Efficient Topology Design in Time-Evolving and Energy-Harvesting Wireless Sensor Networks

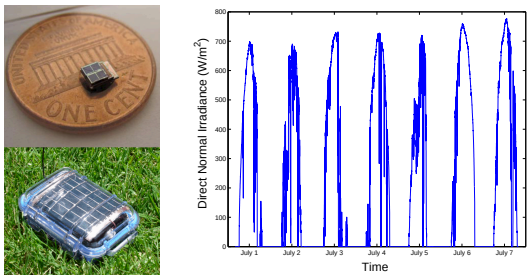
Fan Li* Siyuan Chen† Shaojie Tang‡ Xiao He* Yu Wang†
* School of Computer Science, Beijing Institute of Technology, Beijing, 100081, China.
† Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA.
‡ Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA.

***Abstract*—Recent advances in ambient energy-harvesting technologies have made it possible to power wireless sensor networks (WSNs) from the environment for long durations. However, the energy availability in an energy-harvesting WSN varies with time and thus may cause the network topology to evolve over time. In this paper, we study the topology design problem in a *time-evolving and energy-harvesting WSN* where the time-evolving topology and dynamic energy cost are known a priori or can be predicted. We model such a network as a node-weighted space-time graph which includes both spacial and temporal information. To reduce the cost of supporting time-evolving networks with limited harvesting energy sources, we propose a new *efficient topology design problem* which aims to put more sensors into sleep while still maintaining the network connectivity over time. We prove that the optimization problem of finding the optimal awake sensor set with the minimum total cost is NP-hard. Thus, we propose several topology design algorithms which can significantly reduce the total cost of topology while maintaining the connectivity over time. Simulation results from random time-evolving and energy-harvesting WSNs demonstrate the efficiency of the proposed methods.**

I. Introduction

Wireless sensor networks (WSNs) are commonly powered by batteries. For some applications where the network is expected to operate for long durations, energy consumption becomes a severe bottleneck and the most important issue in the protocol design. Recent advances in ambient energy-harvesting technologies have made it possible to power WSNs from energy generated from the environment [1]–[6]. Various energy sources including light, vibration or heat can be harvested by sensor nodes. Fig. 1(a) illustrates two examples of sensor devices (from [3] and [4]) powered by solar cells.

Even though energy-harvesting technology can power WSNs more perpetually than non-renewable energy sources like batteries, the harvested energy is fundamentally different from battery energy. Usually, it has a limit on the maximum rate at which the energy can be used. Furthermore, the harvested energy availability and supported maximum rate typically vary with time and space. For instance, when harvesting solar energy, the minimum energy output for any solar cell would be near zero at night. Therefore, in energy-harvesting WSNs the energy cost and energy replenishment at each sensor are dynamic over time and space as well. Such temporal and spatial variations of ambient energy sources and consumption impose a great challenge in protocol design for energy-harvesting WSNs.



(a) (b)

Fig. 1. (a) Example sensor nodes in energy-harvesting WSNs from [3] (upper) and [4] (lower). (b) Solar irradiance data at a site in Oak Ridge National Laboratory from July 1 to July 7, 2012. Data is obtained via [7].

The change of energy resources not only affects the communication costs, but also causes network topology to evolve. For example, when a sensor node powered by solar cell temporally runs out of energy or has very low energy at night, it may disappear from the network. Later, when the node is recharged, it reappears. In addition, node mobility may also lead to topological changes. Such dynamic topologies over time domain in energy-harvesting WSNs are often ignored in protocol design or simply modeled by pure randomness.

Fortunately, for certain types of energy-harvesting WSNs, the temporal characteristics of energy resources and dynamic topology could be known a priori or be predicted from historical tracing data. For instance, it is easy to discover temporal patterns of energy resources in a solar energy-harvesting system, since the change of solar irradiance over a place follows regular patterns. Fig. 1(b) plots solar irradiance data at a site in Oak Ridge National Laboratory over seven days (obtained from the Measurement and Instrumentation Data Center at US National Renewable Energy Laboratory [7]). It is obvious that solar irradiance is low during the night and maximized around noon everyday. This implies that it is easy to predict the solar energy resources in an energy-harvesting WSN. Actually, there has been several energy prediction methods [8]–[11] for different energy-harvesting WSNs.

The work of F. Li is partially supported by the National Natural Science Foundation of China under Grant No. 61370192 and 60903151, and the Beijing Natural Science Foundation under Grant No. 4122070. The work of Y. Wang is supported in part by the US National Science Foundation under Grant No. CNS-0915331 and CNS-1050398. Y. Wang (yu.wang@uncc.edu) is the corresponding author.

In this paper, we study the topology design problem for a *Time-evolving and Energy-harvesting Wireless Sensor Network* (TEWSN), by taking *time-varying energy cost and time-domain topological information* into consideration. We assume that the time-evolving topology and dynamic energy cost are known a priori or can be predicted. We first model such a TEWSN as a directed node-weighted space-time graph in which both spacial and temporal information are preserved. We then define the *efficient topology design problem* which aims to build a sparser structure (also a space-time graph) from the original space-time graph by putting a subset of sensors into sleep (removing nodes from the original graph) such that (1) the network is still connected over time and supports routing between any two sensors; (2) the total cost of the structure is minimized. Notice that in energy-harvesting WSNs, it is too expensive to maintain a dense structure and keep every sensor awake for all the time. We formally show that this new topology design problem is NP-hard, by connecting it with an existing topology control problem for delay tolerant networks (DTNs) [13]. We propose five different algorithms to construct new network topologies which can significantly reduce the total cost while maintaining the network connectivity over time. We also discuss how to address the topology design problem under different space-time graph models. Simulation results over random TEWSNs demonstrate the efficiency of our proposed methods.

Topology design has been well studied in ad hoc and sensor networks [12]. Most efforts have been spent on how to construct a power efficient structure from a static and connected topology. Topology design over time-evolving network has not been investigated except for in our recent work [13]–[15], where link-weighted space-time graphs are used to model time-evolving DTNs. The topology design problem over node-weighted space-time graphs is much harder than those in [13]–[15] and more suitable for energy-harvesting WSNs. We believe that this study is the first work to investigate topology design for time-evolving and harvesting WSNs by considering time-varying nature of energy replenishment and dynamic evolution of topology.

The rest of this paper is organized as follows. We first introduce our space-time graph model in Section II, then formally define the efficient topology design problem and prove its NP-hardness in Section III. Five topology design algorithms for TEWSNs are proposed in Section IV. Section V presents the simulation results and Section VI discusses possible variations of space-time graph models. We summarize related work in Section VII and conclude the paper in Section VIII.

