# TOPOLOGY DESIGN FOR TIME-VARYING NETWORKS

by

Minsu Huang

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2012

Approved by:

______________________________

Dr. Yu Wang

______________________________

Dr. Teresa Dahlberg

______________________________

Dr. Jamie Payton

______________________________

Dr. Jiang (Linda) Xie

______________________________

Dr. Yang Cao

## ABSTRACT

MINSU HUANG. Topology design for time-varying networks.
(Under the direction of DR. YU WANG)

Traditional wireless networks seek to support end-to-end communication through either a single-hop wireless link to infrastructure or multi-hop wireless path to some destination. However, in some wireless networks (such as delay tolerant networks, or mobile social networks), due to sparse node distribution, node mobility, and time-varying network topology, end-to-end paths between the source and destination are not always available. In such networks, the lack of continuous connectivity, network partitioning, and long delays make design of network protocols very challenging. Previous DTN or time-varying network research mainly focuses on routing and information propagation. However, with large number of wireless devices' participation, and a lot of network functionality depends on the topology, how to maintain efficient and dynamic topology of a time-varying network becomes crucial. In this dissertation, I model a time-evolving network as a directed time-space graph which includes both spacial and temporal information of the network, then I study various topology control problems with such time-space graphs.

First, I study the basic topology design problem where the links of the network are reliable. It aims to build a sparse structure from the original time-space graph such that (1) the network is still connected over time and/or supports efficient routing between any two nodes; (2) the total cost of the structure is minimized. I first prove that this problem is NP-hard, and then propose several greedy-based methods as solutions.

Second, I further study a cost-efficient topology design problem, which not only requires the above two objective, but also guarantees that the spanning ratio of the topology is bounded by a given threshold. This problem is also NP-hard, and I give several greedy algorithms to solve it.

Last, I consider a new topology design problem by relaxing the assumption of reliable links. Notice that in wireless networks the topologies are not quit predictable and the links are often unreliable. In this new model, each link has a probability to reflect its reliability. The new reliable topology design problem aims to build a sparse structure from the original space-time graph such that (1) for any pair of devices, there is a space-time path connecting them with the reliability larger than a required threshold; (2) the total cost of the structure is minimized. Several heuristics are proposed, which can significantly reduce the total cost of the topology while maintain the connectivity or reliability over time.

Extensive simulations on both random networks and real-life tracing data have been conducted, and results demonstrate the efficiency of the proposed methods.

## ACKNOWLEDGMENTS

First of all, I would like to express my deep appreciation to my Ph.D. advisor Dr. Yu Wang. His vision has led me through my graduate research, and his enthusiasm to work always inspires me to push myself for better. This dissertation would not be possible without his guidance and encouragement.

I would also like to thank Dr. Teresa Dahlberg, Dr. Jamie Payton, Dr. Jiang (Linda) Xie and Dr. Yang Cao for serving on my advisory committee.

Many thanks go to former member of network research Lijuan Cao for her help in literature study, and current Ph.D student Siyuan Chen for his help with his excellent mathematical expertise.

Finally, my special gratitude goes to my parents for their endless love, encouragement, and support throughout my life. To them I dedicate this dissertation.

TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| DTN | Delay Tolerant Network |
| TD | Topology Design |
| CETD | Cost-Efficient Topology Design |
| RTD | Reliable Topology Design |
| DGSN | Directed Generalized Steiner Network |
| DST | Directed Steiner Tree |
| USP | Union of Shortest Path |
| GLCP | Greedy Based on Least Cost Path |
| GLDB | Greedy Based on Least Density Bunch |
| GDL | Greedy Algorithm to Delete Links |
| GAL | Greedy Algorithm to Add Links |
| BFM | Backward-Forward Method |
| UMCRP | Union of Minimum Cost Reliable Path |
| GMCRP | Greedy Algorithm with Minimum Cost Reliable Path |
| LCP/MCRP | Least Cost Path or Minimum Cost Reliable Path |

# CHAPTER 1: INTRODUCTION

Wireless networks often evolve over time: changes of topology structure can occur if some nodes appear, disappear, or move around. In this dissertation, I focus on study of topology design for time-varying wireless networks. Examples of such time-varying wireless networks include mobile ad hoc networks and delay tolerant networks. The following sections briefly introduce characteristics of mobile ad hoc networks and delay tolerant networks.

## 1.1 Mobile Ad Hoc Networks

Mobile ad hoc networks are a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration [1]. Therefore, ad hoc networks are autonomous systems of mobile nodes connected by wireless links. While mobile nodes are free to move arbitrarily, the network topology can consequently change in an unpredictable way. In contrast with widely used cellular networks or wireless LANs, which support single-hop communications with base stations (BSs) and backbone networks, infrastructure-less ad hoc networks support peer-to-peer, multihop communications by using mobile nodes as routers to relay traffic. Thus, if two nodes are close enough to hear each other, they can communicate directly. Otherwise, they need other nodes located between them to set up a route and forward packets for them.

Due to constraints of mobile devices, and network architecture, mobile ad hoc networks have some special characteristics:

- Self-organization: In contrast to cellular networks, which use wired backbone as centralized controller and require tremendous pre-configuration work, such as cell planning, BS deployment, etc, there is no pre-configuration or centralized controller for ad hoc networks. Therefore, with autonomous architecture, the network configuration and management must be automatic and dynamic.

- Multi-hop communication: Due to signal propagation characteristics of wireless transceivers, each mobile node has a limited transmission range. In order to provide communication between nodes that cannot directly hear each other, ad hoc networks adopt multi-hop communication paradigm by exploiting cooperation among mobile nodes in the network.
- Dynamic topology: As nodes can move around freely, the network topology may change randomly and unpredictably. In some cases, there might be unidirectional links incurred by factors such as heterogeneity of receiver and transmitter hardware.
- Energy-constrained operation: Lightweight mobile devices are mainly battery operated, therefore, energy conservation is one of the most important design criteria for ad hoc networks.
- Bandwidth constrained links: As a contrast to wired links, wireless links have significantly low bandwidth. Moreover, due to the effects of multiple access, multipath fading, noise, signal interference, etc., capacity of wireless links can be significantly degraded and the effective throughput might be less than the radio's maximum transmission capacity.
- Limited security: mobile ad hoc networks are more susceptible to security threats due to following reasons: first, there is no central server to manage authentication and access control; second, it is difficult to separate inside from outside in ad hoc networks because each node may operate as a router and access the medium freely, thus, a malicious node can become a router and disrupt network operations; third, the shared medium makes communication unsafe because packets can be easily overheard by others.

## 1.2 Wireless Delay Tolerant Networks

Traditional mobile ad hoc networks seek to support end-to-end communication through multi-hop path. However, in some cases due to nodes sparse distribution, nodes mobility, and varying network connectivity, end-to-end paths are not always available. Such networks are called delay tolerant networks (DTNs) [2, 3], which use asynchronous communications and do not require an end-to-end path between any pair of nodes at any time.

DTNs have wide applications such as pocket switched networks based on human mobility [4–6], vehicular networks based on public buses or taxi cabs [7,8], sensor networks for wildlife tracking [3], mobile social networks [9–13], disaster-relief networks, or space communication [14, 15]. In DTNs, traditional communication protocols designed for wireless networks may be inefficient due to time-varying structures and long delays, or even fail to perform due to the lack of continuous connectivity or network partitioning. Though DTNs share many characteristics with mobile ad hoc networks, they also have the following unique characteristics [16]:

- Nodal mobility: Like in other mobile ad hoc networks, nodes in DTNs are carried by people or vehicles, and their locations can change over time. This also causes time-varying network topology.
- Sparse connectivity: Due to the limited transmission range of wireless devices and sparse node distribution, the connectivity of DTNs is usually intermittent where end-to-end paths do not always exist between source node and destination node.
- Delay tolerability: Due to low connectivity and nodal mobility in DTNs, data delivery delay is usually quite high. Thus, it is necessary for DTN network applications to tolerate high delay.
- Fault tolerability: To guarantee delivery in such sparse networks, redundancy (e.g., multiple copies of message) is a usual approach for DTN routing, thus

dropping of one package may not necessarily cause failure of delivery.

- Limited buffer: DTN nodes have to buffer messages when they are not connected. Usually individual node has limited storage to buffer messages.

## 1.3 Topology Design for Wireless Time-Varying Networks

Both mobile ad hoc networks and delay tolerant networks recently have drawn much attention from networking researchers since they have wide applications in challenging environments. Communication in these time-varying networks is challenging as it must handle time-varying links, long delays, and dynamic topology.

Previously, many routing schemes [17–20,22–26,109] 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 have not precisely use *temporal characteristics* of the network, thus they may lead to poor performance in time-evolving wireless networks. Notice that 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 mobile ad hoc networks, a network is usually modeled as a connected graph with stable end-to-end paths and the effect of dynamic changes in topology on protocol design is handled by continuous monitoring and on-demand updates when the topology changes.

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, 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

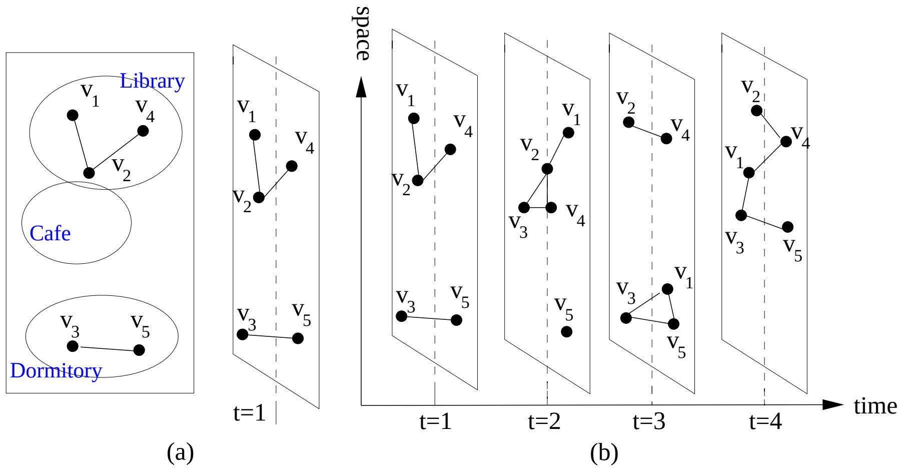


Figure 1.1: A time-evolving DTN: (a) a snapshot of the network, (b) time-evolving topologies of the DTN (a sequence of snapshots).

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 [7] or a mobile social network consist of students who share fixed class schedules. A recent study by Song *et al.* [27] also shows that a human mobility model can achieve a 93% potential predictability. In this kind of time-evolving and predictable networks, 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 1.1 illustrates an example of such time-evolving DTN. It is crucial to design new efficient protocols for this emerging type of DTNs. In this dissertation, I study how to control the dynamic topology for such *time-evolving* and *predictable* wireless networks.

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 [28–33]. The focus of previous research is mainly on how to construct a power efficient structure from a static and connected topology (an underlying communication graph includes all possible links). However, in some time-varying networks (especially DTNs), the underlying topology is lack of continuous connectivity which makes existing topology design algorithms useless. Therefore, how to maintain efficient and dynamic topology of time-varying networks becomes crucial, especially with large number of wireless devices' participation. For my best knowledge, there is no previous results on topology design in time-varying wireless networks. I believe that topology can be controlled more wisely and efficiently if the network evolution over time is known.

## 1.4 Contributions and Organization

In this dissertation, I study the topology design problem in a time-evolving networks by taking *time-domain topological information* into consideration. My major contributions are summarized as following:

- I first model the time-evolving DTN as a directed time-space graph in which both spatial and temporal information is preserved.
- I define a topology design (TD) problem which aims to build a sparser structure (also a time-space graph) from the original time-space graph such that (1) the network is still connected over time and/or supports efficient routing between any two nodes; (2) the total cost of the structure is minimized. I prove this problem is NP-hard.
- Then I propose a cost-efficient topology design (CETD) problem, which not only has the above two objectives but also take in account the spanning ratio of the network during topology design.
- I also modify the time-space model to fit for networks with unreliable connectivity, in which each link has a probability reflects its reliability, and then define a reliable topology design (RTD) problem, which aims to build a sparse structure from the original time-space graph such that (1) for any pair of devices, there is a space-time path connecting them with the reliability larger than a required threshold; (2) the total cost of the structure is minimized.

- For all the three topology design problems I propose various methods, which can significant reduce the total cost of network topology with or without spanner, while maintaining the connectivity or reliability over time.
- To evaluate my algorithms, I have conducted extensive simulations on both random DTN networks and real-world DTN tracing data [35]. Results demonstrate the efficiency of the proposed topology design methods.

The rest of this dissertation is organized as following:

- In Chapter 2, I summarize related works in routing and topology design in time-varying networks (especially DTNs).
- In Chapter 3, I formally define the time-space graph with both temporal and spatial information to model time-varying networks, and define three topology design problems (TD, CETD and RTD) in time-varying networks.
- In Chapter 4, I investigate topology design (TD) problem for time-evolving networks and propose several greedy-based algorithms.
- In Chapter 5, I study cost-efficient topology design (CETD) problem for time-varying networks, which maintains a spanner as the constructed topology.
- In Chapter 6, I study the reliable topology design (RTD) problem in time-space graph with unreliable links, and propose several algorithms for the problem.
- In Chapter 7, I present my extensive simulations of my proposed algorithms for all the topology design problems.
- Finally, in Chapter 8, I conclude the current research progress and point out my future plan.

# CHAPTER 2: RELATED WORK

In recent years, a lot of research has been done to improve the performance of wireless networks. This chapter reviews the related work in literature which inspire my research on time-varying wireless networks.

## 2.1 Routing in Wireless Networks

I first look at classic routing protocols for both mobile ad hoc networks and delay tolerant networks, since routing is one of the most fundamental communication functions in wireless networks. One of the goals of topology control or design is to serve and support efficient routing.

### 2.1.1 Routing in Wireless Ad Hoc Networks

Unicast routing protocols in mobile ad hoc networks are broadly divided into two types: proactive and reactive. Proactive routing maintains routing information between each pair of nodes all the time and performs periodical updates even though they do not have to communicate with each other. Reactive routing, on the other hand, only initializes route discovery when necessary. For example, Destination-Sequenced Distance-Vector Routing (DSDV) [36] uses the distance vector approach, while Optimized Link State Routing (OLSR) [37] adapts the link state algorithm. Both DSDV and OLSR are proactive routing protocols. Some other routings such as, Ad Hoc On-Demand Distance Vector Routing(AODV) [38], Dynamic Source Routing(DSR) [1], Lightweight Mobile Routing [39], Temporally Ordered Routing Algorithm (TORA) [40] are reactive routing protocols. Hybrid routing approaches, which combining proactive and reactive mechanisms, are based on the idea of organizing nodes in groups and then assigning nodes different functionalities inside and outside a group, usually, applying proactive routing inside the group and reactive approach among groups. For example, Hierarchical State Routing (HSR) [41], Clusterhead-Gateway Switch Routing (CGSR) [42], Zone Routing Protocol (ZRP) [43] and Landmark Ad Hoc Routing protocol (LANMAR) [44] are hybrid routings. In addition, there are other types of unicast routing protocols, such as hierarchical routing and geographic routing. For more detail, please refer to surveys on ad hoc routing [45,46].

Besides unicast routing protocols, there are also studies on how to efficiently deliver messages to multiple destinations in mobile ad hoc networks, including broadcast and multicast protocols.

