Robust Connectivity-Aware Energy-Efficient Routing for Wireless Sensor Networks

Charles Pandana and K. J. Ray Liu

***Abstract*—In this paper, we consider a class of energy-aware routing algorithm that explicitly takes into account the connectivity of the remaining sensor network. In typical sensor network deployments, some nodes may be more important than other nodes because the failure of these nodes causes the network disintegration, which results in early termination of information delivery. To mitigate this problem, we propose a class of routing algorithms called *keep-connect* algorithms, that use *computable measures* of network connectivity in determining how to route packets. The proposed algorithms embed the importance of the nodes in the routing cost/metric. The importance of a node is characterized by the algebraic connectivity of the remaining graph when that node fails. We prove several properties of the proposed routing algorithm including the energy consumption upper bound. Using extensive simulations, we demonstrate that the proposed algorithm achieves significant performance improvement compared to the existing routing algorithms. More importantly, we show that our proposed algorithm is more robust in terms of algebraic network connectivity compared to the existing algorithm. Finally, we present the distributed implementation of our proposed algorithm.**

***Index Terms*—Sensor network, graph theory, communication system reliability, communication system routing.**

I. Introduction

ADVANCES in low-power integrated circuit devices and communications technologies have enabled the deployment of low-cost, low-power sensors that can be integrated to form a sensor network. This network has vast important applications, i.e., from battlefield surveillance systems to modern highway and industry monitoring systems; from the emergency rescue systems to early forest fire detection and the very sophisticated earthquake early detection systems, etc. Having the broad range of applications, the sensor network is becoming an integral part of human lives. Moreover, it has been identified as one of the most important technologies nowadays [1], [2].

There are many important characteristics of a sensor network. First, the sensor nodes are typically deployed in an area with high redundancy and each of the sensor node has limited energy, therefore it is prone to failure. In order to accomplish the mission, it is essential for the sensor nodes to collaborate. Second, the nodes in the sensor network typically stay in their original deployed places for their entire lifetime. Hence, it is very important to always keep the remaining network connected, since the disintegrated clusters of nodes are useless for information gathering. Moreover, due to the immobility of the nodes, it may not be possible to reorganize the remaining nodes to create a new connected network.

Due to these characteristics of sensor networks, the design of routing algorithms becomes very different from the typical ad-hoc networks in the following aspect. Instead of minimizing the hop count and delivery delay in the network, the routing algorithms in the sensor networks focus more on extending the scarce battery lifetime of the nodes. Furthermore, most of the existing energy-aware routing algorithms use the time until the first node in the network dies as the definition of network lifetime [3], [4]. Since in many practical sensor applications, the death of the first node may not influence the information collection task; We argue that the definition of the network lifetime should be defined as the time until there is no route from any source to any destination. In other words, the network lifetime should be defined as the time until the network becomes disconnected/disintegrated. Using this definition as the network lifetime, the network connectivity becomes an important criterion to be explicitly considered in the routing algorithm. None of the existing routing algorithm has ever explicitly considered the network connectivity in performing routing task.

To be precise, we employ the notion of algebraic connectivity of a graph in the spectral graph theory [5] to quantify the importance of a node. In particular, the importance of a node is quantified by the Fiedler value [6], [7] of the remaining graph when that particular node fails. One property of the Fiedler value is that the larger it is, the more connected the graph will be. Due to this reason, the Fiedler value is also referred to as algebraic connectivity of a graph [6]. By considering the nodes' importance from the graph connectivity perspective in the routing design, the node with higher importance will be retained in the network, therefore the connectivity of the remaining network is maintained as long as possible.

By embedding the nodes' importance in the routing cost, we propose a class of algorithms called *keep-connect* algorithms to solve the posed problem. Our proposed algorithm has several advantages, namely it is online algorithm, which implies that the algorithm does not require to know ahead of time the sequences of messages to be routed. This characteristic is important, since the information generation typically is not known *a priori* in sensor network. The proposed algorithm is efficient in maintaining the connectivity of the remaining network, we demonstrate the effectiveness of our proposed algorithm using extensive simulations. Finally, the proposed algorithm is flexible and can be used along with any other existing energy-aware routing algorithms that employ distributed

Manuscript received April 30, 2007; revised October 2, 2007 and January 15, 2008; accepted March 17, 2008. The associate editor coordinating the review of this paper and approving it for publication was P. Fan.

C. Pandana is with Arraycomm, San Jose, CA 95131 (e-mail: cpandana@gmail.com).

K. J. Ray Liu is with the Department of Electrical and Computer Engineering, and the Institute of Systems Research, University of Maryland, College Park, MD 20742 (e-mail: kjrliu@umd.edu).

Digital Object Identifier 10.1109/T-WC.2008.070453

Bellman-Ford/Dijkstra algorithms in their implementations.

This paper is organized as follows. We first give the system description and problem formulation in Section II. Several important facts from spectral graph theory are briefly outlined in Section III. In Section IV, our proposed algorithm is explained. Upper bound on the energy consumption of our proposed algorithm is proved in Section V. In Section VI, we present one possible distributed implementation of our proposed algorithm. The effectiveness of our method is presented in Section VII. Finally, conclusions are drawn in Section VIII.

II. System Model and Problem Formulation

In this section, we present the network model, review several definitions for sensor network lifetime, and explain the problem formulation.

A. Network Model

A wireless sensor network can be modeled as an undirected simple finite graph $G(V,E)$, where $V = \{v_1, \cdots, v_n\}$ is the set of nodes in the network, E is the set of all links/edges, $|V| = n$ is the number of vertices in the graph, and $|E| = m$ is the number of edges in the graph. The undirected graph implies that all the links in the network are bidirectional, i.e. node v_i is able to reach node v_j implies the vice versa. The simple graph implies that there is no self-loop in each node and there are no multiple edges connecting two nodes. And the finite graph implies the cardinality of the nodes and edges is finite. The link (v_i,v_j) implies that node $v_j \in S_{v_i}$ can be directly reached by node v_i with a certain transmit power level in the pre-defined dynamic range, where S_{v_i} is the set of nodes that can be directly reached by node v_i. We assume that every node has the initial battery energy of $\mathcal{E}_i$ for $\forall i \in \{1, ..., n\}$. The energy consumption for packet transmission from node v_i to v_j is proportional to $d(i,j)^\alpha$, where $d(i,j)$ is the distance between node v_i and v_j. The path loss exponent α, depends on the transmission environment [8] and typically ranges from 2 to 4. In this paper, we assume $\alpha = 2$ for free space propagation. We will also discuss the performance of our algorithm when $\alpha = 4.0$ in Section VII. When the energy in one node is exhausted, we say that the node has failed.

B. Definitions of Network Lifetime

Depending on the application in the wireless sensor network, there are many definitions of the network lifetime. In [3], [4], the network lifetime is defined as the time until the first node/sensor in the network fails. In contrast, in [9], the network lifetime is defined as the time until all nodes fail. A more general definition on the network lifetime is given in [10]. In [10], they defined the lifetime of sensor networks as the $\min\{t_1, t_2, t_3\}$, where t_1 is the time it takes for the cardinality of the largest connected component to drop below $c_1 \cdot n(t)$, where $n(t)$ is the number of alive nodes at time t, t_2 is the time it takes for $n(t)$ to drop below $c_2 \cdot n(0)$, and t_3 is the time it takes for the area covered to drop below $c_3 \cdot A$, where A is the area covered by the initial deployment of the sensors. In the above definition, c_1, c_2, and c_3 are the pre-defined constants between zero and one. It is well-known that the network connectivity is a very important characteristic in ad-hoc/sensor networks, therefore it should be taken into account in the network lifetime definition and algorithm design. In sensor network applications, the time until the first node/sensor fails may not serve as a good definition of the network lifetime, since the failure of the first node/sensor does not cease the information delivery/collection. In contrast, network disintegration typically causes severe impact in the information delivery. This motivates us to employ the time until the remaining network becomes disconnected as our network lifetime definition. Using this definition, we argue that it is crucial to consider the network connectivity in designing an energy-aware routing algorithm.