II. Node-weighted Space-Time Graphs: Modeling Time-evolving and Energy-Harvesting WSNs

To model the time-evolving and energy-harvesting wireless sensor networks, we adopt the space-time graph model [16], which has been used to model time-evolving DTNs [13]–[15]. In all of these previous work, the space-time graph is link-weighted, however, in our case we use node-weighted version to model the dynamic energy cost at each sensor

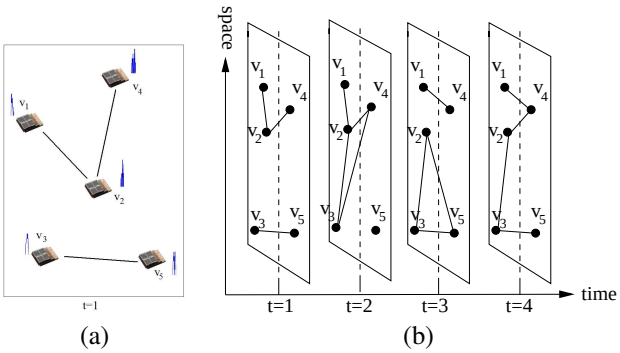


Fig. 2. **A time-evolving and energy-harvesting WSN (TEWSN):** (a) a snapshot of the network at time $t = 1$, (b) the time-evolving topologies of the network over four time slots.

node. Surprisingly, the topology design problem with node-weighted version is much more challenging than those with link-weighted space-time graphs.

We assume that the time is divided into discrete and equal time slots, such as $\{1, \cdots, T\}$. $V = \{v_1, \cdots, v_n\}$ is the set of individual sensors in the network. Fig. 2 illustrates an example of such TEWSNs. Let $G^t = (V^t, E^t)$ be a graph representing the snapshot of the network at time slot t and a link $v_i^t v_j^t \in E^t$ represents that nodes v_i and v_j can communicate with each other at time t. Then, the dynamic network is described by the union of all snapshots $\{G^t | t = 1, \cdots T\}$. For each sensor v_i^t at any time t, we assume that there are two costs $c_{\mathbf{t}}(v_i^t)$ and $c_{\mathbf{r}}(v_i^t)$, which represent the cost to be awake for transmitting packets and the cost to be awake for receiving packets in this time slot, respectively. These costs change with time due to variations of energy replenishment and capture the time-varying and spatial differences properties of energy-harvesting rates.

We then convert the sequence of static graphs $\{G^t\}$ into a node-weighted space-time graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is a directed graph defined in both spacial and temporal spaces. To capture whether a sensor node v_i needs to be awake for transmitting or receiving packets in each time slot t, two nodes $v_i^{t,\mathbf{t}}$ and $v_i^{t,\mathbf{r}}$ are created for v_i in time slot t. The weights of them are $c(v_i^{t,\mathbf{t}}) = c_{\mathbf{t}}(v_i^t)$ and $c(v_i^{t,\mathbf{r}}) = c_{\mathbf{r}}(v_i^t)$. For convenience, we also include two virtual nodes v_i^0 and v_i^{T+1} for sensor v_i as the starting point and ending point of the time span. See Fig. 3 for illustrations. Thus, in the space-time graph $\mathcal{G}$, $2(T+1)$ layers of nodes (2 layers per time slot) are defined and each layer has n nodes. There are $2n(T+1)$ nodes in total. Two kinds of links (spatial links and temporal links) are added between consecutive layers in $\mathcal{G}$. A temporal link $\overrightarrow{v_i^{t,\mathbf{t}} v_i^{t,\mathbf{r}}}$ (a horizontal link inside time slot t) represents buffering packets at the node in the tth time slot[1], while a temporal link $\overrightarrow{v_i^{t,\mathbf{r}} v_i^{t+1,\mathbf{t}}}$ (a horizontal link between time slots t and $t+1$) is a virtual link connecting two consecutive time slots. A spatial link $\overrightarrow{v_i^{t,\mathbf{t}} v_k^{t,\mathbf{r}}}$ (a non-horizontal link inside

[1]Note that under this model the cost for v_i to buffer packets in time slot t is equal to the summation of $c_{\mathbf{t}}(v_i^t)$ and $c_{\mathbf{r}}(v_i^t)$. Later, we will relax such assumption by defining new variations of space-time graphs in Section VI.

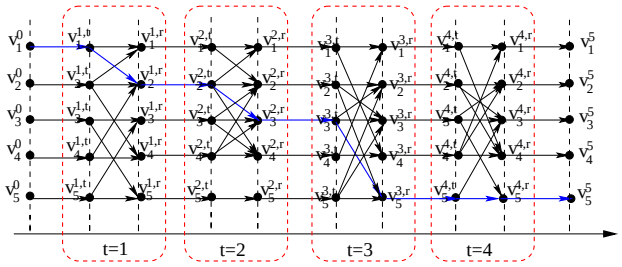


Fig. 3. The corresponding space-time graph $\mathcal{G}$ of the time-evolving sensor network in Fig. 2(b). The blue path shows a space-time route from v_1 to v_5.

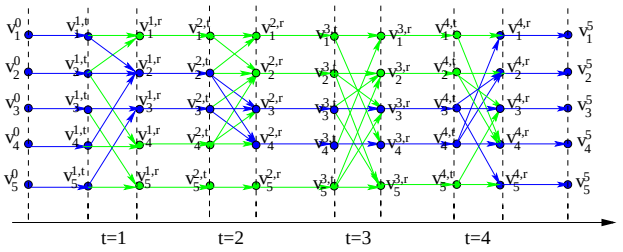


Fig. 4. Efficient topology design on time-evolving and energy-harvesting WSNs (over the one shown in Fig. 3): a new connected subgraph $\mathcal{H}$ of $\mathcal{G}$ where green nodes/links are removed from $\mathcal{G}$).

time slot t) represents forwarding a packet from node v_i to its neighbor v_k in the tth time slot (i.e., $v_iv_k \in E^t$). Notice that the space-time graph defined here is different with those in [13]–[15] for DTNs.

By defining the new space-time graph $\mathcal{G}$, any communication operation in the time-evolving sensor network can be simulated over this directed graph. As highlighted in Fig. 3, the blue space-time path from v_1^0 to v_5^5 shows a particular routing strategy to deliver the packet from v_1 to v_5 in the network using 4 time slots: v_1 sends the packet to v_2 in the first time slot, and v_2 forwards it to v_3 at $t = 2$, then v_3 sends it to v_5 in the third time slot, at last v_5 holds the packet for the last time slot. Notice that in this space-time graph model, only one-hop transmission is allowed within one time slot.

The connectivity of a space-time graph is defined as follows:

Definition 1: A space-time graph $\mathcal{G}$ is *connected* over time period T if and only if there exists at least one directed path between each pair of nodes (v_i^0, v_j^{T+1}) (i and j in $[1, n]$). This guarantees that the packet can be delivered between any two nodes in the time-evolving network over the period of T. Hereafter, we assume that the original space-time graph $\mathcal{G}$ is always connected. In other words, without putting any sensor into sleep, the evolving sensor network is connected over time. Notice that the connectivity of a space-time graph is different with the connectivity of a static graph. A connected space-time graph does not require connectivity in each snapshot.

Given the costs of each sensor node in $\mathcal{G}$, we can also define the total cost of a space-time graph as the summation of costs of all nodes in $\mathcal{G}$, i.e., $c(\mathcal{G}) = \sum_{v \in \mathcal{G}} c(v)$. Similarly, we can define the total cost of a space-time path P as the summation of costs of all nodes in path P (except for the endpoint nodes). In $\mathcal{G}$, the path connecting nodes u and v with the minimum cost is defined as the least cost path $P_{\mathcal{G}}(u, v)$. When the underlying space-time graph is clear, we drop $\mathcal{G}$ from the symbol.