*Broadcasting* has wide applications in wireless ad hoc networks not only for data dissemination, but also for route discovery and maintenance in many ad hoc unicast routing protocols. Blind flooding which is the simplest broadcast protocol, allows each node to forward the message once to its neighbors. However, due to omnidirectional radio propagation and transmission range overlap, blind flooding is usually very costly and will result in serious redundancy, contention, and collision, which is referred to as the broadcast storm problem in [47]. Some broadcast protocols apply probabilistic approaches [47, 48], while some [49–51] use deterministic approaches. In addition, several broadcast protocols [52,53] have been proposed for efficient broadcasting using directional antennas.

*Multicasting* is a very efficient and useful communications paradigm supporting group-oriented applications, especially for mobile ad hoc network, which have constrained bandwidth and energy resource. There has been extensive work to develop multicast protocols for mobile ad hoc networks to provide one-to-many service [54–56]. As a variant of conventional multicasting, geocasting delivers messages to nodes within a geographical region, which can provide new services and applications such as geographic advertising. Protocols proposed in [57,58] are examples of multicasting with a flooding approach and the work in [59] is an example of multicasting without flooding.

### 2.1.2 Routing in Wireless Delay Tolerant Networks

In time-varying DTNs, due to uncertainty of connectivity, mobile nodes are intermittently connected, end-to-end paths between the source and the destination may not be available at certain time. The inherent uncertainty of network conditions and limited resource and information make routing in time-varying networks very challenging. Existing routing protocols in mobile ad hoc networks may fail in time-varying DTNs. To deal with the challenging routing problem, a variety of mechanisms [17–20, 22–26, 109] deferent from those of traditional mobile ad hoc networks have been proposed. Most of these DTN routing protocols belong to three categories: *message-ferry-based*, *opportunity-based* and *prediction-based*.

In message-ferry-based methods [18–20, 63], 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 [3, 22, 109], 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.

Prediction-based routing protocols [23–26] 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.

Another new direction for routing in time-evolving networks is social-based forwarding schemes which attempt to exploit stable social network structures and use these in the design of data forwarding. Notice that social relations among mobile users are usually long-term characteristics and less volatile than node mobility. Most social-based forwarding schemes exploit sociological centrality metrics [64] for relay selections. Daly and HaahrSim [65] proposed SimBet routing which uses the ego-centric betweenness metric and forwards data to nodes with a higher SimBet utility. Hui *et al.* [4] proposed BUBBLE Rap routing which considers node centrality in a hierarchical manner based on social community knowledge. They used cumulative contact length as edge weights for community detection in social networks, but did not exploit such weights for data forwarding. *The binary social network* model considers node pairs with different contact frequencies as equivalent ones, and limits the performance of centrality-based data forwarding schemes because node centrality values do not really characterize the nodes' capabilities of contacting other nodes.

Recently, enormous work has been done for further improving routings in DTNs. Liu and Wu [67,120] have proposed to use estimated *expected minimum delay* (EMD) as a delivery probability metric in DTNs with repetitive mobility. They apply the Markov decision process to derive the EMDs of the messages at particular times, and use EMD as the metric of their opportunistic routing protocol.

Liu and Wu [68] also propose a forwarding method applying a probabilistic forwarding metric derived by modeling each forwarding as an optimal stopping rule problem. In their forwarding model, each time a node has a packet to forward, it compare the joint expected delivery probability of the copies of forwarding it in the current time to that of holding it and forward in a future time, thus can help to choose a time of forwarding to maximize the expected delivery probability.

Yuan *et al.* [69] propose an routing algorithm named *predict and relay* (PER) for DTNs, in which nodes determine the probability distribution of future contact times and choose a proper next hop in order to improve the end-to-end delivery probability. They observe that nodes in DTNs usually have their mobility pattern, usually move around a set of well visited landmarks rather than moving randomly; node mobility behavior is semi-deterministic and could be predicted with enough history information. They also propose a time homogeneous semi-markov process model to describe node mobility as transitions between landmarks. By employing future prediction mechanisms based on mobility history information, nodes could better select the next hop which have higher probability to contract the destination directly or indirectly.

Balasubramanian *et al.* [70] propose a *resource allocation protocol* for DTN routing to explicitly optimize a specified routing metric, such as the worst-case delivery delay or the fraction of packets delivered before a deadline. According the authors in [70], even I have complete knowledge of node meetings and communication requirements a priori, computing an optimal routing schedule that to maximize the number of packets delivered is NP-hard. And there is already an approximation algorithm with a $\Omega(\sqrt{n})$ bound. So the authors have given heuristic algorithm of routing scheduling.

Nelson *et al.* [71] present a new DTN routing algorithm called *encounter-based routing* (EBR), aims to maximize delivery ratio while minimizing overhead and delay. They first roughly predict the future rate of node encounters by past data, and then select the node have higher probability to forward the package. To minimize network resource usage, they limits the number of replicas of any message in the system.

Lu *et al.* [72] propose a social-based privacy preserving packet forwarding protocol called SPRING for vehicular delay tolerant networks (DTNs). The authors defined social degree in intersection of DTNs, based on which they distribute roadside units (RSU). Then, with assistance of RSUs in storing and forwarding packets, they propose a new routing protocol (SPRING) which achieve conditional privacy preservation and resist most attacks existing in vehicular DTNs.

## 2.2 Topology Control in Ad Hoc and Sensor Networks

Network topology is a key functional issue in designing wireless ad hoc and sensor networks. For different network applications, network topology can be controlled under different objectives (such as power efficiency, fault tolerance, and throughput maximization). Topology control protocols let each node adjust its transmission range and select certain neighbors for communication, while maintaining a topology that can support energy-efficient routing and improve overall network performance. In the past several years, topology control [28–31, 33, 73–80] has drawn a significant amount of research interests. Primary topology control algorithms aim to maintain network connectivity, optimize network throughput with power-efficient routing, and conserve energy. Most topology control protocols can be classified into two categories: *geometrical structure-based* and *clustering-based*. In geometrical structure-based methods, a geometrical structure is constructed based on location information by removing many links from the original communication graph. Each node only selects certain neighbors (neighbors in the constructed structure) for communication. A collection of geometrical structures has been proposed to be used as a network topology for wireless networks [31–33,76–79,81–85]. While all these geometric structures are flat structures, another set of hierarchical structures [86–92] is also used as network topologies in wireless networks. Instead of involving all nodes in relaying packets, hierarchical routing protocols [93–96] select a subset of nodes that serve as backbone routers, forwarding packets for other nodes. These methods often construct a virtual backbone by using clustering formation methods. 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 huge.

Traditional topology control problems usually assume that links are all reliable, however this is not always true in real world wireless networks (especially time-varying networks). Recently, Liu *et al.* [97] have proposed novel opportunity-based topology control in wireless sensor networks. For each link in the network they set a variable called *link reliability* to indicate the probability that a package can be successfully relied through the link. Based on *link reliability* they can calculate node reachability, and their opportunistic-based topology problem tries to minimize the total cost of the network while maintain the nodes reachability up a given threshold. The authors show that this problem is NP-hard and give an approximation algorithm. Similarly, fault-tolerant topology control problem has been studied. For example, Hajiaghayi *et al.* [98] propose a power optimization for wireless multi-hop networks which minimizes power while maintaining $k$-fault tolerance. Their methods require all links established by this power setting be symmetric and form a $k$-vertex connected subgraph of the network graph. They prove that it is a NP-hard problem and present an $O(k)$-approximation.

Topology design always can be modeled as a graph theory problem. Here, I review a few related problems in graph theory to my proposed topology control problems in time-varying networks. The *directed Steiner tree* (DST) problem [111] is a known NP-hard problem. Given a directed graph $G = (V, E)$ with weights on the edges, a set of nodes $Q \subseteq V$, and a root node $v_r$, DST tries to find a minimum weight out-branching tree $\mathcal{T}$ rooted at $v_r$, such that all node in $Q$ are included in $\mathcal{T}$. Later we will prove the NP-hardness of our topology design problem by reducing DST problem to our TD problem. The *directed generalized Steiner network* problem is defined as the following: Given a directed graph $G(V, E)$ and a set $X$ of $k$ source and target pairs, find the minimum cost subgraph $H$ of $G$ such that for each source and target pair in $X$, there exists a directed path from the source to the target in $H$. The cost of the subgraph $H$ is the sum of the cost of all the edges in $H$. It is a well-known NP-hard problem. The TD problem we study is a special case of this problem. Charikart *et al.* [34] study directed generalized Steiner network problem, they give an algorithm that achieves an approximation ratio of $O(k^{2/3}\log^{1/3}k)$, where k is the number of pairs of vertices that are to be connected. One of our proposed heuristics is based on their method. Agrawal *et al.* [99] studied the *generalized network stainer problem.* In this problem, the network (a directed graph) has a set of sources and targets, and each pair of source and target have connectivity requirements (number of edge-disjoint paths to connect them). Their goal is to find a minimum cost network with the available links and satisfy all the connectivity requirements. Again, this problem is NP-hard and the authors give an approximation algorithm whose cost is $\log R$ ($R$ is the largest connectivity requirement between any pair of source and target) ratio to the optimal solution.

## 2.3 Modeling for Time-Varying Networks

Modeling the time-evolving networks has been studied in both mobile ad hoc networks [100, 119, 121] and DTNs [120, 122]. Xuan *et al.* [119] first study routing problem in a fixed schedule dynamic network modeled by an evolving graph (i.e., an indexed sequence of subgraphs of a given graph). Then, [121, 122] also use evolving graphs to evaluate various ad hoc and DTN routing protocols. Shashidhar *et al.* [100] study the routing problem in a time-space graph. Liu and Wu [120] also model a cyclic mobispace as a probabilistic time-space 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 time-space graphs. In this dissertation, I investigate the topology design problem in these networks.

## 2.4 Summary

In this chapter, I briefly review some previous research in wireless networks including wireless ad hoc/sensor networks and delay tolerant networks, and topology control study. Some of them are closely related to my topology design problems.

# CHAPTER 3: NETWORK MODEL AND TOPOLOGY DESIGN PROBLEMS

Since traditional static graph cannot model the time-evolving characteristics of wireless networks, I introduce a new time-space graph which take in account both time and space information of the network. Based on the time-space graph model, I define my new topology design problems in time-varying wireless networks. This chapter introduce the time-space model and define my topology design problems.

## 3.1 Network Model: Time-Space Graph

In previously studies, a time-varying network (such as a DTN) usually is represented by a static graph, in which, 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 1.1), traditional static graph model cannot represent such evolution. Thus, a sequence of static graphs is needed to model the time-evolving network. As shown in Figure 1.1(b), each static graph is a snapshot of nodes and their interactions observed at certain time slots. 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 disseration, I use a time-space graph [100] to model such a dynamic time-varying network.

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

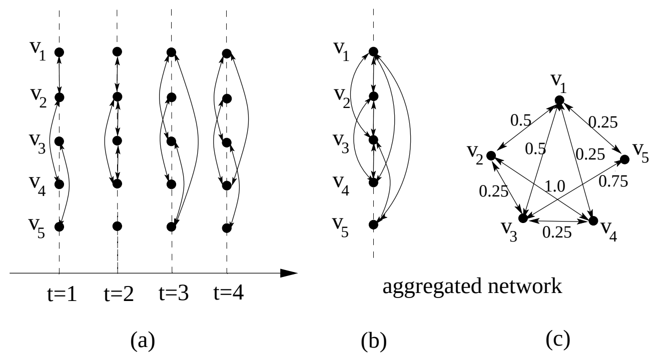


Figure 3.1: Different graph models for the time-evolving network shown in Figure 1.1: (a) a sequence of snapshots of the network at each time slot, denoted by $\{G^t\}$; (b) and (c) its corresponding aggregated graph model.

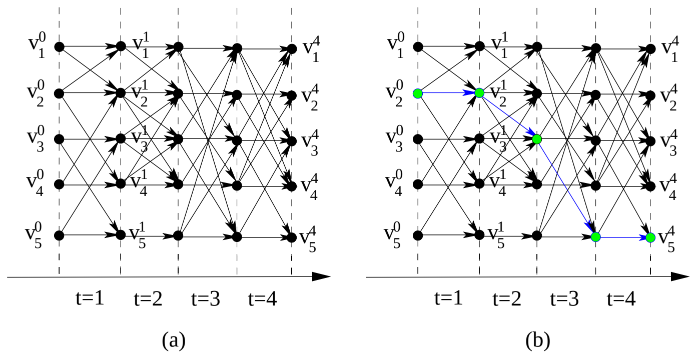


Figure 3.2: Time-space graph model from the same time-evolving network in Figure 3.1: (a) the corresponding time-space graph $\mathcal{G}$; (b) a time-space path (in blue) from the source $v_2$ to the destination $v_5$.

union of series of directed graphs $\{G^t|t = 1 \cdots T\}$. Figure 3.1(a) shows the sequence of snapshots of the network in Figure 1.1 at each time slot. This representation of the network includes all information from both spatial 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 3.1(b) and Figure 3.1(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 $\overrightarrow{v_1v_3}$ exists in two time slots out of total four slots, thus its weight is 0.5. Many DTN routing protocols [17, 23, 24] 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 I convert the sequence of static graphs $\{G^t|t = 1 \cdots T\}$ into a time-space graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is a directed graph defined in both spatial and temporal space. See Figure 3.2 for illustration. In the time-space 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 spatial Figure 3.2) 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 $\overrightarrow{v_j^{t-1}v_k^t}$ represents forwarding a message from one node $v_j$ to its neighbor $v_k$ in the $t$th time slot (i.e., $\overrightarrow{v_jv_k} \in E^t$). By defining the time-space graph $\mathcal{G}$, any communication operation in the time-evolving network can be simulated on this directed graph. As shown in Figure 3.2(b), a path from $v_2^0$ to $v_4^5$ shows a particular routing strategy to deliver the packet from $v_2$ to $v_4$ 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..

Time-space 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 spatial and temporal links). I 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 spatial link or holding a message within one node over a temporal link). The total cost of a time-space 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, I 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 $P(u, v)$.

I also define a *reliable probability* $r(e)$ for each link $e \in \mathcal{E}$, which represents the probability of a successful data transmission over link $e$. Given the reliability of each link, I can then define the reliability of a path $P$ or a structure $H$. In this dissertation, two different types of reliability are considered for DTN topologies: one for single-copy DTN routing, the other for flooding-based DTN routing.

## 3.2 Topology Design Problems

I now define my topology design problems on time-space graphs. In this dissertation I study two topology design problems, one is general topology design (TD) problem, which aims to lower the total cost of the network over the corresponding time-space graph while maintaining connectivity; and the other is cost-efficient topology design (CETD) problem, which aims to lower the total cost while maintain the spanning ratio between any pair of source and target over the time-space graph.

### 3.2.1 Topology Design problem in Time-Space Graph

The aim of topology design is constructing a sparse time-space graph $\mathcal{H}$, which is a subgraph of the original time-space 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. Here *connectivity* has a different definition from that of a static graph. I define that a time-space 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 connected time-space graph does not require connectivity in each snapshot. Hereafter, we assume that the original time-space graph $\mathcal{G}$ is always connected. Figure 3.3 shows an example of

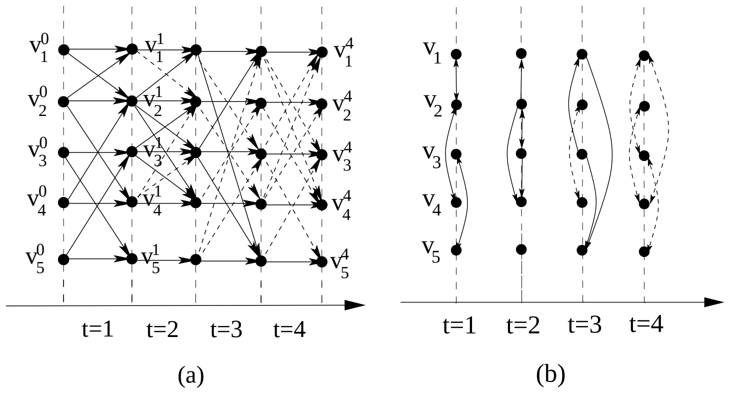


Figure 3.3: Topology design on time-evolving network (the one shown in Figure 1.1): (a) a new connected subgraph $\mathcal{H}$ of $\mathcal{G}$ (green links are removed links from Figure 3.2(a)); (b) its corresponding sequence of static graph with less links than the one of Figure 3.1(a).

topology control on the time-space graph in Figure 3.2(a). After topology design, 16 directed links are removed from the original time-space graph. Not all snapshots are connected, but every node can find the time-space path to any other node. It is interesting to see that there is no spatial 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 design over time domain.