C. Problem Formulation

The problem of maximizing the minimum residual energy of nodes in the network has been studied in [3], [11], [12]. The time until the first node in the network dies can be found using the following linear program

$$\begin{array}{ll}
\text{Maximize} & T \\
\text{s.t.} & \\
\multicolumn{2}{l}{1) f^{(c)}(i,j) \geq 0, \ \forall i \in N, \ \forall j \in S_i, \ \forall c \in C,} \\
\multicolumn{2}{l}{2) \sum_{j \in S_i} e_t(i,j) \sum_{c \in C} f^{(c)}(i,j) +} \\
\multicolumn{2}{l}{\qquad \sum_{j: i \in S_j} e_r(j,i) \sum_{c \in C} f^{(c)}(j,i) \leq \mathcal{E}_i, \forall i \in N,} \\
\multicolumn{2}{l}{3) \sum_{j: i \in S_j} f^{(c)}(j,i) + T Q_i^{(c)} =} \\
\multicolumn{2}{l}{\qquad \sum_{j \in S_i} f^{(c)}(i,j), \ \forall i \in N, \forall c \in C,}
\end{array} \tag{1}$$

where $f^{(c)}(i,j)$ is the amount of information of commodity c that is transmitted from node v_i to its neighbor node $v_j \in S_i$ until time T, S_i is the set of v_i's neighboring nodes. Each commodity $c \in C$ has certain source node $O^{(c)}$ and destination node $D^{(c)}$, where $O^{(c)}$ is the origin/source of commodity c and $D^{(c)}$ being the destination/drain of commodity c. $e(i,j)$ is the energy required to guarantee successful transmission from node v_i to node v_j, and $Q_i^{(c)}$ denotes the information-generation rates at node v_i of commodity c. The first constraint indicates that the number of packets transmitted between any two nodes is nonnegative. The second constraint implies that the total energy used for packet transmission and reception at one particular node should be less that the battery capacity in that node. And this applies for all nodes. The last constraint indicates the flow conservation at each nodes in the network, we note that $Q_i^{(c)} > 0$ for $v_i \in O^{(c)}$, $Q_i^{(c)} < 0$ for $v_i \in D^{(c)}$, and $Q_i^{(c)} = 0$, otherwise. The notation $j : i \in S_j$ denotes the summation of the flow from all nodes v_j whose neighbor is node v_i, therefore, $\sum_{j:i \in S_j} f^{(c)}(j,i)$ denotes the total amount of information going into (inflow to) node v_i until time T. Similarly, $\sum_{j \in S_i} f^{(c)}(i,j)$ indicates the total amount of information going from (outflow from) node v_i until time T. The flow conservation simply states that the total flow coming into node v_i plus any information rate generated from node v_i is equal to the total flow going out from node v_i.

We note that the above formulation has two assumptions; first, the formulation requires the knowledge of all commodities when performing the optimization. This implies that the information generation rate on sources in all commodities during the whole duration of network lifetime is required to solve the linear program. Second, the above formulation does not reflect the sequences of the commodities. We know that the performance of online routing algorithm depends on the sequences of the commodities [11]. By sequences of commodities, we mean the sequences in which the traffic from different commodities appears in the network. These two assumptions imply that the above formulation is only suitable for off-line optimization, which has a limited use in practical scenario. In practice, the routing decision has to be made online; the routing decision is done upon the packet arrival. And it will be very hard, if not impossible, to know the information generation rate of all commodities a priori.

The qualitative performance comparison of online and off-line algorithm for routing algorithm is given in [11]. They show that *there is no online algorithm for message routing that has a constant competitive ratio in terms of network lifetime*, where the competitive ratio is defined as the ratio of the solution to online algorithm with respect to the optimal off-line solution. This implies that the performance of online algorithm is worse compared to the off-line algorithm. Moreover, the formulation in (1) is only suitable for the time until the first node dies. For the above reasons, we focus on designing a robust online algorithm by taking into account the connectivity of the remaining network in making the routing decision. The robustness of our proposed scheme comes from the fact that when the information generation is not known a priori and the routing decision is made on the fly, employing the connectivity weight in the routing decision keeps the remaining network connected as long as possible.

III. Facts from Spectral Graph Theory

Before we describe our proposed solution, we briefly summarize some important facts from spectral graph theory [5]–[7], [13]. We will only state the lemmas since they provide insight for understanding the proposed scheme. And we will use some of these lemmas to prove several properties of the proposed scheme. The complete proofs of the lemmas can be found in the above literatures.

A. Eigenvalues of Laplacian Matrix

In this subsection, we briefly discuss the definition of a Laplacian matrix, its eigenvalues and the relationship between the eigenvalues of Laplacian matrix and the connectivity of the associated graph. The following notations will be used throughout the paper: $G(V,E)$ is the graph with set of vertices V and set of edges E. We recall the number of vertices as $|V| = n$ and the number of edges as $|E| = m$. Moreover, we define G_{-v_i} as a graph resulted from removing vertex v_i and all its adjacent edges from the original graph G. In the rest of the paper, we will use vertex and node, interchangeably. Similarly, we will use link and edge, interchangeably. The Laplacian matrix associated with a graph is defined as follow.

Definition 1 (Laplacian matrix associated with a graph): In a graph $G(V,E)$, the Laplacian matrix associated with a graph, $\mathbf{L}(G)$ is an n by n matrix defined as follows:

$$L(i,j) = \begin{cases} d_{v_i} & \text{if } v_i = v_j, \\ -1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $i, j \in \{1, \cdots, n\}$ are the indices of the nodes. Equivalently, the Laplacian matrix $\mathbf{L}(G)$ can be expressed as:

$$\mathbf{L}(G) = T(G) - A(G), \tag{3}$$

where $T(G)$ is an n by n diagonal matrix associated with graph G with the (i,i)-th entry having value d_{v_i}, which represents the number of the neighboring nodes. $A(G)$ is a n by n adjacent matrix associated with graph G.

The eigenvalues of the Laplacian matrix, $\mathbf{L}(G)$, $(\lambda_0(G) \leq \lambda_1(G) \leq \cdots \leq \lambda_{n-1}(G))$ are usually referred to as the *graph spectra*. The following lemma describes the relationship between the graph spectra and the connectivity of a graph [5].

Lemma 1: Let's denote $0 = \lambda_0(G) \leq \lambda_1(G) \leq \cdots \leq \lambda_{n-1}(G)$ as the eigenvalues of the Laplacian matrix $\mathbf{L}(G)$. If G is connected, then $\lambda_1(G) > 0$. Moreover, if $\lambda_i(G) = 0$ and $\lambda_{i+1}(G) \neq 0$, then G has exactly i+1 disjoint connected components.

It is easy to observe that the Laplacian matrix $\mathbf{L}(G)$ has all zeros row sums, therefore, matrix $\mathbf{L}(G)$ has an eigenvalue 0 with the corresponding eigenvector $(1, \cdots, 1)^T$. Moreover, since $\mathbf{L}(G)$ is real, symmetric and nonnegative semi-definite, thus all the eigenvalues of $\mathbf{L}(G)$ should be real and nonnegative. It is obvious that the smallest eigenvalue of Laplacian matrix $\mathbf{L}(G)$ is zero. The above lemma indicates that if G is strongly connected (there exists a simple path from any initial node v_i to the terminal node v_j, where $i \neq j$) then the Laplacian matrix $\mathbf{L}$ has simple eigenvalue 0 (the eigenvalue 0 has the multiplicity of 1). Moreover, if the eigenvalue 0 of the Laplacian matrix $\mathbf{L}(G)$ has multiplicity n, then there are n disconnected components. Since we always work with the connected graph, we will focus on the second smallest eigenvalue of the Laplacian matrix, $\mathbf{L}(G)$ for the rest of this paper.

B. Fiedler value and vector

Let's denote the eigenvalues of the Laplacian matrix, $\mathbf{L}(G)$ associated with $G(V,E)$ as $\lambda_0(G), \cdots, \lambda_{n-1}(G)$ and the corresponding eigenvectors as $\nu_0(G), \cdots, \nu_{n-1}(G)$. Obviously, $\nu_0(G) = e = (1, \cdots, 1)^T$. Suppose that the graph $G(V,E)$ is strongly connected (the second smallest eigenvalue is strictly larger than zero, $\lambda_1(G) > 0$). This second smallest eigenvalue can be represented as (Courant-Fisher Theorem [14])

$$\lambda_1(G) = \min_{x^T x = 1, x^T \nu_0(G) = 0} x^T \mathbf{L}(G) x \tag{4}$$

The second smallest eigenvalue of the Laplacian matrix is always referred to as the algebraic connectivity of the graph G [6]. It is also called as *Fiedler value* of a graph. The reason for calling the second smallest eigenvalue as the algebraic connectivity of a graph G comes from the following lemmas [6].

Lemma 2: If G_1 and G_2 are edge-disjoint graphs with the same vertices, then $\lambda_1(G_1) + \lambda_1(G_2) \leq \lambda_1(G_1 \cup G_2)$.

Lemma 3: The Fiedler value $\lambda_1(G)$ is non-decreasing for graphs with the same set of vertices, i.e. $\lambda_1(G_1) \leq \lambda_1(G)$, if $G_1(V, E_1)$, $G(V,E)$, and $E_1 \subseteq E$.

We observe that G and G_1 have the same number of vertices. Since G_1 has fewer edges compared to G and $E_1 \subseteq E$, this implies that G_1 is less connected compared to G.

From Lemma 3, we have that the Fiedler value corresponding to G_1 is smaller than G, $\lambda_1(G_1) \leq \lambda_1(G)$. It is in this sense that the Fiedler value represents the degree of connectivity in a graph. Finally, the relation of Fiedler value for graph obtained from removing a vertex and all its adjacent edges is given by the following lemma [6].

Lemma 4: Let G_{v_i} be a graph obtained from removing vertex v_i from G and all the adjacent edges. Then $\lambda_1(G_{v_i}) \geq \lambda_1(G) - 1$.

The following two lemmas give some upper and lower bounds for the Fiedler value. These lemmas will be used to prove some properties of the proposed algorithm in Section V.

Lemma 5: Let $G(V, E)$, d_{v_i} be the degree of node v_i and $|V| = n$ be the number of vertices, then

$$\lambda_1(G) \leq \left[\frac{n}{n-1}\right] \min_{v_i} d_{v_i}. \tag{5}$$

Lemma 6: Let $\varepsilon(G)$ be the edge connectivity of the graph G (the minimal number of edges whose removal would result in losing connectivity of the graph G). Then, we have

$$\lambda_1(G) \geq 2\varepsilon(G)\left[1 - cos\left(\frac{\pi}{n}\right)\right], \tag{6}$$

where n is the number of vertices $|V| = n$.

