different routing objectives for achieving load balancing and reducing congestion. The parameters in the protocol, such as learning rates for Q-values, update rates for NQ-values, parent reset threshold δ, and the maximum number of retransmissions for failed confirmations, can be tuned to make the routing best for a particular application. Lots of research still need to be done on the selection of parameter values and understanding the relationship between different parameters.

References

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed diffusion: A scalable and robust communication paradigm for sensor networks,” in *Proc. 6th Int’l Conf. on Mobile Computing and Networks (ACM Mobicom)*, Boston, MA, 2000.

[2] B. Karp and H. T. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks,” in *Proc. 6th Int’l Conf. on Mobile Computing and Networks (ACM Mobicom)*, Boston, MA, 2000.

[3] M. Chu, H. Haussecker, and F. Zhao, “Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks,” *Int. Journal on High Performance Computing Applications*, June 2002.

[4] Y. Yu, R. Govindan, and D. Estrin, “Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks,” UCLA Computer Science Department,” Technical Report UCLA/CSD-TR-01-0023, May 2001.

[5] E. Royer and C. Toh, “A review of current routing protocols for ad hoc mobile wireless networks,” *IEEE Personal Communications*, April 1999.

[6] Y. Zhang and M. Fromherz, “Message-initiated constraint-based routing for wireless ad-hoc sensor networks,” in *Proc. IEEE Consumer Communication and Networking Conference*, 2004.

[7] Y. Zhang, M. Fromherz, and L. Kuhn, “Smart routing with learning-based qos-aware meta-strategies,” in *Proc. Quality of Service in the Emerging Networking*, ser. Lecture Notes in Computer Science 3266, 2004.

[8] Y. Zhang and M. Fromherz, “Search-based adaptive routing strategies for sensor networks,” in *AAAI04 workshop on Sensor Networks*, 2004.

[9] ——, “Constrained flooding: A robust and efficient routing framework for wireless sensor networks,” in *20th International Conference on Advanced Information Networking and Applications*, 2006.

[10] Y. Zhang and Q. Huang, “Adaptive tree: A learning-based meta-routing strategy for sensor networks,” in *3rd IEEE Consumer Communications and Networking Conference*, 2006.

[11] C. E. Perkins and E. M. Royer, “Ad hoc on-demand distance vector routing,” in *Proc. 2nd IEEE Workshop*

on Mobile Computing Systems and Applications, February 1999, pp. 90–100.

[12] D. B. Johnson and D. B. Maltz, “Dynamic source routing in ad hoc wireless networks,” in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic, 1996, pp. 153–181.

[13] V. D. Park and M. S. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks,” in *Proc. IEEE InfoCom 97*, April 1997.

[14] C. Toh, “A novel distributed routing protocol to support adhoc mobile computing,” in *Proc. 15th IEEE Annual Int’l Phoenix Conf. on Computing and Communication*, March 1996.

[15] R. D. et.al, “Signal stability based adaptive routing (ssa) for ad-hoc mobile networks,” *IEEE Personal Communications*, February 1997.

[16] Y. Zhang, J. Liu, and F. Zhao, “Information-directed routing in sensor networks using real-time reinforcement learning,” in *Combinatorial Optimization in Communication Networks*. Kluwer Academic Publishers, 2006.

[17] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, “Highly-resilient energy-efficient multipath routing in wireless sensor networks,” *Mobile Computing and Communications Review*, vol. 1, no. 2, 2002.

[18] S. D. Servetto and G. Barrenechea, “Constrained random walks on random graphs: Routing algorithms for large scale wireless sensor networks,” in *1st ACM International Workshop on Wireless Sensor Networks and Applications*, September 2002.

[19] J. A. Boyan and M. L. Littman, “Packet routing in dynamically changing networks: A reinforcement learning approach,” in *Advances in Neural Information Processing Systems*, J. A. J. D. Crowan, G. Tesauro, Ed. Morgan Kaufmann Publishers, Inc., 1994, vol. 6, pp. 671–678.

[20] S. Kumar and R. Miikkulainen, “Confidence-based q-routing: An on-line adaptive network routing algorithm,” in *Proceedings of Artificial Neural Networks in Engineering*, 1998.

[21] J. C. D. Subramanian, P. Druschel, “Ants and reinforcement learning: A case study in routing in dynamic networks,” Rice University,” Technical Report TR96-259, July 1998.

[22] S. Lee, B. Bhattacharjee, and S. Banerjee, “Efficient geographic routing in multihop wireless networks,” in *Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc05)*, 2005.