### 3.2.2 Cost-Efficient Topology Design Problem

Comparing to the topology design problem (TD), the cost-efficient topology design (CETD) problem has an additional requirement – maintaining the spanning ratio between any pair of source and destination. The aim of CETD is to construct a sparse time-space graph $\mathcal{H}$, which is a subgraph of the original time-space graph $\mathcal{G}$, such that (1) $\mathcal{H}$ is still connected over the time period $T$; (2) $\mathcal{H}$ is $\delta$-*spanner* of $\mathcal{G}$ over the time period $T$; and (3) the total cost of $\mathcal{H}$ is minimized. Here *spanner* has a *different* definition from the one in static graphs. Given a real number $\delta > 1$, I define that the subgraph $\mathcal{H}$ is $\delta$-*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 $\delta$ times the cost of the shortest path in $\mathcal{G}$, i.e., $\max_{1\leq i,j\leq n}\{\frac{d_{\mathcal{H}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\} \leq \delta$ (Hereafter $\max_{1\leq i,j\leq n}\{\frac{d_{\mathcal{H}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\}$ is called *spanning ratio* of

$\mathcal{H}$). In other words, the constructed subgraph still can support cost-efficient routes for any two devices over the time period. This is obviously a stronger requirement than just connectivity. Thus, CETD is a much harder problem than TD. In addition, if the spanning ratio is set as infinite, CETD becomes TD.

### 3.2.3 Reliable Topology Design Problem

In the time-space model for TD and CETD presented above, the links are assumed reliable. However, in some wireless time-varying networks, the links are not exactly reliable, due to package drop or no guarantee of predictability. To make my network model and topology design problems better fit for those type of netowks, I modify my time-space graph model to take in account the reliability of links. Base on the new model the *reliable topology design problem* (RTD) on weighted time-space graphs can be defined as follows. Given a connected time-space graph $\mathcal{G}$, the aim of *reliable topology design problem* (RTD) is to construct a sparse time-space graph $\mathcal{H}$, which is a subgraph of the original time-space graph $\mathcal{G}$, such that (1) $\mathcal{H}$ is still connected over the time period $T$; (2) the reliability (unicasting or broadcasting reliability) is larger than or equal to a predefined threshold $\gamma$; and (3) the total cost of $\mathcal{H}$ is minimized.

## 3.3 Summary

In this chapter, I introduce the time-space graph model for wireless time-evolving networks. First, I assume links in the time-space graph is reliable, based one which I briefly defined my topology design problems TD, CETD. And later, I assume the links are not exactly reliable, and base on the new assumption I defined the RTD problem.

My topology design problems are different to the standard time-space routing [100, 119], which only aims to find the most cost-efficient time-space path for a pair of source and destination. The topology design problem aims to maintain both cost-efficient and connected time-space routing topology for supporting reliable DTN transmissions between all pairs of nodes.

# CHAPTER 4: TOPOLOGY DESIGN FOR TIME-VARYING NETWORKS

In this chapter, I study the topology design problem in time-space graphs defined in Section 3.2.1. I first show that classic topology design methods for static graph do not work for this new problem and then propose two greedy-based topology design methods which can significantly reduce the total cost of the topology over time-space graph.

## 4.1 Hardness of Topology Design in Time-Varying Networks

The topology design problem for time-space 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 time-space 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 forests of the whole time-space 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. Actually, my topology design problem is a special case of *directed generalized Steiner network* (DGSN) problem [34, 101]. The problem is 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 $H$. Notice that Steiner tree problem is also a special case of DGSN problem, which implies that DGSN problem is a NP-hard problem. The time-space graph $\mathcal{G}$ is a directed graph, and our topology design 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 [101]. However,

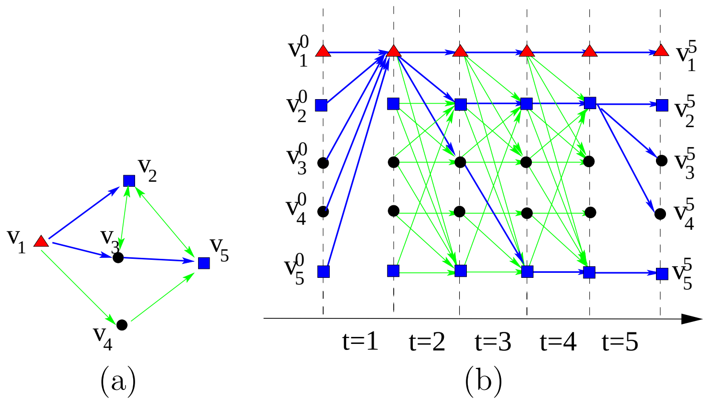


Figure 4.1: Reduction from (a) the directed Steiner tree problem to (b) my TD problem on time-space graphs. Here the blue structures are the corresponding solutions in DST and TD.

their method is very complex.

I now prove the NP-hardness of my TD problem on time-space graphs by using a reduction from the *directed Steiner tree* (DST) problem [111], 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.** *My newly defined* topology design problem on time-space graphs *is a NP-hard problem.*

*Proof.* I first show how to reduce the DST problem into our TD problem on time-space graphs. Given an instance of DST with graph $G = (V, E)$, a root $v_r$, and a set of nodes $Q = \{v_{q_1}, v_{q_2}, \cdots, v_{q_m}\}$ need to reach, I can construct an instance of TD problem on a time-space graph $\mathcal{G}$ as follows. First, all nodes $v_1, \cdots, v_n \in V$ will be the nodes in each snapshot in the time-space $\mathcal{G}$. 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 time-space $\mathcal{G}$ 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 of time slots 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 TD with the same cost in $\mathcal{G}$ for any solution of DST in $G$, and vice versa. Figure 4.1 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 TD on time-space graphs is also NP-hard. $\square$

## 4.2 Topology Design for Directed Time-Varying Networks

In this section, I present two efficient algorithms to construct a sparse structure that fulfills the connectivity requirement over a time-space graph. Both algorithms are based on greedy algorithms, one has theoretical bound on its performance while the other one is practically more efficient.

### 4.2.1 Union of Least Cost Path Algorithm (ULCP)

One naive method for maintaining the network connectivity is to take the union of several least cost path trees $T_i$ rooted at each node $v_i^0$ at the initial snapshot. Each tree $T_i$ is the union of least cost paths from $v_i^0$ to $v_j^T$ for $j = 1, \cdots, n$. Obviously, the resulting graph still keeps the network connectivity. Algorithm 6 shows the detail algorithm. 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 algorithm are running on the time-space graph with $(T+1)n$ nodes and at most $Tn^2$ edges. Hereafter, I refer this method as *union of least cost paths method* (ULCP).

### 4.2.2 Greedy Algorithm Based on Least Cost Path

Even though the structure built by the ULCP method can maintain the connectivity over time, it may contain more links than necessary. Therefore, I propose a new greedy algorithm (as shown in Algorithm 7) to further improve the performance. The basic idea is quite simple and as follows. Initially, I need to connect $n^2$ pair

**Algorithm 1:** Union of Least Cost Path Algorithm

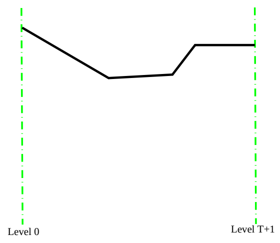
```
1: $\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^T)\}$ for all integer $1 \leq i, j \leq n$.
2: for all pairs $(v_i^0, v_j^T) \in X$ do
3:    Find the least cost path $P_{\mathcal{G}}(v_i^0, v_j^T)$ in $\mathcal{G}$.
4:    if $e \in P_{\mathcal{G}}(v_i^0, v_j^T)$ and $e \notin \mathcal{H}$ then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

of nodes in $X$. In each round I 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 4.2(a). Then I 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 4.2(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 least cost path tree method. I refer this method as *greedy method based on least cost path* (GLCP). The time complexity of this algorithm is $O(Tn^5 + Tn^4 \log(Tn))$ since in each round $n$ times of Dijkstra algorithm are running on the time-space graph and there are $n^2$ rounds.

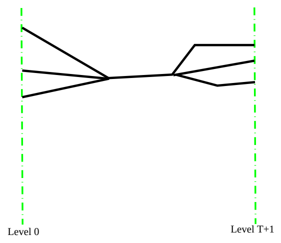
**Algorithm 2:** Greedy Algorithm based on Least Cost Path

```
1: $\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^T)\}$ for all $i$ and $j$ in $[1, n]$.
2: while $X \neq \phi$ do
3:    find the least cost paths for every pair 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 $\mathcal{H}$
```

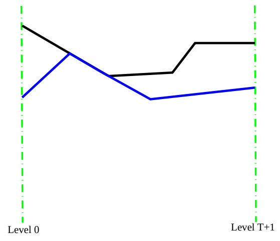
Notice that both ULCP and GLCP may not lead to the optimal solution. Figure 4.3 shows such an example: a network with two nodes ($v_a$ and $v_b$) and 2 time



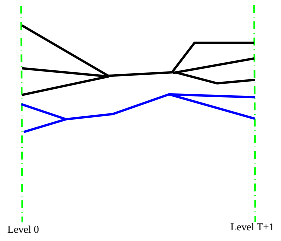
(a) Step 1 - GLCP



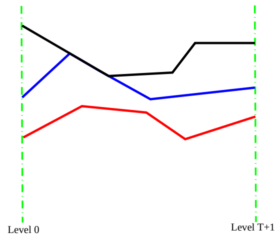
(d) Step 1 - GLDB



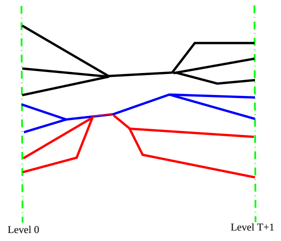
(b) Step 2 - GLCP



(e) Step 2 - GLDB



(c) Step 3 - GLCP



(f) Step 3 - GLDB

Figure 4.2: Illustrations of Algorithm 7 and Algorithm 3: (a-c) Algorithm 7 repeatly adds one least cost path into the topology to connect one pair of nodes in $X$. (d-f) Algorithm 3 repeatly 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.

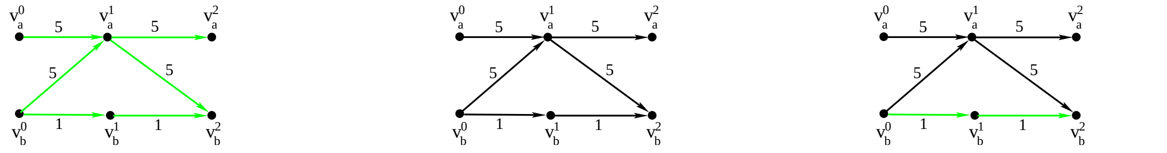
slots. Both ULCP and GLCP generate a structure with total cost 22 as shown in Figure 4.3(b) (since they will first pick the path/bunch including $\overrightarrow{v_b^0 v_b^1}$ and $\overrightarrow{v_b^1 v_b^2}$), while the optimal solution is a structure with total cost 20 (Figure 4.3(c)). However, GLCP and ULCP can perform well when the network is much denser and with more nodes.

### 4.2.3 Greedy Algorithm Based on Least Density Bunch

Next, I present another more complex greedy algorithm (as shown in Algorithm 3) which is inspired by a method proposed by Charikar *et al.* for directed generalized Steiner network problem in [34]. The main idea of the algorithm is to find good bunches repeatedly instead of finding least cost paths repeatedly. Here, a *bunch* $B$

**Algorithm 3:** Greedy Algorithm based on Least Density Bunch

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



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

Figure 4.3: An example: (a) original time-space graph; (b) output of all greedy-based algorithms (ULCP, GLCP, and GLDB); and (c) optimal solution of topology design. Black links are final links in each solution.

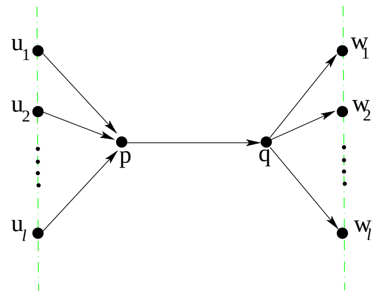


Figure 4.4: Illustration of a bunch connecting $l$ pairs of nodes. Here, each arrow represents a directed least cost path. The total cost of this bunch is $c(B) = c(p, q) + s[u_1, w_1] + s[u_2, w_2] + \cdots + s[u_l, w_l]$.

is a structure connected certain pair of nodes in $X$ as shown in Figure 4.4. In this figure, the edge between any two nodes $u$ and $v$ is a virtual edge which represents the least cost path $P(u, v)$ from $u$ to $v$ in $\mathcal{G}$. The cost of $P(u, v)$ is $c(u, v)$. I 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 4.2(d-f) show the procedure. I refer this method as *greedy method based on least density bunch* (GLDB).

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 $O(T^2n^2)$ times; and the sorting can be done in $O(n^2 \log n)$. Though with larger time complexity than GLCP and ULCP, GLDB algorithm has a nice property in theory: it can achieve approximation-guarantee in term of the total cost compared with the optimal solution. In [34], 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, I have the following theorem for my topology design problem, since $k = n^2$.

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

Notice that Algorithm 3 will also lead to the non-optimal solution for the example shown in Figure 4.3.

## 4.3 Topology Design for Undirected Time-Varying Networks

So far I consider a directed time-varying network where the static graph in each snapshot $G^t$ is a directed graph. Therefore, directed links $\overrightarrow{v_i^{t-1}v_j^t}$ and $\overrightarrow{v_j^{t-1}v_i^t}$ in the time-space 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 4.3) 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 design algorithm can only use one of these two links in the resulting topology (as shown in Figure 3.3). However, in some network applications, the connection between nodes is necessary to be symmetric and undirected. Therefore, in this section, I 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 $\overrightarrow{v_i^{t-1}v_j^t}$ and $\overrightarrow{v_j^{t-1}v_i^t}$ in the time-space 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 time-space 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, I then propose two methods to convert the undirected network into a directed network.

### 4.3.1 Converting Method One - Double Cost

The first method is straightforward. Consider the sequence of static undirected graph $\{G^t\}$. For an undirected link $v_i^t v_j^t \in G^t$, I 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 4.5(a)). I call this converted time-space 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 my algorithm for path/bunch selection (or the total cost of the output structure) is different from the real cost to support such structure.

Next, I prove an important lemma which can still guarantee the approximation of my approach.

**Lemma 1.** *For topology design 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 time-space graph $\mathcal{G}$.*

*Proof.* Consider the solution of $Opt_B$ first. If $\overrightarrow{v_i^{t-1} v_j^t} \in Opt_B$ but $\overrightarrow{v_j^{t-1} v_i^t} \notin Opt_B$, I 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 $S_B$, I can construct a feasible solution $S_A$ of the topology design 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 counts one cost for both directed links. At last, since $Opt_A$ is the optimal solution of the topology design problem in the original undirected graph, $c(Opt_A) \leq c(S_A)$. Therefore, $c(Opt_A) \leq c(S_A) \leq c(S_B) \leq 2c(Opt_B)$. □

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

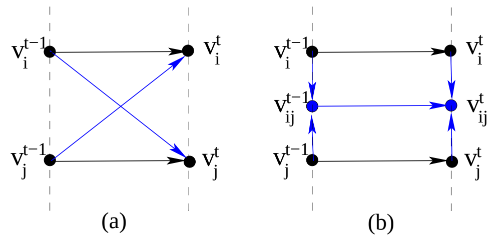


Figure 4.5: Illustration of construction of spatial links in the time-space graph: (a) converting method one - doubling the cost; (b) converting method two - adding new nodes.

### 4.3.2 Converting Method Two - New Graph

The second method converts the sequence of static undirected graph $\{G^t\}$ into a new time-space graph by introducing new nodes. For a undirected link $v_i^t v_j^t \in G^t$, I first add two new nodes $v_{ij}^{t-1}$ at $(t-1)$th layer and $v_{ij}^t$ at $t$th 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 time-space 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 4.5(b) illustrates this procedure. It is obvious that there is one-to-one mapping between the constructed time-space graph and the original undirected network. Then our proposed methods can be directly applied on the constructed directed time-space 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 my algorithms.

## 4.4 Summary

In this chapter, I first study the TD problem in directed graph. I analyze the hardness of the problem and prove that it is NP-hard. Then I propose several greedy-based algorithms to solve the problem. I then show two methods to covert undirected time-space graph to be directed and solve it with the some algorithms.

# CHAPTER 5: COST-EFFICIENT TD FOR TIME-VARYING NETWORKS

In this chapter, I study the cost-efficient topology design (CETD) problem defined in Section 3.2.2. I first show that classic topology design methods for static graph do not work for this new problem and then propose three efficient topology design methods which can significantly reduce the total cost of topology while maintaining the connectivity and cost-efficiency over time.

## 5.1 Hardness of CETD Problem

The cost-efficient topology design problem for time-space graph is much harder than that for a static graph. For a static graph without time domain, there are several efficient methods to construct a spanner (such as [28–31, 33] for unit disk graphs using geometric properties or [102] for general weighted graphs). However, in a time-evolving DTN modeled by a time-space graph, simply applying spanner methods for unit disk graph in each time slot is not feasible since the the network in each single snapshot may not be connected at all. On the other hand, the classic greedy method for general graph [102] over the whole time-space 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 $d_H(u,v) \geq \delta d_G(u,v)$. However, in the time-space 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 TD problem is a special case of CETD problem where $\delta$ is set to be infinity. I showed that TD is also very challenging problem by linking it to a well-known NP-hard problem, *directed generalized Steiner network* (DGSN) problem [34, 101].

## 5.2 Cost-Efficient Topology Design Algorithms

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

### 5.2.1 Union of Least Cost Path Algorithm (ULCP)

Notice that the ULCP can maintain the cost-efficient paths between any two nodes over the time period $T$, since it keeps all the least cost paths from $v_i^0$ to $v_j^T$ for $i, j = 1, \cdots, n$. Obviously, the resulting structure satisfies the spanner property, since $\delta \geq 1$. Algorithm 6 shows the detail algorithm.

### 5.2.2 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 ULCP) in a decreasing order based on link cost. Link $v_i^t v_j^{t+1}$ is removed if without this link the subgraph $\mathcal{H}$ is still a $\delta-$spanner of the original graph $\mathcal{G}$. Algorithm 9 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 least cost paths. Thus, the time complexity of this part is $O(n^2T \cdot n^2T(\log(nT) + n)) = O(n^4T^2(\log(nT) + n))$. Therefore, total time complexity of GDL is also in order of $O(n^4T^2(\log(nT) + n))$, which is much higher than that of ULCP. Notice that since the output of ULCP is much sparser than the original graph, if I use it as the input of GDL it will save certain computations.

### 5.2.3 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, I try to connect $n^2$ pair of nodes in

**Algorithm 4:** Greedy Algorithm to Delete Links

