2011 Eighth IEEE International Conference on Mobile Ad-Hoc and Sensor Systems

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

Minsu Huang* Siyuan Chen* Ying Zhu* Bin Xu† Yu Wang*
*Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA
†Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China

***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 large number of wireless devices' participation, how to maintain efficient and dynamic topology of the DTN becomes crucial. 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 which 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 then propose two greedy-based methods which can significant reduce the total cost of topology while maintain the connectivity over time. Finally, we illustrate how the proposed methods can work for undirected DTNs. Simulations have been conducted on both random DTN networks and real-world DTN tracing data. Results demonstrate the efficiency of the proposed topology control methods.**

I. Introduction

Delay or disruption tolerant networks (DTNs) [1], [2] recently have drawn much attention from networking researchers since they have 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]–[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. 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]–[19]), there are clear socio-temporal patterns for both individual components and network structure. For *certain* type of networks, the temporal characteristics of topology could be known a priori or can be predicted from historical tracing data. For example, it is easy to discovery the temporal pattern of topology for a delay tolerant network formed by public buses [13] or satellites [19] which have fixed tours and schedules, or a mobile social network consisting of students who share fixed class schedules. Recent study [20] also shows that human mobility model can achieve a 93% 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. Figure 1 illustrates an example of such time-evolving DTN. 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 [21]–[25]. 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 applications, network topology can be controlled under different objectives (such as power efficiency, fault tolerance, and throughput maximization). Topology control has been well studied in ad hoc and sensor networks [27]–[32]. The focus of previous research is mainly on how to construct a power efficient structure from a static and connected topology. However, in DTNs, the underlying topology is lack of continuous connectivity which makes existing topology control algorithms useless. Therefore, how to maintain efficient and dynamic topology of DTNs

This work is supported in part by the US National Science Foundation (NSF) under Grant No. CNS-0721666, CNS-0915331, and CNS-1050398, and by Tsinghua National Laboratory for Information Science and Technology (TNList).

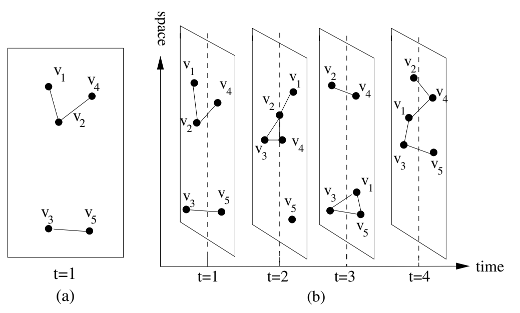


Fig. 1. **A time-evolving DTN:** (a) a snapshot of the network, (b) time-evolving topologies of the DTN (a sequence of snapshots).

becomes crucial, especially with large number of wireless devices' participation.

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 the 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 which aims to build a sparser structure (also a space-time graph) 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. Notice that in some *time-evolving* and *predictable* DTNs (such as the interplanetary space DTN [19]), maintaining dense structure is too expensive. We also show that this new topology control problem is a NP-hard problem, by connecting it with *directed Steiner tree problem* [33].
- We propose two greedy-based methods which can significant 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 topology control problem.
- We also discuss how to address the topology control problem in undirected DTNs using the proposed greedy methods. Basically, we present two methods to convert an undirected DTN to a directed DTN in the format of space-time graphs.
- Simulations have been conducted on both random DTN networks and real-world DTN tracing data [34]. Results demonstrate the efficiency of all proposed methods.

The rest of this paper is organized as follows. In Section II, we summarize related works in topology control and DTNs. In Section III, we formally define the space-time graph and the topology control problem. Two topology control algorithms for time-evolving DTNs are proposed in Section IV. Then, topology control for undirected DTNs are discussed in Section V. Section VI presents the simulation results of all proposed algorithms. Finally, Section VII concludes the paper by pointing out some possible future directions.

II. Related Works

A. 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 [27]–[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]–[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 [35]–[37], a hierarchical structure is built using clustering formation as the virtual backbone for routing and relaying packets for the network. All these topology control protocols deal with topology changes by re-performing 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.

B. Routing in Delay Tolerant Networks

Existing research in DTNs mainly focuses on routing [3]–[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], [38], 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 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]–[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 [21], [39] have proposed to use estimated expected minimum delay as a delivery probability metric in DTNs with repetitive mobility. Liu and Wu [40] also proposed a forwarding method applying a probabilistic forwarding metric derived by modeling each forwarding as an optimal stopping rule problem.