III. Efficient Topology Design Problem in TEWSNs

We now define the *efficient topology design problem* (ETDP) on node-weighted space-time graphs.

Definition 2: Given a connected and node-weighted space-time graph $\mathcal{G}$, the aim of *efficient topology design problem* (ETDP) is to construct a sparse space-time graph $\mathcal{H}$, which is a subgraph of $\mathcal{G}$, such that $\mathcal{H}$ is still connected over the time period T and the total cost of $\mathcal{H}$ is minimized.

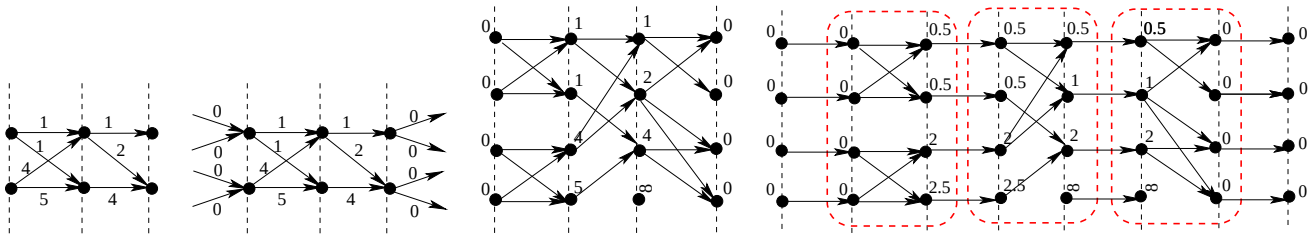
The motivation of ETDP is to keep a few sensors awake while guarantee the overall connectivity over certain time period. In many energy-harvesting sensor networks, it is too expensive to maintain a dense structure over time. Our ETDP focuses on finding cost-efficient and sparse space-time subgraph to be active as the underlying topology for the time-evolving sensor network. Fig. 4 shows a possible solution of the ETDP, where several sensor nodes are removed from the original space-time graph but the connectivity over time is preserved.

The newly defined topology design problem is different from the standard space-time routing [16], [17], which aims to find the most cost-efficient space-time path for a pair of source and destination. The ETDP aims to maintain a cost-efficient and connected space-time graph for all pairs of nodes. The paths inside the constructed graph are not the least cost paths for routing. Therefore, our goal is not to optimize the routing performance but to improve the cost efficiency of the topology.

The topology design problem for a node-weighted space-time graph is also quite different from the same problem for a node-weighted static graph. For a static graph without time domain, a spanning tree can achieve the goal of keeping connectivity and all spanning trees have the same total cost with the original graph. In a space-time graph, nodes in a single snapshot may not be connected at all, thus the spanning tree is not useful. Even assume that all nodes are connected in each snapshot, a spanning tree over the whole space-time graph is not a solution either, since it spans all nodes and does not save any cost comparing with the original graph.

In [13]–[15], we have studied the topology design problem for a link-weighted space-time graph. However, their solutions cannot be directly used in the node-weighted version, since (1) removing a node in the space-time graph causes multiple links to be removed from the graph; (2) node costs are not easy to converted or splitted to link costs; and (3) the construction methods from a sequence of static graphs to the space-time graph are different. In fact, like the directed Steiner tree problem [18], [19], the node-weighted version of topology design problem is much harder than the link-weighted version.

We now prove the NP-hardness of our newly defined topology design problem ETDP by a reduction from a previous topology control problem over link-weighted space-time graphs (TCP) [13], which has been proved an NP-hard



(a) $\mathcal{G}'$ (b) $\mathcal{G}''$ (c) $\mathcal{G}'''$ (d) $\mathcal{G}$

Fig. 5. Illustrations for NP-hardness proof of ETDP: (a) a link-weighted space-time graph $\mathcal{G}'$ in TCP; (b) a link-weighted space-time graph $\mathcal{G}''$ with two sets of free virtual links added in $\mathcal{G}'$; (c) a node-weighted space-time graph $\mathcal{G}'''$ converted from $\mathcal{G}''$; and (d) a node-weighted space-time graph $\mathcal{G}$ converted from $\mathcal{G}'''$ by splitting each node into two nodes.

problem. The TCP problem is defined as follows: given a directed and link-weighted space-time graph $\mathcal{G}$, find a sparser structure $\mathcal{H}$ from the original space-time graph $\mathcal{G}$ such that (1) the network is still connected over time; (2) the total link cost of the structure is minimized.

Theorem 1: The *efficient topology design problem* (ETDP) in a time-evolving and energy-harvesting WSN modeled by a node-weighted space-time graph is NP-hard.

Proof: We first show how to reduce the TCP problem into our ETDP problem. Given an instance of TCP with a link-weighted space-time graph $\mathcal{G}'$ where every link has a cost, as shown in Fig. 5(a), we can construct an instance of ETDP on a new node-weighted space-time graph $\mathcal{G}$ as follows.

Assume that the largest number of links in each time slot in $\mathcal{G}'$ is x. We add two sets of x virtual links at the beginning and the ending of the space-time graph as shown in Fig. 5(b) and assign 0 cost for them. Every node in the starting and ending time slots needs to guarantee having at least one adjacent virtual link. We use $\mathcal{G}''$ to denote the new link-weighted graph.

We then construct a new space-time graph $\mathcal{G}'''$ from $\mathcal{G}''$ by mapping all links in $\mathcal{G}''$ into nodes in $\mathcal{G}'''$. Naturally, the costs of links become costs of nodes. If two links e_1 and e_2 are adjacent in $\mathcal{G}''$ (i.e., share a node), there will be a link in $\mathcal{G}'''$ connecting two nodes who represent e_1 and e_2. See Fig. 5(c). Notice that to make the number of nodes in each time slot in $\mathcal{G}'''$ the same (x), some virtual nodes may be needed. After this step, we now have a node-weighted space-time graph.

The last step is to make the node-weighted space-time graph $\mathcal{G}'''$ into the format of the one we defined in our ETDP problem. A easy node splitting can achieve such goal. Each node in $\mathcal{G}'''$ is split into two nodes with half of the original cost. See Fig. 5(d) for illustrations. Notice that there are a few horizontal links missing. It is easy to add an additional layer with infinite node cost to solve the problem. Due to space limit, we ignore such construction step here.

By this overall construction, it is easy to find a solution of ETDP with the same cost in $\mathcal{G}$ for any solution of TCP in $\mathcal{G}'$, and vice versa. Since the construction of $\mathcal{G}$ can be done in polynomial time and TCP problem is NP-hard, the ETDP on node-weighted space-time graphs is also NP-hard. ■

Notice that the reduction above can only work from TCP to ETDP, but not in the reverse direction. Therefore, ETDP is computationally much harder than TCP.