```
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 $\mathcal{H}$
```

$X = \{(v_i^0, v_j^T)\}$ for all integer $1 \le i, j \le n$. In each round I 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, I have a structure $\mathcal{H}$ which connects all pairs of $(v_i^0, v_j^T)$. In the second step, for each pair $(v_i^0, v_j^T)$, I calculate its least cost path in $\mathcal{H}$. If the cost of $P_{\mathcal{H}}(v_i^0, v_j^T)$ is greater than $\delta$ times of the cost of $P_{\mathcal{G}}(v_i^0, v_j^T)$ in the original graph, I directly add the links in this least cost path into $\mathcal{H}$. See Algorithm 10 for detail. Hereafter, I denote this method as GAL. The complexity of GAL is less than the one of *GDL* algorithm. Both steps need $n^2 \cdot O(n^2T(\log(nT) + n))$ time since they both run computation of least cost paths for $n^2$ rounds. Therefore, the total time complexity is $O(n^4T(\log(nT) + n))$.

## 5.3 Summary

In this chapter, I study the cost efficient topology design problem CETD. Since the problem is NP-hard, I propose three greedy-base algorithms which can significantly reduce the cost of the topology while maintain the connectivity and guarantee spanning ratio of the network.

**Algorithm 5:** Greedy Algorithm to Add Links

```
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 $\mathcal{H}$
```

# CHAPTER 6: RELIABLE TD FOR TIME-EVOLVING NETWORKS

In this chapter, I study the reliable topology design problem in time-space graphs defined in Section 3.2.3. I first show my methods to calculate the combined reliability between a pair of nodes for both single-copy and broadcast routing in the time-space graph. Then I show this RTD problem is NP-hard, and then propose several greedy-based topology design methods which can significantly reduce the total cost of the topology over time-space graph while maintain the required reliability.

## 6.1 Reliability of DTN Topology over Unreliable Links

To consider the reliability of lossy wireless links or unperfected link predications, I also define a *reliable probability* $r(e)$ for each link $e \in \mathcal{E}$, which represents the probability of a successful data transmission over link $e$. The unreliability of the link could come from either link failure due to lossy wireless signal or poor predication of future links. Here, I assume that the reliable probability of each link can be obtained through link estimation techniques at the link and physical layers [104] or mobility predication techniques [105, 106].

Figure 6.1(a) illustrate a simple time-space graph with just two devices. The label of each link $e$ is the pair of cost and reliability, i.e., $(c(e), r(e))$. For example, in the first time slot, the cost of spatial link $\overrightarrow{v_a^0 v_b^1}$ is 3 and its reliability is 0.6. Given the reliability of each link, I can then define the reliability of a path $P$ or a structure $H$. In this dissertation, two different types of reliability are considered for DTN topologies: one for single-copy DTN routing, the other for flooding-based DTN routing.

In a single-copy DTN routing, there is only one copy of each message in the network. When a node with a message meets other nodes, it just picks one node as the forwarding node to relay the message or keeps holding the message. Therefore,

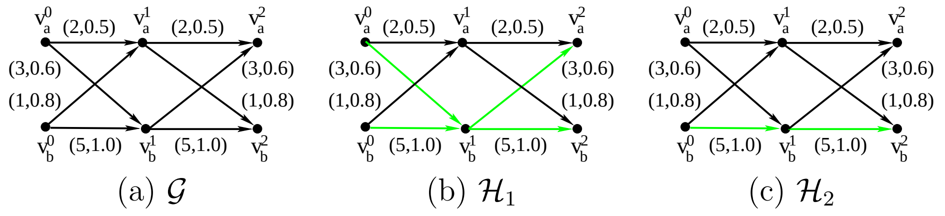


Figure 6.1: Examples of reliable topology design (a) the original space-time graph $\mathcal{G}$ with total cost 22, unicasting reliability 0.36, and broadcasting reliability 0.52; (b) a connected topology $\mathcal{H}_1$ with total cost 6, unicasting reliability 0.25, and broadcasting reliability 0.25; and (c) a connected topology $\mathcal{H}_2$ with total cost 12, unicasting reliability 0.36, and broadcasting reliability 0.4. Here, links are labeled by (cost, reliability), and green links are removed links from $\mathcal{G}$.

the resulting propagation path of a message is basically a single time-space path in $\mathcal{G}$. Given a path $P(u, v)$ connecting nodes $u$ and $v$, the reliability of $P(u, v)$ is the production of reliability of all links in that path. For example, the path $v_a^0 \to v_a^1 \to v_b^2$ in Figure 6.1(a) has reliability of $0.5 \times 0.8 = 0.4$. For a given routing topology $\mathcal{H}$, I can define the *most reliable path* $P_r^{\mathcal{H}}(u, v)$ as the path from $u$ to $v$ in $\mathcal{H}$ with the highest reliability. Let $r^{\mathcal{H}}(u, v) = \Pi_{e \in P_r^{\mathcal{H}}(u,v)} r(e)$ be the reliability of path $P_r^{\mathcal{H}}(u, v)$. For example, between $v_a^0$ and $v_b^2$ in Figure 6.1(a), $v_a^0 \to v_b^1 \to v_b^2$ is the most reliable path with reliability of $0.6 \times 1.0 = 0.6$. Then the *reliability* of the topology $\mathcal{H}$ is defined as follows:

$$r(\mathcal{H}) = \min_{1 \le i,j \le n} r^{\mathcal{H}}(v_i^0, v_j^T). \tag{6.1}$$

The reliability of structures shown in Figure 6.1(a), (b) and (c) are 0.36, 0.25 and 0.36 respectively. Notice that it is easy to calculate $r(\mathcal{H})$ given $\mathcal{H}$ by using any shortest path algorithms. Hereafter, I call this type of reliability *unicasting reliability.*

Empirical and analytical studies [107, 108] have shown that allowing multiple copies propagation in DTN routing can significantly improve the chances of successful delivery. The simplest multi-copies DTN routing is flooding routing or epidemic routing [22, 123], where a node with a message will relay it to every node it encounters. This type of flooding-based routing protocols propagate the copies of the message via multiple paths which leads to larger reliability. Therefore, I define a new type of reliability, *broadcasting reliability*, as follows. For a given source-destination pair $u, v$ in $\mathcal{H}$, the $r^{\mathcal{H}}(u, v)$ is the probability that a packet sent from node $u$ over the routing topology $\mathcal{H}$ reaches node $v$ under flooding-based DTN routing. To efficiently calculate the pair-wise broadcasting reliability is not an easy job. Actually, it is known that the computation of such reliability over general graphs is a problem of NP-hard [110]. Fortunately, the time-space graph in my model is a very special directed acyclic graph where all paths from the sources to the destinations are $T$ hops and there is not any loop. This property allows us to compute the reliability $r^{\mathcal{H}}(u, v)$ efficiently by using dynamic programming. I will present such an algorithm in Section 6.4. With $r^{\mathcal{H}}(u, v)$, the definition of the reliability of $\mathcal{H}$ is straightforward and same as Equation (6.1). The broadcasting reliability of structures shown in Figure 6.1 (a), (b) and (c) are 0.52, 0.25 and 0.4 respectively, which are larger than or equal to their unicasting reliability.

## 6.2 Hardness of RTD Problem

Notice that the topology design problem for time-space graph even without reliability requirement is much harder than the one for a static graph. Plus the reliability requirement, the reliable topology design problem becomes more challenging.As I have proven the topology design problem studied in Chapter 4 is NP-hard, and it a special case of RTD. If I set each link's reliability to be 1, it become TD problem, thus the RTD problem is also NP-hard.

## 6.3 Reliable TD for Single-Copy DTN Protocols

Since RTD over time-space graphs is NP-hard, I now propose five different heuristics to construct a sparse structure that fulfills the connectivity and reliability requirements over time for single-copy DTN routing protocols. The first three heuristics are based on minimum cost reliable routing, while the other two are based on greedy algorithms. For all algorithms, the inputs are the original graph $\mathcal{G}$ and reliability requirement $\gamma \leq 1$, and the output is a subgraph $\mathcal{H}$ of $\mathcal{G}$.

### 6.3.1 Heuristics Based on Min Cost Reliable Path

One natural idea to construct a low-cost reliable subgraph, which guarantees the route reliability between any two nodes over the time period $T$, is keeping all the minimum cost reliable paths from $v_i^0$ to $v_j^T$ for $i, j = 1, \cdots, n$ in $\mathcal{G}$. Here the *minimum cost reliable path* connecting $u$ and $v$, denoted by $P_{cr}^{\mathcal{G}}(u, v)$, is the path with the least cost among all paths between $u$ and $v$ which have reliability larger than or equal to $\gamma$. Obviously, if I can find such paths for every pair of source and destination, the union of all them satisfies the reliability requirement of my topology design problem. However, to find the shortest path with additional constrains in a graph itself is also a well known NP-hard problem, called *restricted shortest path* problem [115,116]. Therefore, in this dissertation, I use one of the existing heuristics for restricted shortest path, Backward-Forward method (BFM) by Reeves and Salama [117].

The basic idea of Backward-Forward method is quite simple. Assume I want to find a reliable path from $s$ to $t$. BFM first determines the *least-cost path* (LCP) and the *most reliable path* (MRP) from every node $u$ to $t$. It then starts from $s$ and explores the graph by concatenating two segments: (1) the so-far explored path from $s$ to an intermediate node $u$, and (2) the LCP or the MRP from node $u$ to $t$. BFM simply uses Dijkstras algorithm with the following modification in the relaxation procedure: a link $uv$ is relaxed if it reduces the total cost from $s$ to $v$ while its approximated end-to-end reliability obeys the reliability constraint. The computational complexity of the BFM is basically three times that of Dijkstra's algorithm. Let $P_{BFM}^{\mathcal{G}}(u, v)$ as the resulting path from $u$ to $v$ based on BFM. Notice that $P_{BFM}^{\mathcal{G}}(u, v)$ is not optimal for restricted shortest path, i.e., not the minimum cost reliable path connecting $u$ and $v$ in $\mathcal{G}$. However, based on simulation study by Kuipers *et al.* [118], BFM is one of the most efficient methods among all existing methods, it has small execution time and often generates good quality path compared with the optimal.

With Backward-Foward method, my first heuristic approach is to find a "minimum" cost reliable path for each pair of source and destination and take the union of them to form the subgraph which guarantee the overall reliability. Algorithm 6 shows the detailed algorithm. Its time complexity is $O(n^2T(\log(nT)+n))$ since I only need to compute $n^2$ minimum cost reliable paths of $\mathcal{G}$ (with $O(nT)$ nodes and at most $O(n^2T)$ links). This can be easily achieved by running $3n$ times of Dijkstra's algorithm whose complexity is $O(nT(\log(nT)+n))$. Hereafter, I refer this method as *union of minimum cost reliable path algorithm* (UMCRP).

**Algorithm 6:** Union of Min Cost Reliable Path (UMCRP)

```
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 "minimum" cost reliable path $P_{BFM}^{\mathcal{G}}(v_i^0, v_j^T)$ in $\mathcal{G}$ using
       Backward-Forward method.
4:     if $e \in P_{BFM}^{\mathcal{G}}(v_i^0, v_j^T)$ and $e \notin \mathcal{H}$ then
5:         $\mathcal{H} = \mathcal{H} \cup \{e\}$.
6:     end if
7: end for
8: return $\mathcal{H}$
```

However, the structure built by the above method may contain more links than necessary. Therefore, I propose a new greedy algorithm (as shown in Algorithm 7) to further improve the performance. The basic idea is quite simple and as follows. Initially, I need to connect $n^2$ pair of nodes in $X$. In each round I pick the minimum cost reliable path between a pair nodes in $X$ which is the minimum among all minimum cost reliable paths connecting any pair of nodes in $X$. Then I add all links in this path into $\mathcal{H}$, clear the costs of these links to zeros, and remove this pair from $X$. This procedure is repeated until all pair nodes $(v_i^0, v_j^T)$ are guaranteed to be connected by reliable paths in $\mathcal{H}$. It is clear that the output of this method is much sparser than the one of UMCRP method. I refer this method as *greedy method based on minimum cost reliable path* (GMCRP). The time complexity of this algorithm is $O(Tn^5 + Tn^4 \log(Tn))$ since in each round $3n$ times of Dijkstra's algorithm are running on the time-space graph and there are $n^2$ rounds.

**Algorithm 7:** Greedy Algorithm with Min Cost Reliable Path (GMCRP)

