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
We propose two greedy-based methods which 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 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 nodes and the topology co-evolve over time (as shown in the example of Figure 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 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.

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 \( v_i v_j \in E^t \) 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 \( v_1 v_4 \) 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 \(v_4v_1\) only exists in the last time slot when link \(v_1v_2\) 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 \(G = (V,E)\), which is a directed graph defined in both spacial and temporal space. See Figure 3 for illustration. In the space-time graph \(G\), \(T+1\) layers of nodes are defined and each layer has \(n\) nodes, thus the vertex set \(V = \{v_{ij}|j = 1, \ldots, n \text{ and } t = 0, \ldots, T\}\) and there are \(n(T+1)\) nodes in \(G\), i.e., \(|V| = n(T+1)\). Two kinds of links (spatial links and temporal links) are added between consecutive layers in \(E\). The space between consecutive layers is a time slot. A temporal link \(v_{ij}^{t-1}v_{jk}^t\) (those horizontal links in spacial Figure 3) connects the same node \(v_j\) across consecutive \((t-1)\)th and \(t\)th layers, which represents the node carrying the message in the \(t\)th time slot. A spatial link \(v_{ij}^{t-1}v_{ij}^t\) (those horizontal links in spacial Figure 3) connects the same node \(v_j\) across consecutive \((t-1)\)th and \(t\)th layers, which represents the node carrying the message in the \(t\)th time slot. A spatial link \(v_{ij}^{t-1}v_{ij}^t\) (those horizontal links in spacial Figure 3) represents forwarding a message from one node \(v_j\) to its neighbor \(v_k\) in the \(t\)th time slot (i.e., \(v_jv_k^t \in E^t\)). By defining the space-time network, any communication operation in the time-evolving network can be simulated on this directed graph. As shown in Figure 3(b), a path from \(v_2^1\) to \(v_2^2\) shows a particular routing strategy to deliver the packet from \(v_2^1\) to \(v_2^2\) in the network using 4 time slots. \(v_2^2\) holds the packet for the first time slot, then passes it to \(v_3^2\) 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 \(e \in E\) there is a cost \(c(e)\), which is the energy cost associated with transmitting 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 \(c(G)\) is the summation of costs of all links in \(G\), i.e., \(c(G) = \sum_{e \in G} c(e)\). 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 \(G\). 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 \(H\), which is a subgraph of the original space-time graph \(G\), such that (1) \(H\) is still connected over the time period \(T\); and (2) the total cost of \(H\) 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 \(H\) 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 connected space-time graph does not require connectivity in each snapshot. Hereafter, we assume that the original space-time graph \(G\) 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.
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 \( T \) rooted at \( v_r \), such that all nodes in \( Q \) are included in \( T \).

**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 \( Q = \{v_{q_1}, v_{q_2}, \ldots, v_{q_m}\} \) need to reach and the root \( v_r \), we can construct an instance of TC problem on a space-time graph \( G \) as follows. First, all nodes \( v_1, \ldots, v_n \in V \) will be the nodes in each snapshot in the space-time \( G \). 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 graph \( G \) 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 \( v_{i}^{1} \) 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 \( v_{q_i}^{D+2} \) 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 \( G \) 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 \( G \) 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 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 \( v_j^D \) for \( j = 1, \ldots, n \). Obviously, the resulting graph still keeps the network connectivity. The time complexity of this method is \( n \times O(|E| + |V| \log(|V|)) = O(T n^2 + T n^2 \log(T n)) \) since \( n \) times of Dijkstra’s algorithm are running on the space-time graph with \( (T + 1)n \) nodes and at most \( T n^2 \) edges. Hereafter, we refer this method as shortest path tree method (SPT).