Algorithm 1 Greedy Algorithm to Add Nodes (GrdAN)

Input: original space-time graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$.
Output: new sparse space-time graph $\mathcal{H}$.

```
Add nodes v_i^0 and v_i^{T+1} into H for all integers 1 ≤ i ≤ n.
Sort all remaining nodes in V based on their costs.
for all v_i^t ∈ V (processed in increasing order of costs) do
    if H is connected then
        return H
    else
        Add v_i^t into H; add all edges between v_i^t and other
        nodes already in H into H if such edges exist in G.
```

Algorithm 2 Greedy Algorithm to Delete Nodes (GrdDN)

Input: original space-time graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$.
Output: new sparse space-time graph $\mathcal{H}$.

```
H = G.
Sort all remaining nodes in V based on their costs.
for all v_i^t ∈ V (processed in decreasing order of costs) do
    if H is connected then
        Remove v_i^t and all its adjacent edges from H.
    else
        return H
```

IV. Topology Design Algorithms for TEWSNs

Since ETDP over node-weighted space-time graphs is NP-hard, in this section, we propose five different heuristics to construct a sparse structure that fulfills the connectivity requirement over a node-weighted space-time graph. Within this section, we use $\tilde{n}$ and $\tilde{m}$ to denote the number of nodes and links in the original space-time graph $\mathcal{G}$, respectively. Notice that $\tilde{n} = O(nT)$ while $\tilde{m} = O(n^2T)$.

A. Simple Greedy Heuristics: Adding or Removing Nodes

The first two heuristics share the same principle: greedily adding or removing single node to/from the graph until the connectivity requirement is achieved or broken. The only difference between them is the processing order of nodes.

The first algorithm starts with a graph only including nodes in the first and last layers (nodes v_i^0 and v_i^{T+1} for $1 \le i \le n$). Then it greedily adds in more nodes until the connectivity of

$\mathcal{G}$ is achieved. During the process, it selects the node with smaller cost first. See Algorithm 1 for detail. The second algorithm starts with the original space-time graph $\mathcal{G}$ and gradually deletes the node with the largest cost if it does not break the connectivity of the graph. Algorithm 2 shows the detail. Hereafter, we denote these two methods as GrdAN and GrdDN, respectively. Both GrdAN and GrdDN can obviously satisfy the connectivity requirement of $\mathcal{H}$. The time complexity of either GrdAN or GrdDN is $O(n\tilde{n}(\tilde{m}+\tilde{n}\log\tilde{n}))$ since there are at most $\tilde{n}$ rounds of connectivity checks.

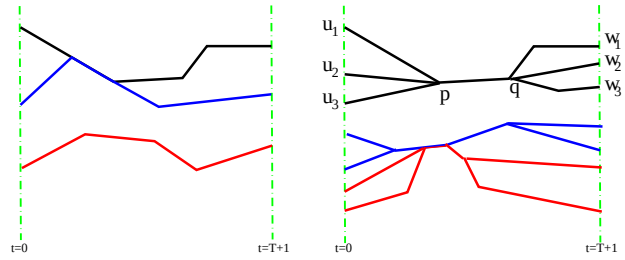
There are a few possible variations of GrdAN and/or GrdDN. For example, instead of sorting nodes based on node costs, both can use node degree as the criterion. For example, in GrdAN, selecting the node with larger node degree first could accelerate the process to meet the connectivity requirement. In addition, GrdDN can start deleting nodes from a sparse connected space-time graph instead of the original graph $\mathcal{G}$. Thus, any output from our proposed algorithms (including those introduced later) can be used as the input of GrdDN to save computation, since they are sparser than $\mathcal{G}$.

B. Greedy Algorithms: Adding Paths or Bunches

While GrdAN adds one node to $\mathcal{H}$ in each round to make it connected, the next two greedy algorithms add an entire path or a group of paths (called a bunch) to $\mathcal{H}$ in each round.

One naive method for maintaining the network connectivity is keeping all least cost paths from v_i^0 to v_j^{T+1} for $i,j = 1,\cdots,n$. However, the output of such method may contain more links than necessary. Hereafter, we use a set X to represent the set of all pairs of (v_i^0, v_j^{T+1}) for all integers $1 \le i,j \le n$. The third algorithm is still based on the union of all least cost paths, but it clears the cost of used nodes in previous rounds so that they are free for reuse in later rounds. Recall that we need to connect n^2 pairs of nodes in X. As shown in Fig. 6(a), in each round the algorithm picks the least cost path between a pair of nodes in X which is the minimum among all least cost paths connecting any pair of nodes in X. Then it adds all nodes and links along this path into $\mathcal{H}$, clears the costs of these nodes to zeros, and removes this pair from X. This procedure is repeated as shown in Fig. 6(a). After n^2 rounds, all pairs of nodes in X are guaranteed to be connected by paths in $\mathcal{H}$. It is obvious that the output of this method is much sparser than the union of all least cost paths. Algorithm 3 gives the detailed algorithm. We refer to this method as *greedy method based on least cost path* (GrdLCP). The time complexity of GrdLCP is $O(n^3(\tilde{m}+\tilde{n}\log\tilde{n}))$ since in each round n times of Dijkstra's algorithm are running on the space-time graph and there are n^2 rounds.

Next, we present a more complex greedy algorithm (as shown in Algorithm 4) which is inspired by a method proposed by Charikar and Chekuri [19] for *directed generalized Steiner network* (DGSN) problem [20]. The DGSN problem is also a NP-hard problem and defined as follows. Given a directed link-weighted graph G and a set of $X = \{(a_i, b_i)\}$ of k node pairs, find the minimum cost subgraph H of G such that for each node pair $(a_i, b_i) \in X$, there exists a directed path from a_i to b_i in H. Since the space-time graph $\mathcal{G}$ is a directed graph, our topology design problem is a special case of a node-weighted version of DGSN problem with $X = \{(v_i^0, v_j^{T+1})\}$ for all $i,j \in [1,n]$. In this case, the number of node pairs is $k = n^2$. For the DGSN problem, the current best approximation guarantee is $O(k^{1/2+\epsilon})$ by [20]. However, it is too complex to be practical. Our fourth algorithm keeps finding a group of paths to connect several



(a) GrdLCP (b) GrdLDB

Fig. 6. Illustrations of GrdLCP (Algorithm 3) and GrdLDB (Algorithm 4): (a) GrdLCP repeatly adds one least cost path into the topology to connect one pair of nodes in X. (b) GrdLDB repeatedly adds one bunch with least density into the topology to connect multiple pairs of nodes in X. Both algorithms terminate when all pairs of nodes in X are connected.

Algorithm 3 Greedy Algorithm based on Least Cost Path

Input: original space-time graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$.
Output: new sparse space-time graph $\mathcal{H}$.

$\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^{T+1})\}$ for all i and j in $[1,n]$.
while $X \neq \phi$ **do**
  Find the least cost path for every pair nodes in X, and assume path $P_{\mathcal{G}}(v_i^0, v_j^{T+1})$ has the least cost among these paths.
  Add all nodes and links in $P_{\mathcal{G}}(v_i^0, v_j^{T+1})$ to $\mathcal{H}$.
  Set the costs of all nodes in $P_{\mathcal{G}}(v_i^0, v_j^{T+1})$ in $\mathcal{G}$ to zeros.
  $X \leftarrow X - (v_i^0, v_j^{T+1})$.
return $\mathcal{H}$

Algorithm 4 Greedy Algorithm based on Least Density Bunch

Input: original space-time graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$.
Output: new sparse space-time graph $\mathcal{H}$.