C. Modeling Time-Evolving Networks

How to model the time-evolving networks has been studied in both mobile ad hoc networks [22]–[24] and delay tolerant networks [21], [25]. Xuan *et al.* [22] first study the 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, [24], [25] also use evolving graphs to evaluate various ad hoc and DTN routing protocols. Shashidhar *et al.* [23] study the routing problem in dynamic networks as well. They use a *space-time graph* to model the network. The space-time graph is a directed graph which 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. In [21], Liu and Wu also model a cyclic mobispace as a probabilistic space-time graph in which an edge between two nodes contains a set of discretized probabilistic contacts. All of these work only focus 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.

For our best knowledge, there is no previous results on topology control in DTNs except for our recent work [26], which studies how to build time-evolving structures preserving spanner properties. This paper and [26] are the first attempts 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.

III. Model and The Problem

A. 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 co-evolve over time (as shown in the example of Figure 1), traditional static graph model can not represent such evolution. Thus, a sequence of static graphs is needed to model the time-evolving DTN. As shown in Figure 1(b), 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 [23] to model such a dynamic DTN.

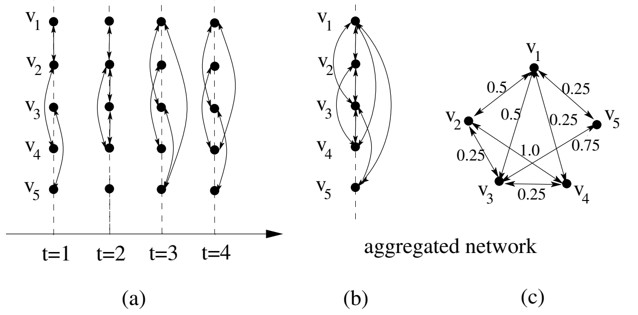


Fig. 2. **Different graph models for the time-evolving network shown in Figure 1:** (a) a sequence of snapshots of the network at each time slot, denoted by $\{G^t\}$; (b) and (c) its corresponding aggregated graph model.

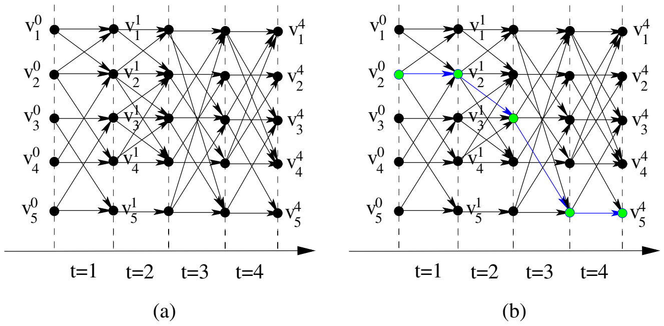


Fig. 3. Space-time graph model from the same time-evolving network in Figure 2: (a) the corresponding space-time graph $\mathcal{G}$; (b) a space-time path (in blue) from the source v_2 to the destination v_5.

Assume that the time is divided into discrete and equal time slots, such as $\{1, \cdots, T\}$. Let $V = \{v_1, \cdots, v_n\}$ 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 series of directed graphs $\{G^t | t = 1 \cdots T\}$. Figure 2(a) shows the sequence of snapshots of the network in Figure 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 Figure 2(b) and Figure 2(c) 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 forwarding nodes. However, such 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 $\{G^t|t = 1\cdots T\}$ into a space-time graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is a directed graph defined in both spacial and temporal space. See Figure 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 $\mathcal{V} = \{v_j^t | j = 1, \cdots, n \textit{ and } t = 0, \cdots, T\}$ 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 Figure 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 Figure 3(b), 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 4 time slots. v_2 holds the packet for the first time slot, then passes it to v_3 at $t = 2$, etc.

Space-time graph model captures both the space and time dimensions of the network topology. It 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 costs of all links in $\mathcal{G}$, i.e., $c(\mathcal{G}) = \sum_{e\in\mathcal{G}} c(e)$. Given the cost of links, we can also define the shortest path $P(u, v)$ as the least cost path from u to v in $\mathcal{G}$. The total cost of such path is denoted by $c(u, v)$, which is the summation of costs of all links in path $P(u, v)$.