IV. Keep-Connect Algorithms

In this section, we use the facts of spectral graph theory described in the previous section to develop online routing algorithms to maximize the network lifetime, which is defined as the time until the network becomes disconnected. Before describing the routing algorithms, let's first consider how to measure the degree of connectivity of the remaining graph in the routing metric. We propose to quantify the connectivity of the remaining graph based on the Fiedler value. Recall from Section III-B, the Fiedler value qualitatively represents the connectivity of a graph in the sense that the larger the Fiedler value is, the more connected the graph will be. The degree of connectivity of the remaining graph can be quantified by the Fiedler value of the graph resulted from removing that particular node and all the edges connected to that node from the original graph. We design the connectivity weight of each node as $1/\lambda_1(G_{-v_i})$, where G_{-v_i} denotes a graph resulted from removing node v_i and all its adjacent edges from the original graph G. In this way, the node that causes severe reduction in the remaining algebraic network connectivity will be avoided when performing the routing decision. This is due to the fact that $\lambda_1(G_{-v_i})$ measures the degree of connectivity of the graph after removing node v_i and all the edges connected to that node. We note that if v_i is an articulation node [15], (i.e. node after its removal results in disconnected network), then $\lambda_1(G_{-v_i})$ theoretically equals to zero. This will cause numerical problem when we embed the weight in the routing algorithm. To avoid this problem, we introduce a small threshold, ϵ. If $\lambda_1(G_{-v_i}) <= \epsilon$, we set $\lambda_1(G_{-v_i}) = \epsilon$. The detail of the algorithm is given in Table I. In short, the proposed algorithm avoids using the articulation nodes to keep the remaining network connected. We set the threshold as $\epsilon = 10^{-5}$ throughout this paper.

TABLE I
Keep-Connect using Fiedler value

```
Let $G(V, E)$ be the original graph. Let's define the graph obtained after
removing node $v_i$ and all its adjacent edges as $G_{-v_i}(\{V - v_i\}, E_{-v_i})$.
Let also denote $\epsilon$ as small threshold.
  1. Initialization: Set nodes' weights as zeros $W(v_i) = 0, \forall v_i \in V$
  2. For each node $v_i$:
     a. Form the Laplacian matrix $\mathbf{L}(G_{-v_i})$ of graph as (2).
     b. Find the Fiedler value $\lambda_1(G_{-v_i})$ as (4).
     c. Set the weight of node as:
        $W(v_i) = 1/\lambda_1(G_{-v_i})$, if $\lambda_1(G_{-v_i}) > \epsilon$.
        Else, set $W(v_i) = 1/\epsilon$.
  End for
```

We note that in order to maintain the remaining algebraic connectivity of a network as long as possible in the routing algorithm, route metric should reflect the effect of selecting one particular route on the remaining (algebraic) connectivity of the network. In other words, the severity effect of the selection of one particular route on the connectivity measure of a network should be quantified. Since the route metric is the sum of the links' costs that constitute the route [3], [4], [16], the link cost should also reflect the reduction in the remaining connectivity of the network. In the following subsections, we present proper modification on the existing routing algorithms by incorporating the connectivity weight into the routing metrics.

A. MMKC, MHKC and MMREKC routing algorithms

In this subsection, we first review the idea of existing routing algorithms and proceed with the modification to include the keep-connect algorithm. The minimum hop routing is usually used in ad-hoc or wire-line network to minimize the delay in the packet delivery. The routing algorithm chooses the route with the minimum hop (MH) [16]. The link cost between two nodes is 1. To modify this algorithm to reflect the remaining connectivity in the network, we embed the connectivity weight in the link cost between two nodes, so that it becomes $c(u, v) = \frac{1}{2}[W(u)^y + W(w)^y]$. We call this algorithm minimum hop while keeping the connectivity, (MHKC(y)). Both minimum hop and MHKC routing algorithms can be computed using standard Dijkstra algorithm.

The max-min residual energy (MMRE) algorithm [4] selects the route that maximizes the minimum residual energy of nodes in the route. The link cost for MMRE is represented as

$$c(u, w) = \min\{\underline{\mathcal{E}}_u(t), \underline{\mathcal{E}}_w(t)\}, \tag{7}$$

where $\underline{\mathcal{E}}_u(t)$ and $\underline{\mathcal{E}}_w(t)$ represent the residual energy at time t for transmitting node u and receiving node w, respectively. Another variant of MMRE is to maximize the minimum link cost that is represented as

$$c(u, w) = \underline{\mathcal{E}}_u(t) + \underline{\mathcal{E}}_w(t). \tag{8}$$

The max-min route can be implemented using modified Dijkstra algorithm [4]. Following the same way as before, we modify the variant of MMRC by embedding the connectivity into the link cost as

$$c(u, w) = \underline{\mathcal{E}}_u(t)W(u)^y + \underline{\mathcal{E}}_w(t)W(w)^y, \tag{9}$$

TABLE II
LINK COSTS FOR SEVERAL ROUTING ALGORITHMS WITH/OUT CONNECTIVITY CRITERION

Algorithm / Method	Link cost
MH / Standard Dijkstra	$c(u,w)=1$
MHKC(y) / Standard Dijkstra	$c(u,w)=\frac{1}{2}[W(u)^y+W(w)^y]$
MMRE / Modified Dijkstra	$c(u,w)=\min\{\underline{\mathcal{E}}_u(t),\underline{\mathcal{E}}_w(t)\}$
Variant of MMRE / Modified Dijkstra	$c(u,w)=\underline{\mathcal{E}}_u(t)+\underline{\mathcal{E}}_w(t)$
MMREKC(y) / Modified Dijkstra	$c(u,w)=$ $\underline{\mathcal{E}}_u(t)W(u)^y+\underline{\mathcal{E}}_w(t)W(w)^y$
MMKC(y) / Modified Dijkstra	$c(u,w)=W(u)^y+W(w)^y$

where $W(u)$ and $W(w)$ are the connectivity weights for node u and v, respectively. We name this algorithm as MMREKC(y). This algorithm selects the routing decision based on the remaining energy in nodes and the connectivity criterion. We note that maximizing the minimum residual energy of nodes is not the same as avoiding articulation nodes. Using MMREKC(y), one hopes to balance the choices of avoiding minimum residual energy of nodes and avoiding articulation nodes. If we set the residual energy of nodes in MMRECK(y) as one, we will get the max-min remaining connectivity (MMKC(y)) algorithm. We note that the MMKC is similar to MHKC except that the MMKC uses the modified dijkstra algorithm and MHKC uses the standard dijkstra algorithm. MMKC algorithm only avoids overusing some articulation nodes. Precisely, the routing algorithms that use connectivity criterion avoid using nodes that result in severe reduction in the remaining connectivity after they failed.

In all the keep-connect algorithms, y determines how important the connectivity weight should affect the routing cost. Since in MMREKC(y) and MHKC(y) the routing metrics try to simultaneously two different objectives (e.g., both minimum hop and connectivity in the MHKC algorithm), this variable y controls the importance of connectivity criterion in the routing metric. The link cost for the exiting algorithms and the proposed modifications are summarized in Table II. The connectivity weight, $W(\cdot)$, in the algorithms is calculated using the keep-connect algorithm (Table I). During the routing computation, if there are more than one nodes having weights of $1/\epsilon$, the route metric will be determined by the link cost of the rest of nodes in the route. In the case where there is a tie in route metric, any tie breaker rule is sufficient.

B. Minimum total energy while keeping connectivity (MTEKC) routing

In previous subsection, the routing metrics are determined either by the number of hops (minimum hop routing/MH) or minimum residual energy of nodes (MMRE), with/without the connectivity criterion. The minimum total energy (MTE) algorithm minimizes the total energy used for packet transmission and reception along the route [4]. By embedding the connectivity weights of nodes to the MTE algorithm, we obtain MTEKC(y). In particular, the link cost of MTEKC(y) is represented by

$$c(u,w)=e_t(u,w)\cdot W(u)^y+e_r(u,w)\cdot W(w)^y, \qquad (10)$$

where $e_t(u,w)$ and $e_r(u,w)$ are the transmit and receive energy for delivering a packet from node u to w. The

TABLE III
MTEKC(Y)

1. For any source-destination pairs, find the minimum total energy path with edge cost as: $e_t(v_i,v_j)W(v_i)^y+e_r(v_i,v_j)W(v_j)^y$ for $v_i\in V$, $v_j\in S_{v_i}$, where $e_t(v_i,v_j)$ and $e_r(v_i,v_j)$ are the transmit and receive energy for delivering a packet from node v_i to v_j. S_{v_i} denotes the neighbors of node v_i. $W(v_i)$ is the weight of node v_i.
2. If node dies, recompute the alive nodes' weight using *Keep-Connect* algorithm. Redo step 1.