$\mathcal{H} \leftarrow \phi$; $k = n^2$; $X = \{(v_i^0, v_j^T)\}$ for all $1 \le i,j \le n$.
while $X \neq \phi$ **do**
  $d \leftarrow \infty$; $B1 \leftarrow \phi$.
  for all pairs $(p,q) \in \mathcal{V} \times \mathcal{V}$ **do**
    for all pairs $(v_i^0, v_j^T) \in X$ **do**
      $s[v_i^0, v_j^T] \leftarrow c(v_i^0, p) + c(q, v_j^T)$.
    Sort all $s[v_i^0, v_j^T]$ in increasing order of s, and let (u_l, w_l) refer to the lth pair in this sorted list.
    for l going from 1 to k **do**
      Let B be the bunch connecting first l node-pairs.
      if $d(B) \le d$ **then**
        $d \leftarrow d(B)$; $k1 = l$; $B1 \leftarrow B$.
  $\mathcal{H} \leftarrow \mathcal{H} + B1$; $k = k - k1$;
  $X \leftarrow X - \{(u_1, w_1), \cdots, (u_{k1}, w_{k1})\}$.
return $\mathcal{H}$

pairs of nodes in X. Each group of paths is defined as a structure, called a "bunch", where these paths share a portion of paths (i.e., have the same subpath formed by a single or multiple links). Assume a bunch B connects l pairs of nodes in X (assume the node pairs are (u_i, w_i) for $i = 1, \cdots, l$) and the l paths from u_i to w_i share a portion from node p to q. See Fig. 6(b) for illustrations. Recall that $P_{\mathcal{G}}(u, v)$ is the least cost path between u and v. We use $c(u, v)$ to represent its cost, i.e., $c(u, v) = c(P_{\mathcal{G}}(u, v))$.Let $s[u_i, w_i]$ represent the cost of $P_{\mathcal{G}}(u_i, p)$ plus the cost of $P_{\mathcal{G}}(q, w_i)$, i.e., $s[u_i, w_i] = c(u_i, p) + c(q, w_i)$. Then, the total cost of bunch B which connects l-pair of nodes in X is $c(B) = c(p) + c(q) + c(p, q) + s[u_1, w_1] + s[u_2, w_2] + \cdots + s[u_l, w_l]$. We then can define the density of this bunch as $d(B) = c(B)/l$, which implies how much cost per connection is used to connect l pairs of nodes. The fourth greedy algorithm considers all possible bunches and greedily selects the bunch with the smallest density in each round. After a bunch is selected, all nodes and links in the bunch are added to the subgraph $\mathcal{H}$ and X is also updated accordingly. The algorithm terminates until all n^2 pairs of nodes are connected by bunches. The output is the union of selected bunches. Fig. 6(b) shows the procedure. We refer to this method as *greedy method based on least density bunch* (GrdLDB). GrdLDB's time complexity is $O(n^4 \tilde{n}^2 \log n)$, since the outer while-loop runs n^2 times in the worst case; the outer for-loop runs $O(\tilde{n}^2)$ times; and the sorting can be done in $O(n^2 \log n)$. Notice that an all-to-all shortest path algorithm needs to be performed once to prepare $c(u, v)$ for all nodes u and v in the beginning of the algorithm. Although the overall time complexity is high, this algorithm can achieve approximation-guarantee in term of the total cost compared with the optimal solution. In [19], Charikar et al. proved that the greedy algorithm based on bunch selection can give an approximation ratio of $O(k^{2/3} log^{1/3} k)$ for the DGSN problem. Therefore, we have the following theorem for our topology design problem, since $k = n^2$.

Theorem 2: Algorithm 4 (GrdLDB) gives an approximation ratio of $O(n^{4/3} \log^{1/3} n)$ for the efficiency topology design problem in node-weighted space-time graph.

Similar to GrdLCP, GrdLDB can be modified to the following version. In each round, reset the cost of nodes to zero once they are added to the current solution. Nonetheless, doing so will lose the approximation ratio (in Theorem 2).

C. Simple Heuristic: Search over Least Cost Path Trees

Last, we give a simple heuristic based on the least cost path tree. The basic idea is quite simple and as follows. For a node v_i^t in $\mathcal{G}$, we construct two least cost path trees rooted at it and reaching all nodes at the beginning and ending of the space-time graph: one least cost path tree to reach every node v_i^{T+1} and one least cost path tree to reach all nodes v_i^0 (all directed links need to be reversed). If both trees can be founded, the union of them can be a possible solution for the topology design problem. Our algorithm tries all intermediate nodes v_i^t in $\mathcal{G}$, and chooses the one with the minimum cost as the output. See Algorithm 5 for the detail. We call this algorithm

Algorithm 5 Search Algorithm over Least Cost Path Trees

Input: original space-time graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$.
Output: new sparse space-time graph $\mathcal{H}$.

```
Calculate all-to-all least cost paths in G.
H ← φ.
for all v_i^t ∈ G, 1 ≤ t ≤ T do
    Let LCPT_1 be the least cost path tree rooted at v_i^t and reaching v_j^0, 1 ≤ j ≤ n.
    Let LCPT_2 be the least cost path tree rooted at v_i^t and reaching v_j^{T+1}, 1 ≤ j ≤ n.
    Let LCPTs = LCPT_1 ∪ LCPT_2 and c(LCPTs) be the cost of LCPTs.
    if c(H) > c(LCPTs) then
        H = LCPTs.
return H
```

search algorithm over least cost path trees (LCPT). Clearly, it is possible that LCPT cannot find any solution, e.g., in the example shown in Fig. 4. However, our simulation results show nice performances of this algorithm especially when the graph is dense. Since an all-to-all shortest path algorithm needs to be performed only once to prepare all least cost path trees, the time complexity of LCPT is $O(\tilde{n}\tilde{m} + \tilde{n}^2 \log \tilde{n})$. Hence, this algorithm is very practical.

