Topology Control for Time-Evolving and Predictable Delay-Tolerant Networks

Minsu Huang, Siyuan Chen, Ying Zhu, and Yu Wang, *Senior Member*, *IEEE*

Abstract—In delay tolerant networks (DTNs), the lack of continuous connectivity, network partitioning, and long delays make design of network protocols very challenging. Previous DTN research mainly focuses on routing and information propagation. However, with a large number of wireless devices' participation, it becomes crucial regarding how to maintain efficient and dynamic topology of the DTN. In this paper, we study the topology control problem in a predictable DTN, where the time-evolving network topology is known a priori or can be predicted. We first model such time-evolving network as a directed space-time graph that includes both spacial and temporal information. The aim of topology control is to build a sparse structure from the original space-time graph such that 1) the network is still connected over time and supports DTN routing between any two nodes; 2) the total cost of the structure is minimized. We prove that this problem is NP-hard, and propose two greedy-based methods that can significantly reduce the total cost of topology while maintaining the connectivity over time. We also introduce another version of the topology control problem by requiring that the least cost path for any two nodes in this constructed structure is still cost-efficient compared with the one in the original graph. Two greedy-based methods are provided for such a problem. Simulations have been conducted on both random DTN networks and real-world DTN tracing data. Results demonstrate the efficiency of the proposed methods.

Index Terms—Topology control, delay tolerant networks, greedy algorithm, spanner, space-time graph

1 Introduction

Delay or disruption tolerant networks (DTNs) [1], [2] have drawn much attention from networking researchers due to their wide applications in challenging environments, such as space communications, military operations, and sensor networks. In DTNs, the lack of continuous connectivity, network partitioning, and very long delays are the norm, not the exception. Communication in DTNs is challenging as it must handle time-varying links, long delays, and dynamic topology. Recently, many routing schemes [3], [4], [5], [6], [7], [8], [9], [10] have been proposed for DTNs to take the intermittent connectivity and time-varying topology into consideration. However, current solutions for DTNs either use stochastic routing techniques in which replicating messages are randomly forwarded or use static prediction of future contact to select the next hop for forwarding. All these solutions ignore *temporal characteristics* of the network, thus they may lead to poor performance in time-evolving DTNs.

Delay tolerant networks often evolve over time: changes of topology can occur if some nodes appear, disappear, or move around. Such dynamics over time domain are often ignored in protocol design or simply modeled by pure randomness such as in the well-known random walk mobility model and the random graph model. For example, in traditional ad hoc networks, a network is usually modeled as a connected graph with stable end-to-end paths and the effect of dynamic changes in topology on protocol design is handled by continuous monitoring and on-demand updates when the topology changes. However, in real-world wireless networks, node mobility and the evolution of topology heavily depend on both social and temporal characteristics of the network and network participants. In many wireless network applications (such as pocket switched networks based on human mobility [11], [12], vehicular networks based on public buses or taxi cabs [13], [14], sensor networks for wildlife tracking [2], mobile social networks [15], [16], disaster-relief networks, or space communication [17], [18], [19]), there are clear socio-temporal patterns for both individual components and network structure. For *certain* types of networks, the temporal characteristics of topology could be known a priori or can be predicted from historical tracing data. For instance, it is easy to discover the temporal pattern of topology for a delay tolerant network formed by public buses [13] or satellites [19], [20], [21] with fixed tours and schedules, or a mobile social network consisting of students who share fixed class schedules. Recent study [22] shows that human mobility model can achieve a 93 percent potential predictability. In this kind of *time-evolving* and *predictable* DTNs, traditional communication protocols designed for wireless networks may be inefficient due to time-varying structure and long delays, or even fail to perform due to the lack of continuous connectivity or network partitioning. Fig. 1 illustrates an example of such time-evolving DTN. Hence, it is crucial to design new efficient protocols for this emerging type of DTNs. Routing in these time-evolving and predictable networks has been studied [23], [24], [25], [26], [27]. In this paper, we are interested in how to smartly control the dynamic topology for such *time-evolving* and *predictable* DTNs.

Network topology is always a key functional issue in design of wireless networks. For different network

• *The authors are with the Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223.*
E-mail: {mhuang4, schen40, yzhu17, yu.wang}@uncc.edu.

Manuscript received 22 Aug. 2011; revised 15 May 2012; accepted 7 Aug. 2012; published online 11 Sept. 2012.
Recommended for acceptance by S. Fahmy.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-08-0566.
Digital Object Identifier no. 10.1109/TC.2012.220.

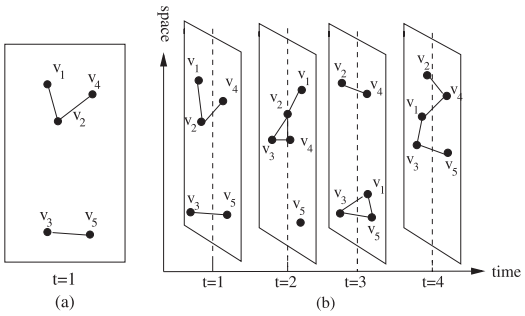


Fig. 1. *A time-evolving DTN:* (a) a snapshot of the network, (b) time-evolving topologies of the DTN (a sequence of snapshots).

applications, network topology can be controlled under different objectives (such as power efficiency, fault tolerance, and throughput maximization). Topology control (TC) has been well studied in ad hoc and sensor networks [28], [29], [30], [31], [32]. The focus of previous research is mainly on how to construct a power-efficient structure from a static and connected topology (an underlying communication graph includes all possible links). However, in DTNs, the underlying topology lacks continuous connectivity that makes existing topology control algorithms useless. Therefore, it becomes crucial regarding how to maintain efficient and dynamic topology of DTNs, especially with high-cost communication links or a large number of wireless devices' participation. For example, in satellite DTNs [19], [20], [21], the network has intermittent connectivity among multiple satellites but all of their orbits are fixed, and thus, the dynamic of topology is known a priori. When two satellites have an opportunity to communicate to each other, it may be costly, in terms of energy or other costs, to maintain the communication link and perform transmission. Moreover, the energy sources of these satellites entail solar energy, which becomes very limited considering such a system travels farther away from the Sun or is shadowed (sunlight is blocked) for long periods for planet orbiting platforms [33]. The level of power availability is a critical factor on the performance of the deployed DTN network. To save energy or other costs, it is important and efficient to carefully plan the topology of satellite DTNs by limiting the communication links. Another example is low-duty-cycle wireless sensor networks [34], [35]. In such wireless sensor networks, to resolve the conflict between limited energy and application lifetime requirements, sensors have very low communication and sensing duty cycles. Different sensors may have different cycles but they are known a priori. This type of networks can be modeled as time-evolving and predicable DTNs. Since sensor devices (e.g., MicaZ and Telos) are normally equipped with limited power sources due to their small form factor and low-cost requirements, power-efficient topology control becomes crucial. This is also true when considering mobile sensor networks [36].

In this paper, we study the topology control problem in a time-evolving and predictable DTN by taking *time-domain topological information* into consideration. Our major contributions are summarized as follows:

- We first model a time-evolving DTN as a directed space-time graph in which both spacial and temporal information is preserved. We then define the *topology control problem* that aims to build a sparser structure (also a space-time graph) from the original space-time graph such that
  - the network is still connected over time and supports DTN routing between any two nodes;
  - the total cost of the structure is minimized.

  Notice that in some *time-evolving* and *predictable* DTNs (such as the interplanetary space DTN [19]), it is too expensive to maintain dense structure. We also show that this new topology control problem is NP-hard, by connecting it with the *directed Steiner tree (DST) problem* [37].
- We propose two greedy-based methods that can significantly reduce the total cost of network topology while maintaining the connectivity over time. Both methods repeatedly add a set of edges into the topology to connect one or multiple pairs of nodes in the space-time graph. In each round, the first method basically adds one least cost path to connect one pair of nodes, while the second method adds a bunch of paths to connect multiple pairs. The latter method can theoretically achieve an approximation of the optimal solution for the topology control problem. We discuss how to address the topology control problem in undirected DTNs using the proposed greedy methods.
- We then define the *cost-efficient topology design problem* that has an additional requirement on the constructed structure: the cost of the least cost path for any two nodes in this constructed structure is at most δ times of that in the original graph, i.e., cost-efficient DTN routing is possible between any two nodes. We propose two methods to reduce the total cost of network topology while maintaining the connectivity and the cost-efficient paths between any two nodes over time.
- Simulations have been conducted on both random DTN networks and real-world DTN tracing data [38]. Results demonstrate the efficiency of all proposed methods.

The rest of this paper is organized as follows: In Section 2, we summarize related work in topology control and DTNs. In Section 3, we formally define the space-time graph and the topology control problem. Two topology control algorithms for time-evolving DTNs are proposed in Section 4. Topology control for undirected DTNs is discussed in Section 5. Section 6 introduces the cost-efficient topology control problem (CETC) and provides two methods for such a problem. Section 7 presents the simulation results of all proposed algorithms. Finally, Section 8 concludes the paper by pointing out some limitations of our study and a few possible future directions. A preliminary version of this paper appeared in [39].

2 Related Works

2.1 Topology Control in Ad Hoc and Sensor Networks

Topology control has drawn a significant amount of research interests in wireless ad hoc and sensor networks [28], [29], [30], [31], [32]. Primary topology control algorithms aim to maintain network connectivity and conserve

energy. Most topology control protocols for ad hoc and sensor networks can be classified into two categories: *geometrical structure based* and *clustering based*. In geometrical structure-based methods [30], [31], [32], a geometrical structure is constructed based on location information by removing as many links as possible from the original communication graph. Each node only selects certain neighbors (neighbors in the constructed structure) for communication. In clustering-based methods [40], [41], [42], a hierarchical structure is built using clustering formation as the virtual backbone for routing and relaying packets for the network. All of these topology control protocols deal with topology changes by reperforming the construction algorithm. Fortunately, most of the construction algorithms are localized or distributed algorithms, therefore the update cost is not expensive. However, all of these methods assume that the underlying communication graph is fully connected and they do not consider the time domain knowledge of network evolution.

