Cost-Efficient Topology Design Problem in Time-Evolving Delay-Tolerant Networks

Minsu Huang, Siyuan Chen, Ying Zhu, Yu Wang
Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA.
Email:{mhuang4,schen40,yzhu17,yu.wang}@uncc.edu

***Abstract*—In this paper, we study the cost efficient topology design (CETD) problem in a predictable delay tolerant networks (DTN) where the time-evolving network topology is known a priori or can be predicted. We model such time-evolving network as a weighted space-time graph which includes both spacial and temporal information. The aim of CETD is to build a sparse structure from the original space-time graph such that (1) the network is still connected over time between any two nodes, (2) the cost of the least cost path for any two nodes in this constructed structure is at most δ times of that in the original graph, i.e., cost efficient DTN routing is possible between any two nodes; and (3) the total cost of the structure is minimized. We first show that classic topology control methods for static graph do not work for this new problem and then propose three efficient topology control methods which can significantly reduce the total cost of topology while maintain the connectivity and cost-efficiency over time. Extensive simulations have been conducted on random DTN networks and results demonstrate the efficiency of the proposed methods.**

I. Introduction

In *delay or disruption tolerant networks* (DTNs), the lack of continuous connectivity, network partitioning, and long delays make design of network protocols very challenging. Previous DTN research [1]–[3] mainly focuses on routing and information propagation by taking the intermittent connectivity and time-varying topology into consideration. However, there is little research on how to efficiently maintain the dynamic topology of DTNs. Network topology is always a key functional issue in design of large scale wireless networks. For different applications, network topology can be constructed under different objectives, such as power efficiency, fault tolerance, and throughput maximization. Topology design has been well studied in wireless ad hoc and sensor networks [4]–[6]. The existing topology control methods are mainly on how to construct a topology structure from a static and connected communication graph with all possible links. However, in DTNs, the lack of continuous connectivity makes these existing algorithms unsuitable. Therefore, how to maintain efficient and dynamic topology of DTNs becomes crucial, especially with participation of large number of wireless devices.

DTNs often evolve over time: changes of topology can occur when some nodes appear, disappear, or move around. Such dynamics over time domain are often ignored in protocol design or modeled by simple unrealistic models, e.g., random walk mobility model. In real-world wireless networks (such as pocket switched networks based on human mobility [7], vehicular networks based on public buses or taxi cabs [8], sensor networks for wildlife tracking [1], mobile social networks [10] or space communication [9]), node mobility and the evolution of topology heavily depend on both social and temporal characteristics of the network and network participants. For certain type of these networks, the temporal characteristics of their topology could be known a priori or can be predicted from historical tracing data. E.g., it is easy to discover the temporal pattern of topology for a DTN formed by public buses or a mobile social network consist of students who share fixed class schedules. In such networks, protocols designed for traditional wireless networks may be inefficient due to the 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 type of DTNs.

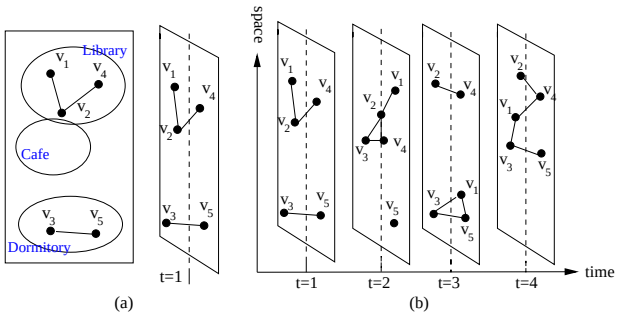


Fig. 1. **A time-evolving DTN:** (a) a snapshot of the network, (b) time-evolving topologies of the DTN (a sequence of snapshots).

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

- In Section II, we first model the time-evolving DTN as a weighted space-time graph and define the *cost-efficient topology design problem* which aims to build a sparser structure (also a space-time graph) from the original space-time graph such that the structure preserves the connectivity and path efficiency while minimizes the total cost. We show that the classic topology control algorithms for building spanners on unit disk graphs or general graphs do not work for this new problem.