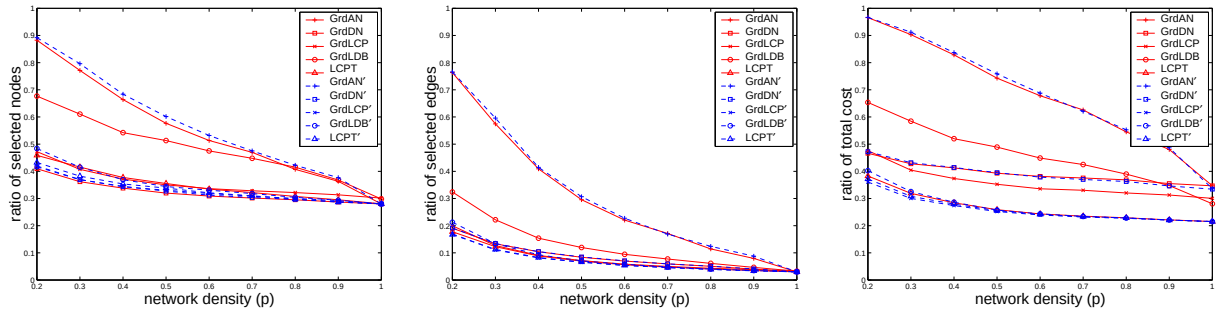
D. Refining Node Cost with Node Degree

Note that in all of algorithms above, we use the node cost as the metric to select nodes or paths to add. The intuitive behind it is trying to use nodes with less cost in the constructed $\mathcal{H}$. However, adding nodes with less cost may not improve the connectivity significantly. On the other hand, adding a node with highest node degree in $\mathcal{G}$ may significantly improve the connectivity of $\mathcal{H}$. Based on this observation, we can refine the node cost $c(v)$ of v using its node degree $d(v)$ in $\mathcal{G}$. The new cost $c'(v) = c(v)/d(v)$, which implies the cost per node degree. By this simple modification, we can have a set of new algorithms. For any algorithm Y, we use Y' to denote the new version with the refined node cost.

V. Simulations

We evaluate our proposed topology design algorithms, namely, GrdAN, GrdDN, GrdLCP, GrdLDB, and LCPT, over random time-evolving and energy-harvesting WSNs. We implement all these algorithms (each with two versions: one uses cost $c(v)$ and the other uses refined cost $c'(v)$) in a simulator developed by our group. In all simulations, we take four metrics as the performance measurements for any topology design algorithm:

- **Total Number of Selected Nodes:** the total number of nodes in the constructed $\mathcal{H}$, denoted by $n(\mathcal{H})$.
- **Total Number of Selected Edges:** the total number of edges in the constructed $\mathcal{H}$, denoted by $m(\mathcal{H})$.
- **Total Cost:** the total cost of the constructed topology $\mathcal{H}$ (output of the algorithm), i.e., $c(\mathcal{H}) = \sum_{v \in \mathcal{H}} c(v)$.



(a) $n(\mathcal{H})/n(\mathcal{G})$ (b) $m(\mathcal{H})/m(\mathcal{G})$ (c) $c(\mathcal{H})/c(\mathcal{G})$

Fig. 7. Simulation results on random networks ($n = 10$ and $T = 10$) with different densities. The number of nodes/edges and total cost of $\mathcal{H}$ are divided by those of $\mathcal{G}$, which illustrates how much saving is achieved by the proposed algorithms, compared with the original network without topology design.

- **Running Time:** the total time to generate the output topology $\mathcal{H}$.

For all the simulations, we repeat the experiment for multiple times and report the average values of these metrics. It is clear that a desired topology should have small total cost, small edge number, and small node number.

Generating Random Space-Time Graphs: To simulate random TEWSNs, we first generate a sequence of static random graphs G^t to denote the time-evolving network. We consider a network with n sensors and spreading over T time slots. For each time slot t, we randomly generate the graph G^t using the classical random graph generator. Basically, for each pair of nodes v_i, v_j, we insert the edge of v_iv_j with a fixed probability p. This probability can control the density of the network. The larger value of p is, the denser is the network. $p = 1.0$ implies that the topology in each time slot is a complete graph. For each node, we randomly pick its costs from 1 to 100 for each time slot. After generating $\{G^t\}$, we then convert it into its corresponding space-time graph $\mathcal{G}$ with $2n(T + 1)$ nodes. All topology design algorithms take the same $\mathcal{G}$ as the input.

Simulation Results: We first test our algorithms over a set of 10-node 10-time-slot time-evolving networks (i.e., $n = 10$ and $T = 10$). We vary the network density parameter p from 0.2 to 1.0 and generate 100 time-evolving networks for each case. Fig. 7 and Fig. 8 show the results.

Figs. 7 (a)-(c) show the ratio between the number of selected nodes/edges or total cost of the generated graph $\mathcal{H}$ and that of the original graph $\mathcal{G}$ when p increases. This ratio implies how much saving is achieved by the topology design algorithm, compared with the original network without topology design. From the results, all proposed algorithms can significantly reduce the cost of maintaining the connectivity over time. Even with the least density ($p = 0.2$), most of algorithms (except for GrdAN, GrdAN', and GrdLDB) can save more than 50% cost, 50% nodes and around 60% edges. For $p = 1.0$, more than 60% cost, around 70% nodes and 90% edges are saved by all algorithms. With increasing density of the network, the ratios of used cost/node/edge decrease. This indicates that more saving can be achieved by all algorithms with dense networks. For all of the results, the algorithms with refined cost (using the node degree) usually have better performance than those with original node cost. This shows that the refinement is effective, especially for the total cost measurement. The improvement of such refinement is significant for GrdLDB. For the number of selected nodes/edges, both versions of LCPT, GrdDN, and GrdLCP plus GrdLDB' have nice and similar performances. However, in term of total cost, GrdLCP', GrdLDB' and LCPT/LCPT' have the best performances. Notice that GrdDN/GrdDN' can delete many nodes and links, but not save a lot of costs. In addition, GrdAN performs poorly over all measurements, since it adds too many nodes/edges until it achieves the connectivity. Finally, even though GrdLDB has the theoretical approximation bound, GrdLCP and LCPT perform much better in practice.

Fig. 8 shows the average running time of each algorithm. It is clear that only GrdDN/GrdDN' needs a lot of time with a denser space-time graph. Other methods are kind of stable. Compared with LCPT/LCPT' and GrdLCP/GrdLCP', GrdLDB/GrdLDB' uses more time, which is consistent with our theoretical analysis of time complexity. Obviously, LCPT/LCPT' and GrdLCP/GrdLCP' are nice choices for both running time and performances. However, recall that LCPT/LCPT' may find no solution for certain networks.

We also perform simulations on a set of random networks with larger size and longer time period to test the scalability of our algorithms. Due to space limit, we do not include the detailed results here. The results and conclusions are consistent with the previous set of simulations. With larger networks, all algorithms can save more costs while spend more time.

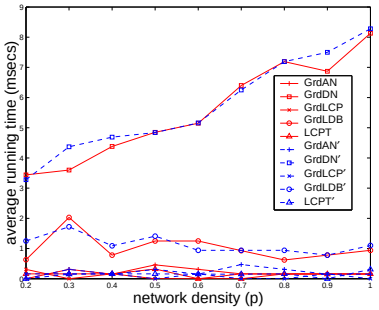


Fig. 8. Average running time of each algorithm on the random networks.

VI. Variations on Space-Time Graph Model

Notice that in our node-weight space-time graph model (presented in Section II) we assume that the cost of buffering packets at time slot t is equal to the summation of $c_{\mathbf{t}}(v_i^t)$ and $c_{\mathbf{r}}(v_i^t)$. This may not be true in real sensor networks. In this section, we discuss three possible relaxations on this model. In all of them, directed links from $v_i^{t,\mathbf{t}}$ to $v_i^{t,\mathbf{r}}$ are removed.