MTEKC(y) minimizes the total transmit energy while trying to keep the remaining network as connected as possible. The complete algorithm for MTEKC is shown in Table III. One of the purposes of introducing the variable y is to limit the energy consumption of the MTEKC. In principle, the algorithm will never achieve both the best energy efficient and the most robust connectivity route. Introducing y provides a way to trade-off between the two objective functions. This is particularly true when the network is large as we will prove the bound of the energy consumption in the MTEKC algorithm in Section V. This bound can be easily controlled by the variable y. Similar to the previous keep-connect algorithms, in the case when there are more than one nodes having weights of $1/\epsilon$, the rest of the nodes in these different routes will determine which overall routing metrics are smaller. In the tie situation, any tie breaker rule is sufficient.

It is important to emphasize the MTE algorithm is less effective compared to MMRE type algorithms *only* when one considers the first node dies as the network lifetime definition. In our case, since we employ the time until the network becomes disconnected as the network lifetime, the above argument is not necessary true in general. In fact this phenomena can be observed in [4]. In [4], after more than one node dies, the expiration time (network lifetime) for the MTE is higher than MMRE. We will further discuss this phenomenon in the Section VII. In our simulation, we typically observe that more than one node fails before the network becomes disconnected (cf. Figure 4).

V. UPPER BOUND ON THE ENERGY CONSUMPTION OF MTEKC(Y) ALGORITHM

In this section, we continue to show the upper bound on the energy consumption of the MTEKC(y) algorithm. It is well-known from [11] that there is a tradeoff between the energy consumption in the route and avoiding overuses of popular nodes in achieving the maximum lifetime/total delivered packets before the network becomes disconnected. Similarly, there is a trade-off between the energy efficiency and the connectivity robustness. Hence, it is important to show the bound on the energy consumption of our proposed algorithm. In this section, we first derive some properties of our proposed algorithm and we employ these properties to prove the upper bound of the energy consumption. For simplicity we only include the transmit energy between two nodes in calculating the energy of the route. We denote r^* as the minimum total energy route connecting any fixed source node v_0 and destination node v_d. Equivalently, the MTE route is represented as

$$r^* = \arg \min_{r \in R(v_0,v_d)} \sum_{i=1}^{d(r)-1} e_t(v_i, v_{i+1}), \tag{11}$$

where $R(v_0, v_d)$ is the set of all routes connecting source node v_0 and destination node v_d, $d(r)$ is the number of hops in the route. Furthermore, we denote $r^\dagger$ as the MTEKC(y) route obtained using Fiedler value and this route satisfies

$$r^\dagger = \arg \min_{r \in R(v_0,v_d)} \sum_{i=1}^{d(r)-1} \frac{e_t(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^y}. \tag{12}$$

In the following, we give some simple lemmas on the lower bound and upper bound of the metric used in the keep-connect algorithm.

Lemma 7: [Lower bound of MTEKC(y) metric] For each route, the MTEKC(y) employing the Fiedler value metric has the following property

$$\sum_{i=0}^{d-1} e_t(v_i, v_{i+1}) W(v_i)^y \geq \left(\frac{n-2}{n-1} \frac{1}{\min_i d_{v_i}(G)}\right)^y \sum_{i=0}^{d-1} e_t(v_i, v_{i+1}), \tag{13}$$

$$\geq \left(\frac{(n-2)n}{2(n-1)m}\right)^y \cdot \sum_{i=0}^{d-1} e_t(v_i, v_{i+1}), \tag{14}$$

where d_{v_i} is the degree of node v_i in the graph, n is the number of vertices in the graph, and m is the number of edges in the graph.

Proof: To prove these inequalities, we require the upper bound of the Fiedler value as follows (stated in Lemma 5). Consider $G(V,E)$ and let $d_{v_i}(G)$ be the degree of node v_i in graph G. Then,

$$0 < \lambda_1(G) \leq \frac{n}{n-1} \min_i d_{v_i}(G). \tag{15}$$

Now, consider the graph G_{-v_i} obtained from graph G by removing node v_i and all edges connecting to node v_i. Obviously,

$$\lambda_1(G_{-v_i}) \leq \frac{n-1}{n-2} \min_i d_{v_i}(G_{-v_i}) \leq \frac{n-1}{n-2} \min_i d_{v_i}(G), \tag{16}$$

since the minimum degree of graph G_{-v_i} is smaller or equal to minimum degree of graph G. Now, using keep-connect algorithm with Fiedler value, we have

$$\sum_{i=0}^{d-1} e_t(v_i, v_{i+1}) \cdot W(v_i)^y = \sum_{i=0}^{d-1} \frac{e_t(v_i,v_{i+1})}{\lambda_1(G_{-v_i})^y}$$
$$\geq \left(\frac{n-2}{(n-1)\min_i d_{v_i}(G)}\right)^y \cdot \sum_{i=0}^{d-1} e_t(v_i, v_{i+1}). \tag{17}$$

Since $n \cdot \min_i d_{v_i} \leq \sum_i d_{v_i} = 2m$, we have

$$\left(\frac{1}{\min_i d_{v_i}(G)}\right)^y \geq \left(\frac{n}{2m}\right)^y. \tag{18}$$

Then, we obtain the second inequality

$$\left(\frac{n-2}{(n-1)\min_i d_{v_i}(G)}\right)^y \cdot \sum_{i=0}^{d-1} e_t(v_i, v_{i+1}) \geq$$
$$\left(\frac{(n-2)n}{2(n-1)m}\right)^y \cdot \sum_{i=0}^{d-1} e_t(v_i, v_{i+1}). \tag{19}$$

■

Lemma 8: [Upper bound of MTEKC(y) metric] For each route, the MTEKC(y) employing the Fiedler value metric has the following property

$$\sum_{i=0}^{d-1} e_t(v_i, v_{i+1}) \cdot W(v_i)^y \leq$$
$$\left[\frac{1}{2\left(\varepsilon(G)-1\right)\left(1-\cos\left(\frac{\pi}{n-1}\right)\right)}\right]^y \sum_{i=0}^{d-1} e_t(v_i, v_{i+1}), \tag{20}$$

where $\varepsilon(G)$ is the edge-cut or edge connectivity of the graph. The edge-cut/edge connectivity is defined as the minimal number of edges whose removal would result in disconnected graph. n is the number of vertices in the graph.

Proof: Similar to Lemma 7, we use the lower bound of Fiedler value in Lemma 6. Consider $G(V,E)$ and let $\varepsilon(G)$ be the edge-cut of the graph. Then, $\lambda_1(G) \geq 2\varepsilon(G)[1 - \cos(\pi/n)]$. Now, consider the graph G_{-v_i} obtained from graph G by removing node v_i and all edges connecting to node v_i. From upper bound of Fiedler value, we have

$$\lambda_1(G_{-v_i}) \geq 2\varepsilon(G_{-v_i})[1 - \cos(\pi/(n-1))]. \tag{21}$$

Since, $\varepsilon(G_{-v_i}) \geq \varepsilon(G) - 1$. Hence,

$$\lambda_1(G_{-v_i}) \geq 2[\varepsilon(G) - 1][1 - \cos(\pi/(n-1))]. \tag{22}$$

Follow the similar proof in Lemma 7, we can obtain

$$\sum_{i=0}^{d-1} e_t(v_i, v_{i+1}) \cdot W(v_i)^y = \sum_{i=0}^{d-1} \frac{e_t(v_i,v_{i+1})}{\lambda_1(G_{-v_i})^y}$$
$$\leq \left[\frac{1}{2[\varepsilon(G)-1][1-\cos(\frac{\pi}{n-1})]}\right]^y \cdot \sum_{i=0}^{d-1} e_t(v_i, v_{i+1}).$$

■

Lemma 9: [Complete Graph] For a complete graph, the MTEKC(y) route employing the Fiedler value metric is the same as the MTE route.

Proof: From the definitions of MTE route and MTEKC(y) with Fiedler value route, we have the following inequalities

$$\sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1}) \leq \sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1}) \tag{23}$$

$$\sum_{i=1}^{d(r^\dagger)-1} \frac{e_t(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^y} \leq \sum_{i=1}^{d(r^*)-1} \frac{e_t(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^y} \tag{24}$$

We note that removing one nodes from a complete graph with n nodes results in another complete graph with $n-1$ nodes. Therefore, we have $\lambda_1(G_{-v_i}) = n-1$. Simplifying (24), we have

$$\sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1}) \leq \sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1}). \tag{25}$$

Combining with (23), we have

$$\sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1}) = \sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1}). \tag{26}$$

Since it is impossible to have two different routes with the same total transmit energy in random network deployment for fixed source node v_0 and destination node v_d, we conclude that $r^* = r^\dagger$. ■

Now we are ready to develop the upper bound on the energy consumed in the MTEKC(y) algorithm with Fiedler value. The following theorem gives the upper bound on the consumed energy

Theorem 1: The energy consumed in the MTEKC(y) using Fiedler value satisfies the following upper bound

$$\sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1}) \leq \left[\frac{(n-1)m}{n(n-2)(\varepsilon(G)-1)(1-\cos(\pi/(n-1)))}\right]^y \sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1}) \quad (27)$$

Proof: From inequality (24), Lemma 7, and Lemma 8, we have

$$\left(\frac{(n-2)n}{2(n-1)m}\right)^y \sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1}) \leq \sum_{i=1}^{d(r^\dagger)-1} \frac{e_t(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^y} \quad (28)$$

$$\sum_{i=1}^{d(r^*)-1} \frac{e_t(v_i,v_{i+1})}{\lambda_1(G_{-v_i})^y} \leq \left[\frac{1}{2[\varepsilon(G)-1][1-\cos(\frac{\pi}{n-1})]}\right]^y \sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1}) \quad (29)$$

Combining the above two inequalities and (24), we have

$$\left[\frac{(n-2)n}{2(n-1)m}\right]^y \sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1}) \leq \left[\frac{1}{2(\varepsilon(G)-1)(1-\cos(\frac{\pi}{n-1}))}\right]^y \sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1}) \quad (30)$$

$$\sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1}) \leq \left[\frac{m(n-1)}{n(n-2)(\varepsilon(G)-1)(1-\cos(\frac{\pi}{n-1}))}\right]^y \sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1}) \quad (31)$$

■

The above theorem gives the upper bound on the energy consumed in MTEKC(y) route compared to the minimum total energy for routing the packet. The following theorem gives the bound on the ratio of energy consumed by MTEKC(y) using Fiedler value when the number of nodes becomes large.