This work is supported in part by the US National Science Foundation (NSF) under Grant No. CNS-0721666 and No. CNS-0915331.

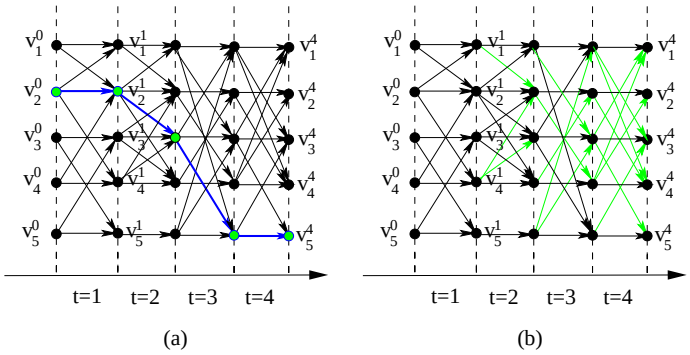


Fig. 2. (a) The corresponding space-time graph $\mathcal{G}$ of the time-evolving network in Figure 1(b). Here, a space-time path (in blue) from the source v_2 to the destination v_5. (b) A new connected subgraph $\mathcal{H}$ of $\mathcal{G}$ (green links are removed links from original graph).

- In Section III, we propose several methods which can significantly reduce the total cost of network topology while maintaining the connectivity and cost-efficient paths between any two nodes over time.
- In Section IV, extensive simulations have been conducted on random DTN networks. Results demonstrate the efficiency of all proposed topology control methods.

To our best knowledge, there is no previous results on topology control in DTNs except for our recent work [11]. In [11], we study how to build topology for time-evolving DTNs so that the network is connected over time (without the requirement on path efficiency). This paper is the first attempt to build topology to support cost-efficient routing beyond just connectivity.

II. Model and The Problem

A. Network Model: Time-Space 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 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 [12] 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). we now define a space-time graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is a directed graph defined in both spacial and temporal space. See Figure 2 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 $\mathcal{V} = \{v_j^t | j = 1, \cdots, n \text{ and } t = 0, \cdots, T\}$ 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 2) connects the same node v_j across consecutive $(t-1)$th and tth layers, which represents that the node is able to carry the message in the tth time slot. A spatial link $\overrightarrow{v_j^{t-1} v_k^t}$ represents that node v_j can forward a message to its neighbor v_k in the tth time slot. This representation of the network includes all information from both spacial and temporal information of this time-evolving network. 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 2(a), 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 transmitting a message on that link (transmitting it from one node to another 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 $P_{\mathcal{G}}(u, v)$ as the least cost path from u to v in $\mathcal{G}$. The total cost of such shortest path $P_{\mathcal{G}}(u, v)$ in $\mathcal{G}$ is denoted by $d_{\mathcal{G}}(u, v) = \sum_{e \in P_{\mathcal{G}}(u,v)} c(e)$, which is the summation of costs of all links in path $P_{\mathcal{G}}(u, v)$.

B. Topology Design Problem: Building Spanner over Time

We now define the *cost-efficient topology design problem* (CETD) on space-time graphs.

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

Here *connectivity* and *spanner* have *different* definitions from those of a static graph.

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

This guarantees 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. Figure 2(b) shows an example. Hereafter, we always assume that the original space-time graph $\mathcal{G}$ is connected over time period T.

Definition 3: Given a real number $\delta > 1$, a subgraph $\mathcal{H}$ of $\mathcal{G}$ is *δ-spanner* of $\mathcal{G}$ if and only if for any pairs of nodes (v_i^0, v_j^T), the cost of the shortest path in $\mathcal{H}$ is at most δ times the cost of the shortest path in $\mathcal{G}$, i.e., $\max_{1\le i,j\le n}\{\frac{d_{\mathcal{H}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\} \le \delta$. Here, $\max_{1\le i,j\le n}\{\frac{d_{\mathcal{H}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\}$ is called *spanning ratio* of $\mathcal{H}$.