```
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 minimum cost reliable paths for every pair nodes in $X$ using BFM in
       $\mathcal{G}$, and assume $P_{BFM}(v_i^0, v_j^T)$ has the least cost among these paths.
4:     if $e \in P_{BFM}(v_i^0, v_j^T)$ 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 $\mathcal{H}$
```

**Algorithm 8:** Least Cost Path or Min Cost Reliable Path (LCP/MCRP)

```
1: $\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^T)\}$ for all integer $1 \leq i, j \leq n$.
2: for all pairs $(v_i^0, v_j^T) \in X$ do
3:     Find the least cost path $P_c^{\mathcal{G}}(v_i^0, v_j^T)$ in $\mathcal{G}$.
4:     if $e \in P_c^{\mathcal{G}}(v_i^0, v_j^T)$ and $e \notin \mathcal{H}$ then
5:         $\mathcal{H} = \mathcal{H} \cup \{e\}$.
6:     end if
7: end for
8: for all pairs $(v_i^0, v_j^T) \in X$ do
9:     if $r^{\mathcal{H}}(v_i^0, v_j^T) < \gamma$ then
10:        Add all links $e \in P_{BFM}^{\mathcal{G}}(v_i^0, v_j^T)$ into $\mathcal{H}$ if $e \notin \mathcal{H}$.
11:    end if
12: end for
13: return $\mathcal{H}$
```

The third method basically combines the least cost path and the minimum cost reliable path. It includes two steps. In the first step, I connect $n^2$ pair of nodes in $X = \{(v_i^0, v_j^T)\}$ for all integer $1 \leq i, j \leq n$ by adding all of the least cost paths between them. After this step, I have a structure $\mathcal{H}$ which connects all pairs of $(v_i^0, v_j^T)$ but may not satisfy the reliability requirements yet. In the second step, for each pair $(v_i^0, v_j^T)$, I calculate its reliability in $\mathcal{H}$. If the cost of $r^{\mathcal{H}}(v_i^0, v_j^T)$ is less than $\gamma$, I directly add the links in the minimum cost reliable path between this two nodes in $\mathcal{G}$ into $\mathcal{H}$. See Algorithm 8 for detail. Hereafter, I denote this method as LCP/MCRP. Both steps of LCP/MCRP need $n^2 \cdot O(n^2T(\log(nT) + n))$ time since they both run computation of shortest paths for $O(n^2)$ rounds. Therefore, the total time complexity is $O(n^4T(\log(nT) + n))$.

## 6.3.2 Greedy-based Heuristics

Both the fourth and fifth algorithms are based on the same greedy principle, deleting or adding edges in order and checking whether the reliability is achieved or void.

The basic idea of the fourth method is deleting edges from input graph (either the original graph $\mathcal{G}$ or the output graph from the first two algorithms) in a decreasing order based on link cost. Link $v_i^t v_j^{t+1}$ is removed if without this link the subgraph $\mathcal{H}$ can still achieve reliability of $\gamma$. Algorithm 9 shows the detail, I 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 those of methods based on minimum cost reliable path. Notice that since the outputs of my first two algorithms are much sparser than the original graph, if I use them as the input of GDL it will save certain computation.

**Algorithm 9:** 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 $r(\mathcal{H} - \{e\}) \geq \gamma$ then
5:       $\mathcal{H} = \mathcal{H} - \{e\}$.
6:    end if
7: end for
8: return $\mathcal{H}$
```

**Algorithm 10:** Greedy Algorithm to Add Links (GAL)

```
1: $\mathcal{H} \leftarrow \phi$; $X = \{(v_i^0, v_j^T)\}$ for all integer $1 \leq i, j \leq n$.
2: for all pairs $(v_i^0, v_j^T) \in X$ do
3:    Find the least cost path $P_c^{\mathcal{G}}(v_i^0, v_j^T)$ in $\mathcal{G}$.
4:    if $e \in P_c^{\mathcal{G}}(v_i^0, v_j^T)$ and $e \notin \mathcal{H}$ then
5:       $\mathcal{H} = \mathcal{H} \cup \{e\}$.
6:    end if
7: end for
8: for all $e \in \mathcal{E}$ but $\notin \mathcal{H}$ (processed in increasing order of costs) do
9:    if $r(\mathcal{H}) < \gamma$ then
10:      $\mathcal{H} = \mathcal{H} + \{e\}$.
11:   end if
12: end for
13: return $\mathcal{H}$
```

The fifth algorithm is also a greedy algorithm but processes links in the reverse order compared with the fourth algorithm. It starts from building a connected sparse structure and then adds more links into it to satisfy the reliability requirement. It also includes two steps. In the first step, again I take the union of all least cost paths between $(v_i^0, v_j^T)$ for all integer $1 \leq i, j \leq n$ as $\mathcal{H}$. In the second step, remaining links are processed in the increasing order of cost. Link $v_i^t v_j^{t+1}$ is added to $\mathcal{H}$ if the current subgraph $\mathcal{H}$ cannot achieve reliability of $\gamma$ yet. See Algorithm 10 for detail. Hereafter, I denote this method as GAL. The complexity of GAL is the same as the one of GDL ($O(n^4T^2(\log(nT) + n))$) since the second step is almost the same with GDL and the first step cost less time.

## 6.4 Reliable TD for Flooding-based DTN Protocols

In the above RTD heuristics, I consider the unicasting reliability of a topology for single-copy DTN routing protocols. If multiple copies are allowed during the DTN propagation, the reliability of the topology (the probability of success DTN delivery over the topology) will increases. Therefore, in this section, I study the RTD heuristics under broadcasting reliability for flooding-based DTN protocols.

I first introduce a dynamic programming algorithm (DP) to compute the broadcasting reliability $r^{\mathcal{H}}(u,v)$ of a structure $\mathcal{H}$ for a given source-destination pair $u, v$. Even though such problem is NP-hard in general graphs [110], the nice loop-free property of my time-space graph model allows us to compute the reliability very efficiently with a dynamic programming algorithm. Basically, for any node $v_i^t$ in $\mathcal{H}$, its broadcasting reliability from a source node $s$ can be calculated as follows:

$$r^{\mathcal{H}}(s, v_i^t) = 1 - \prod_{\overrightarrow{v_j^{t-1} v_i^t} \in \mathcal{H}} (1 - r^{\mathcal{H}}(s, v_j^{t-1}) r(\overrightarrow{v_j^{t-1} v_i^t})).$$

For example, in Figure 6.1(c), $r^{\mathcal{H}_2}(v_a^0, v_a^2) = 1 - (1 - 0.5 \times 0.5)(1 - 0.6 \times 0.6) = 0.52$. Given the structure $\mathcal{H}$, starting from a source node, the dynamic programming algorithm can compute the broadcasting reliability of all other nodes within time of $O(nT(\log(nT) + n))$. Notice that the time complexity of DP algorithm is the same with that of Dijkstra's algorithm. I use this DP as a building block in my two RTD heuristics.

Notice that with broadcasting reliability, the union of minimum cost reliable paths or the union of most reliable paths based on unicasting reliability between all source-destination pairs may not achieve the required broadcasting reliability. In addition, there is no effect algorithm to generate a multi-path structure which can minimize the cost while guarantee the reliability for a given source-destination. Therefore, the only possible heuristics I have are two greedy based algorithms (Algorithm 9 and Algorithm 10). Both algorithms can be used for broadcasting reliability. The only difference is now using the DP algorithm to check the reliability of a topology.

## 6.5 Summary

In this chapter, I study the reliable topology design problem in DTN modeled by a probabilistic time-space graph. I first show the problem is NP-hard. And then propose a set of heuristic algorithms which reduce the cost of topology while maintain the reliability of paths over time.

# CHAPTER 7: SIMULATIONS

To evaluate my topology design algorithms in time-varying networks, I have conducted some simulations and are planning to extend my simulations for more advanced work. In this chapter I introduce a topology simulator developed by our research team and present extensive simulation results.

## 7.1 Our Topology Simulator

I have developed a simple simulator in java for our proposed topology design problems. Since our work focuses on topology design of wireless networks, I ignore all details on networking layers and simply assume idea MAC layer and perfect shortest path routing protocol. Our simulations only focus on graphs representing network topologies and topology design or routing algorithms on these graphs.

In our simulator, I model a network as a graph, in which an edge representing a link in the network, and a node representing a wireless device. Various classes of graph are defined to represent different types of wireless networks, such as directed or undirected graphs to represent heterogenous or homogenous wireless networks, unit disk graphs, or arbitrary graphs to simulate networks of different distribution. The simulator also defines various classes for different type nodes and edges. In our simulation a node can hold information of position, list of neighbors, transmission power and can be extended to include other properties to consider in simulation. An edge could be directed or undirected, weighted or unweighted. I simulate interference between any nodes by defining interference range for each node , and detecting when nodes are within each other's interference range. With this design, it is possible to simulate various properties of a node or link by setting the attributes of the corresponding objects. The simulator can generate random graphs or load real-world wireless tracing data and format it as a time-space graph for further use.

I have also implemented the basic algorithms I need during our simulations, such as algorithm of single source shortest path, algorithms to construct broad first search tree and connected dominating set. And when performing a topology design algorithm on a graph, I can track the whole procedure of the algorithm and record performance metrics such as the total cost, total edge number and spanning ratio of the graph generated by topology design algorithms.

## 7.2 Simulations for TD

I have conducted extensive simulations to evaluate the performance of my methods for TD problem presented in Chapter 4. In this section, I present the simulations settings and results which verified the efficiency of my methods.

### 7.2.1 Simulation Settings and Metrics

I will evaluate my proposed topology design algorithms, namely, *Greedy Algorithm with Least Cost Path* (GLCP) and *Greedy Algorithm with Least Density Bunch* (GLDB), by comparing their performances with *Least Cost Path Tree* method (ULCP). 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 [35]. All the networks are directed DTNs, thus I only test my algorithms on directed networks in this section. In all simulations, I take two metrics as the performance measurement for any topology design algorithm:

- **Total Cost:** the total cost of the constructed topology $\mathcal{H}$ (output of the algorithm), i.e., $c(\mathcal{H}) = \sum_{e\in\mathcal{H}} c(e)$.
- **Total Number of Edges:** the total number of edges in the constructed $\mathcal{H}$, i.e., $|\mathcal{H}|$. Here, $|\mathcal{G}|$ denotes the number of edges of graph $\mathcal{G}$.

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

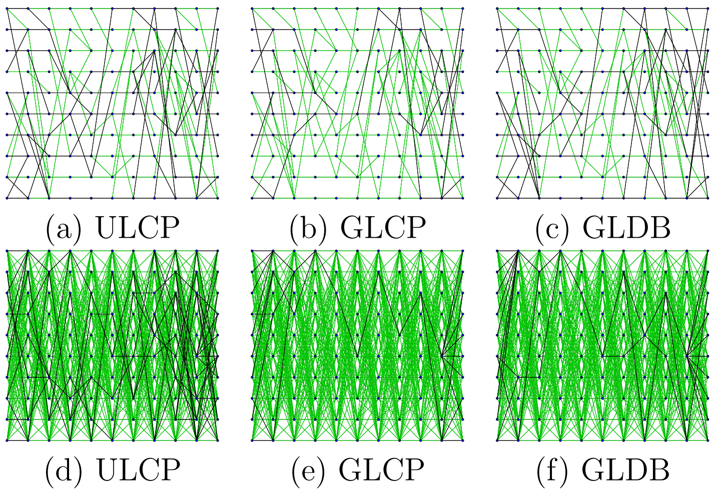


Figure 7.1: Topologies generated over the same random network: green links are removed links from the original graph $\mathcal{G}$, while black links are the ones kept by each algorithm. (a-c): $p = 0.1$, (d-f): $p = 0.8$.

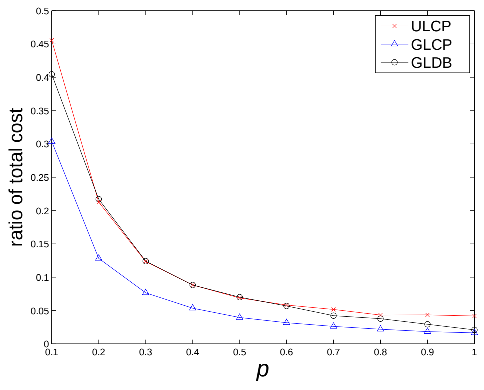
total cost and small number of edges.

### 7.2.2 Simulations on Random Networks

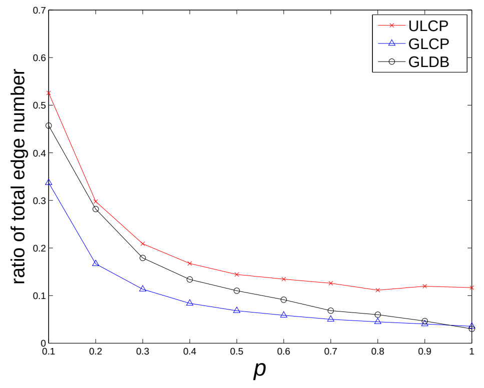
I first generate a sequence of static random graphs $\{G^t\}$ to represent the time-evolving DTN. Here, I first consider a DTN with 10 nodes ($n$=10) and spreading over 10 time slots ($T = 10$). For each time slot $t$, I randomly generate the graph $G^t$ using the classical random graph generator. Basically, for each pair of nodes $v_i, v_j$, I 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 $\{G^t\}$, I convert it into its corresponding time-space graph $\mathcal{G}$ with $n(T+1)$ nodes. Then all three topology design algorithms (ULCP, GLCP, GLDB) take the same $\mathcal{G}$ as the input.

Figure 7.1 shows the constructed topologies in time-space graph format from all three algorithms when $p = 0.1$ and 0.8. It is clear that GLCP 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 time-space 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.

I then increase the network density by rising $p$ from 0.1 to 1.0. Small value of



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



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

Figure 7.2: 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 design algorithm, compared with the original network without topology design.

$p$ leads to a sparse DTN, and $p = 1.0$ implies that the topology in each time slot is a complete graph. For each setting, I generate 50 random networks and report the average performance of topology design among them. Figure 7.2 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 implies how much saving achieved by the topology design algorithm, compared with the original network without topology design. From the results, all topology design 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, GLDB and GLCP can achieve better efficiency than ULCP. However, GLCP has a clear advantage over GLDB. Even though GLDB has the theoretical approximation bound, GLCP 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 design algorithms with dense networks.

I 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.

### 7.2.3 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, I also use real-world wireless tracing data to evaluate my topology design algorithms. Particularly, I select a data set from the Cambridge Haggle data set [35] which is available at CRAWDAD [103]. 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 my simulations, I only consider the 78 mobile nodes and the first half of the period in the tracing data. I 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, I add a spatial link between the two corresponding nodes in $G^t$. For each round of simulation, I extract a slice of the network which contains 10 mobile nodes. Costs are again randomly generated from 1 to 5 for both spatial and temporal links. Then topology design algorithms are performed over these 10-node DTNs. Figure 7.3 shows two examples of those networks. One is relatively sparse and the other is dense.