**Algorithm 1 Greedy Algorithm based on Least Cost Path**

1. \( \mathcal{H} \leftarrow \phi; X = \{(v_i^0, v_j^D)\} \) 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 \( P(v_i^0, v_j^D) \) has the least cost among these paths.  
4. if \( e \in P(v_i^0, v_j^D) \) then  
5. \( \mathcal{H} \leftarrow e; c(e) \leftarrow 0. \)  
6. end if  
7. \( X \leftarrow X - (v_i^0, v_j^D) \).  
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 \( (v_i^0, v_j^D) \) 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 \( O(T n^5 + T n^4 \log(T n)) \) 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 \((v_0\) and \(v_9\) and two time slots. Both SPT and GrdLCP generate a structure with total cost 22 as shown in...
Figure 7(b) (since they will first pick the path/bunch including $v_i^0, v_b^1$, and 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 $a_i$ to $b_i$ in $H$. Since the space-time graph $G$ 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 pairs of nodes in $X$ as shown in Figure 8. In this figure, the edge between any two nodes $u$ and $v$ (such as $a_i$ and $b_i$, $p$, $p$ and $q$, or $q$ and $w_i$) is a virtual edge which represents the shortest path $P(u, v)$ from $u$ to $v$ in $G$. The cost of $P(u, v)$ is $c(u, v)$. We use $s(u, v)$ to represent the cost of $P(u, v)$. 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 $O(T^2 n^5 \log n)$, 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. $\mathcal{H} \leftarrow \emptyset; \; k = n^2; \; X = \{(v^0_i, v^T_j)\}$ for all $i$ and $j$ in $[1, n]$.
2. while $X \neq \emptyset$ do
3. \hspace{1em} $d \leftarrow \infty; \; B \leftarrow \emptyset$.
4. \hspace{2em} for all pairs $(p, q) \in \mathcal{V} \times \mathcal{V}$ do
5. \hspace{3em} for all pairs $(v^0_i, v^T_j) \in X$ do
6. \hspace{4em} $s[v^0_i, v^T_j] \leftarrow c(v^0_i, p) + c(q, v^T_j)$.
7. \hspace{2em} end for
8. \hspace{2em} sort all $s[v^0_i, v^T_j]$ in increasing order of $s$, and let $(u_i, v_l)$ refer to the $l$th pair in this sorted list.
9. \hspace{2em} for $l$ going from 1 to $k$ do
10. \hspace{3em} $C \leftarrow c(p, q) + s[u_1, w_1] + s[u_2, w_2] + \ldots + s[u_k, w_l]$.
11. \hspace{3em} if $C/l \leq d$ then
12. \hspace{4em} $d \leftarrow C/l; \; k = l$.
13. \hspace{4em} $B \leftarrow \{P(p, q) + P(u_1, p) + P(q, w_1) + \ldots + P(u_l, p) + P(q, w_l)$.
14. \hspace{3em} end if
15. \hspace{2em} end for
16. \hspace{2em} end for
17. $\mathcal{H} \leftarrow \mathcal{H} + B; \; k = k - k_1$.
18. $X \leftarrow X - \{(u_1, w_1), \ldots, (u_{k_1}, w_{k_1})\}$.
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 $c(B) = c(p, q) + s[u_1, w_1] + s[u_2, w_2] + \ldots + s[u_k, w_l]$.

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 $O(T^2n^6 \log n)$. 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 $G^t$ is a directed graph. Therefore, directed links $v_i^{t-1}v_j^t$ and $v_j^{t-1}v_i^t$ in the space-time graph $G$ are independent to each other and have individual costs. In other words, it is possible that only one of such link exists in $G$ (as the example in Figure 7) or $c(v_i^{t-1}v_j^t) = c(v_j^{t-1}v_i^t)$ 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 $v_i^tv_j^t$ in $G^t$, it only has one cost. Let $c_{i,j}^t$ represent this cost. In this situation, both directed links $v_i^{t-1}v_j^t$ and $v_j^{t-1}v_i^t$ 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 $G$ 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 $v_i^tv_j^t \in G^t$, we set the costs of the corresponding pair of spatial links $(v_i^{t-1}v_j^t$ and $v_j^{t-1}v_i^t$ both to $c_{i,j}^t$ (as shown in Figure 9(a)). We call this converted space-time graph $G$. 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 $G$.

Proof: Consider the solution of $Opt_B$ first. If $v_i^{t-1}v_j^t \in Opt_B$ but $v_j^{t-1}v_i^t \notin Opt_B$, we can construct a new solution $S_B$ by adding $v_j^{t-1}v_i^t$ 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 $v_i^tv_j^t \in S_A$ if and only if both $v_i^{t-1}v_j^t$ and $v_j^{t-1}v_i^t$ are in $S_B$. It is clear that $c(S_A) \leq c(S_B)$ since the total cost of $S_A$ only
counts one cost for both directed links. At last, since \( O_{\text{pt}A} \) is the optimal solution of the topology control problem in the original undirected graph, \( c(O_{\text{pt}A}) \leq c(S_A) \). Therefore, \( c(O_{\text{pt}A}) \leq c(S_A) \leq c(S_B) \leq 2c(O_{\text{pt}B}) \).

This lemma implies that Algorithm 2 can still achieve \( O(n^{1/3}\log^{1/3} n) \) 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 \( v_i^{t-1} \) at \( (t-1) \)th layer and \( v_j^t \) at \( t \)th layer. Then five directed links \( v_i^{t-1} v_j^t, v_j^{t-1} v_i^t, v_i^{t-1} v_j^{t-1}, v_j^{t-1} v_i^{t-1}, v_i^t v_j^t \) are added in the space-time graph. The costs of the first four links are set to zero, while \( c(v_i^{t-1} v_j^t) = c_{i,j}^t \). 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., \( c(\mathcal{H}) = \sum_{e \in \mathcal{H}} c(e) \).

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 \( v_i v_j \) 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
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 $G'$. 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