In other words, the constructed subgraph still can support cost-efficient routes for any two devices over the time period. This is obviously a stronger requirement than just connectivity. In [11], we study the topology design problem which only requires connectivity (denoted by TDC hereafter).

The cost-efficient topology design problem for space-time graph is much harder than the one for a static graph. For a static graph without time domain, there are several efficient methods to construct a spanner (such as [4]–[6] for unit disk graphs using geometric properties or [13] for general weighted graphs). However, in a time-evolving DTN modeled by a space-time graph, simply applying spanner methods for unit disk graph in each time slot is not feasible since the network in each single snapshot may not be connected at all. On the other hand, the classic greedy method for general graph [13] over the whole space-time graph is neither a solution of CETD. The greedy method basically sorts all links in term of their cost, then in increasing order of cost it adds link uv into H if $d_H(u,v) \ge \delta d_G(u,v)$. However, in the space-time graph $\mathcal{G}$, each link v_i^t, v_j^{t+1} is the only path connected v_i^t and v_j^{t+1}, thus the greedy algorithm will keep all links of $\mathcal{G}$ in $\mathcal{H}$. This is unacceptable for CETD. Actually, The TDC problem is a special case of CETD problem where δ is set to be infinity. In [11], we showed that TDC is also very challenging problem by linking it to a well-known NP-hard problem, *directed generalized Steiner network* (DGSN) problem [14], [15].

III. Cost-Efficient Topology Control Algorithms

In this section, we will propose three efficient algorithms to construct a sparse structure that fulfills the connectivity and spanner requirements over a space-time graph. For all algorithms, the inputs are the original graph $\mathcal{G}$ and a real number $\delta \ge 1$, and the output is a subgraph $\mathcal{H}$ of $\mathcal{G}$.

A. Union of Shortest Path Algorithm (USP)

One simple method to construct a subgraph, which maintains the cost-efficient paths between any two nodes over the time period T, is keeping all the shortest paths from v_i^0 to v_j^T for $i, j = 1, \cdots, n$. Obviously, the resulting structure satisfies the spanner property, since $\delta \ge 1$. Algorithm 1 shows the detail algorithm. Its time complexity is $O(n^2T(\log(nT)+n))$ since we only need to compute n^2 shortest paths of $\mathcal{G}$ (with $O(nT)$ nodes and at most $O(n^2T)$ links). This can be easily achieved by running n times of Dijkstra algorithm whose complexity is $O(nT(\log(nT)+n))$. Hereafter, we refer this method as *union of shortest path algorithm* (USP).

Algorithm 1 Union of Shortest Path (USP) Algorithm

1: $\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^T)\}$ for all integer $1 \le i, j \le n$.
2: **for all** pairs $(v_i^0, v_j^T) \in X$ **do**
3: Find the shortest path $P_{\mathcal{G}}(v_i^0, v_j^T)$ in $\mathcal{G}$.
4: **if** $e \in P_{\mathcal{G}}(v_i^0, v_j^T)$ and $e \notin \mathcal{H}$ **then**
5: $\mathcal{H} = \mathcal{H} \cup \{e\}$.
6: **end if**
7: **end for**
8: **return** $\mathcal{H}$

B. Greedy Algorithm to Delete Links (GDL)

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

Algorithm 2 Greedy Algorithm to Delete Links (GDL)

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

C. Greedy Algorithm to Add Links (GAL)

The third algorithm is also a greedy algorithm which starts from building a connected sparse structure and then adds more links into it to satisfy the spanner property. It includes two steps. In the first step, we try to connect n^2 pair of nodes in $X = \{(v_i^0, v_j^T)\}$ for all integer $1 \le i, j \le n$. In each round we pick the least cost path between a pair nodes in X whose cost is the minimum among all least cost paths connecting any pair of nodes in X, add all links in this path into $\mathcal{H}$, clear their link costs to zeros, and remove this pair from X. After n^2 rounds, we have a structure $\mathcal{H}$ which connects all pairs of (v_i^0, v_j^T). In the second step, for each pair (v_i^0, v_j^T), we calculate its shortest path in $\mathcal{H}$. If the cost of $P_{\mathcal{H}}(v_i^0, v_j^T)$ is greater than δ times of the cost of $P_{\mathcal{G}}(v_i^0, v_j^T)$ in the original graph, we directly add the links in this shortest path into $\mathcal{H}$. See Algorithm 3

for detail. Hereafter, we denote this method as GAL. The complexity of GAL is less than the one of GDL algorithm. Both steps need $n^2 \cdot O(n^2 T(\log(nT)+n))$ time since they both run computation of shortest paths for n^2 rounds. Therefore, the total time complexity is $O(n^4 T(\log(nT)+n))$.