2.2 Routing in Delay Tolerant Networks

Existing research in DTNs mainly focuses on routing [3], [4], [5], [6], [7], [8], [9], [10]. Most of these DTN routing protocols belong to three categories: *message-ferry based*, *opportunity based,* and *prediction based*. In message-ferry-based methods [4], [5], [43], systems usually employ extra mobile nodes as ferries for message delivery. The trajectory of these message ferries is controlled to improve delivery performance with store-and-carry. All of these approaches improve routing performance with additional mobile nodes, although controlling these nodes leads to extra cost and overhead. In opportunity-based schemes [2], [6], [7], nodes forward messages randomly hop by hop with the expectation of eventual delivery, but with no guarantee. Generally, messages are exchanged only when two nodes meet at the same place, and multiple copies of the same message are flooded in the network to increase the chance of delivery. Approaches in this category usually distribute multiple copies in the network, to ensure a high reliability of delivery. But they also bring in a high cost of buffer occupancy and bandwidth consumption. Some DTN routing protocols [8], [9], [10] use delivery estimation to determine a metric for contacts relative to successful delivery, such as delivery probability or delay, based on a history of observations. Most of these protocols focus on whether two nodes will have a contact without sufficiently considering when the contact happens. Recently, Liu and Wu [23] used the estimated expected minimum delay as a delivery probability metric in DTNs with repetitive mobility.

2.3 Modeling Time-Evolving Networks

The problem of how to model the time-evolving networks has been studied in both mobile ad hoc networks [24], [25], [26] and delay tolerant networks [23], [27]. Xuan et al. [24] first studied routing problem in a fixed schedule dynamic network, where the topology dynamics at different time intervals can be predicted. They use evolving graphs to capture the evolving characteristic of such dynamic networks. Here, an evolving graph is an indexed sequence of subgraphs of a given graph, where the subgraph at a given index point corresponds to the network connectivity at the time interval indicated by the index number. Recently, [26], [27] also use evolving graphs to evaluate various ad hoc and DTN routing protocols. Merugu et al. [25] studied the routing problem in dynamic networks as well. They use a *space-time graph* to model the network. A space-time graph is a directed graph that captures both the space and time dimensions of the network topology. We adopt this model for this paper, and will introduce more about it in the next section. Liu and Wu [23] model a cyclic mobispace as a probabilistic space-time graph in which an edge between two nodes contains a set of discretized probabilistic contacts. However, all of these previous works only focuses on the routing problem in the dynamic networks modeled by either evolving graphs or space-time graphs. In this paper, we will investigate the topology control problem in these networks.

To our best knowledge, there are no previous results on topology control in DTNs. This paper is the first attempt to study topology control for time-evolving DTNs. We believe that topology can be controlled more wisely and efficiently if the network evolution over time is known.

3 Model and the Problem

3.1 Space-Time Graph

In a DTN, different nodes corresponding to different individual devices and edges represent interactions between them over time. Since positions of individual nodes and the topology coevolve over time (as shown in the example of Fig. 1), traditional static graph model cannot represent such evolution. Thus, a sequence of static graphs is needed to model the time-evolving DTN. As shown in Fig. 1b, each static graph is a snapshot of nodes and their interactions observed at certain time step. In this example, there is no end-to-end path between some node pairs inside one snapshot, and the network is not connected in some snapshots ($t = 1, 2, 3$). The dynamic network with a sequence of snapshots then describes the evolution of interactions among nodes over a period of time. In this paper, we use a space-time graph [25] to model such a dynamic DTN.

Assume that the time is divided into discrete and equal time slots, such as $\{1, \ldots, T\}$. Let $V = \{v_1, \ldots, v_n\}$ be the set of all individual nodes in the network (which represents the set of wireless devices). Let $G^t = (V^t, E^t)$ be a directed graph representing the snapshot of the network at time slot t and $\overrightarrow{v_i^t v_j^t} \in E^t$ if node v_i can communicate to v_j at time t. Then, the dynamic network can be modeled as a union of a series of directed graphs $\{G^t | t = 1, \ldots, T\}$. Fig. 2a shows the sequence of snapshots of the network in Fig. 1 at each time slot. This representation of the network includes all information from both spacial and temporal information of this time-evolving network. However, most existing DTN protocols are not designed for this model. Instead, they usually use a standard aggregated representation as shown in Figs. 2b and 2c, where an edge exists between a pair of nodes if they have interacted at any point during the time period. The weight of such a link is the probability of the occurrence (or the fraction of the occurrence over the total/historical period) of the link. For example, link $\overrightarrow{v_1 v_3}$ exists in two time slots out of total four slots, thus its weight is 0.5. Many DTN routing protocols [3], [8] estimate this contact probability based on past history and use it to select

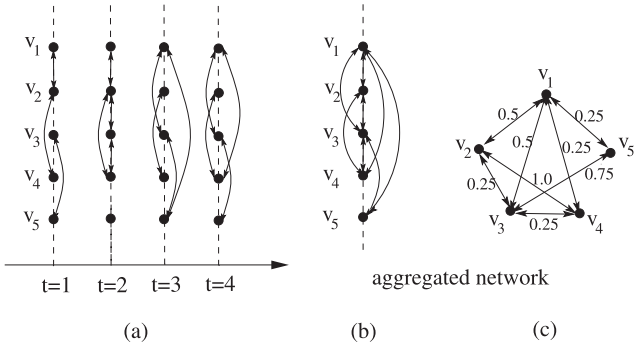


Fig. 2. Different graph models for the time-evolving network shown in Fig. 1: (a) a sequence of snapshots of the network at each time slot, denoted by $\{G^t\}$; (b) and (c) its corresponding aggregated graph model.

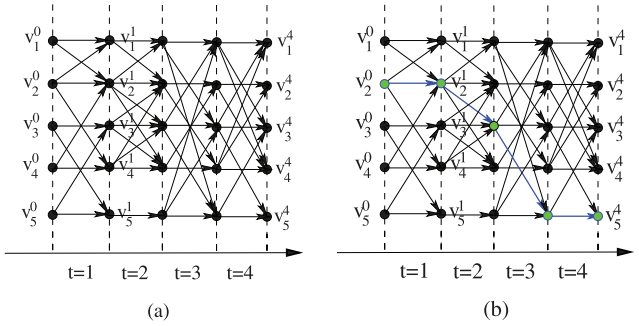


Fig. 3. Space-time graph model from the same time-evolving network in Fig. 2: (a) the corresponding space-time graph $\mathcal{G}$; (b) a space-time path (in blue and bold) from the source v_2 to the destination v_5.

forwarding nodes. However, such an aggregated view discards temporal information about the timing and order of interactions. For example, based on the aggregated model, there is a path from v_4 to v_2 via v_1. But link $\overrightarrow{v_4v_1}$ only exists in the last time slot when link $\overrightarrow{v_1v_2}$ is already broken. Thus, path $v_4v_1v_2$ cannot be used for packet delivery from v_4 to v_2.

Now, we convert the sequence of static graphs $\{G^t|t = 1,\ldots,T\}$ into a space-time graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is a directed graph defined in both spacial and temporal space. See Fig. 3 for illustration. In the space-time graph $\mathcal{G}$, $T+1$ layers of nodes are defined and each layer has n nodes; thus, the vertex set $\mathcal{V} = \{v_j^t | j = 1,\ldots,n \text{ and } t = 0,\ldots,T\}$ and there are $n(T+1)$ nodes in $\mathcal{G}$, i.e., $|\mathcal{V}| = n(T+1)$. Two kinds of links (spatial links and temporal links) are added between consecutive layers in $\mathcal{E}$. The space between consecutive layers is a time slot. A temporal link $\overrightarrow{v_j^{t-1}v_j^t}$ (those horizontal links in spacial Fig. 3) connects the same node v_j across consecutive $(t-1)$th and tth layers, which represents the node carrying the message in the tth time slot. A spatial link $\overrightarrow{v_j^{t-1}v_k^t}$ represents forwarding a message from one node v_j to its neighbor v_k in the tth time slot (i.e., $\overrightarrow{v_jv_k} \in E^t$). By defining the space-time graph $\mathcal{G}$, any communication operation in the time-evolving network can be simulated on this directed graph. As shown in Fig. 3b, a path from v_2^0 to v_5^4 shows a particular routing strategy to deliver the packet from v_2 to v_5 in the network using four time slots. v_2 holds the packet for the first time slot, then passes it to v_3 at $t = 2$, and so on. Notice that in this space-time graph model, only one-hop transmission is allowed within one time slot. In other words, we assume that each time slot is only long enough for one packet transmission. For example, v_1 cannot send the packet to v_4 via v_2 in the first time slot, and such a transmission needs two time slots. To allow multihop transmissions within a single time slot, we can modify the space-time graph by adding virtual links to represent multihop paths within the same time slot. Hereafter, we assume multihop transmission within one time slot is not allowed.

Space-time graph model captures both the space and time dimensions of the network topology. This model increases the complexity of protocol design (introducing a new dimension of time domain), but also provides more choices of routes/links for selection (enjoying efficient combinations of spacial and temporal links). We further assume that for each direct link $e \in \mathcal{E}$ there is a cost $c(e)$, which is the energy cost associated with transiting a message on that link (transiting it from one node to the other node on a spacial link or holding a message within one node over a temporal link). The total cost of a space-time graph $c(\mathcal{G})$ is the summation of the costs of all links in $\mathcal{G}$, i.e., $c(\mathcal{G}) = \sum_{e\in\mathcal{G}} c(e)$. Given the costs of links, we can define the least cost path $P_{\mathcal{G}}(u,v)$ from u to v in $\mathcal{G}$. The total cost of such a least cost path $P_{\mathcal{G}}(u,v)$ in $\mathcal{G}$ is denoted by $d_{\mathcal{G}}(u,v) = \sum_{e\in P_{\mathcal{G}}(u,v)} c(e)$, which is the summation of costs of all links in path $P_{\mathcal{G}}(u,v)$.

3.2 New Topology Control Problem

We now define the *topology control problem* on space-time graphs.

Definition 1. *Given a connected space-time graph $\mathcal{G}$, the aim of* topology control problem *is to construct a sparse space-time graph $\mathcal{H}$, which is a subgraph of the original space-time graph $\mathcal{G}$, such that 1) $\mathcal{H}$ is still connected over the time period T; and 2) the total cost of $\mathcal{H}$ is minimized.*

Notice that in some *time-evolving* and *predictable* DTNs (such as the interplanetary space DTN [19]), it is too expensive to maintain dense structure, and thus a connected subgraph is more cost-efficient. Here, *connectivity* has a *different* definition from that of a static graph.