B. New Topology Control Problem

We now can define the *topology control problem* (TC) on space-time graphs. The aim of topology control is constructing 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]), maintaining dense structure is too expensive, thus a connected subgraph is more cost efficient. Here *connectivity* has a *different* definition from that of a static graph. We define that a space-time graph $\mathcal{H}$ is *connected* 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

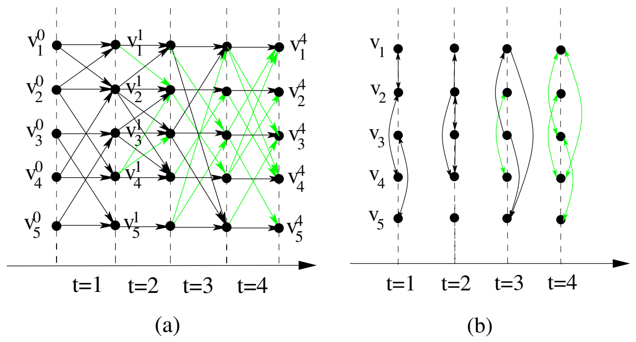


Fig. 4. Topology control on time-evolving network (the one shown in Figure 1): (a) a new connected subgraph $\mathcal{H}$ of $\mathcal{G}$ (green links are removed links from Figure 3(a)); (b) its corresponding sequence of static graph with less links than the one of Figure 2(a).

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. Figure 4 shows an example of topology control on the space-time graph in Figure 3(a). 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, since 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.

Notice that the newly defined topology control problem is different with the standard space-time routing [22], [23], which aims to find the most cost-efficient space-time path for a pair of source and destination. The topology control 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, studying the tradeoff between topology control and cost-efficient routing is one of our future work. In addition, we only consider the connectivity from the first time slot 0 to the last time slot T, thus we assume that 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 in the end of T. However, in many *time-evolving* and *predictable* DTNs (such as the interplanetary space DTN [19] or the cyclic mobispace [21]), the mobility process of network is periodic and the routing is repeated. Thus, the delivery of packets is still guaranteed in such case.

The topology control problem for 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, simply applying the spanning tree in each snapshot is not a solution, since 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, since it connects each node in every snapshot, which is not necessary

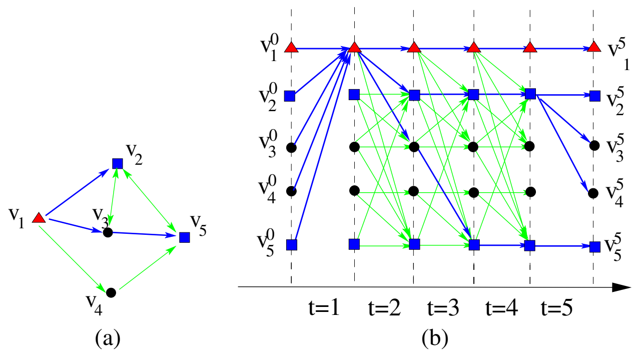


Fig. 5. Reduction from (a) the directed Steiner tree problem to (b) our TC problem on space-time graphs. Here the blue structures are the corresponding solutions in DST and TC.

and a waste of many links. Such method is neither optimal nor even near optimal.

We now prove the NP-hardness of our TC problem on space-time graphs by using a reduction from the *directed Steiner tree* (DST) problem [33], which is a known NP-hard problem. The DST problem is the 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 node 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 show how to reduce the DST problem into the TC problem on space-time graphs. Given an instance of DST with graph $G = (V, E)$, the set of nodes $Q = \{v_{q_1}, v_{q_2}, \cdots, v_{q_m}\}$ need to reach and the root v_r, we can construct an instance of TC problem on a space-time graph $\mathcal{G}$ as follows. First, all nodes $v_1, \cdots, v_n \in V$ will be the nodes in each snapshot in the space-time $\mathcal{G}$. Assume that D is the longest hop count of a directed path (no loop) from v_r to other nodes, thus $D < n$. The constructed space-time $\mathcal{G}$ has $D + 2$ time slots, i.e., $D + 3$ layers of nodes. For the first time slot ($t = 1$), all nodes v_i^0 are connected to v_r^1 with cost 0. 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 last time slot have cost of 0. For each time slot except for the first and the last, both temporal links and spatial links are added based on G. All temporal links have cost of 0, while spatial links have the same costs as those 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. Figure 5 shows an example with v_1 as the root and $Q = \{v_2, v_5\}$. The blue structures are one set of solutions. Since the construction of $\mathcal{G}$ can be done in polynomial time and DST problem is NP-hard, the TC on space-time graphs is also NP-hard. ■

IV. Topology Control for Directed Delay-Tolerant Networks

Since TC over space-time graphs is NP-hard, in this section, we propose two efficient heuristics to construct a sparse structure that fulfills the connectivity requirement over a space-time graph. Both algorithms are based on greedy algorithms, one has theoretical bound on its performance while the other one is practically more efficient.