In my simulation, 13 rounds of simulations are conducted and average measurements are plotted in Figure 7.4. The same conclusions can be drawn from these results: (1) all algorithms can reduce the cost remarkably (more than 95%); (2) ULCP uses the largest cost among three methods, while GLCP has the best performance in practice.

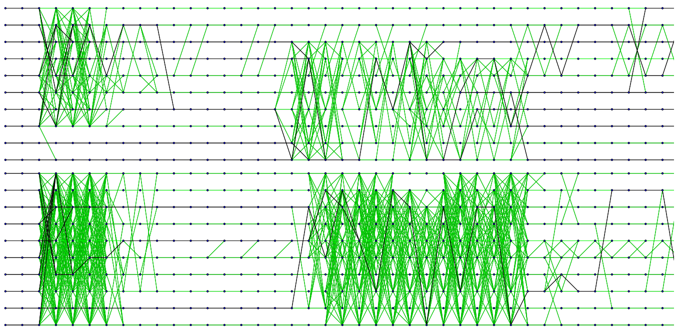


Figure 7.3: Topologies generated over two 10-node/50-timeslot time-evolving networks from Cambridge Haggle [35] tracing data. Here green links are removed links by the algorithm, while black links are the links in the resulting topology. It is clear that more cost can be saved for denser networks.

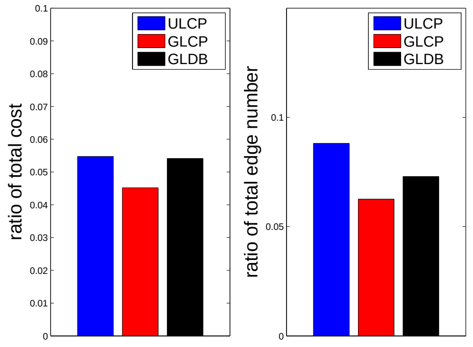


Figure 7.4: Simulation results on networks from Cambridge Haggle [35] tracing data. Results are averages over 13 small networks.

## 7.3 Simulations for CETD

I have conducted extensive simulations to evaluate my proposed algorithms devoted to the CETD problem. I implement and test the following five algorithms: ULCP (Algorithm 6), GDL (Algorithm 9 running on the original graph $\mathcal{G}$), GDL-ULCP (Algorithm 9 running on the output of ULCP), GAL (Algorithm 10), and GLCP (a least cost path based greedy algorithm for TDC problem, basically is the first half of GAL [Lines 1-8 of Algorithm 10]). Recall that UPS has a fix spanning ratio 1 since it preserve the least cost paths for all pairs of nodes in $X$. GLCP algorithm builds a connected topology over time but do not have bounded spanning ratio. I use it as a reference to my proposed algorithms.

### 7.3.1 Simulation Settings and Metrics

During my simulations, the underlying time-evolving networks are either randomly generated from random graph model or directly extracted from Cambridge Haggle tracing data [35]. In all simulations, I take three metrics total cost, number of edges and spanning ratio as the performance measurement for any algorithm devoted to *cost-efficient topology design problem.* The total cost and number of edges are defined and used as metrics for the TD problem. And the additional metrics spanning ratio is defined as following:

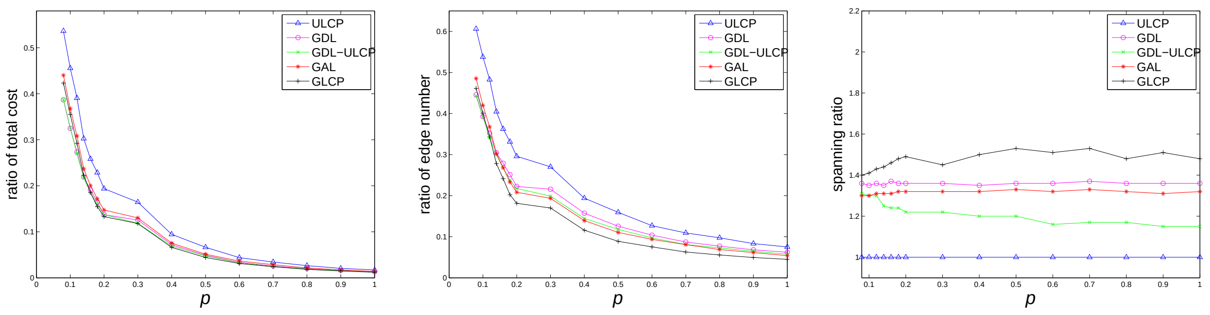
- **Spanning Ratio:** $\max_{1\leq i,j\leq n}\{\frac{d_{\mathcal{H}}(v_i^0,v_j^T)}{d_{\mathcal{G}}(v_i^0,v_j^T)}\}$.

Recall that the objective of my topology design methods is to construct topology structures with small total cost, small edge number and small spanning ratio. For all the simulations, I repeat the experiment for multiple times and report the average values of these metrics.

### 7.3.2 Simulations on Random Networks

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

For the first set of simulations, I increase the network density by rising $p$ from 0.1 to 1.0, and keep spanning ratio requirement $\delta$ at 1.4. Figure 7.5(a) and (b) show the

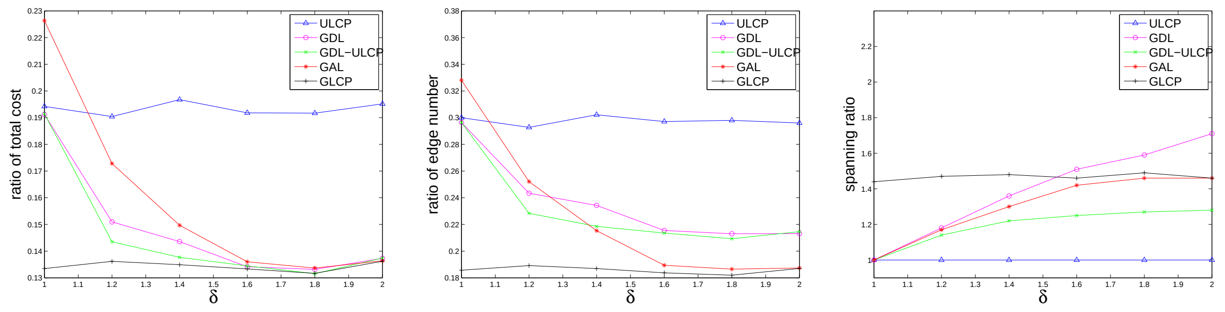


(a) $c(\mathcal{H})/c(\mathcal{G})$ (b) $|\mathcal{H}|/|\mathcal{G}|$ (c) Spanning Ratio

Figure 7.5: 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 design algorithm, compared with the original network without topology design. (c): The spanning ratio shows the path efficiency of $\mathcal{H}$ compared with $\mathcal{G}$.

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 design algorithm, compared with the original network without topology design. From the results, all topology design 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 ULCP. For $p = 1.0$, more than 95% cost is saved. When the network is denser, the saving of topology design is larger. Overall, ULCP has the largest cost and edge number, whille LCP usually has the least. Figure 7.5(c) shows the spanning ratios of all methods. Clearly, LCP is the only algorithm cannot satisify the spanning ratio requirement $\delta = 1.4$. ULCP can always achieve spanning ratio of 1 but with higher cost than other algorithms. The other three algorithms have very close performance in total cost. GDL-ULCP has lower spanning ratio than GDL and GAL, yet they are all bounded by $\delta$.

In the second set of simulations, I fixed the network density $p = 0.2$ and run my algorithms with defferent spanning ratio requirement $\delta = 1.0$ to 2.0. From results shown in Figure 7.6, I can observe that with tighter spanning ratio requirement my



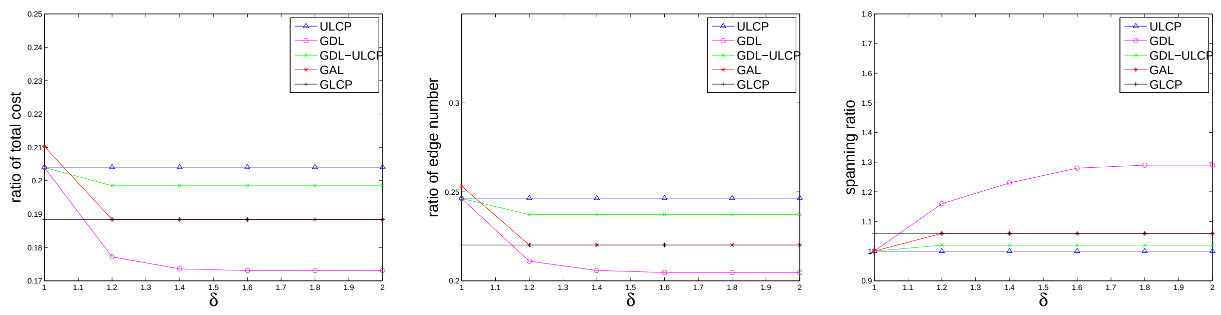
(a) $c(\mathcal{H})/c(\mathcal{G})$ (b) $|\mathcal{H}|/|\mathcal{G}|$ (c) Spanning Ratio

Figure 7.6: Simulation results on random networks with fixed network density $p = 0.2$ and varying spanner ratio.

topology algorithms use higher cost and larger edge number. When the spanning ratio requirement becomes larger, the restrction on removing links becomes weaker. In extreme case, my CETD problem converges to TD. Notice that performance of GLCP and ULCP do not affect by the spanning ratio requirements.

### 7.3.3 Simulations on Tracing Data

To evaluate the performance of CETD, I also conduct simulations on networks extracted from real wold tracing data. Particularly, I select a data set from the Cambridge Haggle data set [35] which is available at CRAWDAD [103]. 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 my simulations, I only consider the 78 mobile nodes and the first half of the period in the tracing data. I 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, I add a spatial link between the two corresponding nodes in $G^t$. For each round of simulation, I extract a slice of the network which contains 10 mobile nodes. Costs are again randomly generated from 1 to 50 for both spatial and temporal links. Then topology design algorithms are performed over these 10-node DTNs. In my simulation, 13 networks have been retrieved from the trace data set, I run my algorithms with spanning ratio requirement from 1.0 to 2.0 on all these 13 networks,



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

Figure 7.7: Simulation results on networks from Cambridge Haggle [35] tracing data. Results are averages over 13 small networks.

and average measurements are plotted in Figure 7.7. The same conclusions can be drawn from these results: all algorithms can reduce the cost remarkably (more than 50%) and my proposed methods can guarantee the spanning ratio requirement.

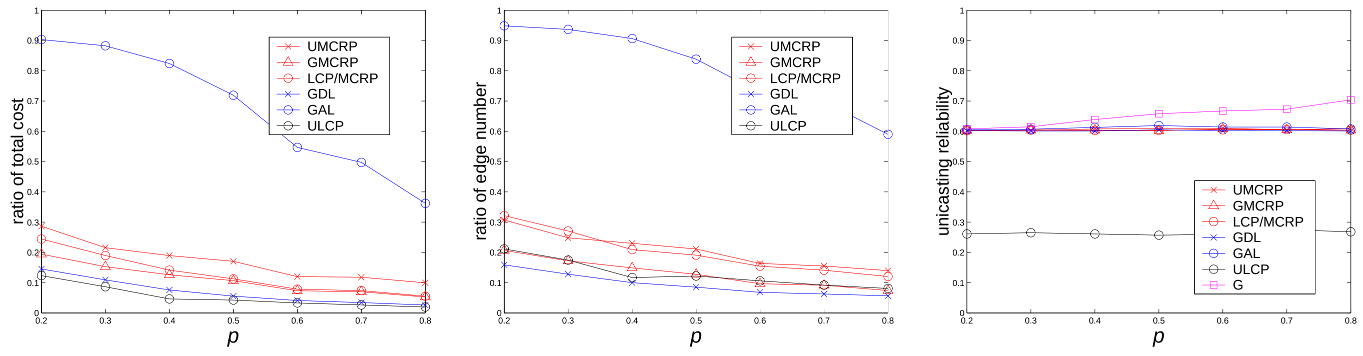
## 7.4 Simulations for RTD

I have conducted extensive simulations to evaluate my proposed algorithms devoted to the RTD problem. I implement and test the following six algorithms: UMCRP (Alg 6), GMCRP (Alg. 7), LCP/MCRP (Alg. 8), GDL (Alg. 9), GAL (Alg. 10), and ULCP. Here, ULCP is the union of least cost paths from $v_i^0$ to $v_j^T$ for $i, j = 1, \cdots, n$), which maintains the connectivity over time but cannot guarantee the reliability. I use ULCP as a reference.

### 7.4.1 Simulation Settings and Metrics

During my simulations, the underlying time-evolving networks are either randomly generated from random graph model or directly extracted from Cambridge Haggle tracing data [35]. In all simulations, I take three metrics total cost, number of edges and reliability as the performance measurement for any algorithm devoted to *reliable topology design problem*. The total cost and number of edges are defined and used as metrics for the TD problem. And the additional metric reliability is defined as following:

- **Reliability:** $r(\mathcal{H}) = \min_{1 \le i,j \le n} r^{\mathcal{H}}(v_i^0, v_j^T)$.



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

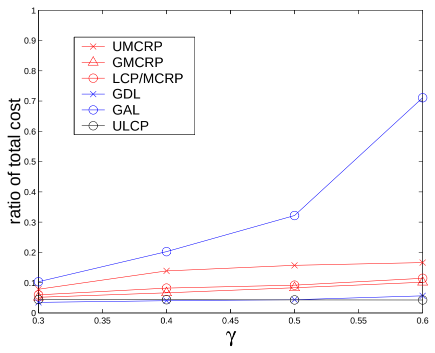
Figure 7.8: Simulation results on random networks with different density and $\gamma = 0.6$. (a)-(b): The cost (or edge number) of $\mathcal{H}$ is divided by the cost (or edge number) of $\mathcal{G}$, which illustrates how much saving achieved by the topology design, compared with the original network. (c): The reliability $r(\mathcal{H})$ shows the unicasting reliability of $\mathcal{H}$.

The objective of my topology design methods is to construct topology structures with small total cost, small edge number and larger reliability.

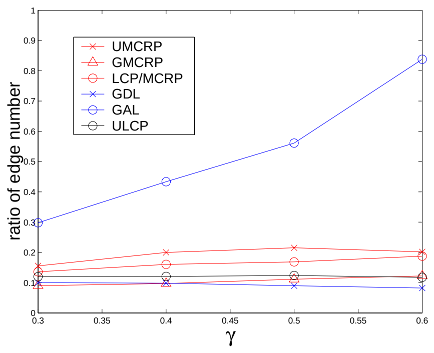
### 7.4.2 Simulations on Random Networks

I first conduct simulations over time-evolving networks are randomly generated from random graph model. I generate a sequence of static random graphs $\{G^t\}$ with 10 nodes ($n$=10), spreading over 10 time slots ($T = 10$). For each time slot $t$, the static graph $G^t$ is generated using the classical random graph generator. For each pair of nodes $v_i$ and $v_j$, I insert edge $\overrightarrow{v_iv_j}$ with a fixed probability $p$ into $G^t$. 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 $\{G^t\}$, I convert it into its corresponding time-space graph $\mathcal{G}$ with $n(T+1)$ nodes. The cost and reliability of each inserted edge are randomly chosen from 1 to 50 and from 0.8 to 1, respectively. Then I perform proposed topology design algorithms on $\mathcal{G}$. For each setting, I generate 100 random networks and report the average performance of topology design algorithms among them.