Theorem 2: Suppose that the network generated satisfies $m = a_1 \cdot n$ and $\varepsilon(G) - 1 = a_2$, where a_1 and a_2 are some constants. Then the upper bound on the ratio of energy consumed can be presented as follow

$$\frac{\sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1})}{\sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1})} = O((n^2)^y) \quad (32)$$

Proof: Using the assumption of the theorem, we have

$$\frac{\sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1})}{\sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1})} \leq \left(\frac{a_1 n(n-1)}{a_2 n(n-2)(1-\cos(\frac{\pi}{n-1}))}\right)^y$$

$$\frac{\sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1})}{\sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1})} \leq C\left(\frac{n(n-1)(1+\cos(\frac{\pi}{n-1}))}{n(n-2)\sin^2(\frac{\pi}{n-1})}\right)^y, \quad (33)$$

where $C = (a_1/a_2)^y$. As $n \to \infty$, we have

$$\frac{\sum_{i=1}^{d(r^\dagger)-1} e_t(v_i, v_{i+1})}{\sum_{i=1}^{d(r^*)-1} e_t(v_i, v_{i+1})} \leq C\left(\frac{(n-1)^2}{\pi^2}\right)^y, \quad (34)$$

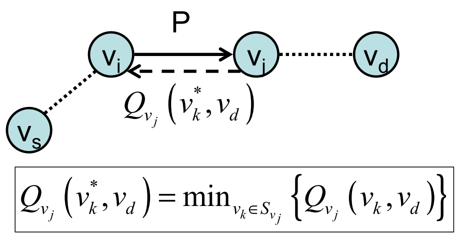
P
v_i
v_j
v_d
v_s
$Q_{v_j}(v_k^*, v_d)$
$Q_{v_j}(v_k^*, v_d) = \min_{v_k \in S_{v_j}} \{Q_{v_j}(v_k, v_d)\}$

Fig. 1. Exchange and Update Q-value

where we have used small angle approximation in sinusoidal function, $sin(\theta) \approx \theta$, as $\theta \ll 1$. Hence, we have (32). ■

From this theorem, we see that when the network is very large, the ratio of energy increases as a quadratic function of number of nodes, compared to the minimum energy used to route a packet. This ratio of energy can be easily controlled by the parameter y, for instance if $y = 1/2$, then the ratio of consumed energy increases as a linear function of number of nodes in the network. In the extreme case, setting $y = O(1/n)$ makes the proposed algorithm approaching to MTE as $n \to \infty$.

VI. Distributed Implementation and Learning Algorithm

In this section, we present the distributed implementation of the proposed robust connectivity routing algorithm. The method is based on the distributed reinforcement learning routing algorithm [17]. We note that the reinforcement learning algorithm has been shown to be an effective online decision making procedure in sensor network application [18]. The resulting algorithm can be characterized as a version of distributed Bellman-Ford algorithm that performs its path relaxation step asynchronously and online with the edge cost defined as weighted energy required to transmit packet in that hop [16]. Each node learns the routing decision by itself and maintains the best packet delivery cost to its destinations. In particular, each node v_i maintains a table of Q-values $Q_{v_i}(v_j, v_d)$, for $v_j \in S_{v_i}$, where v_j is in the set of node v_i neighbors, S_{v_i}, and node v_d is the destination. The $Q_{v_i}(v_j, v_d)$ has the interpretation of node v_i's best estimated cost that a packet would incur to reach its destination node v_d from node v_i when the packet is sent via node v_i's neighbor node v_j. Whenever node v_i wants to send packets to node v_d, it simply sends the packets to its neighbor that satisfies

$$v_j^* = \arg\min_{v_j \in S_{v_i}} Q_{v_i}(v_j, v_d). \quad (35)$$

The value in the Q-table will be exchanged between node v_i and v_j, whenever a packet is sent from node v_i to node v_j, and vice versa. The exchange mechanism is illustrated as in Figure 1. Whenever node v_i transmits a packet P to node v_j, node v_j feeds back

$$Q_{v_j}(v_k^*, v_d) = \min_{v_k \in S_{v_j}} Q_{v_j}(v_k, v_d) \quad (36)$$

to node v_i as shown in the figure. We note that $Q_{v_j}(v_k^*, v_d)$ represents the best estimated cost incurred when a packet is sent from node v_j to the destination node v_d. The node i uses this value to update its own Q-value as follow

$$Q_{v_i}(v_j, v_d) = (1-\delta)Q_{v_i}(v_j, v_d) + \delta[Q_{v_j}(v_k^*, v_d) + c(v_i, v_j)], \quad (37)$$

where $c(v_i, v_j)$ is the cost for sending packet from node v_i to node v_j, and $\delta \in [0,1]$ is the learning rate for the algorithm. It is important to point out that the storage requirement for the distributed algorithm is $O(d_{v_i}(n-1))$, since each node requires to store at most $d_{v_i}(n-1)$ Q-values, where d_{v_i} is the number of neighbors, node v_i has. Moreover, the distributed algorithm has $O(1)$ computational complexity per packet transmission, since for every packet transmission, the sender and the receiver update their corresponding Q-values according to (37).

The cost of sending a packet between node v_i and node v_j is related to the energy consumption for sending the packet. In particular, the costs of sending a packet for MTE and MTEKC routing algorithms are

$$\begin{aligned} &[\text{MTE}]: && c(v_i, v_j) = e_t(v_i, v_j) + e_r(v_i, v_j), \\ &[\text{MTEKC(y)}]: && c(v_i, v_j) = e(v_i, v_j)W(v_i)^y + e_r(v_i, v_j)W(v_j)^y, \end{aligned}$$

For MTE, $Q_{v_i}(v_j, v_d)$ represents the total energy consumption used to guarantee delivery a packet from node v_i to node v_d via node v_i's neighbor node, v_j. In contrast, $Q_{v_i}(v_j, v_d)$ in MTEKC represents the total energy consumption in delivering a packet from node v_i to v_d via v_j, while considering the connectivity of the remaining network. The procedure for implementing the MTEKC is summarized in Table IV. We note that when $\delta = 1.0$, the algorithm becomes the distributed Bellman-Ford iterations [16].

A. Improvement on distributed algorithm

In previous description, it is obvious that the quality of the routing decision depends on the accuracy of the Q-table. The more accurate the Q-table represents the network state, the better the routing decision will be. Moreover, the quality of the Q-table depends on how frequently it is updated. In the distributed scheme described in previous subsection, the Q-value is updated whenever a packet is transmitted from the transmitter to the receiver. This existing distributed algorithm can be enhanced by improving the accuracy of the Q-table. The accuracy of the Q-table can be improved by updating the Q-values not only when there is a packet transmission between two nodes, but nodes periodically send their neighboring nodes small packets containing the request on updating $Q_{v_i}(v_j, v_d)$. This can be straightforwardly done by updating $Q_{v_i}(N(v_i), v_d)$ as if there are packets to be sent from node v_i to nodes $N(v_i)$. This approach is similar to the use of packet radio organization packet (PROP) [19]. We note that the overhead encountered in each request on updating $Q_{v_i}(v_j, v_d)$ is only $Q_{v_j}(v_k^*, v_d)$ as in (36), which will be fed back by node v_j. This overhead depends on how many bits the Q-value is represented. Typically, 8 bits will be sufficient. It is important also to note that this small packet consumes negligible energy compared to the energy for the packet transmission. Therefore, these small packets can be sent more frequently to improve the learning speed of the Q-table. In a real application, there is a trade-off between how fast the link cost changes and how frequently the small packets should be exchanged. We will show in the simulation that the improvement in the distributed implementation achieves the near-centralized solution.

B. Distributed weight computation and discussion on the scalability

In the network initialization, each node will power up and start to broadcast control/organization packet similar to the DARPA packet radio network protocol [19]. This packet contains information about its neighboring nodes (nodes that are connected/reachable from this node). This packet will be broadcasted to all of its neighbors. The nodes that are one hop away from the origin node will also ripple this information outward in the tiered fashion by appending its own neighbor information. This is similar to the method of building tier table in [19]. Therefore, all the nodes have the correct view of the network topology. Before performing routing task, each node calculates its connectivity weight based on this topology view as in Table I. Precisely, the final topology view consists of the adjacency matrix of the graph. Since, nodes stay in their original places for their entire lifetime in the sensor network, the flooding of information occurs rarely. This event occurs whenever there are nodes joint the network or there are nodes that have failed. We assume that this event rarely happens after the network initialization.