Definition 2. *A space-time graph $\mathcal{H}$ is* connected *over time period T if and only if there exists at least one directed path for each pair of nodes (v_i^0, v_j^T) (i and j in $[1,n]$).*

This can guarantee that the packet can be delivered between any two nodes in the network over the period of T. Notice that a connected space-time graph does not require connectivity in each snapshot. Hereafter, we assume that the original space-time graph $\mathcal{G}$ is always connected. Fig. 4 shows an example of topology control on the space-time graph in Fig. 3a. After topology control, 16 directed links are removed from the original space-time graph. Not all snapshots are connected, but every node can find the space-time path to any other node. It is interesting to see that there is no spacial link remaining in the last time slot because every node can be reached by any other node using only three time slots. This example demonstrates the efficiency of topology control over time domain.

Note that the newly defined topology control problem is different from the standard space-time routing [24], [25], which aims to find the most cost-efficient space-time path for a pair of source and destination. The topology control

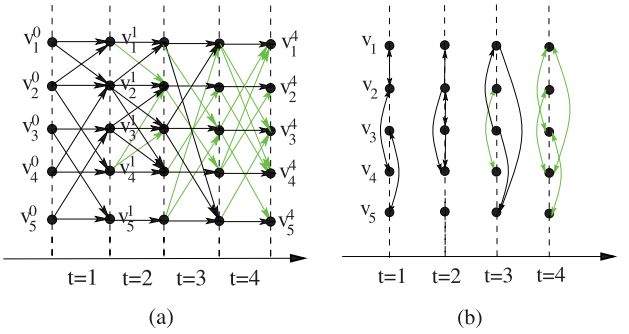


Fig. 4. Topology control on time-evolving network (over the one shown in Fig. 1): (a) a new connected subgraph $\mathcal{H}$ of $\mathcal{G}$ (green links are removed links from Fig. 3a); (b) its corresponding sequence of static graph with fewer links than the one of Fig. 2a.

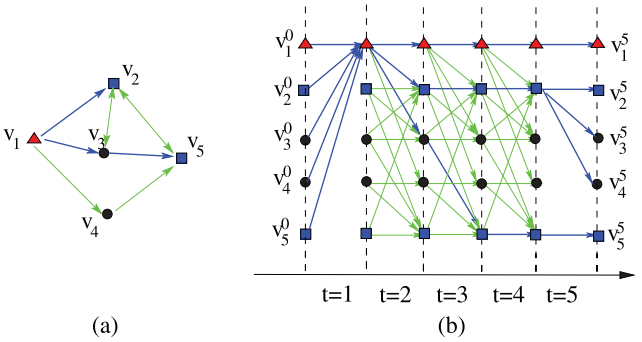


Fig. 5. Reduction from (a) the directed Steiner tree problem to (b) our TC problem on space-time graphs. Here, the blue and bold structures are the corresponding solutions in DST and TC.

problem 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 the cost efficiency of the topology. However, we will study the tradeoff between topology control and cost-efficient routing in Section 6 by introducing a new version of TC problem (CETC).

The topology control problem for a space-time graph is much harder than the one for a static graph. For a static graph without time domain, a spanning tree can achieve the goal of keeping connectivity. However, in a space-time graph, it is not a solution to simply apply the spanning tree in each snapshot because the network in each snapshot may not be connected at all. On the other hand, a spanning tree or spanning forest of the whole space-time graph is not a direct solution either because it connects each node in every snapshot, which is not necessary and a waste of many links. Such a method is neither optimal nor even near optimal.

We now prove the NP-hardness of TC on space-time graphs by a reduction from the *directed Steiner tree* problem [37], which is a known NP-hard problem. The DST problem is formed as following: given a directed graph $G = (V, E)$ with weights on the edges, a set of nodes $Q \subseteq V$, and a root node v_r, find a minimum weight out-branching tree $\mathcal{T}$ rooted at v_r, such that all nodes in Q are included in $\mathcal{T}$.

Theorem 1. *Our newly defined* topology control problem on space-time graphs *is a NP-hard problem.*

Proof. We first show how to reduce the DST problem into our TC problem. Given an instance of DST with graph $G = (V, E)$, the set of nodes $Q = \{v_{q_1}, \ldots, v_{q_m}\}$ need to reach and the root v_r, we can construct an instance of TC on a space-time graph $\mathcal{G}$ as follows.

Assume that D is the longest hop count of a directed path (no loop) from v_r to other nodes in G, thus $D < n$. We first construct a space-time $\mathcal{G}$ with $D + 2$ time slots, i.e., $D + 3$ layers of nodes. Each snapshot of $\mathcal{G}$ has n nodes $v_1^t, \ldots, v_n^t$.

For the first time slot ($t = 1$), all nodes v_i^0 are connected to v_r^1. For the last time slot ($t = D + 2$), node v_r^{D+2} and nodes $v_{q_i}^{D+2}$ are connected to their corresponding nodes in $D + 3$ layer, and node $v_{q_1}^{D+2}$ is also connected all other nodes (not v_r and not in Q) in $D + 3$ layer. All links in the first and the last time slots have the cost of 0. For each time slot except for the first and the last, we add temporal link $\overrightarrow{v_i^t v_i^{t+1}}$ with cost 0 for all nodes v_i, and add spacial link $\overrightarrow{v_i^t v_j^{t+1}}$, if there is a link $v_i v_j$ in G, with the same cost of that in G.

By this construction, it is easy to find a solution of TC with the same cost in $\mathcal{G}$ for any solution of DST in G, and vice versa. Fig. 5 shows an example with v_1 as the root and $Q = \{v_2, v_5\}$. The blue and bold structures are from one set of solutions. Since the construction of $\mathcal{G}$ can be done in polynomial time and the DST problem is NP-hard, the TC on space-time graphs is also NP-hard. □

4 Topology Control for Directed Delay-Tolerant Networks

Since TC over space-time graphs is NP-hard, we will propose several heuristics to construct a sparse structure that fulfills the connectivity requirement over a space-time graph.

4.1 Union of Least Cost Path Algorithm (ULCP)

One naive method for maintaining the network connectivity is keeping all the least cost paths from v_i^0 to v_j^T for $i, j = 1, \ldots, n$. Obviously, the resulting graph still keeps network connectivity. Algorithm 1 shows the detailed algorithm. Its time complexity is $O(n^2 T(\log(nT) + n)) = O(Tn^3 + Tn^2 \log(Tn))$ because we only need to compute n^2 least cost paths of $\mathcal{G}$ (with $O(nT)$ nodes and at most $O(n^2 T)$ links). This can be easily achieved by running n times of Dijkstra's algorithm whose complexity is $O(nT(\log(nT) + n))$. Hereafter, we refer to this method as the *union of least cost path algorithm* and use it as the reference method.

Algorithm 1. Union of Least Cost Path (ULCP) Algorithm.

```
1: H ← φ; X = {(v_i^0, v_j^T)} for all integer 1 ≤ i, j ≤ n.
2: for all pairs (v_i^0, v_j^T) ∈ X do
3:     Find the least cost path P_G(v_i^0, v_j^T) in G.
4:     if e ∈ P_G(v_i^0, v_j^T) and e ∉ H then
5:         H = H ∪ {e}.
6:     end if
7: end for
8: return H
```

4.2 Greedy Algorithm Based on Least Cost Path

Although the structure built by ULCP method can guarantee the connectivity over time, it may contain more

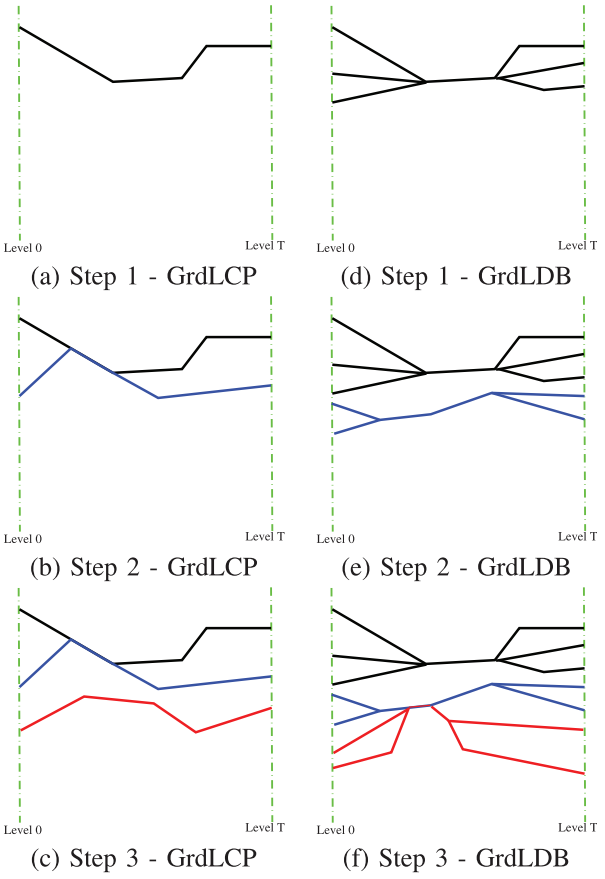


Fig. 6. Illustrations of Algorithm 2 and Algorithm 3: (a-c) Algorithm 2 repeatedly adds one least cost path into the topology to connect one pair of nodes in X. (d-f) Algorithm 3 repeatedly adds one bunch with least density into the topology to connect multiple pairs of nodes in X. Both algorithms terminate when all pairs of nodes in X are connected.

links than necessary. Therefore, we propose a new greedy algorithm (as shown in Algorithm 2) to further improve the performance. The basic idea is quite simple and presented as follows: Initially, we need to connect n^2 pair of nodes in X. As shown in Fig. 6a, in each round we pick 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, we add all edges in this path into $\mathcal{H}$, clear the costs of these edges to zeros, and remove this pair from X. This procedure is repeated as shown in Figs. 6a, 6b, and 6c. After n^2 rounds, all pairs of nodes (v_i^0, v_j^T) are guaranteed to be connected by paths in $\mathcal{H}$. It is obvious that the output of this method is much sparser than the one of ULCP method. We refer to this method as the *greedy method based on least cost path* (GrdLCP). The time complexity of this algorithm is $O(Tn^5 + Tn^4 \log(Tn))$ because in each round n times of Dijkstra's algorithm are running on the space-time graph and there are n^2 rounds.

Algorithm 2. Greedy Algorithm based on Least Cost Path.