Fig. 9(a) illustrates the first modified model by adding new horizontal links from $v_i^{t,\mathbf{t}}$ to $v_i^{t+1,\mathbf{t}}$ and from $v_i^{t,\mathbf{r}}$ to $v_i^{t+1,\mathbf{r}}$ for each node v_i at any time slot t. In the figure, we use dash lines to represent these new links. Note that we do not include all of them to avoid messing the figure. Instead, we only include those for v_1 and v_5. By adding these links, node v_i can buffer the packet for one time slot with cost of either $c_{\mathbf{t}}(v_i^t)$ or $c_{\mathbf{r}}(v_i^t)$. In other words, the buffering cost is $min(c_{\mathbf{t}}(v_i^t), c_{\mathbf{r}}(v_i^t))$.

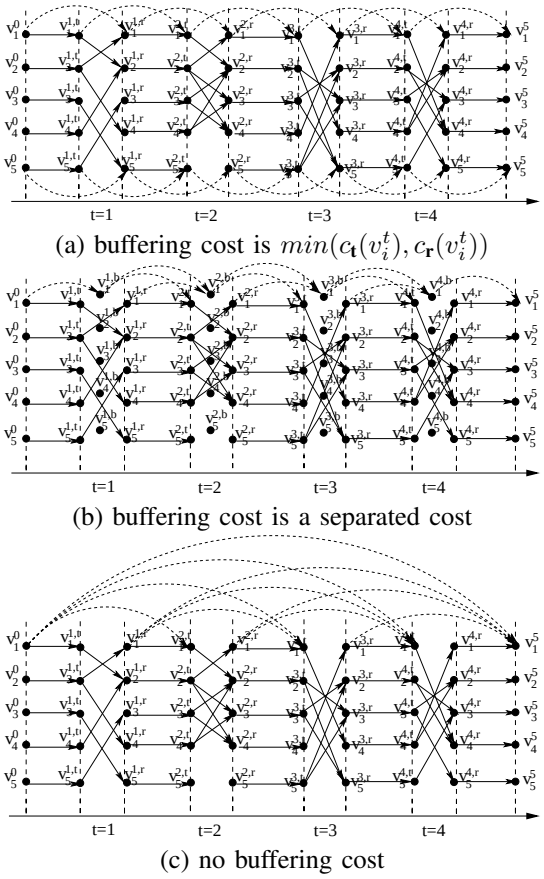
However, in some cases, the buffering cost could be much smaller than either $c_{\mathbf{t}}(v_i^t)$ or $c_{\mathbf{r}}(v_i^t)$. In such cases, each sensor can have a separated buffering cost as $c_{\mathbf{b}}(v_i^t)$ which is different from either $c_{\mathbf{t}}(v_i^t)$ or $c_{\mathbf{r}}(v_i^t)$. Then a new space-time graph can be defined as shown in Fig. 9(b). Now three sets of nodes are in each time slot representing sensors awake for transiting, receiving and buffering packets, respectively. For each new node $v_i^{t,\mathbf{b}}$, four horizontal links are added: $\overrightarrow{v_i^{t-1,\mathbf{t}} v_i^{t,\mathbf{b}}}$, $\overrightarrow{v_i^{t-1,\mathbf{b}} v_i^{t,\mathbf{b}}}$, $\overrightarrow{v_i^{t,\mathbf{b}} v_i^{t+1,\mathbf{t}}}$, and $\overrightarrow{v_i^{t,\mathbf{b}} v_i^{t+1,\mathbf{b}}}$. Similarly, we only draw new links of v_1 in the figure for the clarity.

In all models above, a sensor node needs to be awake and cost energy to buffer packets. However, in some cases, sleeping sensor can still buffer packets without costing any energy (or at least ignorable). Therefore, we also give a model where buffering packets does not have any cost. See Fig. 9(c) for illustrations. In this model, whenever a node receives a packet, it can buffer it freely for all the remaining time periods. This is implemented by adding a set of new virtual links. Fig. 9(c) shows those links for node v_1.

For all three variations, our proposed algorithms can still be used to build the sparse space-time structures to maintain the connectivity over time.

VII. Related Work

Modeling Time-Evolving Networks: How to model time-evolving networks has been studied in both mobile ad hoc networks (MANETs) [16], [17], [25] and DTNs [26], [27]. Xuan *et al.* [17] first study routing problem in a fixed schedule dynamic network and use an evolving graph (an indexed sequence of static subgraphs of a given graph) to capture the evolving characteristic of such a dynamic network. [25], [27] also use evolving graphs to evaluate various ad hoc and DTN routing protocols. Shashidhar *et al.* [16] study the routing problem in dynamic networks modeled by space-time graphs. Liu and Wu [26] model a cyclic mobispace as a probabilistic space-time graph in which an edge between two nodes contains a set of probabilistic contacts. However, all of these previous work only focus on the routing issue in dynamic networks.



(a) buffering cost is $min(c_{\mathbf{t}}(v_i^t), c_{\mathbf{r}}(v_i^t))$

(b) buffering cost is a separated cost

(c) no buffering cost

Fig. 9. Other space-time graph models for the time-evolving and energy-harvesting WSNs: the corresponding new space-time graphs $\mathcal{G}$ of the network in Fig. 2(b) under different models.

Topology Control in Wireless Sensor Networks: Topology control has drawn a significant amount of research interests in MANETs and WSNs [12]. Primary topology control algorithms aim to maintain network connectivity and conserve energy. Most of them can be classified into two categories: *geometrical structure-based* [21], [22] and *clustering-based* [23], [24]. These topology control protocols deal with topology changes by re-performing the construction algorithm. However, they all assume that the underlying communication graph is fully connected at any moment and they do not consider the time domain knowledge of network evolution.

Topology Design for Time-Evolving Networks: Topology design over time-evolving networks has been studied in our recent work [13]–[15], where a link-weighted space-time graph is used to model the time-evolving DTN. Multiple heuristics have been proposed to build sparse topologies over time which can maintain the network connectivity with possible additional properties (such as satisfying spanner or reliability requirements). In this paper, we focus on topology design for a node-weighted space-time graph, which is a much harder optimization problem than the one with link-weighted. Removing a single node in the space-time graph may cause multiple links to be removed. Previous solutions for link-weighted problem cannot be directly used in this new problem.

Sleep Scheduling in Wireless Sensor Networks: Various sleep/wakeup scheduling schemes [28]–[30] have been proposed to save energy by employing scheduled duty cycles in WSNs. E.g., Lu *et al.* [28] propose several techniques for minimizing communication latency while providing energy-efficient periodic sleep cycles for nodes in WSNs. Keshavarzian *et al.* [29] introduce a multi-parent forwarding technique and propose a heuristic algorithm for assigning parents to the nodes in the network. Recently, there are also several studies on sleep scheduling for low-duty-cycle wireless sensor networks [31]–[33]. Most of these studies aim to design new data forwarding methods to optimize data delivery, end-to-end delay, or energy consumptions. Usually, they assume static networks with constant and uniform energy resources. In this paper, we focus on the overall topology design over time-evolving and energy-harvesting networks where network topology could change over time and energy resources are dynamic and nonuniform.