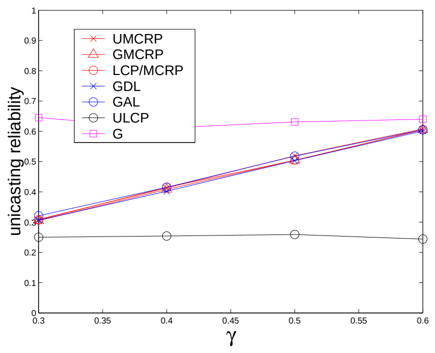
**Topology Design for Unicasting Reliability:** For the first set of simulations, I increase the network density by rising $p$ from 0.2 to 0.8, and keep reliability



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



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



(c) $r(\mathcal{H})$

Figure 7.9: Simulation results on random networks with fixed network density $p = 0.5$ and varying reliability requirement $\gamma$.

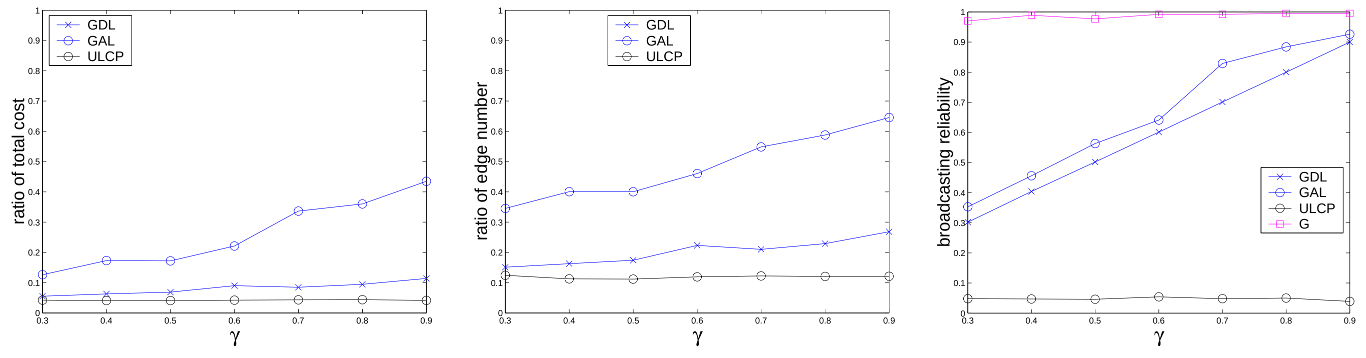
requirement $\gamma$ at 0.6. Figure 7.8(a) and 7.8(b) show the ratios between the total cost/number-edges of the generated graph $\mathcal{H}$ and that of the original graph $\mathcal{G}$ when $p$ increases. This ratio implies how much saving achieved by the topology design algorithm, compared with the original network. From the results, all algorithms can significantly reduce the cost of maintaining the connectivity. Even with the least density ($p = 0.2$), all algorithms except for GAL can save more than 70% cost and around 65% edges. When the network is denser, the saving of topology design is larger. For $p = 0.8$, more than 85% cost is saved. Comparing all methods, I can find the following observations. (1) ULCP has the least cost since it only guarantee the connectivity while other proposed algorithms guarantee the reliability; (2) GAL has the largest cost due to simply adding many low-cost links; (3) GMCRP achieves smaller cost than UMCRP since it reuse some links in previous rounds; (4) LCP/MCRP's cost is between ULCP's and UMCRP's since it combines both LCP and MCRP; (5) Overall, GDL and GMCRP have the best costs among the proposed five methods. Figure 7.8(c) shows the reliability of each structure generated by all methods. Clearly, GLCP is the only algorithm that cannot satisfy the reliability requirement $\gamma = 0.6$. It is an algorithm only for maintaining the connectivity over time. All other five algorithms can always achieve the desired reliability, and they have very close performance in term of reliability. But considering their costs, GDL, GMCRP, LCP/MCRP are better choices. When $\mathcal{G}$ is sparse (e.g. $p = 0.2$), its reliability is just over 0.6. In that case, my proposed algorithms remove around 70% edges but still keep the reliability at the desired level. This clearly demonstrates the power of topology design.

In the second set of simulations, I fixed the network density $p = 0.5$ and run my algorithms with different reliability requirement $\gamma$ increasing from 0.3 to 0.6. From results shown in Figure 7.9, I can observe that tighter reliability requirement results in higher cost and larger number of edge-usage of my topology algorithms. When the reliability requirement becomes looser, the restriction on removing links becomes weaker. In the extreme case with $\gamma = 0$, my RTD problem converges to the basic topology design problem [113] which only preserves the connectivity. Notice that performances of GLCP do not affect by the reliability requirement. In addition, GAL performs better when the reliability requirement is small.

**Topology Design for Broadcasting Reliability:** I also implement the dynamic programming algorithm to compute the broadcasting reliability, and test the performances of GDL (Alg. 9), GAL (Alg. 10), and ULCP in random networks. Here,I increase broadcasting reliability $\gamma$ from 0.3 to 0.9 and keep $p$ at 0.5. The results are plotted in Figure 7.10. It is obvious that both GDL and GAL can guarantee the reliability, but GDL is more efficient in term of cost. Compared with unicasting reliability, less links and smaller cost are needed to achieve the same level of reliability. I also preform simulations on networks with different settings (various reliability, size and time length). The results and conclusions are similar, thus they are ignored here due to space limit.

### 7.4.3 Simulations on Tracing Data

I also conduct simulation on real-world wireless tracing data to evaluate my reliable topology design algorithms. Particularly, I select a data set from the Cambridge Haggle data set [35] which is available at CRAWDAD [103]. In this data set, connec-



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

Figure 7.10: Simulation results on broadcasting reliability on random networks with network density $p = 0.5$ and varying reliability requirement $\gamma$.

tions among 78 mobile iMote Bluetooth nodes carried by researchers and additional 20 stationary nodes are recorded over 4 days during IEEE Infocom 2006. In my simulations, I only consider the 78 mobile nodes and the whole period in the tracing data. I divide the time period of the tracing data into 20 time slots. For each time slot $t$, if there is a contact trace which is overlapping with this slot, I add a spatial link between the two corresponding nodes in $G^t$. For each round of simulation, I extract a slice of the network which contains 10 mobile nodes. Costs are again randomly generated from 1 to 50, and reliability is uniformly distributed between 0.8 to 1.0 for both spatial and temporal links. Then topology design algorithms are performed over these 10-node DTNs. In my simulation, 14 subnetworks have been extracted, simulations are conducted over them, and average measurements are plotted.

**Topology Design with Unicasting Reliability:** I at first conduct simulations for my unicast algorithms for RTD problem. In this set of simulation the required unicast reliability is set to be 0.15. Figure 7.11 shows the ratio of total cost of result graphs from all our algorithms over that of the original graphs. Figure 7.12 shows ratio of edge number of results from all our algorithms over that of the original graphs. And Figure 7.13 shows the reliability $\gamma$ of results from all algorithms. The same conclusions can be drawn as simulations on random networks, and it shows once again that my algorithms efficiently solve my RTD problem.

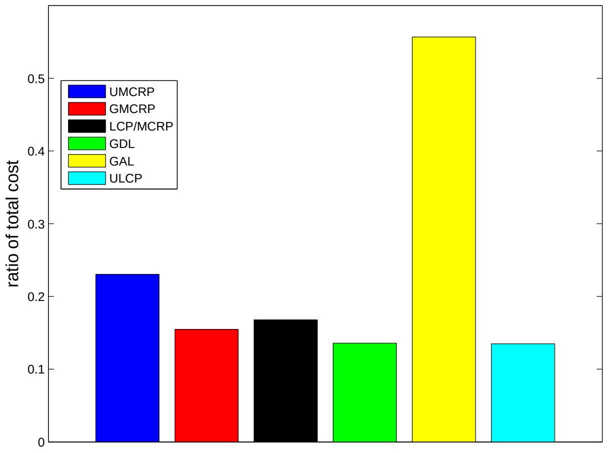


Figure 7.11: Ratio of total cost of unicast algorithms for RTD on tracing data.

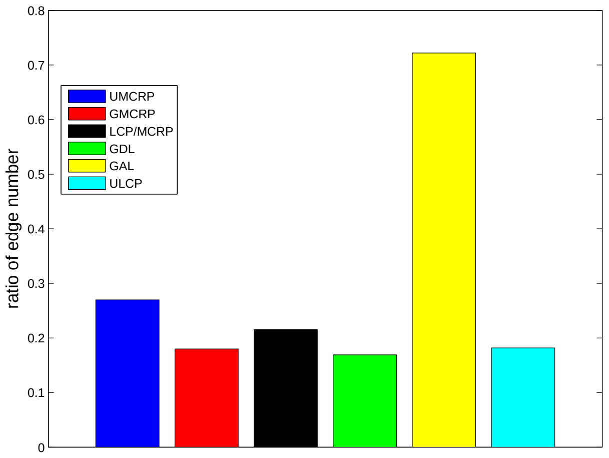


Figure 7.12: Ratio of edge number of unicast algorithms for RTD on tracing data.

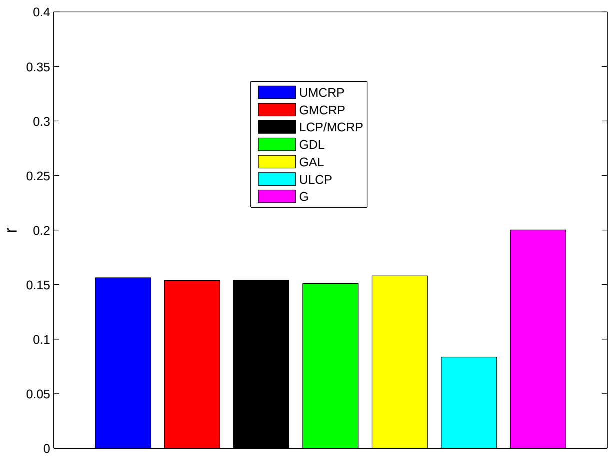


Figure 7.13: Ratio of $r$ of unicast algorithms for RTD on tracing data.

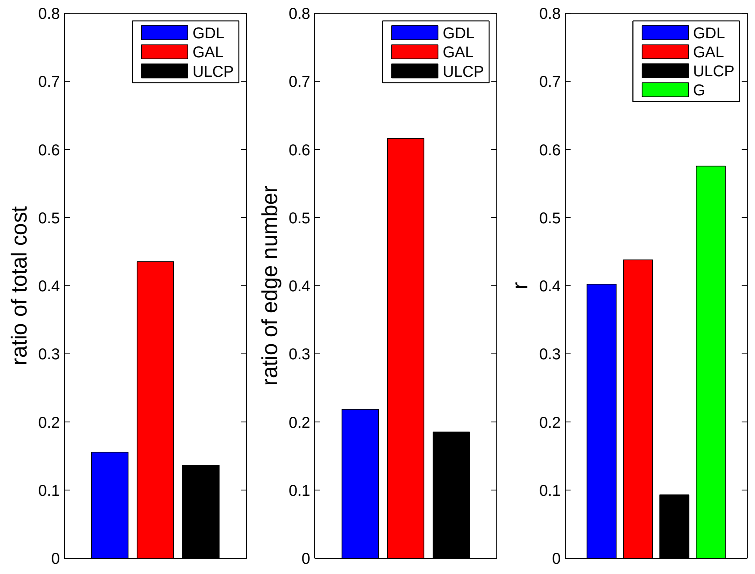


Figure 7.14: Results of broadcast algorithms for RTD on tracing data.

**Topology Design for Broadcasting Reliability:** I then conduct simulations for my broadcasting algorithms for RTD. In this set of simulations, the required broadcast reliability is set to be 0.4. All data for three metrics are shown in Figure 7.14. It is easy to see my broadcasting algorithms also works well in real wold tracing data.

## 7.5 Summary

In this chapter, I present the simulations for all three of my topology design problems, TD, CETD, RTD. All of the simulation results show the effectiveness of my algorithms. For each problem, I have one or more algorithm can significantly reduce the cost of topology while maintain the required objectives.

# CHAPTER 8: CONCLUSION

In this dissertation, I study topology design problems for time-varying wireless networks. By introducing a time-space graph, a predictable time-evolving DTN can be modeled with both spatial and temporal information.

I at first assume the links in the time-space graph are reliable, and then define two topology control problems in time-space graph for time-varying networks: TD and CETD. Both problems are very challenging.

TD problem is a special case of the well-known NP-hard problem, directed generalized Steiner network. By reducing the generalized steiner network to the TD problem, I prove that TD is NP-hard. Since the hardness of the problem, I propose two greedy-based methods for TD, which can significant reduce the cost of topology while maintain the connectivity over time. I also present two techniques which can convert undirected networks into directed networks such that my proposed methods can still work.

CETD problem is even harder than TD, since it has an additional spanning ratio requirement while aims to minimize the total cost of the time-space graph. In fact TD is a special case of CETD, when the spanning ratio requirement is loose enough, CETD become TD. Thus CETD is obviously NP-hard. For the hardness of the problem, rather than finding the optimal solution, I propose several algorithms to construct efficient spanner for time-space graphs.

Both TD and CETD problems study the connectivity of the network. After that, I modify my time-space graph with unreliable or weighted links, base on that I define a new topology design problem RTD, which minimize the cost of the network while maintain its connectivity as well as reliability of the paths. As the TD problem is a special case of the RTD problem, RTD is also NP-hard. And then I give several heuristic algorithms to solve the RTD problem.

To evaluate the performance of my proposed algorithms for all my three topology design problems, I have conducted extensive simulations. At first, I have developed a topology simulator which can simulate the topology structure of wireless networks. With the simulator, I have done a lot set of simulations based on graphs randomly generated and from real world tracing data. The simulation results show the efficiency of my algorithms.

I believe that my work presents the first step in exploiting topology control for time-evolving networks. The following are some possible future works:

- Design more efficient algorithms with lower complexity to achieve connectivity over time-space graphs.
- Extend our simulator and conduct more sophisticated simulations to evaluation our topology control algorithms.
- Study more tracing data of DTNs or social networks from CRAWDAD, and detect their mobility patterns which may be useful for my simulations or design of topology control.

## REFERENCES

[1] David B Johnson and David A Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds., vol. 353. Kluwer Academic Publishers, 1996.

[2] Yu Wang, Ha Dang, and Hongyi 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.

[3] 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.

[4] Pan Hui, Jon Crowcroft, and Eiko Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, New York, NY, USA, 2008, pp. 241–250, ACM.

[5] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot, "Pocket switched networks and human mobility in conference environments," in *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, New York, NY, USA, 2005, pp. 244–251, ACM.

[6] Shu-Yan Chan, Pan Hui, and Kuang Xu, "Community detection of time-varying mobile social networks," in *Proc. of First International Conference on Complex 2009, LNICST 4*, 2009.

[7] 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 *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, New York, NY, USA, 2007, pp. 195–206, ACM.

[8] Michal Piorkowski, Natasa Sarafijanovoc-Djukic, and Matthias Grossglauser, "A Parsimonious Model of Mobile Partitioned Networks with Clustering," in *THE First International Conference on COMmunication Systems and NETworkS (COMSNETS)*, 2009, pp. 1–10.

[9] Augustin Chaintreau, Pierre Fraigniaud, and Emmanuelle Lebhar, "Opportunistic spatial gossip over mobile social networks," in *WOSP '08: Proceedings of the first workshop on Online social networks*, New York, NY, USA, 2008, pp. 73–78, ACM.

[10] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi, "Gossiping (via mobile?) in social networks," in *DIAL M-POMC '08: Proceedings of the fifth international workshop on Foundations of mobile computing*, New York, NY, USA, 2008, pp. 27–28, ACM.

[11] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, New York, NY, USA, 2008, pp. 337–350, ACM.

[12] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, M. Terada, and R. Han, "WhozThat? evolving an ecosystem for context-aware mobile social networks," *Network, IEEE*, vol. 22, no. 4, pp. 50–55, 2008.