1: $\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^T)\}$ for all i and j in $[1, n]$.
2: **while** $X \neq \phi$ **do**
3: find the least cost paths for every pair of nodes in X, and assume $P_{\mathcal{G}}(v_i^0, v_j^T)$ has the least cost among these paths.
4: **if** $e \in P_{\mathcal{G}}(v_i^0, v_j^T)$ **then**
5: $\mathcal{H} \leftarrow e$; $c(e) \leftarrow 0$.
6: **end if**
7: $X \leftarrow X - (v_i^0, v_j^T)$.
8: **end while**
9: **return** $\mathcal{H}$

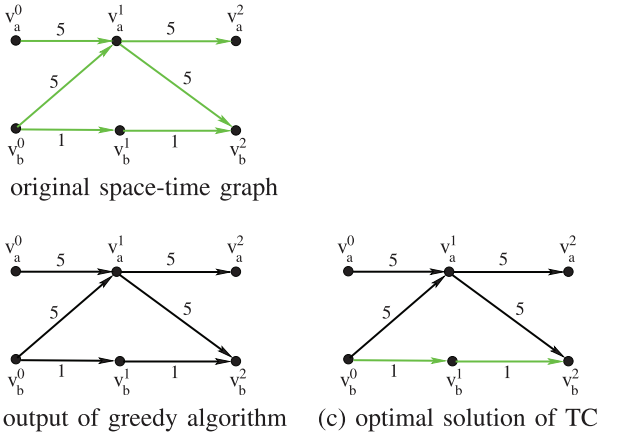


Fig. 7. An example: (a) original space-time graph, (b) output of all greedy-based algorithms (ULCP, GrdLCP, and GrdLDB), and (c) optimal solution of topology control. Black links are final links in each solution.

Notice that both ULCP and GrdLCP may not lead to the optimal solution. Fig. 7 shows such an example: a network with two nodes (v_a and v_b) and two time slots. Both ULCP and GrdLCP generate a structure with total cost 22 as shown in Fig. 7b (because they will first pick the path/bunch including $\overrightarrow{v_b^0 v_b^1}$ and $\overrightarrow{v_b^1 v_b^2}$), while the optimal solution is a structure with total cost 20 (Fig. 7c). However, GrdLCP and ULCP can perform well when the network is much denser and with more nodes. Our simulations in Section 7 confirm this.

4.3 Greedy Algorithm Based on Least Density Bunch

Next, we present a more complex greedy algorithm (as shown in Algorithm 3) which is inspired by a method proposed by Charikar and Chekuri [44] for *directed generalized Steiner network* (DGSN) problem [46]. The DGSN problem is also a NP-hard problem and defined as follows: Given a directed 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 control problem is a special case of DGSN problem with $X = \{(v_i^0, v_j^T)\}$ 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 Chekuri et al. [46]. However, their method is too complex.

Algorithm 3. Greedy Algorithm based on Least Density Bunch.

1: $\mathcal{H} \leftarrow \phi$; $k = n^2$; $X = \{(v_i^0, v_j^T)\}$ for all i and j in $[1, n]$.
2: **while** $X \neq \phi$ **do**
3: $d \leftarrow \infty$; $B \leftarrow \phi$.
4: **for all** pairs $(p, q) \in \mathcal{V} \times \mathcal{V}$ **do**
5: **for all** pairs $(v_i^0, v_j^T) \in X$ **do**
6: $s[v_i^0, v_j^T] \leftarrow d_{\mathcal{G}}(v_i^0, p) + d_{\mathcal{G}}(q, v_j^T)$.

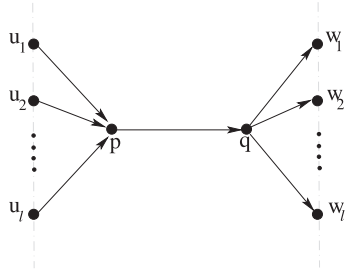


Fig. 8. Illustration of a bunch connecting l pairs of nodes. Here, each arrow represents a directed least cost path. The total cost of this bunch is $c(B) = d_{\mathcal{G}}(p,q) + s[u_1,w_1] + s[u_2,w_2] + \cdots + s[u_l,w_l]$.

```
    end for
    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
        C ← d_G(p,q) + s[u_1,w_1] + s[u_2,w_2] + ··· + s[u_l,w_l].
        if C/l ≤ d then
            d ← C/l; k1 = l;
            B ← P_G(p,q) + P_G(u_1,p) + P_G(q,w_1) + ··· + P_G(u_l,p) + P_G(q,w_l).
        end if
    end for
  end for
  H ← H + B; k = k − k1;
  X ← X − {(u_1,w_1), ..., (u_k1, w_k1)}.
end while
return H
```

The main idea of our greedy algorithm (Algorithm 3) is to find good bunches repeatedly instead of finding least cost paths repeatedly. Here, a *bunch* B is a structure connected certain pair of nodes in X as shown in Fig. 8. In this figure, the edge between any two nodes u and v (such as u_i and p, p and q, or q and w_i) is a virtual edge that represents the least cost path $P_{\mathcal{G}}(u,v)$ from u to v in $\mathcal{G}$. Assume that the cost of $P_{\mathcal{G}}(u,v)$ is $d_{\mathcal{G}}(u,v)$. We use $s[u_i,w_i]$ to represent the cost of $P_{\mathcal{G}}(u_i,p)$ plus $P_{\mathcal{G}}(q,w_i)$, i.e., $s[v_i^0,v_j^T] = d_{\mathcal{G}}(v_i^0,p) + d_{\mathcal{G}}(q,v_j^T)$. Thus, the total cost of bunch B that connects l-pair of nodes in X is $c(B) = d_{\mathcal{G}}(p,q) + s[u_1,w_1] + s[u_2,w_2] + \cdots + s[u_l,w_l]$. Further, let $d(B) = c(B)/l$ be the density of this bunch, which implies how much cost is used to connect l pairs of nodes. The greedy algorithm considers all possible bunches and greedily selects the bunch with the smallest density in each round. After a bunch is selected, all edges 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. Figs. 6d, 6e, and 6f show the procedure. We refer to this method as the *greedy method based on least density bunch* (GrdLDB).

The time complexity of this algorithm is roughly $O(T^2n^6\log n)$ because the outer while-loop runs n^2 times in the worst case; the outer for-loop runs $O(T^2n^2)$ 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. However, the time needed for such an algorithm is much less than $O(T^2n^6\log n)$. Although with larger time complexity than GrdLCP and ULCP, the GrdLDB algorithm has a nice property in theory: it can achieve approximation-guarantee in terms of the total cost compared with the optimal solution. Charikar and Chekuri [44] 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 directed generalized Steiner network problem. Therefore, we have the following theorem for our topology control problem because $k = n^2$.

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

Note that Algorithm 3 will also lead to the nonoptimal solution for the example shown in Fig. 7.

Similar to GrdLCP, the least density bunch-based algorithm can be modified to the following version. In each round, reset the cost of links to zero once they are added to the current solution. Nonetheless, by doing so, it loses the approximation ratio (in Theorem 2) as well.

5 Topology Control for Undirected Delay-Tolerant Networks

So far, we consider a directed delay-tolerant network where the static graph in each snapshot G^t is a directed graph. Therefore, directed links $\overrightarrow{v_i^{t-1}v_j^t}$ and $\overrightarrow{v_j^{t-1}v_i^t}$ in the space-time graph $\mathcal{G}$ are independent to each other and have individual costs. In other words, it is possible that only one of such link exists in $\mathcal{G}$ (as in the example in Fig. 7) or $c(\overrightarrow{v_i^{t-1}v_j^t}) \neq c(\overrightarrow{v_j^{t-1}v_i^t})$ even hey both exist. Similarly, the output of a topology control algorithm can use just one of these two links in the resulting topology (as shown in Fig. 4). However, in some network applications, the connection between nodes is necessary to be symmetric and undirected. Therefore, in this section, we consider an undirected delay-tolerant network, where each G^t is an undirected graph. For each edge $v_i^tv_j^t$ in G^t, it has only one cost. Let $c_{i,j}^t$ represent this cost. In this situation, both directed links $\overrightarrow{v_i^{t-1}v_j^t}$ and $\overrightarrow{v_j^{t-1}v_i^t}$ in the space-time graph need to be kept or removed simultaneously. The cost of such two links is one value $c_{i,j}^t$ instead of two values. Notice that the space-time graph $\mathcal{G}$ is always a directed graph. Here, *undirected* or *directed* is defined for the static graph of G^t in each snapshot. To make the proposed algorithms work on the undirected delay-tolerant networks, we propose two methods to convert the undirected network into a directed network.

5.1 Converting Method One—Double Cost

The first method is straightforward. Consider the sequence of static undirected graph $\{G^t\}$. For an undirected link $v_i^tv_j^t \in G^t$, we set the costs of the corresponding pair of spatial links ($\overrightarrow{v_i^{t-1}v_j^t}$ and $\overrightarrow{v_j^{t-1}v_i^t}$) to $c_{i,j}^t$ (as shown in Fig. 9a). We call this converted space-time graph $\mathcal{G}$. Then, the proposed methods can be still applied. If the output of such an algorithm only uses one of the spatial links, we add the other in the final output too. Notice that now the cost

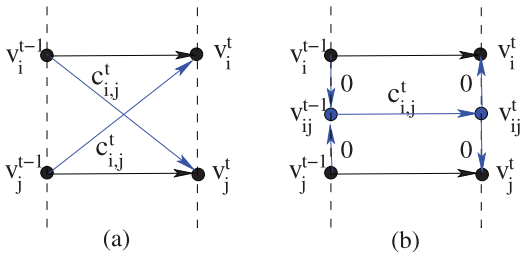


Fig. 9. Illustration of construction of spacial links in the space-time graph: (a) converting method one—doubling the cost; (b) converting method two—adding new nodes.

considered by our algorithm for path/bunch selection is different from the real cost to support such a structure. Next, we prove a lemma that can guarantee the approximation of our approach.

Lemma 1. *For topology control problem, the cost of optimal solution Opt_A in the original undirected graph $\{G^t\}$ is less than or equal to twice of that of optimal solution Opt_B in the converted space-time graph $\mathcal{G}$.*

Proof. Consider the solution of Opt_B first. If $\overrightarrow{v_i^{t-1}v_j^t} \in Opt_B$ but $\overrightarrow{v_j^{t-1}v_i^t} \notin Opt_B$, we can construct a new solution S_B by adding $\overrightarrow{v_j^{t-1}v_i^t}$ into Opt_B. Clearly, $c(S_B) \leq 2c(Opt_B)$. Note that from this new solution S_B, we can construct a feasible solution S_A of the topology control problem in the original undirected graph $\{G^t\}$ by keeping $v_i^t v_j^t \in S_A$ if and only if both $\overrightarrow{v_i^{t-1}v_j^t}$ and $\overrightarrow{v_j^{t-1}v_i^t}$ are in S_B. It is clear that $c(S_A) \leq c(S_B)$ because the total cost of S_A only counts one cost for both directed links. At last, since Opt_A is the optimal solution of the topology control problem in the original undirected graph, $c(Opt_A) \leq c(S_A)$. Putting everything together, $c(Opt_A) \leq c(S_A) \leq c(S_B) \leq 2c(Opt_B)$. □

This lemma implies that Algorithm 3 can still achieve $O(n^{4/3}\log^{1/3} n)$ for the topology control problem in undirected graph by using our converting method one.