A. Greedy Algorithm based on Least Cost Path

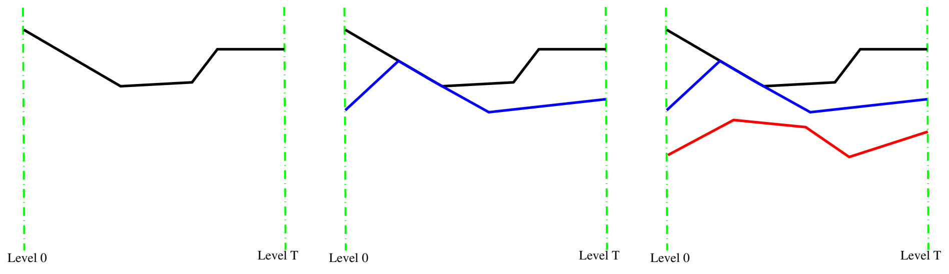
One naive method for maintaining the network connectivity is to take the union of several shortest path trees T_i rooted at each node v_i^0 at the initial snapshot. Each tree T_i is the union of shortest paths from v_i^0 to v_j^T for $j = 1, \cdots, n$. Obviously, the resulting graph still keeps the network connectivity. The time complexity of this method is $n \times O(|\mathcal{E}| + |\mathcal{V}| \log(|\mathcal{V}|)) = O(Tn^3 + Tn^2 \log(Tn))$ since n times of Dijkstra's algorithm are running on the space-time graph with $(T + 1)n$ nodes and at most Tn^2 edges. Hereafter, we refer this method as *shortest path tree method* (SPT).

Algorithm 1 Greedy Algorithm based on Least Cost Path

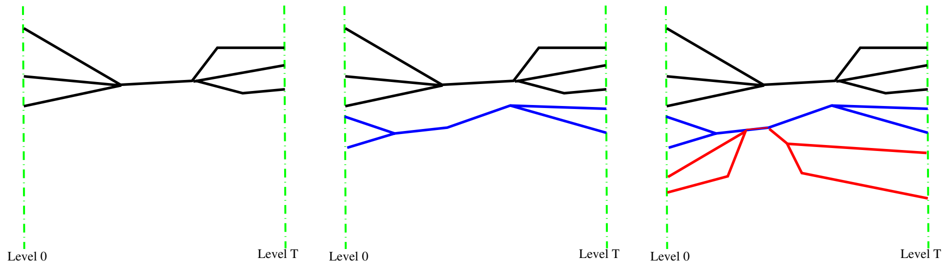
```
1: $\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^T)\}$ for all $i$ and $j$ in $[1, n]$.
2: while $X \neq \phi$ do
3:     find the least cost paths for every pair nodes in $X$, and
       assume $P(v_i^0, v_j^T)$ has the least cost among these paths.
4:     if $e \in P(v_i^0, v_j^T)$ then
5: 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

However, the structure built by the shortest path tree method may contain more links than necessary. Therefore, we propose a new greedy algorithm (as shown in Algorithm 1) to further improve the performance. The basic idea is quite simple and as follows. Initially, we need to connect n^2 pair of nodes in X. In each round we pick the least cost path between a pair nodes in X which is the minimum among all least cost paths connecting any pair of nodes in X as shown in Figure 6(a). 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 Figure 6(a)-(c). After n^2 rounds, all pair nodes (v_i^0, v_j^T) are guaranteed to be connected by paths in $\mathcal{H}$. It is clear that the output of this method is much sparser than the one of shortest path tree method. We refer this method as *greedy method based on least cost path* (GrdLCP). The time complexity of this algorithm is $O(Tn^5 + Tn^4 \log(Tn))$ since in each round n times of Dijkstra's algorithm are running on the space-time graph and there are n^2 rounds.

Notice that both SPT and GrdLCP may not lead to the optimal solution. Figure 7 shows such an example: a network with two nodes (v_a and v_b) and two time slots. Both SPT and GrdLCP generate a structure with total cost 22 as shown in

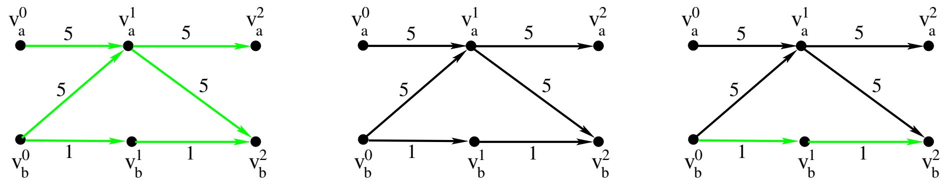


(a) Step 1 - GrdLCP (b) Step 2 - GrdLCP (c) Step 3 - GrdLCP



(d) Step 1 - GrdLDB (e) Step 2 - GrdLDB (f) Step 3 - GrdLDB

Fig. 6. Illustrations of Algorithm 1 and Algorithm 2: (a-c) Algorithm 1 repeatedly adds one least cost path into the topology to connect one pair of nodes in X. (d-f) Algorithm 2 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.



(a) original space-time graph (b) output of greedy algorithm (c) optimal solution of TC

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

Figure 7(b) (since 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 (Figure 7(c)). However, GrdLCP and SPT can perform well when the network is much denser and with more nodes. Our simulations in Section VI confirm this.

B. Greedy Algorithm based on Least Density Bunch

Next, we present another more complex greedy algorithm (as shown in Algorithm 2) which is inspired by a method proposed by Charikar *et al.* [41] for *directed generalized Steiner network* (DGSN) problem [42]. 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 u_i to v_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 [42]. However, their method is very complex.

The main idea of our greedy algorithm (Algorithm 2) 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 Figure 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 which represents the shortest path $P(u, v)$ from u to v in $\mathcal{G}$. The cost of $P(u, v)$ is $c(u, v)$. We use $s[u_i, w_i]$ to represent the cost of $P(u_i, p)$ plus $P(q, w_i)$. Thus, the total cost of bunch B which connects l-pair of nodes in X is $c(B) = c(p, q) + s[u_1, w_1] + s[u_2, w_2] + \cdots + s[u_l, w_l]$. 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 is 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. Figure 6(d-f) show the procedure. We refer this method as *greedy method based on least density bunch* (GrdLDB).

The time complexity of this algorithm is roughly $O(T^2 n^6 \log n)$, since the outer while-loop runs n^2 times in the worst case; the outer for-loop runs $O(T^2 n^2)$ times; and

Algorithm 2 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:       $s[v_i^0, v_j^T] \leftarrow c(v_i^0, p) + c(q, v_j^T)$.
7:     end for
8:     sort all $s[v_i^0, v_j^T]$ in increasing order of $s$, and let $(u_l, w_l)$ refer to the $l$th pair in this sorted list.
9:     for $l$ going from 1 to $k$ do
10:      $C \leftarrow c(p,q)+s[u_1,w_1]+s[u_2,w_2]+\cdots+s[u_l,w_l]$.
11:      if $C/l \leq d$ then
12:        $d \leftarrow C/l$; $k1 = l$;
13:        $B \leftarrow P(p,q) + P(u_1,p) + P(q,w_1) + \cdots + P(u_l,p) + P(q,w_l)$.
14:      end if
15:    end for
16:  end for
17:  $\mathcal{H} \leftarrow \mathcal{H} + B$; $k = k - k1$;
18:  $X \leftarrow X - \{(u_1,w_1), \cdots, (u_{k1}, w_{k1})\}$.
19: end while
20: return $\mathcal{H}$
```