Algorithm 3 Greedy Algorithm to Add Links (GAL)

1: $\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^T)\}$ for all integer $1 \le i, j \le n$.
2: **while** $X \neq \phi$ **do**
3: find the least cost paths for every pair nodes from X in $\mathcal{G}$, and assume $P_{\mathcal{G}}(v_i^0, v_j^T)$ has the least cost among these paths.
4: **if** $e \in P_{\mathcal{G}}(v_i^0, v_j^T)$ **then**
5: $\mathcal{H} = \mathcal{H} \cup \{e\}$; $c(e) \leftarrow 0$.
6: **end if**
7: $X = X - (v_i^0, v_j^T)$.
8: **end while**
9: Recover all link costs of $\mathcal{G}$ to their original values.
10: **for all** $(v_i^0, v_j^T), 1 \le i, j \le n$ **do**
11: **if** $d_{\mathcal{H}}(v_i^0, v_j^T) > \delta \cdot d_{\mathcal{G}}(v_i^0, v_j^T)$ **then**
12: Add all links $e \in P_{\mathcal{G}}(v_i^0, v_j^T)$ into $\mathcal{H}$ if $e \notin \mathcal{H}$.
13: **end if**
14: **end for**
15: **return** $\mathcal{H}$

IV. Simulations

We have conducted extensive simulations to evaluate our proposed algorithms devoted to the CETD problem. We implement and test the following five algorithms: USP (Algorithm 1), GDL-G (Algorithm 2 running on the original graph $\mathcal{G}$), GDL-USP (Algorithm 2 running on the output of USP), GAL (Algorithm 3), and LCP (a least cost path based greedy algorithm for TDC problem, which basically is the first half of GAL [Lines 1-8 of Algorithm 3]). Recall that USP has a fix spanning ratio 1 since it preserves the shortest paths for all pairs of nodes in X. LCP algorithm builds a connected topology over time but the constructed topology does not have bounded spanning ratio. We use it as a reference to our proposed algorithms.

In all simulations, we take three metrics as the performance measurement for any algorithm devoted to *cost-efficient topology design problem*:

- **Total Cost:** the total cost of the constructed topology $\mathcal{H}$ (output of the algorithm), i.e., $c(\mathcal{H}) = \sum_{e \in \mathcal{H}} c(e)$.
- **Total Number of Edges:** the total number of edges in the constructed $\mathcal{H}$, i.e., $|\mathcal{H}|$. Here, $|\mathcal{G}|$ denotes the number of edges of graph $\mathcal{G}$.
- **Spanning Ratio:** $\max_{1 \le i,j \le n} \{\frac{d_{\mathcal{H}}(v_i^0, v_j^T)}{d_{\mathcal{G}}(v_i^0, v_j^T)}\}$.

The objective of our topology control methods is to construct topology structures with small total cost, small edge number and small spanning ratio. For all simulations, we repeat the experiment for multiple times and report the average values of these metrics.