5.2 Converting Method Two—New Graph

The second method converts the sequence of static undirected graph $\{G^t\}$ into a new space-time graph by introducing new nodes. For an undirected link $v_i^t v_j^t \in G^t$, we first add two new nodes v_{ij}^{t-1} at $(t-1)$th layer and v_{ij}^t at tth layer. Then, five directed links $\overrightarrow{v_i^{t-1}v_{ij}^{t-1}}$, $\overrightarrow{v_j^{t-1}v_{ij}^{t-1}}$, $\overrightarrow{v_{ij}^t v_i^t}$, $\overrightarrow{v_{ij}^t v_j^t}$, $\overrightarrow{v_{ij}^{t-1}v_{ij}^t}$ are added in the space-time graph. The costs of the first four links are set to zero, while $c(\overrightarrow{v_{ij}^{t-1}v_{ij}^t}) = c_{i,j}^t$. Fig. 9b illustrates this procedure. It is obvious that there is one-to-one mapping between the constructed space-time graph and the original undirected network. Then, our proposed methods can be directly applied to the constructed directed space-time graph and their output can be easily converted back to an undirected topology for the undirected graph sequence. The only drawback of this method is that additional nodes and links may increase the input size for our algorithms.

6 Cost-Efficient Topology Control for Delay-Tolerant Networks

So far, our topology control problem aims to minimize the cost of topology by removing links from the original space-time graph, while the network connectivity is maintained. However, with fewer links, the performance of routing protocols maybe hurt because the route used by them may lead to larger costs. Therefore, in this section, we introduce a new version of topology control problem that also considers the cost efficiency of paths in the constructed structure.

6.1 Cost-Efficient Topology Control: Spanner over Time

We now define the *cost-efficient topology control problem* on space-time graphs.

Definition 3. *Given a connected space-time graph $\mathcal{G}$, the aim of* cost-efficient topology control problem *is to construct a sparse space-time graph $\mathcal{H}$, which is a subgraph of the original space-time graph $\mathcal{G}$, such that 1) $\mathcal{H}$ is still* connected *over the time period T; 2) $\mathcal{H}$ is δ-spanner of $\mathcal{G}$ over the time period T; and 3) the total cost of $\mathcal{H}$ is minimized.*

Compared with the TC problem, CETC adds an additional requirement on spanner property. Here, *spanner* has a *different* definition from the one of a static graph.

Definition 4. *Given a real number $\delta > 1$, a subgraph $\mathcal{H}$ of $\mathcal{G}$ is δ-spanner of $\mathcal{G}$ if and only if for any pairs of nodes (v_i^0, v_j^T), the cost of the least cost path in $\mathcal{H}$ is at most δ times the cost of the least cost path in $\mathcal{G}$, i.e., $\max_{1\leq i,j\leq n}\{\frac{d_{\mathcal{H}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\} \leq \delta$. Here, $\max_{1\leq i,j\leq n}\{\frac{d_{\mathcal{H}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\}$ is called* spanning ratio *of $\mathcal{H}$.*

In other words, the constructed subgraph can still support cost-efficient routes for any two devices over the time period. This is obviously a stronger requirement than connectivity and makes the problem harder than the TC problem. Since the TC problem (which we already prove to be NP-hard) is a special case of the CETC problem, where δ is set to be infinity, CETC is also NP-hard.

For a static graph without time domain, there are several efficient methods to construct a spanner (such as [28], [29], [30], [32] for unit disk graphs using geometric properties or [45] for general weighted graphs). However, on the one hand, in a time-evolving DTN modeled by a space-time graph, it is not feasible to simply apply spanner methods for unit disk graph in each time slot because the network in each single snapshot may not be connected at all. On the other hand, the classic greedy method for a general graph [45] over the whole space-time graph is neither a solution of CETC. The greedy method basically sorts all links in terms of their costs, then in increasing order of cost it adds link uv into H if $d_H(u,v) \geq \delta d_G(u,v)$. Nonetheless, in the space-time graph $\mathcal{G}$, each link v_i^t, v_j^{t+1} is the only path connected v_i^t and v_j^{t+1}, and thus the greedy algorithm will keep all links of $\mathcal{G}$ in $\mathcal{H}$. This is unacceptable for CETC.

6.2 Cost-Efficient Topology Control Algorithms

In this section, we will study efficient algorithms to construct a sparse structure that fulfills the connectivity and spanner requirements over a space-time graph. For each algorithm, the inputs are the original graph $\mathcal{G}$ and a real number $\delta \geq 1$, and the output is a subgraph $\mathcal{H}$ of $\mathcal{G}$. Notice that the ULCP method can build a 1-spanner of $\mathcal{G}$ by keeping all least cost paths. Since $\delta \geq 1$, the resulting structure satisfies the spanner property. Next, we propose two more efficient greedy algorithms.

6.2.1 Greedy Algorithm to Delete Links (GrdDL)

The basic idea of the first greedy algorithm is deleting edges from input graph (either the original graph $\mathcal{G}$ or the output graph from ULCP) in a decreasing order based on link cost. Link $v_i^t v_j^{t+1}$ is removed if without this link the subgraph $\mathcal{H}$ is still a δ—spanner of the original graph $\mathcal{G}$. Algorithm 4 shows the details, and we denote this algorithm as GrdDL hereafter. The time complexity analysis is straightforward. The sorting could be done in $O(n^2T\log(n^2T))$ with $O(n^2T)$ links. The *for*-loop includes n^2T rounds of computations of least cost paths. Thus, the time complexity of this part is $O(n^2T \cdot n^2T(\log(nT)+n)) = O(n^4T^2(\log(nT)+n))$. Therefore, the total time complexity of GrdDL is in order of $O(n^4T^2(\log(nT)+n))$, which is much higher than that of ULCP. Notice that because the output of ULCP is much sparser than the original graph, it will save computation if we use it as the input of GrdDL.

Algorithm 4. Greedy Algorithm to Delete Links (GrdDL).

1: $\mathcal{H} \leftarrow \mathcal{G}$.
2: Sort all links in link set $\mathcal{E}$ of $\mathcal{G}$ based on their costs.
3: **for all** $e \in \mathcal{E}$ (processed in decreasing order of costs) **do**
4: **if** $\max_{1\le i,j\le n}\{\frac{d_{\mathcal{H}-\{e\}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\} \le \delta$ **then**
5: $\mathcal{H} = \mathcal{H} - \{e\}$.
6: **end if**
7: **end for**
8: **return** $\mathcal{H}$

6.2.2 Greedy Algorithm to Add Links (GrdAL)

The second greedy algorithm starts from building a connected sparse structure and then adds more links into it to satisfy the spanner property. We can use GrdLCP to build the sparse structure $\mathcal{H}$ that connects all pairs of (v_i^0, v_j^T). Then, for each pair (v_i^0, v_j^T), we calculate its least cost path in $\mathcal{H}$. If the cost of $P_{\mathcal{H}}(v_i^0, v_j^T)$ is greater than δ times of the cost of $P_{\mathcal{G}}(v_i^0, v_j^T)$ in the original graph, we directly add the links in this least cost path into $\mathcal{H}$. See Algorithm 5 for details. Hereafter, we denote this method as GrdAL. The complexity of GrdAL is less than the one of GrdDL algorithm. Both GrdLCP and the remaining steps need $n^2 \cdot O(n^2T(\log(nT)+n))$ time because they both run the computation of least cost paths for n^2 rounds. Therefore, the total time complexity is $O(n^4T(\log(nT)+n))$.

Algorithm 5. Greedy Algorithm to Add Links (GrdAL).

1: $X = \{(v_i^0, v_j^T)\}$ for all integer $1 \le i, j \le n$.
2: $\mathcal{H} \leftarrow$ the output of GrdLCP (Algorithm 2);
3: **for all** $(v_i^0, v_j^T), 1 \le i, j \le n$ **do**
4: **if** $d_{\mathcal{H}}(v_i^0, v_j^T) > \delta \cdot d_{\mathcal{G}}(v_i^0, v_j^T)$ **then**
5: Add all links $e \in P_{\mathcal{G}}(v_i^0, v_j^T)$ into $\mathcal{H}$ if $e \notin \mathcal{H}$.
6: **end if**
7: **end for**
8: **return** $\mathcal{H}$

Both GrdDL and GrdAL can obviously satisfy the spanning ratio requirement δ.

7 Simulations

In this section, we will evaluate our proposed topology control algorithms, namely, *Greedy Algorithm based on Least Cost Path Greedy Algorithm based on Least Density Bunch*, *Greedy Algorithm to Delete Links* running on the original graph $\mathcal{G}$ (GrdDL-G), *Greedy Algorithm to Delete Links* running on the output of ULCP (GrdDL-U), and *Greedy Algorithm to Add Links* by comparing their performances with *Union of Least Cost Path* method. We implement all these algorithms in a simulator developed by our group. The underlying time-evolving networks are either randomly generated from a random graph model or directly extracted from Cambridge Haggle tracing data [38]. All of the networks are directed DTNs, and thus, we only test our algorithms on directed networks in this section. In all simulations, we take four metrics as the performance measurement for any topology control algorithm:

- *Total Cost.* The total cost of the constructed topology $\mathcal{H}$ (output of the algorithm), i.e., $c(\mathcal{H}) = \sum_{e\in\mathcal{H}} c(e)$.
- *Total Number of Edges.* The total number of edges in the constructed $\mathcal{H}$, i.e., $|\mathcal{H}|$. Here, $|\mathcal{G}|$ denotes the number of edges of graph $\mathcal{G}$.
- *Spanning Ratio.* $\max_{1\le i,j\le n}\{\frac{d_{\mathcal{H}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\}$. This metric is for the CETC problem.
- *Running Time.* The total time to generate the output topology $\mathcal{H}$.

For all the simulations, we repeat the experiment 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 spanning ratio.