In practical network, maintaining the algebraic connectivity of the whole large network may not be necessary. Based on different applications of sensor network, the sensor network typically can be decomposed into several smaller subnetworks which are interconnected by access points (APs). These APs may not have power limitation. Each AP is responsible for its own subnetwork. In order to exchange information between nodes within each subnetwork, it is very important to maintain the connectivity within each subnetwork. It is within this smaller subnetworks, the keep-connect algorithm is applied. In this way, the neighboring nodes information is only disseminated over the smaller subnetworks and the improved distributed algorithm can be done on the fly. Using this hierarchical structure, the routing across subnetworks can be handled via the APs. And, the proposed method can be used in this way in a very large network where the connectivity of the whole network is maintained via combination of local connectivity and the APs.

VII. Simulation Results

We simulate the routing algorithms in a packet level discrete-event simulator. The simulator initially deploys nodes in the network. All events are time-stamped and queued. The most current event will be dequeued and some task will be performed according to the type of the event. There are three types of events; the packet arrival event, the reporting event, and the sending event. In packet arrival event, packets are injected from source node to destination node. We assume the packet arrival follows the Poisson arrival process with mean μ. The reporting event occurs periodically to retrieve the simulation parameters such as the average delivery delay per packet, average hops per packet transmission, energy consumed per one delivered packet, and the number of packets

TABLE IV
DISTRIBUTED ASYNCHRONOUS MTEKC(Y)

```
1. Network Initialization: forward the adjacent neighbors information to all
   nodes in the network
2. Each node computes its own connectivity weight
3. Whenever node $v_i$ sends a packet to node $v_j$:
   a. Node $v_j$ informs $v_i$ its minimum cost transmitting a packet to the
      destination $v_d$, $Q_{v_j}(v_k^*, v_d)$
   b. Node $v_i$ updates its metric as:
      $Q_{v_i}(v_j, v_d) = (1-\delta)Q_{v_i}(v_j, v_d) + \delta[Q_{v_j}(v_k^*, v_d) + c(v_i, v_j)]$
   c. Node $v_i$ leaves other estimates unchanged.
4. When a node dies, the neighboring nodes flood this information to the
   rest of the network and repeat step 2
```

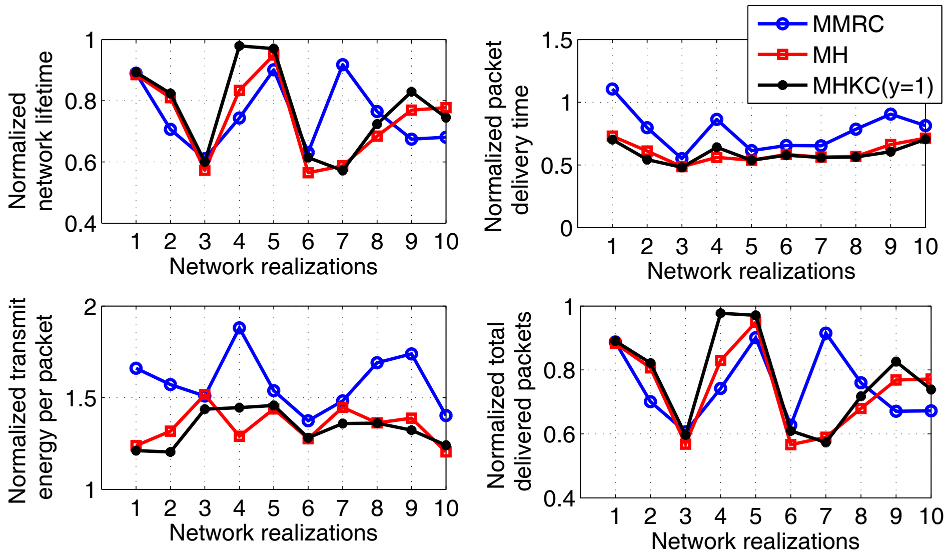

Fig. 2. Comparison of normalized metric for MMRC, MH, and MHKC(y=1) w.r.t. MTE algorithm, when the packet arrival follows the Poisson process with mean $\mu = 1.0$.

delivered in this report interval. All events that are neither the packet arrival events nor reporting events are the sending events. In the sending event, a packet is sent to the next hop. The next hop is determined based on the routing algorithm used. Whenever a packet arrives at a node, it is queued in the node's buffer and will be sent in the next transmission time. Whenever a packet reaches its destination, the number of delivered packets is incremented and the event associated with that packet is freed.

The channel between two adjacent nodes in the network is modeled as fading process that attenuates the transmit signal proportionally to the distance, $d^{-\alpha}$. We note that the main focus of this paper is on the static sensor network. In the static sensor network, the effect of fast fading is negligible, since practically the Doppler shift experienced by the receivers is zero. Therefore, this model is general enough to describe both the free space propagation and the two ray ground propagation model, which has been typically used in many ad-hoc simulators [20]. In this simulation section, we first generate 10 uniform random networks in the area of 100m by 100m with 36 nodes. The networks have average edges per node from 4.4 to 7.5. All nodes in the network initially have the same amount of energy. The source and destination are selected uniformly from the alive nodes. We use the network lifetime (time before the remaining network becomes disconnected), packet delivery time, average transmit energy per packet, and total delivered packet before the remaining network becomes disconnected as our performance metrics.

In Figure 2, we compare the performance of the normalized max-min residual connectivity (MMRC) routing, normalized the minimum hop (MH) routing, and normalized minimum hop while keeping connectivity (MHKC(y=1)) routing. All metrics are normalized with respect to the performance of MTE algorithm. From the figure, we observe that all the algorithms have lower network lifetime compared to the MTE algorithm. This also verifies that MTE is not less effective than MMRE type algorithm when one defines the network lifetime as the time before the remaining network becomes disconnected as discussed in Section IV. This is partly due to the fact that MH, MHKC, and MMRE do not use the transmission energy to guide the routing decision, hence the route is not energy efficient at all. This can be observed from the figure that all the algorithms consume 20% to 90% more energy per packet. However, the MH and MHKC take only 50% less time to deliver one packet, this is clear because MH based algorithms select route that has the minimum hops from source to destination, hence the resulting packet delivery delay is the smallest.

Since considering the energy efficient route is very important, we will focus on MTE type algorithm for the rest of

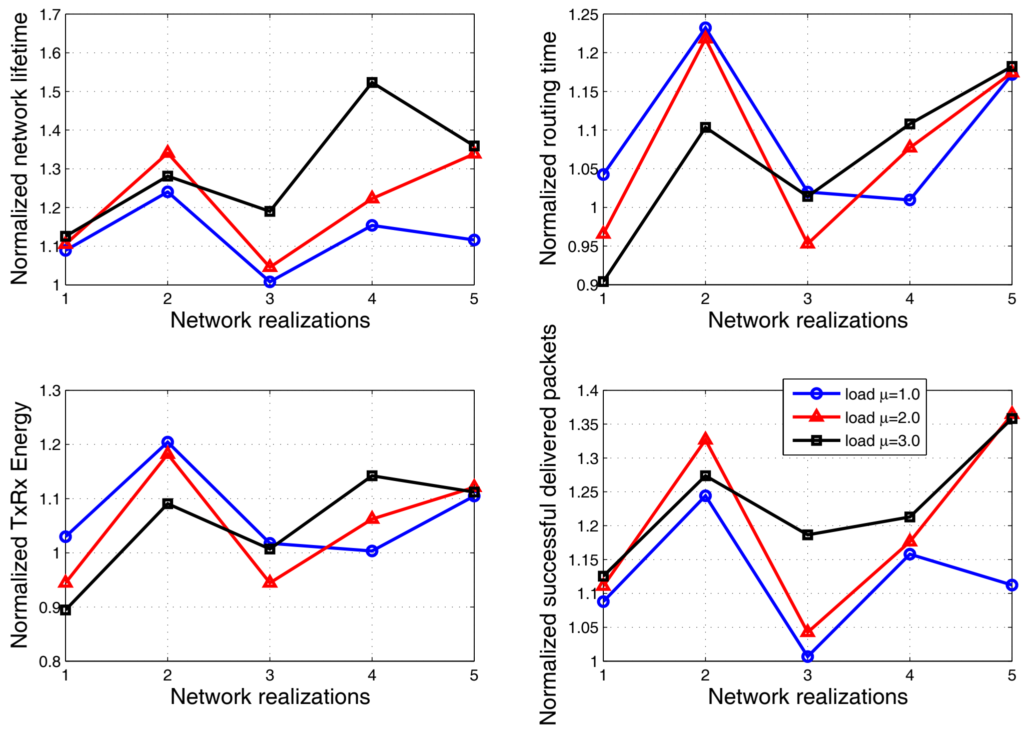

Fig. 3. Comparison of normalized metrics for MTEKC(y=1) w.r.t. to MTE for different packet arrival rate.