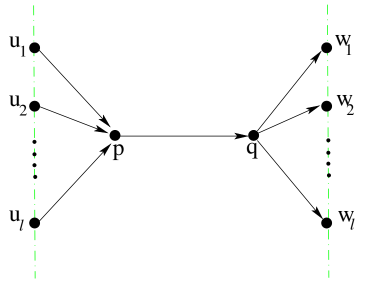


Fig. 8. Illustration of a bunch connecting l pairs of nodes. Here, each arrow represents a directed shortest path. The total cost of this bunch is $c(B) = c(p,q) + s[u_1,w_1] + s[u_2,w_2] + \cdots + s[u_l,w_l]$.

the sorting can be done in $O(n^2 \log n)$. Notice that an all-to-all shortest path algorithm need to be performed once to prepare $c(u,v)$ for all nodes u and v in the beginning of the algorithm. However, time needed for such algorithm is much less than $O(T^2 n^6 \log n)$. Though with larger time complexity than GrdLCP and SPT, GrdLDB algorithm has a nice property in theory: it can achieve approximation-guarantee in term of the total cost compared with the optimal solution. In [41], 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 directed generalized Steiner network problem. Therefore, we have the following theorem for our topology control problem, since $k = n^2$.

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

Notice that Algorithm 2 will also lead to the non-optimal solution for the example shown in Figure 7.

V. Topology Control for Undirected Delay-Tolerant Networks

So far we consider a directed delay-tolerant network where the static graph in each snapshot $\overrightarrow{G^t}$ 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 the example in Figure 7) or $c(\overrightarrow{v_i^{t-1}v_j^t}) \neq c(\overrightarrow{v_j^{t-1}v_i^t})$ even they both exist. Similarly, the output of topology control algorithm can use just one of these two links in the resulting topology (as shown in Figure 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 $v_i^t v_j^t$ in G^t, it only has 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 just 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 for the static graph of G^t in each snapshot. To make the proposed algorithms work on the undirected delay-tolerant networks, we then propose two methods to convert the undirected network into a directed network.