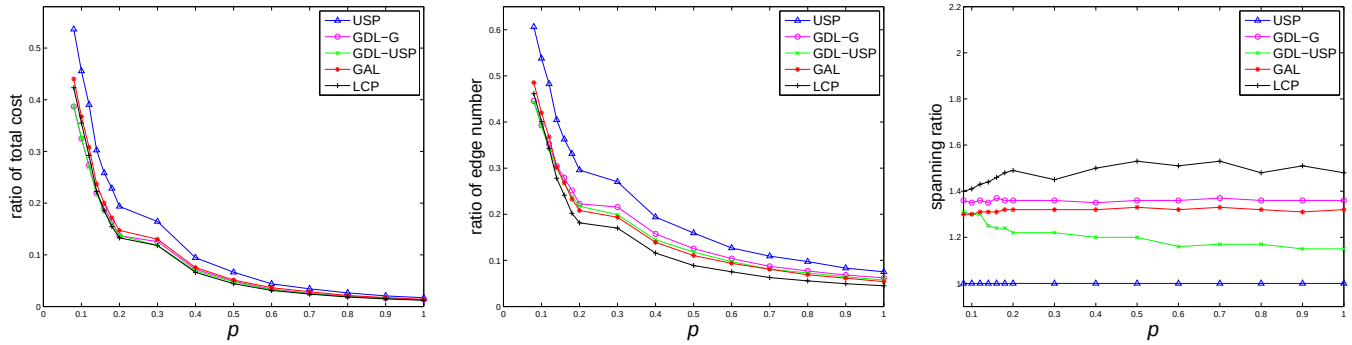
The underlying time-evolving networks are randomly generated from random graph model. We first generate a sequence of static random graphs $\{G^t\}$ to represent the time-evolving DTN. Here, we consider a DTN with 10 nodes (n=10) and spreading over 10 time slots ($T = 10$). For each time slot t, we randomly generate a static graph G^t using the classical random graph generator. Basically, for each pair of nodes v_i and v_j, we insert edge $\overrightarrow{v_i v_j}$ with a fixed probability p. 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. After generating $\{G^t\}$, we convert it into its corresponding space-time graph $\mathcal{G}$ with $n(T+1)$ nodes. The cost of each inserted edge is randomly chosen from 1 to 50. Then we perform proposed topology control algorithms on $\mathcal{G}$. For each setting, we generate 100 random networks and report the average performance of topology design algorithms among them.

For the first set of simulations, we increase the network density by rising p from 0.1 to 1.0, and keep spanning ratio requirement δ at 1.4. Figure 3(a) and 3(b) show the ratios between the total cost/number-edges of the generated graph $\mathcal{H}$ and that of the original graph $\mathcal{G}$ when p increases. This ratio implies how much saving achieved by the topology control algorithm, compared with the original network without topology control. From the results, all topology control algorithms can significantly reduce the cost of maintaining the connectivity. Even with the least density ($p = 0.1$), all algorithms can save more than 50% cost and around 50% edges except for USP. When the network is denser, the saving of topology control is larger. For $p = 1.0$, more than 95% cost is saved. Overall, USP has the largest cost and edge number, while LCP usually has the least. Figure 3(c) shows the spanning ratios of all methods. Clearly, LCP is the only algorithm that cannot satisfy the spanning ratio requirement $\delta = 1.4$. It is an algorithm only for maintaining the connectivity over time. USP can always achieve spanning ratio of 1 but with higher cost than other algorithms since it keeps all edges in shortest paths. The other three algorithms have very close performance in total cost. GDL-USP has lower spanning ratio than GDL and GAL, yet they are all bounded by δ.

In the second set of simulations, we fixed the network density $p = 0.2$ and run our algorithms with different spanning ratio requirement δ increasing from 1.0 to 2.0. From results shown in Figure 4, we can observe that tighter spanning ratio requirement results in higher cost and larger number of edge-usage of our topology algorithms. When the spanning ratio requirement becomes looser, the restriction on removing links becomes weaker. In the extreme case with infinite δ, our CETD problem converges to TDC which only preserves the connectivity. Notice that performances of LCP and USP do not affect by the spanning ratio requirement.