VIII. Conclusion

Harvesting energy from the ambient environment is a promising approach to solve the energy problem in WSNs. However, the energy availability and cost in energy-harvesting WSNs vary across time and space, thus may cause the network topology to evolve over time. In this paper, we study the new topology design problem in time-evolving and energy-harvesting WSNs, modeled by node-weighted space-time graphs. We first prove that this problem is NP-hard, and then propose several algorithms to reduce the cost of topology while maintaining the connectivity over time. Simulation results from random networks demonstrate the efficiency of our methods. We believe that this paper presents the first step in exploiting topology design problem for TEWSNs.

The topology design problem studied here has certain limitations. (1) In our ETDP problem, only connectivity between nodes in first time slot and the last time slot is considered. If a packet arrives in the middle of T, it may not be able to reach the destination at the end of T via the constructed topology. However, in most TEWSNs (such as the ones shown in Fig. 1 and those in [4]–[6], [26], [34]), the energy patterns and topology evolution are periodic. Thus, the delivery of packets is guaranteed in such cases. (2) Our model assumes that all communication links are *reliable*. However, this may not be true due to lossy nature of wireless channels. One possible way to relax this assumption is introducing a probability for each link to reflect its "reliability". Then a new topology design problem can be defined by adding reliable constraint over the topology, which is more complex and challenging. We have obtained some preliminary results [15] in non-energy-harvesting networks and leave the complete study of such a problem as our future work. (3) We may also consider the effect of interferences among multiple transmissions within the same time slot, which is ignored in this study. (4) We will perform experiments over real testbeds of TEWSNs and evaluate the effects of topology design over their performances.

References

[1] K. Lin, J. Hsu, S. Zahedi, et al., "Heliomote: Enabling long-lived sensor networks through solar energy harvesting," in *ACM Sensys*, 2005.

[2] C. Park and P. Chou, "Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes," in *IEEE SECON*, 2006.

[3] G. Chen, M. Fojtik, et al., "A millimeter-scale nearly-perpetual sensor system with stacked battery and solar cells," in *IEEE ISSCC*, 2010.

[4] A. Kansal, et al., "Power management in energy harvesting sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 6, no. 4, 2007.

[5] C. Vigorito, D. Ganesan, et al., "Adaptive control of duty-cycling in energy-harvesting wireless sensor networks," in *IEEE SECON*, 2007.

[6] A. Kansal, et al., "Harvesting aware power management for sensor networks," in *ACM/IEEE Design Automation Conference*, 2006.

[7] The Measurement and Instrumentation Data Center (MIDC), http://www.nrel.gov/midc/ornl_rsr/.

[8] J. Piorno, C. Bergonzini, D. Atienza, et al., "Prediction and management in energy harvested wireless sensor nodes," in *Wireless VITAE*, 2009.

[9] C. Bergonzini, et al., "Algorithms for harvested energy prediction in batteryless wireless sensor networks," in *IEEE IWASI*, 2009.

[10] J. Lu, et al., "Accurate modeling and prediction of energy availability in energy harvesting real-time embedded systems," in *IEEE IGCC*, 2010.

[11] A. Cammarano, et al., "Pro-energy: A novel energy prediction model for solar and wind energy harvesting WSNs," in *IEEE MASS*, 2012

[12] Y. Wang, "Topology control for wireless sensor networks," Book chapter in "Wireless Sensor Networks and Applications", edited by Y. Li, M. Thai and W. Wu, Springer, 2007.

[13] M. Huang, S. Chen, Y. Zhu, B. Xu, et al., "Topology control for time-evolving and predictable delay-tolerant networks," in *IEEE MASS*, 2011.

[14] M. Huang, S. Chen, et al., "Cost-efficient topology design problem in time-evolving delay-tolerant networks," in *IEEE Globecom*, 2010.

[15] M. Huang, S. Chen, F. Li, et al., "Topology design in time-evolving delay-tolerant networks with unreliable links," in *IEEE Globecom*, 2012.

[16] S. Merugu, M. Ammar, et al., "Routing in space and time in networks with predictable mobility," GaTech, Tech. Rep. GIT-CC-04-07, 2004.

[17] B. Xuan, A. Ferreira, and A. Jarry, "Computing shortest, fastest, and foremost journeys in dynamic networks," *J. of Foundations of Computer Science*, vol. 14, no. 2, pp. 267–285, 2003.

[18] P. Klein and R. Ravi, "A nearly best-possible approximation algorithm for node-weighted Steiner trees," *J. Algorithms*, 19(1): 104–115, 1995.

[19] M. Charikar and C. Chekuri, "Approximation algorithms for directed Steiner problems," *J. Algorithms*, vol. 33, no. 1, pp. 73–91, 1999.

[20] C. Chekuri, et al., "Set connectivity problems in undirected graphs and the directed steiner network problem," in *ACM-SIAM SODA*, 2008

[21] N. Li, J. C. Hou, and L. Sha, "Design and analysis of a MST-based topology control algorithm," in *IEEE INFOCOM*, 2003.

[22] Y. Wang and X.-Y. Li, "Localized construction of bounded degree and planar spanner for wireless ad hoc networks," *Mobile Networks and Appli*, vol. 11, no. 2, pp. 161–175, 2006.

[23] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong, "Max-min d-cluster formation in wireless ad hoc networks," in *IEEE INFOCOM*, 2000

[24] Y. Wang, W. Wang, and X.-Y. Li, "Efficient distributed low-cost backbone formation for wireless networks," in *ACM MobiHoc*, 2005.

[25] J. Monteiro, et al., "Performance evaluation of dynamic networks using an evolving graph combinatorial model," in *IEEE WiMob*, 2006.

[26] C. Liu, J. Wu, "Routing in a cyclic mobispace," in *ACM MobiHoc*, 2008.

[27] L. Arantes, A. Goldman, and M. dos Santos, "Using evolving graphs to evaluate DTN routing protocols," in *ExtremeCom Workshop*, 2009.

[28] G. Lu, N. Sadagopan, B. Krishnamachari, et al., "Delay efficient sleep scheduling in wireless sensor networks," in *IEEE INFOCOM*, 2005.

[29] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *ACM MobiHoc*, 2006.

[30] Y. Zhou and M. Medidi, "Sleep-based topology control for wakeup scheduling in wireless sensor networks," in *IEEE SECON*, 2007.

[31] Y. Gu and T. He, "Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 12, pp. 1741–1754, 2011.

[32] Z. Li, Y. Peng, et al., "LBA: Lifetime balanced data aggregation in low duty cycle sensor networks," in *IEEE INFOCOM*, 2012.

[33] Y. Cao, S. Guo, and T. He, "Robust multi-pipeline scheduling in low-duty-cycle wireless sensor networks," in *IEEE INFOCOM*, 2012.

[34] T. Al-Khdour and U. Baroudi, "An energy-efficient distributed schedule-based communication protocol for periodic wireless sensor networks," *Arab. Jour. for Sci. and Eng.*, vol. 35, no. 2B, pp. 155-168, 2010.