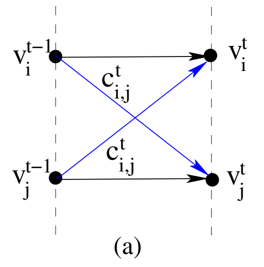
A. Converting Method One - Double Cost

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

Next, we prove an important lemma which can still 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, $c(S_B) \leq 2c(Opt_B)$. Notice 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 $c(S_A) \leq c(S_B)$ since the total cost of S_A only



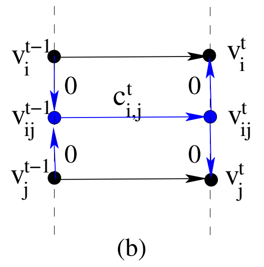


Fig. 9. Illustration of construction of spacial links in the space-time graph: (a) converting method one - doubling the cost; (b) converting method two - adding new nodes.

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, $c(Opt_A) \le c(S_A)$. Therefore, $c(Opt_A) \le c(S_A) \le c(S_B) \le 2c(Opt_B)$. ■

This lemma implies that Algorithm 2 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.

B. 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 a 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 $c(\overrightarrow{v_{ij}^{t-1}v_{ij}^t}) = c_{i,j}^t$. Figure 9(b) 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 on 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 introducing additional nodes and links which may increase the size of the input for our algorithms.

VI. Simulations

In this section, we will evaluate our proposed topology control algorithms, namely, *Greedy Algorithm with Least Cost Path* (GrdLCP) and *Greedy Algorithm with Least Density Bunch* (GrdLDB), by comparing their performances with *Shortest Path Tree* method (SPT). We implement all these three algorithms in a simulator developed by our group. The underlying time-evolving networks are either randomly generated from random graph model or directly extracted from Cambridge Haggle tracing data [34]. All the networks are directed DTNs, thus we only test our algorithms on directed networks in this section. In all simulations, we take two 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., $c(\mathcal{H}) = \sum_{e\in\mathcal{H}} c(e)$.

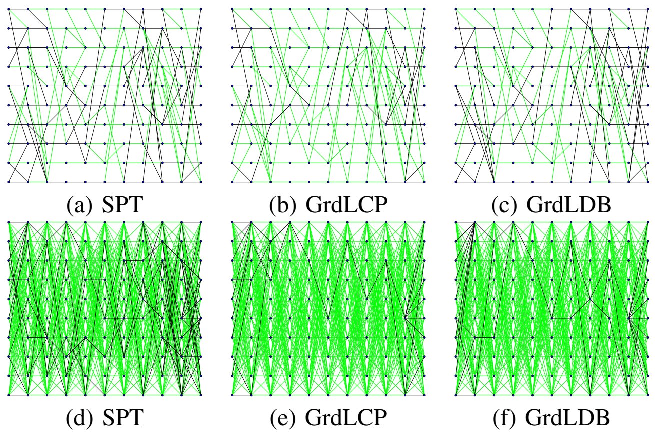


Fig. 10. Topologies generated over the same random network: green links are removed links from the original graph $\mathcal{G}$, while black links are the ones kept by each algorithm. (a-c): $p = 0.1$, (d-f): $p = 0.8$.

- **Total Number of Edges:** the total number of edges in the constructed topology $\mathcal{H}$, i.e., $|\mathcal{H}|$. Here, $|\mathcal{G}|$ denotes the number of edges of graph $\mathcal{G}$.

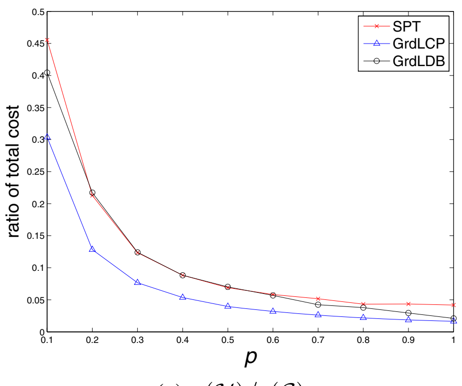
For all the simulations, we repeat the experiment for multiple times and report the average values of these metrics. It is clear that a desired topology should have small total cost and small number of edges.