7.1 Simulations for TC Problem

7.1.1 Simulations on Random Networks

We first generate a sequence of static random graphs $\{G^t\}$ to represent the time-evolving DTN. Here, we consider a DTN with 10 nodes ($n = 10$) and spreading over 10 time slots ($T = 10$). 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 $\overrightarrow{v_i v_j}$ with a fixed probability p. The cost of each inserted edge is randomly chosen from 1 to 5.[1] After generating $\{G^t\}$, we convert it into its corresponding space-time graph $\mathcal{G}$ with $n(T+1)$ nodes. Then, three topology control algorithms (ULCP, GrdLCP, GrdLDB) for the TC problem take the same $\mathcal{G}$ as the input.

Fig. 10 shows the constructed topologies in space-time graph format from all three algorithms when $p = 0.1$ and 0.8. It is obvious that GrdLCP selects fewer edges than the other two algorithms. However, all of them can remove large portions of links to save energy while guarantee the connectivity over space-time graph (there are paths from all nodes in the first time slot to those in the last time slot). Notice that when the density of network is larger (with larger p), such saving is more visible.

We then increase the network density by increasing p from 0.1 to 1.0. Small value of p leads to a sparse DTN, and $p = 1.0$ implies that the topology in each time slot is a complete graph. For each setting, we generate 50 random networks and report the average performance of topology

1. In this paper, we consider topology control problems with general link costs (could be energy, delay, or other costs, depending on the application). Therefore, we use random cost in all simulations.

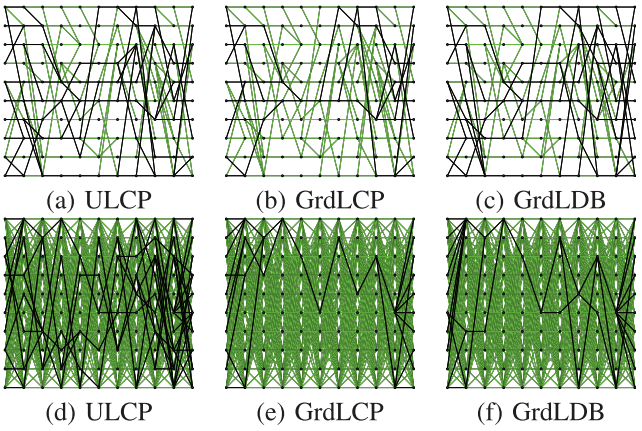


Fig. 10. Topologies generated over the same random network from proposed TC methods: green links are removed links from the original graph $\mathcal{G}$, while black links are the ones kept by each algorithm. (a-c): $p = 0.1$, (d-f): $p = 0.8$.

control among them. Figs. 11a and 11b show the ratio between the total cost/number-edges 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 control algorithm, compared with the original network without topology control. From the results, all topology control algorithms can significantly reduce the cost of maintaining the connectivity. Even with the least density ($p = 0.1$), all algorithms can save more than 50 percent cost and around 50 percent edges. For $p = 1.0$, more than 95 percent cost is saved. In most cases, GrdLDB and GrdLCP can achieve better efficiency than ULCP. However, GrdLCP has a clear advantage over GrdLDB. Even though GrdLDB has the theoretical approximation bound, GrdLCP performs much better in practice. With increasing density of the network, the ratio of cost decreases. This indicates that more saving can be achieved by all TC algorithms with dense networks. Fig. 11c shows the running time of each algorithm. It is clear that all algorithms use more time with a denser space-time graph. Compared with ULCP, GrdLDB and GrdLCP use more time, which is consistent with our theoretical analysis of time complexity. Obviously, there is a tradeoff between the running time and the cost saving from TC algorithms.

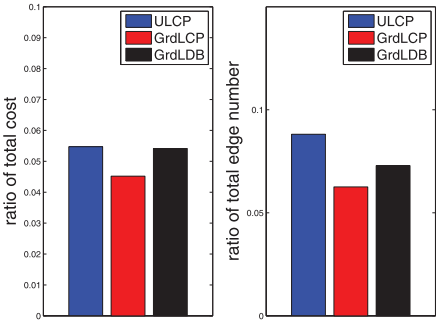


Fig. 12. Simulation results of TC on networks from Cambridge Haggle [38] tracing data. Results are averages over 13 small networks.

We also preform simulations on networks with different size and different time length. The results and conclusions are similar, and thus they are excluded here due to space limit.

7.1.2 Simulations on Tracing Data

Recently, there are tremendous efforts in the wireless network research community on measuring, recording, and releasing tracing data from real-world wireless networking systems. Taking such advantages, we use real-world wireless tracing data to evaluate our topology control algorithms. Particularly, we select a data set from the Cambridge Haggle data [38]. In this data set, connections among 78 mobile iMote Bluetooth nodes carried by researchers and additional 20 stationary nodes are recorded over four days during IEEE Infocom 2006. In our simulations, we only consider the 78 mobile nodes and the first half of the period in the tracing data. We divide the time period of the tracing data into 50 time slots. For each time slot t, if there is a contact trace which is overlapping with this slot, we add a spacial link between the two corresponding nodes in G^t. For each round of simulation, we extract a slice of the network that contains 10 mobile nodes. Costs are again randomly generated from 1 to 5 for both spacial and temporal links. Topology control algorithms are performed over these 10-node DTNs. In our simulation, 13 rounds of simulations are conducted and the average measurements are plotted in Fig. 12. The same conclusions can be drawn from these results: 1) all algorithms can reduce the cost remarkably (more than 95 percent); 2) ULCP uses the largest cost among three methods, while GrdLCP has the best performance in practice.

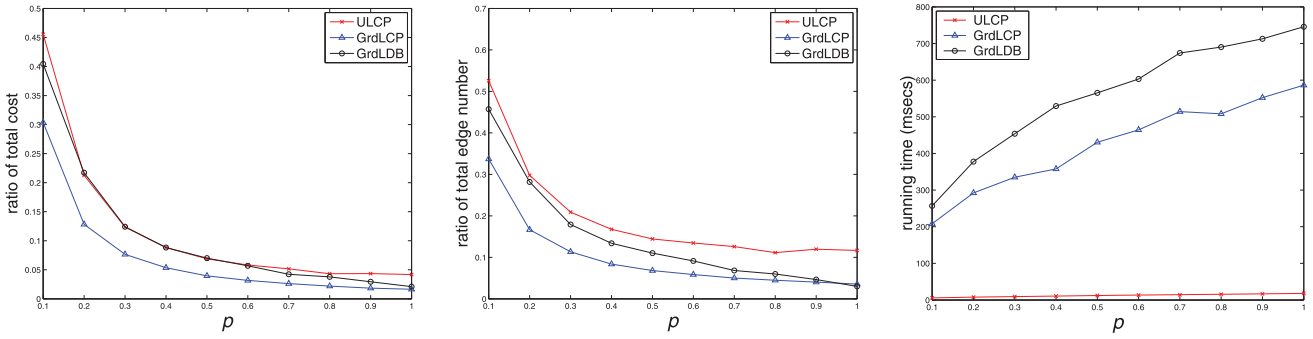
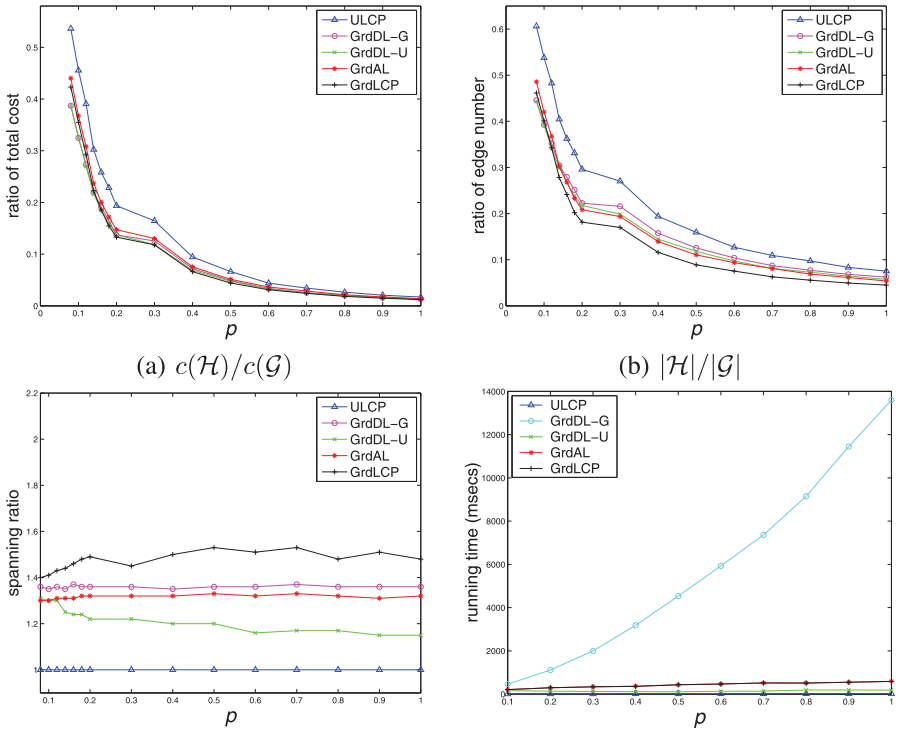


Fig. 11. Simulation results of TC on random networks with different densities. The cost (or edge number) of $\mathcal{H}$ in (a) or (b) is divided by the cost (or edge number) of $\mathcal{G}$, which illustrates how much saving is achieved by the topology control algorithm, compared with the original network without topology control. (c) The running time shows how fast each algorithm runs in average over the space-time graph.



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

(c) Spanning Ratio (d) Running Time

Fig. 13. Simulation results of CETC on random networks with different densities and $\delta = 1.4$. (a)-(b): The cost (or edge number) of $\mathcal{H}$ is divided by the cost (or edge number) of $\mathcal{G}$. (c): The spanning ratio shows the path efficiency of $\mathcal{H}$ compared with $\mathcal{G}$. (d) The running time shows how fast an algorithm runs.

7.2 Simulations for CETC Problem

7.2.1 Simulations on Random Networks

We perform two sets of simulations of CETC on the same random graphs.