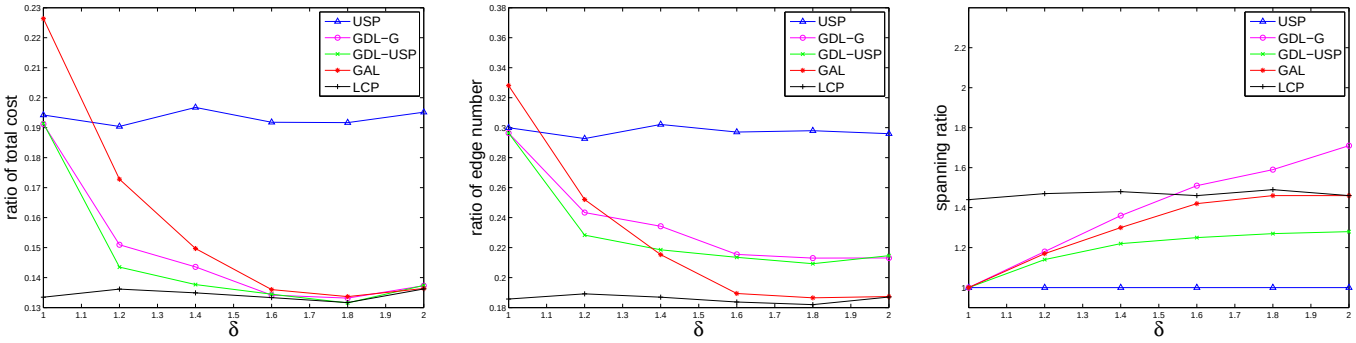
V. Conclusion

We study cost efficient topology control problem in a predictable time-evolving DTN modeled by a space-time graph. We propose three greedy-based methods which can significant reduce the cost of topology while maintain the connectivity



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

Fig. 3. Simulation results on random networks with different density and $\delta = 1.4$. (a)-(b): The cost (or edge number) of $\mathcal{H}$ is divided by the cost (or edge number) of $\mathcal{G}$, which illustrates how much saving achieved by the topology control algorithm, compared with the original network without topology control. (c): The spanning ratio shows the path efficiency of $\mathcal{H}$ compared with $\mathcal{G}$.



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

Fig. 4. Simulation results on random networks with fixed network density $p = 0.2$ and varying spanner ratio requirement δ.

and power efficiency of paths over time. Simulation results from random networks demonstrate the efficiency of our methods. We believe that this paper presents the first step in exploiting topology control for time-evolving DTNs.

REFERENCES

[1] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 5, pp. 96–107, 2002.

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

[3] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc: Proc. of ACM int'l symp. on Mobile ad hoc networking and computing*, 2004.

[4] Xiang-Yang Li, Peng-Jun Wan, and Yu Wang, "Power efficient and sparse spanner for wireless ad hoc networks," in *Proc. IEEE Int. Conf. on Computer Communications and Networks (ICCCN)*, 2001.

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

[6] Roger Wattenhofer, Li Li, Paramvir Bahl, and Yi-Min Wang, "Distributed topology control for wireless multihop ad-hoc networks," in *Prof. of IEEE INFOCOM*, 2001.

[7] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *MobiHoc '08: Proc. of ACM int'l symp. on Mobile ad hoc networking and computing*, 2008.

[8] Xiaolan Zhang, Jim Kurose, Brian Neil Levine, Don Towsley, and Honggang Zhang, "Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing," in *Proc. of ACM international conf on Mobile computing and networking (MobiCom)*, 2007.

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

[10] Augustin Chaintreau, Pierre Fraigniaud, and Emmanuelle Lebhar, "Opportunistic spatial gossip over mobile social networks," in *WOSP '08: Proceedings of the first ACM workshop on Online social networks*, 2008.

[11] Minsu Huang, Siyuan Chen, Ying Zhu, and Yu Wang, "Topology Control for Time-Evolving and Predictable Delay-Tolerant Networks," Submitted for Publication, 2010.

[12] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Tech. Rep., 2004.

[13] B. Chandra, G. Das, G. Narasimhan, and J. Soares, "New sparseness results on graph spanners," in *Proceedings of the Eighth Annual ACM Symposium on Computational Geometry*, 1992.

[14] Moses Charikar and Chandra Chekuri, "Approximation algorithms for directed steiner problems," *J. Algorithms*, 33(1), pp. 73–91, 1999.

[15] Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev, "Set connectivity problems in undirected graphs and the directed steiner network problem," in *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, 2008.