TABLE V
SIMULATION PARAMETERS SET 2

Network	1	2	3	4	5
Nodes	102	101	115	124	86
Algebraic network connectivity	0.3994	0.5679	0.7282	0.5543	0.1141

the simulation section. Next, we simulate both 2D Poisson network and uniform random network. The nodes in the 2D Poisson network are deployed/generated based on 2D Poisson process, while the nodes in uniform random network are generated based on uniform distribution. We also consider 2 types of source and destination pairs, namely converge-cast traffic and random traffic. The converge-cast traffic refers to the pattern where the sources are generated from alive nodes to a fixed destination (sink). And, the random traffic refers to the pattern where the source and destination are selected randomly from the nodes that have not failed. We generate 5 networks, where nodes are generated using 2D Poisson point process with the following parameters. Networks with averaged 100 nodes are generated using 2D Poisson point process in the area of 100m by 100m. The transmit energy per packet between two nodes is equivalent to $3 \times 10^{-3} d^2$, where d is the distance between the nodes. The maximum transmit power equals to 1.4 Watt per packet, receive power equals to 0.7 Watt per packet. This implies that the farthest node that can be reached by a node is about 21.602m away. The initial energy of all nodes is equal to 10000.0 energy unit. We also show the algebraic network connectivity of the original graph in Table V. Figure 3 shows the simulation results for normalized MTEKC(y=1) w.r.t. MTE algorithm for the generated networks. As before, this figure shows the normalized performance metrics with respect to MTE algorithm. These metrics are collected from the initialization of the network until the network becomes disconnected. The x-axes of these sub-figures denote the network number shown in Table V. From this figure, we observe that the improvement of keep-connect algorithm w.r.t MTE is about 20% for load ($\mu = 1.0$) in network 2. In summary, one can observe that the proposed algorithm achieves from 10% to 53% increase in the network lifetime for broad network loads. Similarly, the proposed algorithm achieves around 10% $\sim$ 36% improvement in the total delivered packets. We note that the keep-connect algorithm is about 20% less energy efficient compared to MTE. Besides being able to have longer lifetime and larger total delivered packets, our proposed algorithm is also more robust in terms of the algebraic network connectivity. This can be seen from Figure 4. This figure shows the decrease in the algebraic network connectivity whenever a node dies. The x-axis is the number of failed nodes before the network becomes disconnected and the y-axis is the algebraic network connectivity. We note that the zero algebraic network connectivity implies the disconnected network. It is obvious that our algorithm is more robust in terms of algebraic network connectivity. As shown in Figure 4, there is typically more than one node dies before the network becomes disconnected. We also simulate converge-cast traffic in network 1 of Table V and run the simulation for 100 realizations. We randomly select the destination to be node 94 and assume that it has infinite energy. Figure 5 shows that our proposed algorithm constantly outperforms the MTE algorithm.

Next, we show the performance of the distributed imple-

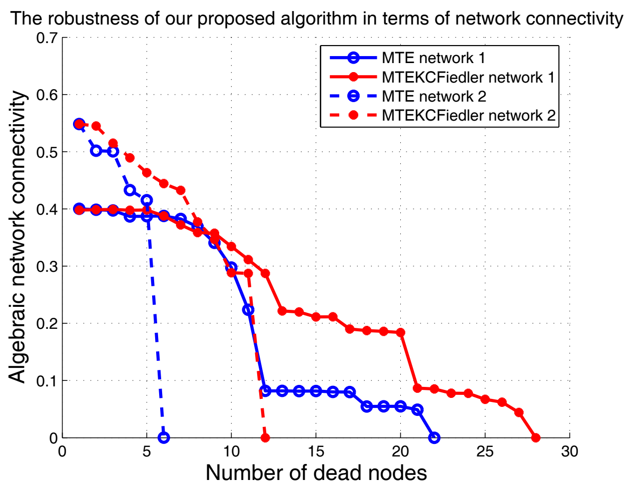

Fig. 4. Robustness of our proposed algorithm in terms of network connectivity

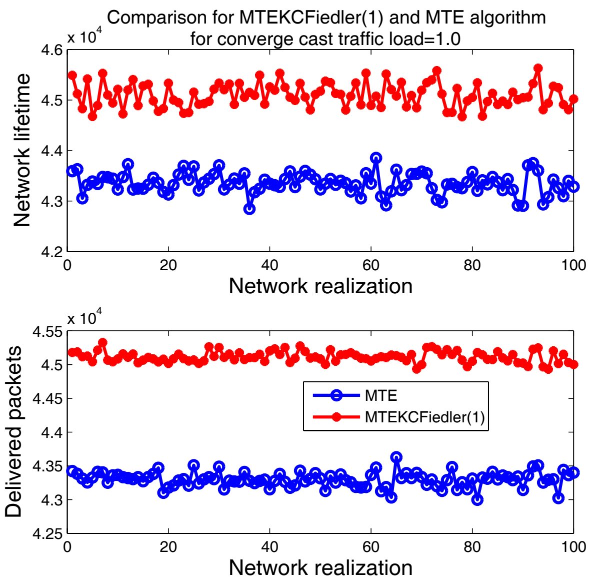

Fig. 5. Comparison of normalized metrics for MTEKC(y=1) w.r.t. to MTE for converge cast traffic.

mentation presented in Section VI in Figure 6. The distributed solution sends around 10 small packets in between sending the actual packets to facilitate better learning result. If one node dies, each node increases the sending of small packets to quickly learn the correct route. The implementation is straightforward. The learning process for the distributed solution for the converge cast traffic in network 1 is shown in Figure 6. This figure shows that the improved distributed solution can nearly achieve the centralized solution.

Finally, in order to get statistics of improvement of keep-connect algorithm, we study the effect of routing algorithm on random network where nodes are randomly placed uniformly in an area of 100m by 100m with random traffic. For each number of nodes (100 to 200 nodes), we generated 20 networks, we averaged the normalized network lifetime for MTEKC($y = 1, 2, 3$) with respect to MTE algorithm. The results are plotted in Figure 7 to 9. In these Figures we show both the mean, m and unbiased standard deviation, s of the performance metrics given below by means of error bar,

$$m = \frac{1}{N}\sum_{n=1}^{N} x_n,$$

$$s = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_n-\mu)^2},$$

where $N = 20$ represents 20 network realizations. x_n denotes the normalized network lifetime and normalized number of nodes failed before the network becomes disconnected in the left and right sub-figures of Figures 7 to 9, respectively.

We note that Figure 7 shows the case where the attenuation is $\alpha = 2.0$ and load is $\mu = 1.0$. In contrast, we also simulate the case where $\alpha = 4.0$ and $\mu = 1.0$ in Figure 9. Comparing these figures, we conclude that the proposed algorithm is not sensitive to the choice of attenuation coefficient α. From all these figure, we show that the proposed algorithm achieves about 10% to 20% performance improvement compared to MTE algorithm. Moreover, the proposed algorithm is more robust in terms of algebraic network connectivity, which may be a much more important criterion to be considered in the network protocol design. This can be seen from these Figures, there is more number of nodes failed before the network becomes disconnected in our proposed algorithm.

VIII. Conclusions

In this paper, we employ the time before the network becomes disconnected as our definition of network lifetime. Using this definition, we propose a class of routing algorithms called *keep-connect* algorithms that use *computable measures* of network connectivity in determining how to route packets. The algorithm embeds the importance of a node when making the routing decision. The importance of a node is quantified by the Fiedler value of the remaining network when that particular node dies. The proposed algorithm achieves $10 \sim 20\%$ better network lifetime (time before the network becomes disconnected) and total delivered packets compared to MTE algorithm in low to high arrival rate. More importantly, the proposed algorithm is more robust in terms of algebraic network connectivity. We also present a distributed implementation that nearly achieves the centralized solution.

References

[1] Tech. Review, “10 emerging technologies that will change the world,” *Technology Review*, pp. 33–49, Feb. 2003.
[2] C.-Y. Chong and S. P. Kumar, “Sensor networks: evolution, opportunities, and challenges,” *IEEE Wireless Commun.*, vol. 91, no. 8, pp. 1247–1256, Aug. 2003.
[3] J.-H. Chang and L. Tassiulas, “Maximum lifetime routing in wireless sensor networks,” *IEEE/ACM Trans. Networking*, vol. 12, no. 4, pp. 609–619, Aug. 2004.
[4] C.-K. Toh, “Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks,” *IEEE Commun. Mag.*, vol. 39, no. 6, pp. 138–147, June 2001.
[5] F. R. K. Chung, *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics, no. 92, American Mathematical Society, 1997.
[6] M. Fiedler, “Algebraic connectivity of graphs,” *Czechoslovak Mathematical J.*, vol. 23, pp. 298–305, 1973.
[7] ——, “A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory,” *Czechoslovak Mathematical J.*, vol. 25, pp. 619–633, 1975.

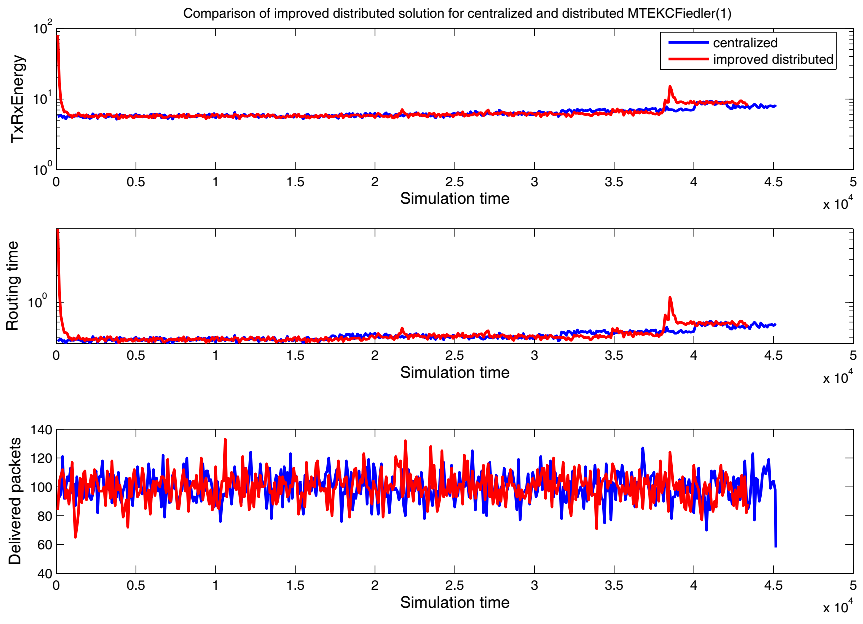

Fig. 6. Performance of Tx and Rx Energy, Routing time, and the delivered packets for improved distributed solution when $\mu = 1.0$.