For the first set of simulations, we increase the network density by raising p from 0.1 to 1.0, and keep spanning ratio requirement δ at 1.4. Figs. 13a and 13b show the ratios between the total cost/number-edges of the generated graph $\mathcal{H}$ and that of the original graph $\mathcal{G}$ when p increases. From the results, all topology control algorithms can significantly reduce the cost of maintaining the connectivity. Overall, ULCP has the largest cost and edge number, while GrdLCP usually has the least. Fig. 13c shows the spanning ratios of all methods. Clearly, GrdLCP is the only algorithm that cannot satisfy the spanning ratio requirement $\delta = 1.4$. It is an algorithm solely for maintaining the connectivity over time. ULCP can always achieve a spanning ratio of 1 but with higher cost than other algorithms because it keeps all edges in least cost paths. The other three algorithms have very close performance in total cost. GrdDL-U has lower spanning ratio than GrdDL-G and GrdAL, yet they are all bounded by δ. Fig. 13d shows the running time of every algorithm. Again, ULCP is the fastest algorithm but it cannot satisfy the spanning ratio and has the highest cost. It is interesting to see that GrdDL-G uses significantly longer time than all of the other algorithms. This is mainly because to iterate almost all links in decreasing order of costs until the spanning ratio requirement cannot be satisfied. To overcome this problem, using the output of ULCP as the input (as GrdDL-U) can significantly reduce the running time because ULCP has much less links than $\mathcal{G}$.

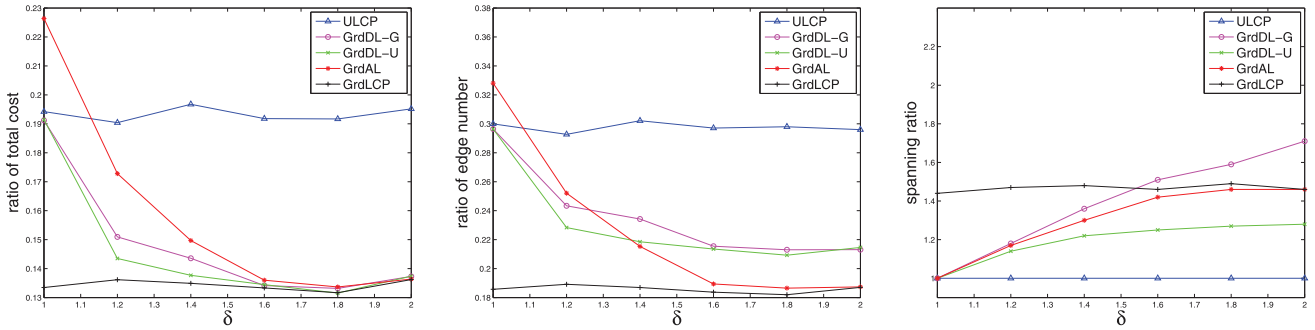
In the second set of simulations, we fixed the network density $p = 0.2$ and run our algorithms with different spanning ratio requirement δ increasing from 1.0 to 2.0. From the results shown in Fig. 14, we observe that tighter spanning ratio requirement results in higher cost and larger number of edge-usage of our topology algorithms. When the spanning ratio requirement becomes stronger, the restriction on removing links becomes weaker. In the extreme case with infinite δ, the CETC problem converges to TC that only preserves the connectivity. Notice that performances of GrdLCP and ULCP are not affected by the spanning ratio requirement.

7.2.2 Simulations on Tracing Data

We also apply our algorithms on the 13 networks retrieved from the trace data set with spanning ratio requirement from 1.0 to 2.0. The average measurements are plotted in Fig. 15. The same conclusions can be drawn from these results: all algorithms reduce the cost remarkably (more than 70 percent) and maintain the connectivity, while our proposed methods can guarantee the spanning ratio.

8 Conclusion

We study the topology control problem in a predictable time-evolving DTN modeled by a space-time graph. The newly defined TC problem is different from the standard space-time routing [24], [25], which only aims to find the most cost-efficient space-time path for a pair of source and destination. In contrast, our TC problem aims to maintain



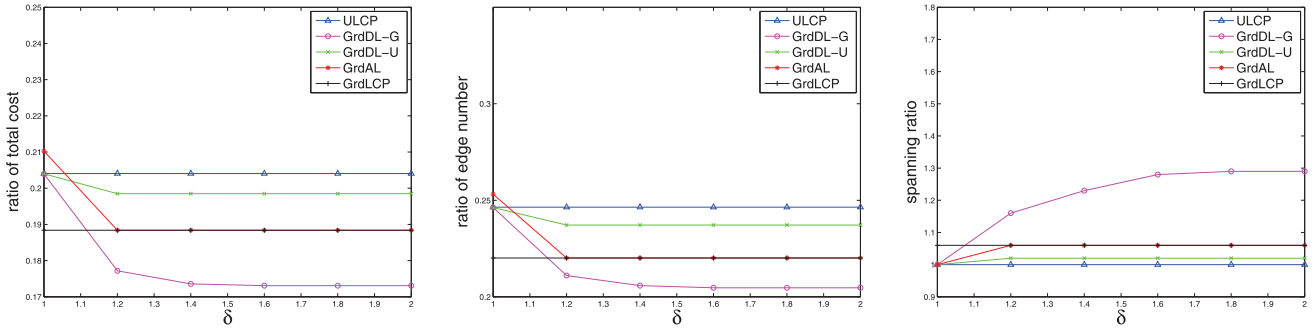
(a) $c(\mathcal{H})/c(\mathcal{G})$ (b) $|\mathcal{H}|/|\mathcal{G}|$ (c) Spanning Ratio

Fig. 14. Simulation results of CETC on random networks with fixed network density $p = 0.2$ and varying spanner ratios.

both cost-efficient and connected space-time routing topology for supporting DTN transmissions between all pairs of nodes. We first prove that this problem is NP-hard, and then propose several greedy-based methods that can significantly reduce the cost of topology while maintaining the connectivity (and power efficiency) over time. We also study another TC problem (CETC) where the least cost path for any two nodes needs to be cost-efficient. Simulation results from random networks and real-world tracing data demonstrate the efficiency of our methods. We believe that this paper presents the first step in exploiting topology control for time-evolving DTNs.

Discussions. The TC problem we study here and our proposed algorithms have several limitations and weakness:

- In our TC problem, we only consider the connectivity from the first time slot 0 to the last time slot T, i.e., packets are generated at time 0. In reality, if a packet arrives in the middle of the period T, it may not be able to reach the destination at the end of T via the constructed topology. However, in many time-evolving DTNs (such as the interplanetary space DTN [19] or the cyclic mobispace [23]), the mobility process of network is periodic and the routing is repeated. Thus, the delivery of packets is guaranteed in such cases. If we want the packet generated at any time be able to reach the destination at the end of T (which is a very strong requirement and the underlying original space-time graph needs to be very dense especially for the last few time slots), we can naively run the proposed methods for every possible starting time. This will roughly add a factor of T to the complexity. However, such a problem is actually much easier to solve than the TC problem we studied here. Since every node in each slot needs a space-time path to any node in the last time slot, this problem can be converted to a minimum spanning tree problem, where an optimal solution can be easily found. We can first consider the connectivity in the last time slot. Since the packets arrived at the beginning of the last time slot need to arrive at their destination, all links are needed in the last time slot (i.e., a complete graph). Then, for the second-to-last time slot, each node only needs one link to connect to any node (could be a temporal link connecting to itself) and we can pick the link with the least cost. This process continues for all time slots until the structure reaches the nodes at the beginning of the first time slot. This is like the Prim algorithm starting from a virtual node at the last time slot and spanning all nodes over time. In the end, an optimal solution can be found.
- Several proposed algorithms have quite high complexities. With the increase of the number of nodes n and the length of time period T, they may become slow. This limits the application of these algorithms in large-scale DTNs. However, due to the hardness of the problem and the complexity of space-time graph itself (with $O(nT)$ nodes and possible $O(n^2T)$ links), it is challenging to find efficient algorithms to achieve good performance. Dynamic programming could be a solution if the intermediate results can be



(a) $c(\mathcal{H})/c(\mathcal{G})$ (b) $|\mathcal{H}|/|\mathcal{G}|$ (c) Spanning Ratio

Fig. 15. Simulation results of CETC on networks from Cambridge Haggle [38] tracing data. Results are averages over 13 small networks.

used for speeding up the process. However, it is hard (at least for us) to define the recursive relationship between a solution of the problem and a solution of its subproblems. Nonetheless, we do believe there should be ways to speed up the proposed algorithms by smartly implementing or revising them (e.g., reusing the intermediate least cost path information). We leave such improvement as our future work.

- Our TC problems are defined for time-evolving and predictable DTNs and we assume that the predication of future links are *perfect* and links are *reliable* for communications, i.e., the packet delivery over these spacial or temporal links is guaranteed and without errors. This may be true for certain type of DTNs. However, it is clearly too optimistic in general DTNs. In practice, the wireless communications are unreliable due to the lossy nature of wireless channels. In addition, even though certain type of network mobility could be predicated based on user behavior or historical data, sometimes the link predication could be inaccurate. Therefore, it will be more interesting to study the TC problems without such strong assumptions on reliable links and perfect predication. One possible way to do so is assuming that each link in the space-time graph has a probability to reflect its "reliability," either based on link quality estimation or mobility predication. The larger the reliable probability of a link, the higher chance the DTN transmissions over that link will be successful. Then, a new version TC problem can be defined by introducing the reliability of DTN routing over the topology. Clearly, the new TC problem with reliable constraint becomes more complex and challenging. We have obtained some preliminary results [47] and leave the complete study of such a problem as our future work.
- In this paper, we mainly consider the topology control problems for connectivity over time. However, as in all previous topology control protocols, removing links may hurt network performance. In our simulations presented here, we assume dynamic topology (original space-time graph) is known a priori and every link is reliable. Thus, we did not measure how the network performance (such as routing performance) suffers due to topology control. Note that the CETC problem in Section 6 indeed considers the reduction of path quality due to TC (beyond pure connectivity). However, it is more interesting to see the tradeoff between the cost saved from TC and the reduction of routing performance due to TC in realistic settings. We leave such study as our future work.

To address the limitations listed above, we will continue our study on topology control for DTNs in the following directions:

1. Design more efficient algorithms with lower complexity to achieve connectivity over space-time graphs;
2. Investigate how to adapt the constructed structure to unexpected changes in the network such as node failures;
3. Study how to perform efficient topology control when the links are unreliable; and
4. Perform simulations over more realistic settings with different DTN routing protocols and evaluate the tradeoff between the cost saved from TC and the reduction of routing performance due to TC.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under Grant No. CNS-0915331 and CNS-1050398.

REFERENCES

[1] Y. Wang, H. Dang, and H. Wu, "A Survey on Analytic Studies of Delay-Tolerant Mobile Sensor Networks: Research Articles," *Wireless Comm. Mobile Computing,* vol. 7, no. 10, pp. 1197-1208, 2007.

[2] P. Juang et al., "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with Zebranet," *ACM SIGOPS Operating System Rev.,* vol. 36, no. 5, pp. 96-107, 2002.