[23] F. Ye, G. Zhong, S. Lu, and L. Zhang, “A robust data delivery protocol for large scale sensor networks,” in *Information Processing in Sensor Networks, Lecture Notes in Computer Science*, no. 2634, 2003, p. 658.

[24] R. Poor, “Gradient routing in ad hoc networks,” http://www.media.mit.edu/pia/Research/ESP/texts /poorieeepaper.pdf.

[25] Y. Choi, M. G. Gouda, H. Zhang, and A. Arora, “Routing on a Logical Grid in Sensor Networks,” Department of Computer Sciences, The University of Texas at Austin, Technical Report TR04-49, 2004.

[26] M. Maroti, “Directed Flood-Routing Framework,” Institute for Software Integrated Systems, Vanderbilt University, Technical Report ISIS-04-502, 2004.

[27] A. Woo, T. Tong, and D. Culler, “Taming the underlying challenges of reliable multihop routing in sensor networks,” in *Proc. 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys03)*, 2003.

[28] T. He, S. Krishnamurthy, J. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, “An energy-efficient surveillance system using wireless sensor networks,” in *Proc. 2nd International Conference on Mobile Systems, Applications and Services (MobiSys04)*, 2004.

[29] Y. Zhang, L. Kuhn, and M. Fromherz, “Improvements on ant-routing for sensor networks,” in *Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science 3172, 2004.

[30] U. Berkeley, “Berkeley wireless embedded systems,” http://webs.cs.berkeley.edu/.

[31] B. Schwartz, A. W. Jackson, W. T. Strayer, W. Zhou, R. D. Rockwell, and C. Partridge, “Smart packets for active networks,” *ACM Transaction on Computer Systems*, vol. 18, no. 1, Feb. 2000.

[32] X. Liu, Q. Huang, and Y. Zhang, “Combs, needles, haystacks: Balancing push and pull for discovering in large-scale sensor networks,” in *Proc. 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys04)*, 2004.

[33] R. S. Sutton and A. G. Barto, Eds., *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press, 1998.

[34] S. Koenig and R. G. Simmons, “Complexity analysis of real-time reinforcement learning applied to finding shortest path in deterministic domains,” in *National Conference on Artificial Intelligence*, 1993, pp. 99–105.

[35] Y. Zhang, G. Simon, and G. Balogh, “High-level sensor network simulations for routing performance evaluations,” in *Third International Conference on Networked Sensing Systems*, 2006.

[36] G. Simon, “Probabilistic wireless network simulator,” http://www.isis.vanderbilt.edu/projects/nest/prowler/.

[37] J. Zhao and R. Govindan, “Understanding packet delivery performance in dense wireless sensor networks,” in *Proc. of 1st International Conference on Embedded Networked Sensor Systems (SenSys03)*, 2003.

[38] Y. Zhang, M. Fromherz, and L. Kuhn, “Rmase: Routing modeling application simulation environment,” http://www.parc.com/era/nest/Rmase.

Ying Zhang was born in Hangzhou, PR China. She obtained her PhD in computer science from University of British Columbia, Vancouver, Canada in 1994, and her Bachelor and Master degrees in Computer Science from Zhejiang University, Hangzhou, China, in 1984 and 1987, respectively.

After graduation, she first worked at Xerox Wilson Center for Research and Technology as a technical specialist/project manager, then Xerox Architecture Center, and starting 1999, she joined Palo Alto Research Center as a member of research staff. Her current research interests are in sensor/actuator networks and embedded control architectures.

Dr. Zhang is a member of IEEE and ACM. She is information director and associate editor of ACM Transactions on Sensor Networks, and associate editor of International Journal of Ad Hoc and Ubiquitous Computing. She has served in various technical program committees, guest editors, and NSF review panels on sensor networks.

QingFeng Huang was born in China. He received his D.Sc. degree in computer science from Washington University in St. Louis in the United States of America in 2003. He also received A.M. degree in physics from Washington University in St. Louis in 1998 and B.S. degree in Physics from Fudan University in China in 1992.

He is a research scientist at the Palo Alto Research Center (PARC) in Palo Alto, California, USA. He has published more than 30 papers in areas including software engineering, mobile computing, sensor networks, intelligent transportation systems, neuroscience, and quantum physics. His current research interests include algorithms and middleware services for transportation networks, sensor actuator networks, and social networks.

Dr. Huang is an editor for the International Journal of Ad Hoc and Ubiquitous Computing and is a member of the IEEE Computer Society.