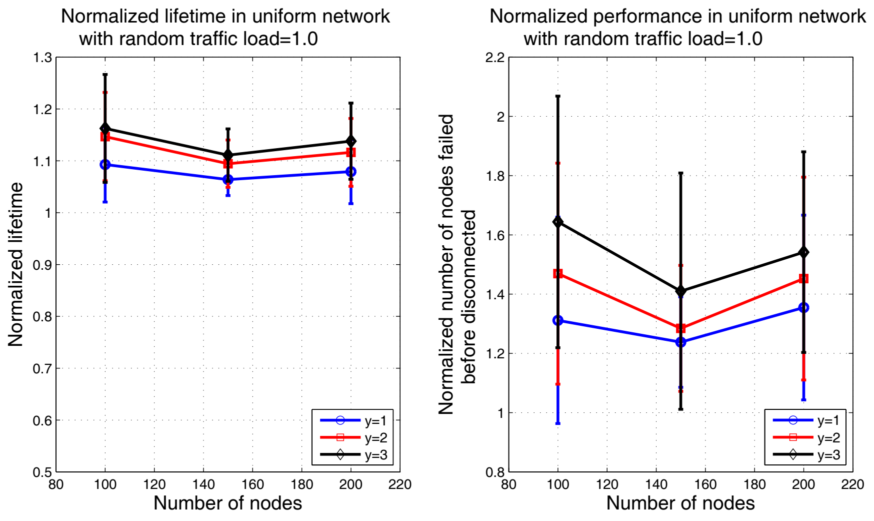

Fig. 7. Normalized performance in random network with random traffic when the packet arrival load is $\mu = 1.0$ and $\alpha = 2.0$

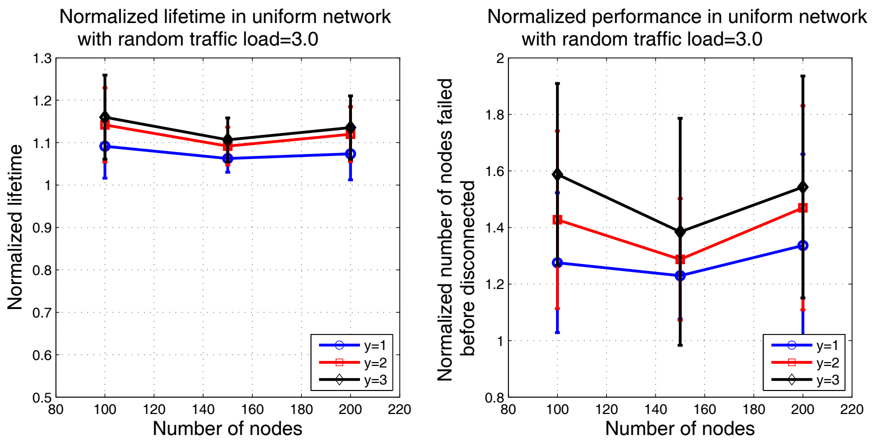

Fig. 8. Normalized performance in random network with random traffic when the packet arrival load is $\mu = 3.0$ and $\alpha = 2.0$

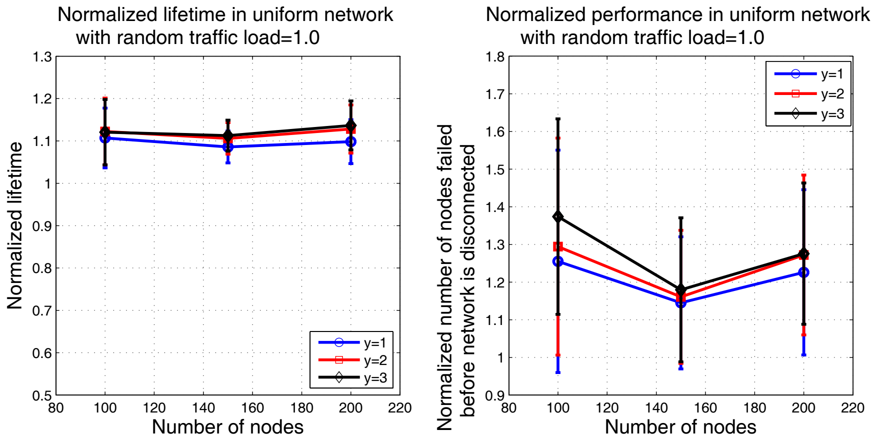

Fig. 9. Normalized performance in random network with random traffic when the packet arrival load is $\mu = 1.0$ and $\alpha = 4.0$

[8] T. S. Rappaport, *Wireless Communications: Principle and Practice*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[9] A. Chandrakasan, W. Heinzelman, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annual Hawaii International Conference on System Sciences*, pp. 1–10, Jan. 2000.

[10] D. M. Blough and P. Santi, "Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks," in *Proc. 8th Annual International Conference on Mobile Computing and Networking*, pp. 183–192, 2002.

[11] Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless ad hoc networks," in *Proc. 4th Annual IEEE/ACM MOBICOM*, Rome, Italy, pp. 97–107, July 2001.

[12] A. Sankar and Z. Liu, "Maximum lifetime routing in wireless ad-hoc networks," in *Proc. IEEE INFOCOM*, pp. 1089–1097, Mar. 2004.

[13] B. Mohar, "Some applications of Laplace eigenvalues of graphs," Hahn and G. Sabidussi, eds., *Graph Symmetry: Algebraic Methods and Applications*, NATO ASI Series C, Kluwer, Dordrecht, vol. 497, pp. 227–275, 1997.

[14] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 1st ed. Cambridge, UK.: Cambridge University Press, 1985.

[15] R. Diestel, *Graph Theory*, 3rd ed. Springer-Verlag, 2005.

[16] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, MA.: Athena Scientific, 1999.

[17] M. L. Littman and J. A. Boyan, "A distributed reinforcement learning scheme for network routing," *Advances in Neural Infromation Processing Systems*, vol. 6, pp. 670–678, 1993.

[18] C. Pandana and K. J. R. Liu, "Near-optimal reinforcement learning framework for energy-aware sensor communications," *IEEE J. Select. Areas Commun.*, vol. 23, no. 4, pp. 788–797, Apr. 2005.

[19] J. Jubin and J. D. Tornow, "The darpa packet radio network protocols," *Proc. IEEE*, vol. 95, no. 1, pp. 21–34, Jan. 1987.

[20] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad-hoc routing," in *Proc. MOBICOM*, pp. 70–84, July 2001.

Charles Pandana received his B.S. degree and M. S. degree in electronics engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1998 and 2000, and the Ph.D. degree in electrical and computer engineering from University of Maryland in 2005. He is currently working as a system research engineer in ArrayComm LLC. His research interests include stochastic modeling/learning and system level/network performance analysis for next generation wireless networks. He is a member of the IEEE Signal Processing and Communication societies.

K. J. Ray Liu (F-03) is Professor and Associate Chair, Graduate Studies and Research, of Electrical and Computer Engineering Department, and Distinguished Scholar-Teacher of University of Maryland, College Park. He leads the Maryland Signals and Information Group conducting research encompassing broad aspects of information technology including signal processing, communications, networking, information forensics and security, biomedical and bioinformatics.

Dr. Liu is the recipient of numerous honors and awards including best paper awards from IEEE Signal Processing Society (twice), IEEE Vehicular Technology Society, and EURASIP; IEEE Signal Processing Society Distinguished Lecturer, EURASIP Meritorious Service Award, and National Science Foundation Young Investigator Award. He also received various teaching and research recognitions from University of Maryland including university-level Invention of the Year Award and college-level Poole and Kent Company Senior Faculty Teaching Award.

Dr. Liu is Vice President Publications and on the Board of Governor of IEEE Signal Processing Society. He was the Editor-in-Chief of IEEE SIGNAL PROCESSING MAGAZINE and the founding Editor-in-Chief of EURASIP JOURNAL ON APPLIED SIGNAL PROCESSING.

His recent books include *Cooperative Communications and Networking*, Cambridge University Press, 2008; *Resource Allocation for Wireless Networks: Basics, Techniques, and Applications*, Cambridge University Press, 2008; *Ultra-Wideband Communication Systems: The Multiband OFDM Approach*, IEEE-Wiley, 2007; *Network-Aware Security for Group Communications*, Springer, 2007; *Multimedia Fingerprinting Forensics for Traitor Tracing*, Hindawi, 2005.