[3] Q. Yuan, I. Cardei, and J. Wu, "Predict and Relay: An Efficient Routing in Disruption-Tolerant Networks," *Proc. ACM MobiHoc,* 2009.

[4] B. Burns, O. Brock, and B.N. Levine, "MV Routing and Capacity Building in Disruption Tolerant Networks," *Prof. IEEE INFOCOM,* 2005.

[5] W. Zhao, M. Ammar, and E. Zegura, "Controlling the Mobility of Multiple Data Transport Ferries in a Delay-Tolerant Network," *Proc. IEEE INFOCOM,* 2005.

[6] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," Technical Report CS-200006, Duke Univ., 2000.

[7] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," *Proc. ACM SIGCOMM Workshop Delay-Tolerant Networking (WDTN '05),* 2005.

[8] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic Routing in Intermittently Connected Networks," *ACM SIGMOBILE Mobile Computing and Comm. Rev.,* vol. 7, no. 3, pp. 19-20, 2003.

[9] J. Burgess et al., "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks," *Proc. IEEE INFOCOM,* 2006.

[10] J. Leguay, T. Friedman, and V. Conan, "Evaluating Mobility Pattern Space Routing for DTNs," *Proc. IEEE INFOCOM,* 2006.

[11] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble Rap: Social-Based Forwarding in Delay Tolerant Networks," *Proc. ACM MobiHoc,* 2008.

[12] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket Switched Networks and Human Mobility in Conference Environments," *Proc. ACM SIGCOMM Workshop Delay-Tolerant Networking (WDTN '05),* 2005.

[13] X. Zhang, J. Kurose, B.N. Levine, D. Towsley, and H. Zhang, "Study of a Bus-Based Disruption-Tolerant Network: Mobility Modeling and Impact on Routing," *Proc. ACM MobiCom,* 2007.

[14] M. Piorkowski, N. Sarafijanovoc-Djukic, and M. Grossglauser, "A Parsimonious Model of Mobile Partitioned Networks with Clustering," *Proc. First Int'l Conf. Comm. Systems and Networks and Workshops (COMSNETS '09),* 2009.

[15] A. Chaintreau, P. Fraigniaud, and E. Lebhar, "Opportunistic Spatial Gossip over Mobile Social Networks," *Proc. ACM Workshop Online Social Networks (WOSN '08),* 2008.

[16] Z.-B. Dong, G.-J. Song, K.-Q. Xie, and J.-Y. Wang, "An Experimental Study of Large-Scale Mobile Social Network," *Prof. 18th Int'l Conf. World Wide Web (WWW '09),* 2009.

[17] X. Hong, M. Gerla, R. Bagrodia, T.J. Kwon, P. Estabrook, and G. Pei, "The Mars Sensor Network: Efficient, Energy Aware Communications," *Proc. IEEE Military Comm. Conf. Comm. Network-Centric Operations: Creating the Information Force (MILCOM '01),* 2001.

[18] NASA, "Disruption Tolerant Networking (DTN)," https://www.spacecomm.nasa.gov/spacecomm/programs/technology/dtn/, 2013.

[19] S. Burleigh and A. Hooke, "Delay-Tolerant Networking: An Approach to Interplanetary Internet," *IEEE Comm. Magazine*, vol. 41, no. 6, pp. 128-136, June 2003.
[20] C. Caini et al., "Delay- and Disruption-Tolerant Networking (DTN): An Alternative Solution for Future Satellite Networking Applications," *Proc. IEEE*, vol. 99, no. 11, pp. 1980-1997, Nov. 2011.
[21] C. Caini et al., "A DTN Approach to Satellite Communications," *IEEE J. Selected Areas in Comm.*, vol. 26, no. 5, pp. 820-827, June 2008.
[22] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi, "Limits of Predictability in Human Mobility," *Science*, vol. 327, pp. 1018-1021, 2010.
[23] C. Liu and J. Wu, "Routing in a Cyclic Mobispace," *Proc. ACM MobiHoc*, 2008.
[24] B.B. Xuan, A. Ferreira, and A. Jarry, "Computing Shortest, Fastest, and Foremost Journeys in Dynamic Networks," *Int'l J. Foundations of Computer Science*, vol. 14, no. 2, pp. 267-285, 2003.
[25] S. Merugu et al., "Routing in Space and Time in Networks with Predictable Mobility," Technical Report GIT-CC-04-07, Georgia Inst. of Technology, 2004.
[26] J. Monteiro, A. Goldman, and A. Ferreira, "Performance Evaluation of Dynamic Networks Using an Evolving Graph Combinatorial Model," *Proc. IEEE Int'l Conf. Wireless and Mobile Computing, Networking and Comm. (WiMob '06)*, 2006.
[27] L. Arantes, A. Goldman, and M.V. dos Santos, "Using Evolving Graphs to Evaluate DTN Routing Protocols," *Proc. Extreme Workshop Comm. (ExtremeCom '09)*, 2009.
[28] X.-Y. Li, P.-J. Wan, and Y. Wang, "Power Efficient and Sparse Spanner for Wireless Ad Hoc Networks," *Proc. IEEE 10th Int'l Conf. Computer Comm. Networks (ICCCN '01)*, 2001.
[29] R. Rajaraman, "Topology Control and Routing in Ad Hoc Networks: A Survey," *ACM SIGACT News*, vol. 33, pp. 60-73, 2002.
[30] R. Wattenhofer et al., "Distributed Topology Control for Wireless Multihop Ad-Hoc Networks," *Proc. IEEE INFOCOM*, 2001.
[31] N. Li, J.C. Hou, and L. Sha, "Design and Analysis of a MST-Based Topology Control Algorithm," *Proc. IEEE INFOCOM*, 2003.
[32] Y. Wang and X.-Y. Li, "Localized Construction of Bounded Degree and Planar Spanner for Wireless Ad Hoc Networks," *Mobile Networks and Applications*, vol. 11, no. 2, pp. 161-175, 2006.
[33] F. Alagoz, "Energy Efficiency and Satellite Networking: A Holistic Overview," *Proc. IEEE*, vol. 99, no. 11, pp. 1954-1979, Nov. 2011.
[34] Y. Gu and T. He, "Dynamic Switching-Based Data Forwarding for Low-Duty-Cycle Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 10, no. 12, pp. 1741-1754, Dec. 2011.
[35] L. Chen et al., "Group-Based Discovery in Low-Duty-Cycle Mobile Sensor Networks," *Proc. IEEE Ninth Ann. Comm. Soc. Conf. Sensor, Mesh and Ad Hoc Comm. Networks (SECON '12)*, 2012.
[36] Y. Wang, H. Wu, F. Lin, and N.-F. Tzeng, "Cross-Layer Protocol Design and Optimization for Delay/Fault-Tolerant Mobile Sensor Networks," *IEEE J. Selected Areas in Comm.*, vol. 26, no. 5, pp. 809-819, June 2008.
[37] L. Zosin and S. Khuller, "On Directed Steiner Trees," *Proc. ACM/SIAM 13th Ann. Symp. Discrete Algorithms (SODA '02)*, 2002.
[38] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD Data Set Cambridge/Haggle (v. 2006-09-15)," http://crawdad.cs.dartmouth.edu/cambridge/haggle, Sept. 2006.
[39] M. Huang, S. Chen, Y. Zhu, B. Xu, and Y. Wang, "Topology Control for Time-Evolving and Predictable Delay-Tolerant Networks," *Proc. IEEE Eight Int'l Conf. Mobile Adhoc and Sensor Systems (MASS '11)*, 2011.
[40] A.D. Amis, R. Prakash, D. Huynh, and T. Vuong, "Max-Min D-Cluster Formation in Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM*, 2000.
[41] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE J. Selected Areas in Comm.*, vol. 15, no. 7, pp. 1265-1275, Sept. 1997.
[42] Y. Wang, W. Wang, and X.-Y. Li, "Efficient Distributed Low-Cost Backbone Formation for Wireless Networks," *Proc. ACM MobiHoc*, 2005.
[43] R.C. Shah et al., "Data MULEs: Modeling a Three-Tier Architecture for Sparse Sensor Networks," *Proc. IEEE Int'l Workshop Sensor Network Protocols and Applications (SNPA '03)*, 2003.
[44] M. Charikar and C. Chekuri, "Approximation Algorithms for Directed Steiner Problems," *J. Algorithms*, vol. 33, no. 1, pp. 73-91, 1999.
[45] B. Chandra, G. Das, G. Narasimhan, and J. Soares, "New Sparseness Results on Graph Spanners," *Proc. ACM Eighth Ann. Symp. Computational Geometry (SCG '92)*, 1992.
[46] C. Chekuri, G. Even, A. Gupta, and D. Segev, "Set Connectivity Problems in Undirected Graphs and the Directed Steiner Network Problem," *Proc. 19th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '08)*, 2008.
[47] M. Huang, S. Chen, F. Li, and Y. Wang, "Topology Design in Time-Evolving Delay-Tolerant Networks with Unreliable Links," *Proc. IEEE GlobeCom*, 2012.

Minsu Huang received the BEng degree in mechanical engineering from Central South University, in 2003 and the MS degree in computer science from Tsinghua University, in 2006. He is currently working toward the PhD degree in computer science at the University of North Carolina at Charlotte. His current research focuses on wireless networks, ad hoc and sensor networks, delay-tolerant networks, and algorithm design.

Siyuan Chen received the BS degree from Peking University, China, in 2006. He is currently working toward the PhD degree in computer science at the University of North Carolina at Charlotte. His current research focuses on wireless networks, ad hoc and sensor networks, delay-tolerant networks, and algorithm design.

Ying Zhu received the BEng degree in automatic control and the MEng degree in pattern recognition from the University of Electronic Science and Technology of China in 2002 and 2005, respectively. She is currently working toward the PhD degree computer science at the the University of North Carolina at Charlotte. Her current research focuses on wireless networks, ad hoc and sensor networks, delay-tolerant networks, and algorithm design.

Yu Wang received the BEng degree in 1998 and the MEng degree in 2000 in computer science from Tsinghua University, China. He received the PhD degree in computer science from the Illinois Institute of Technology in 2004. He is an associate professor of computer science at the University of North Carolina at Charlotte. His research interest includes wireless networks, ad hoc and sensor networks, delay-tolerant networks, mobile computing, and algorithm design. He has published more than 100 papers. He received the Ralph E. Powe Junior Faculty Enhancement Award from Oak Ridge Associated Universities in 2006. He is a senior member of the ACM, the IEEE, and the IEEE Communications Society.