A. Simulations on Random Networks

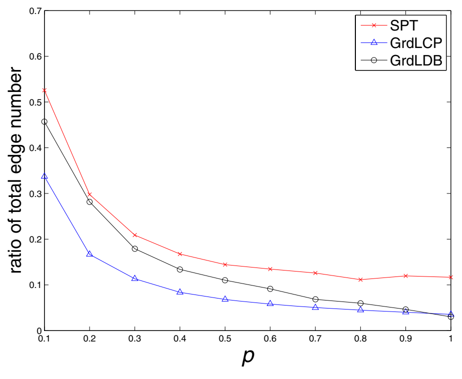
We first generate a sequence of static random graphs $\{G^t\}$ to represent the time-evolving DTN. Here, we first consider a DTN with 10 nodes (n=10) and spreading over 10 time slots ($T = 10$). For each time slot t, we randomly generate the graph G^t using the classical random graph generator. Basically, 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. After generating $\{G^t\}$, we convert it into its corresponding space-time graph $\mathcal{G}$ with $n(T+1)$ nodes. Then all three topology control algorithms (SPT, GrdLCP, GrdLDB) take the same $\mathcal{G}$ as the input.

Figure 10 shows the constructed topologies in space-time graph format from all three algorithms when $p = 0.1$ and 0.8. It is clear 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 rising 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 control among them. Figure 11 shows 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



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



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

Fig. 11. Simulation results on random networks with different density. The cost (or edge number) of $\mathcal{H}$ is divided by the cost (or edge number) of $\mathcal{G}$, which illustrates how much saving achieved by the topology control algorithm, compared with the original network without topology control.

implies how much saving achieved by the topology control algorithm, comparing to 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% cost and around 50% edges. For $p = 1.0$, more than 95% cost is saved. In most cases, GrdLDB and GrdLCP can achieve better efficiency than SPT. 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 topology control algorithms with dense networks.

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

B. 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 advantage, we also use real-world wireless tracing data to evaluate our topology control algorithms. Particularly,

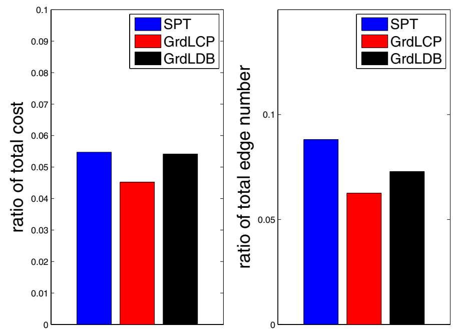


Fig. 12. Simulation results on networks from Cambridge Haggle [34] tracing data. Results are averages over 13 small networks.

we select a data set from the Cambridge Haggle data set [34] which is available at CRAWDAD [43]. In this data set, connections among 78 mobile iMote Bluetooth nodes carried by researchers and additional 20 stationary nodes are recorded over 4 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 which contains 10 mobile nodes. Costs are again randomly generated from 1 to 5 for both spacial and temporal links. Then topology control algorithms are performed over these 10-node DTNs. In our simulation, 13 rounds of simulations are conducted and average measurements are plotted in Figure 12. The same conclusions can be drawn from these results: (1) all algorithms can reduce the cost remarkably (more than 95%); (2) SPT has the largest cost among three methods, while GrdLCP has the best performance in practice.

VII. Conclusion

We study topology control problem in a predictable time-evolving DTN modeled by a space-time graph. We first prove that it is NP-hard, then propose two greedy-based methods which can significant reduce the cost of topology while maintain the connectivity over time. We also present two techniques which can convert undirected DTNs into directed DTNs such that our proposed methods can still work. Simulation results from random networks and real-world tracing data both demonstrate the efficiency of our methods. We believe that this paper presents the first step in exploiting topology control for time-evolving DTNs.

Some possible future works include the following problems: (1) to design more efficient algorithms with lower complexity to achieve connectivity over space-time graphs; (2) to investigate how to adapt the constructed structure to unexpected changes in the network such as node failures; (3) to study how to perform efficient topology control when the links are unreliable.

References

[1] Y. Wang, H. Dang, and H. Wu, "A survey on analytic studies of delay-tolerant mobile sensor networks: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 7, no. 10, pp. 1197–1208, 2007.

[2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," *SIGOPS Oper. Syst. Rev.*, 36(5):96–107, 2002.

[3] Q. Yuan, I. Cardei, and J. Wu, "Predict and relay: an efficient routing in disruption-tolerant networks," in *Proc of ACM Mobihoc*, 2009.

[4] B. Burns, O. Brock, and B.N. Levine, "MV routing and capacity building in disruption tolerant networks," in *Prof. of IEEE INFOCOM*, 2005.

[5] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *Proc. of IEEE INFOCOM*, 2005.

[6] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Technical Report CS-200006, 2000.

[7] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proc. of ACM SIGCOMM workshop on DTN*, 2005.

[8] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3)19–20, 2003.

[9] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *Proc. of IEEE INFOCOM*, 2006.

[10] J. Leguay, T. Friedman, and V. Conan, "Evaluating mobility pattern space routing for DTNs," in *Proc. of IEEE INFOCOM*, 2006.

[11] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *Proc. of 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," in *Proc. of ACM SIGCOMM workshop on DTN*, 2005.

[13] X. Zhang, J., B. N. Levine, D. Towsley, and H. Zhang, "Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing," in *Proc. of ACM MobiCom*, 2007.

[14] M. Piorkowski, N. Sarafijanovoc-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering," in *Proc. of Int'l Conf on Commun. Sys. and Networks*, 2009.

[15] A. Chaintreau, P. Fraigniaud, and E. Lebhar, "Opportunistic spatial gossip over mobile social networks," in *Proc. of ACM workshop on Online social networks*, 2008.

[16] Z.-B. Dong, G.-J. Song, K.-Q. Xie, and J.-Y. Wang, "An experimental study of large-scale mobile social network," in *Prof. of WWW*, 2009.

[17] X. Hong, M. Gerla, R. Bagrodia, T.J. Kwon, P. Estabrook, and G. Pei, "The Mars sensor network: efficient, energy aware communications," in *Proc. of IEEE MILCOM*, 2001.

[18] NASA Disruption tolerant networking (DTN) porject, https://www.spacecomm.nasa.gov/spacecomm/programs/technology/dtn/.

[19] S. Burleigh and A. Hooke, "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Commun. Mag.*, 41(6):128–136, 2003.

[20] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi, "Limits of predictability in human mobility," *Science*, vol. 327, pp. 1018–1021, 2010.

[21] C. Liu and J. Wu, "Routing in a cyclic mobispace," in *Proc. of ACM MobiHoc*, 2008.

[22] B.B. Xuan, A. Ferreira, and A. Jarry, "Computing shortest, fastest, and foremost journeys in dynamic networks," *Int'l J. of Foundations of Computer Science*, 14(2):267–285, 2003.

[23] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Georgia Institute of Technology, CC Technical Report, GIT-CC-04-07, 2004.

[24] J. Monteiro, A. Goldman, and A. Ferreira, "Performance evaluation of dynamic networks using an evolving graph combinatorial model," in *Proc. of IEEE WiMob*, 2006.

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

[26] M. Huang, S. Chen, Y. Zhu, and Y. Wang, "Cost-efficient topology design problem in time-evolving delay-tolerant networks," in *Proc. of IEEE Globecom*, 2010.

[27] X.-Y. Li, P.-J. Wan, and Y. Wang, "Power efficient and sparse spanner for wireless ad hoc networks," in *Proc. of IEEE ICCCN*, 2001.

[28] R. Rajaraman, "Topology control and routing in ad hoc networks: A survey," *SIGACT News*, 33:60–73, 2002.

[29] R. Ramanathan and R. Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *Proc. of IEEE INFOCOM*, 2000.

[30] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "Distributed topology control for wireless multihop ad-hoc networks," in *Prof. of IEEE INFOCOM*, 2001.

[31] N. Li, J.C. Hou, and L. Sha, "Design and analysis of a MST-based topology control algorithm," in *Proc. of 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 Appli*, 11(2):161–175, 2006. A shorter version in *Proc. of ACM DIALM-POMC*, 2003.

[33] L. Zosin and S. Khuller, "On directed Steiner trees," in *Proc. of ACM-SIAM SODA*, 2002.

[34] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2006-09-15)," Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, Sept. 2006.

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

[36] C.R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE J. of Sel. Areas in Commun.*, 15(7):1265–1275, 1997.

[37] Y. Wang, W. Wang, and X.-Y. Li, "Efficient distributed low-cost backbone formation for wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, 17(7):681–693, 2006. A shorter version in *Proc. of ACM MobiHoc*, 2005.

[38] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: modeling a three-tier architecture for sparse sensor networks," in *Proc. of IEEE SNPA Workshop*, 2003.

[39] C. Liu and J. Wu, "Scalable routing in delay tolerant networks," in *Proc. of ACM MobiHoc*, 2007.

[40] C. Liu and J. Wu, "An optimal probabilistic forwarding protocolin delay tolerant networks," in *Proc. of ACM MobiHoc*, 2009.

[41] M. Charikar and C. Chekuri, "Approximation algorithms for directed steiner problems," *J. Algorithms*, 33(1):73–91, 1999.

[42] C. Chekuri, G. Even, A. Gupta, and D. Segev, "Set connectivity problems in undirected graphs and the directed steiner network problem," in *Proc. of ACM SODA*, 2008.

[43] J. Yeo, D. Kotz, and T. Henderson, "Crawdad: a community resource for archiving wireless data at dartmouth," *SIGCOMM Comput. Commun. Rev.*, 36(2):21–22, 2006.