[13] Zheng-Bin Dong, Guo-Jie Song, Kun-Qing Xie, and Jing-Yao Wang, "An experimental study of large-scale mobile social network," in *18th International World Wide Web Conference (WWW2009)*, April 2009.

[14] X. Hong, M. Gerla, R. Bagrodia, T.J. Kwon, P. Estabrook, and G. Pei, "The Mars sensor network: efficient, energy aware communications," in *IEEE Military Communications Conference (MILCOM 2001)*, 2001.

[15] NASA, "Disruption tolerant networking porject," 2009, https://www.spacecomm.nasa.gov/spacecomm/programs/technology/dtn/.

[16] Yu Wang and Hongyi Wu, "Delay/fault-tolerant mobile sensor network (dft-msn): A new paradigm for pervasive information gathering," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 9, pp. 1021 –1034, sept. 2007.

[17] Quan Yuan, Ionut Cardei, and Jie Wu, "Predict and relay: an efficient routing in disruption-tolerant networks," in *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, New York, NY, USA, 2009, pp. 95–104, ACM.

[18] Brendan Burns, Oliver Brock, and Brian N. Levine, "Mv routing and capacity building in disruption tolerant networks," 2005, vol. 1, pp. 398–408 vol. 1.

[19] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, New York, NY, USA, 2004, pp. 187–198, ACM.

[20] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *in IEEE INFOCOM*, 2005.

[21] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, 2006.

[22] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, New York, NY, USA, 2005, pp. 252–259, ACM.

[23] Anders Lindgren, Avri Doria, and Olov Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.

[24] Xiaolan Zhang, Giovanni Neglia, Jim Kurose, and Don Towsley, "Performance modeling of epidemic routing," *Comput. Netw.*, vol. 51, no. 10, pp. 2867–2891, 2007.

[25] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *In Proc. IEEE INFOCOM*, 2006.

[26] Jérémie Leguay, Timur Friedman, and Vania Conan, "Evaluating mobility pattern space routing for DTNs," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, Barcelona, Spain, April 2006.

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

[28] Xiang-Yang Li, Peng-Jun Wan, and Yu Wang, "Power efficient and sparse spanner for wireless ad hoc networks," in *IEEE Int. Conf. on Computer Communications and Networks (ICCCN01)*, 2001, pp. 564–567.

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

[30] Ram Ramanathan and Regina Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *IEEE INFOCOM (2)*, 2000, pp. 404–413.

[31] Roger Wattenhofer, Li Li, Paramvir Bahl, and Yi-Min Wang, "Distributed topology control for wireless multihop ad-hoc networks," in *IEEE INFOCOM'01*, 2001.

[32] Ning Li, Jennifer C. Hou, and Lui Sha, "Design and analysis of a MST-based topology control algorithm," in *Proc. of IEEE INFOCOM 2003*, 2003.

[33] Yu Wang and Xiang-Yang Li, "Localized construction of bounded degree and planar spanner for wireless ad hoc networks," *ACM/Springer Mobile Networks and Appli (MONET)*, vol. 11, no. 2, pp. 161–175, 2006.

[34] Moses Charikar and Chandra Chekuri, “Approximation algorithms for directed steiner problems,” *J. Algorithms*, vol. 33, no. 1, pp. 73–91, 1999.

[35] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau, “CRAWDAD data set cambridge/haggle (v. 2006-09-15),” Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, Sept. 2006.

[36] C. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing,” in *Proc. of the ACM SIGCOMM, October*, 1994.

[37] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, “Optimized link state routing protocol for ad hoc networks,” in *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, 2001, pp. 62 – 68.

[38] C. Perkins, “Ad-hoc on-demand distance vector routing,” in *MILCOM ’97*, Nov. 1997.

[39] M. Scott Corson and Anthony Ephremides, “A distributed routing algorithm for mobile wireless networks,” *Wirel. Netw.*, vol. 1, pp. 61–81, February 1995.

[40] V. Park and M. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks,” in *IEEE Infocom*, 1997.

[41] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang, “A wireless hierarchical routing protocol with group mobility,” in *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, 1999, pp. 1538 –1542 vol.3.

[42] Ching-Chuan Chiang and M. Gerla, “Routing and multicast in multihop, mobile wireless networks,” in *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, oct 1997, vol. 2, pp. 546 –551 vol.2.

[43] Mario Joa-Ng and I-Tai Lu, “A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks,” *IEEE Journal on Selected Areas in Communication*, vol. 17, no. 8, pp. 1415–1425, August 1999.

[44] Kaixin Xu, Xiaoyan Hong, and Mario Gerla, “Landmark routing in ad hoc networks with mobile backbones,” *J. Parallel Distrib. Comput.*, vol. 63, pp. 110–122, February 2003.

[45] E. Royer and C. Toh, “A review of current routing protocols for ad-hoc mobile wireless networks,” *IEEE Personal Communications*, Apr. 1999.

[46] Xiaoyan Hong, Kaixin Xu, and M. Gerla, “Scalable routing protocols for mobile ad hoc networks,” *Network, IEEE*, vol. 16, no. 4, pp. 11 –21, jul/aug 2002.

[47] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, pp. 153–167, 2002, Short version in MOBICOM 99.

[48] Julien Cartigny and David Simplot, "Border node retransmission based probabilistic broadcast protocols in ad-hoc networks," in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9*, Washington, DC, USA, 2003, HICSS '03, pp. 303–, IEEE Computer Society.

[49] Wei Lou and Jie Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 111 – 122, apr-jun 2002.

[50] Wei Peng and Xi-Cheng Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, Piscataway, NJ, USA, 2000, MobiHoc '00, pp. 129–130, IEEE Press.

[51] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9 - Volume 9*, Washington, DC, USA, 2002, HICSS '02, pp. 298–, IEEE Computer Society.

[52] Romit Roy Choudhury and Nitin H. Vaidya, "Performance of ad hoc routing using directional antennas," *Ad Hoc Netw.*, vol. 3, pp. 157–173, March 2005.

[53] Chien-Chung Shen, Zhuochuan Huang, and Chaiporn Jaikaeo, "Directional broadcast for mobile ad hoc networks with percolation theory," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 317–332, April 2006.

[54] Elizabeth M. Royer and Charles E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, New York, NY, USA, 1999, MobiCom '99, pp. 207–218, ACM.

[55] J.J. Garcia-Luna-Aceves and E.L. Madruga, "The core-assisted mesh protocol," *Selected Areas in Communications, IEEE Journal on*, vol. 17, no. 8, pp. 1380 –1394, aug 1999.

[56] T. Ozaki, Jaime Bae Kim, and T. Suda, "Bandwidth-efficient multicast routing for multihop, ad-hoc wireless networks," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2001, vol. 2, pp. 1182 –1191 vol.2.

[57] I. Stojmenovic, "Voronoi diagram and convex hull based geocasting and routing in wireless networks," 2000.

[58] J. Boleng, T. Camp, and V. Tolety, "Mesh-based geocast routing protocols in an ad hoc network," in *Parallel and Distributed Processing Symposium., Proceedings 15th International*, apr 2001, pp. 1924 –1933.

[59] Young-Bae Ko and N.H. Vaidya, "Geotora: a protocol for geocasting in mobile ad hoc networks," in *Network Protocols, 2000. Proceedings. 2000 International Conference on*, 2000, pp. 240 –250.

[60] N. Thepvilojanapong, Y. Tobe, and K. Sezaki, "Har: hierarchy-based anycast routing protocol for wireless sensor networks," in *Applications and the Internet, 2005. Proceedings. The 2005 Symposium on*, jan.-4 feb. 2005, pp. 204 – 212.

[61] Y.T. Hou, Yi Shi, and H.D. Sherali, "On base station selection for anycast flow routing in energy-constrained wireless sensor networks," in *Quality of Service in Heterogeneous Wired/Wireless Networks, 2005. Second International Conference on*, aug. 2005, pp. 9 pp. –8.

[62] Jianxin Wang, Yuan Zheng, and Weijia Jia, "An aodv-based anycast protocol in mobile ad hoc network," in *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*, sept. 2003, vol. 1, pp. 221 – 225 Vol.1.

[63] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: modeling a three-tier architecture for sparse sensor networks," 2003, pp. 30–41.

[64] Peter V. Marsden, "Egocentric and sociocentric measures of network centrality," *Social Networks*, vol. 24, no. 4, pp. 407–422, October 2002.

[65] Elizabeth M. Daly and Mads Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, New York, NY, USA, 2007, pp. 32–40, ACM.

[66] Cong Liu and Jie Wu, "Routing in a cyclic mobispace," in *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, New York, NY, USA, 2008, pp. 351–360, ACM.

[67] Cong Liu and Jie Wu, "Scalable routing in delay tolerant networks," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, New York, NY, USA, 2007, pp. 51–60, ACM.

[68] Cong Liu and Jie Wu, "An optimal probabilistic forwarding protocolin delay tolerant networks," in *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, New York, NY, USA, 2009, pp. 105–114, ACM.

[69] Quan Yuan, Ionut Cardei, and Jie Wu, "Predict and relay: an efficient routing in disruption-tolerant networks," in *Proceedings of the tenth ACM international*

*symposium on Mobile ad hoc networking and computing*, New York, NY, USA, 2009, MobiHoc '09, pp. 95–104, ACM.

[70] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani, "Dtn routing as a resource allocation problem," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2007, SIGCOMM '07, pp. 373–384, ACM.

[71] S.C. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in dtns," in *INFOCOM 2009, IEEE*, 2009, pp. 846 –854.

[72] Rongxing Lu, Xiaodong Lin, and Xuemin Shen, "Spring: A social-based privacy-preserving packet forwarding protocol for vehicular delay tolerant networks," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1 –9.

[73] M. Grünewald, T. Lukovszki, C. Schindelhauer, and K. Volbert, "Distributed maintenance of resource efficient wireless network topologies," in *Proc. of the 8th European Conference on Parallel Computing (Euro-Par'02)*, 2002.

[74] Li Li, Joseph Y. Halpern, Paramvir Bahl, Yi-Min Wang, and Roger Wattenhofer, "Analysis of a cone-based distributed topology control algorithms for wireless multi-hop networks," in *ACM Symposium on Principle of Distributed Computing (PODC)*, 2001.

[75] Xiang-Yang Li, Peng-Jun Wan, Yu Wang, and Ophir Frieder, "Sparse power efficient topology for wireless networks," in *IEEE Hawaii Int. Conf. on System Sciences (HICSS)*, 2002.

[76] Yu Wang and Xiang-Yang Li, "Localized construction of bounded degree planar spanner for wireless networks," in *ACM DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2003.

[77] Wen-Zhan Song, Yu Wang, Xiang-Yang Li, and Ophir Frieder, "Localized algorithms for energy efficient topology in wireless ad hoc networks," in *5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2004)*, Tokyo, Japan, 2004.

[78] Xiang-Yang Li, Wen-Zhan Song, and Yu Wang, "Localized topology control for heterogeneous wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 1, pp. 129–153, 2006.

[79] Wen-Zhan Song, Yu Wang, and Xiang-Yang Li, "Localized algorithms for energy efficient topology in wireless ad hoc networks," *ACM/Springer Mobile Networks and Appli (MONET)*, vol. 10, no. 6, pp. 911–923, 2005.

[80] Xiang-Yang Li, Wen-Zhan Song, and Weizhao Wang, "A unified energyefficient topology for unicast and broadcast," in *11th ACM Annual International Conference on Mobile Computing and Networking (MobiCom 2005)*, 2005.

[81] P. Bose and P. Morin, “Online routing in triangulations,” in *Proc. of the 10 th Annual Int. Symp. on Algorithms and Computation ISAAC*, 1999.

[82] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, “Routing with guaranteed delivery in ad hoc wireless networks,” *ACM/Kluwer Wireless Networks*, vol. 7, no. 6, 2001.

[83] J. Gao, L. J. Guibas, J. Hershburger, L. Zhang, and A. Zhu, “Geometric spanner for routing in mobile networks,” in *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, 2001.

[84] Brad Karp and H.T. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks,” in *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[85] Ivan Stojmenovic and Xu Lin, “Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, 2001.

[86] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder, “New distributed algorithm for connected dominating set in wireless ad hoc networks,” in *HICSS, Hawaii*, 2002.

[87] Alan D. Amis and Ravi Prakash, “Load-balancing clusters in wireless ad hoc networks,” in *Proc. of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, 2000.

[88] Alan D. Amis, Ravi Prakash, Dung Huynh, and Thai Vuong, “Max-min d-cluster formation in wireless ad hoc networks,” in *Proc. of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM*, 2000, vol. 1, pp. 32–41.

[89] I. Chlamtac and A. Farago, “A new approach to design and analysis of peer to peer mobile networks,” *Wireless Networks*, vol. 5, pp. 149–156, 1999.

[90] Chunhung Richard Lin and Mario Gerla, “Adaptive clustering for mobile wireless networks,” *IEEE Journal of Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.

[91] Ji-Cherng Lin, Shi-Nine Yang, and Maw-Sheng Chern, “An efficient distributed algorithm for minimal connected dominating set problem,” in *Proc. of the Tenth Annual International Phoenix Conference on Computers and Communications 1991*, 1991, pp. 204–210.

[92] Yu Wang, Weizhao Wang, and Xiang-Yang Li, “Efficient distributed low-cost backbone formation for wireless networks,” in *6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2005)*, Urbana-Champaign, Illinois, 2005.

[93] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *1997 IEEE International Conference on on Communications (ICC'97)*, 1997, vol. 1, pp. 376–380.

[94] P. Sinha, R. Sivakumar, and V. Bharghavan, "Cedar: Core extraction distributed ad hoc routing algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1454 –1465, August 1999.

[95] Ivan Stojmenovic, Mahtab Seddigh, and Jovisa Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14–25, 2002.

[96] Jie Wu and Hailan Li, "A dominating-set-based routing scheme in ad hoc wireless networks," *the special issue on Wireless Networks in the Telecommunication Systems Journal*, vol. 3, pp. 63–84, 2001.

[97] Yunhuai Liu, Qian Zhang, and L.M. Ni, "Opportunity-based topology control in wireless sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 3, pp. 405 –416, march 2010.

[98] MohammadTaghi Hajiaghayi, Nicole Immorlica, and Vahab S. Mirrokni, "Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. 2003, pp. 300–312, ACM Press.

[99] Ajit Agrawal, Philip Klein, and R. Ravi, "When trees collide: an approximation algorithm for the generalized steiner problem on networks," in *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, New York, NY, USA, 1991, STOC '91, pp. 134–144, ACM.

[100] Shashidhar Merugu, Mostafa Ammar, and Ellen Zegura, "Routing in space and time in networks with predictable mobility," Tech. Rep., 2004.

[101] 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*, Philadelphia, PA, USA, 2008, pp. 532–541, Society for Industrial and Applied Mathematics.

[102] Barun Chandra, Gautam Das, Giri Narasimhan, and Jose Soares, "New sparseness results on graph spanners," in *Proc. 8th Annual ACM Symposium on Computational Geometry*, 1992, pp. 192–201.

[103] Jihwang Yeo, David Kotz, and Tristan Henderson, "Crawdad: a community resource for archiving wireless data at dartmouth," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 21–22, 2006.

[104] N. Baccour, A. Koubaa, M.B. Jamaa, and et al., "A comparative simulation study of link quality estimators in wireless sensor networks," in *IEEE MASCOTS*, 2009.

[105] W. Su, S. Lee, and M. Gerla, "Mobility prediction in wireless networks," in *IEEE MILCOM*, 2000.

[106] W.-S. Soh and H.S. Kim, "Dynamic bandwidth reservation in cellular networks using road topology based mobility predictions," in *IEEE INFOCOM*, 2004.

[107] B. Gallagher, D. Jensen, B.N. Levine, "Explaining routing performance in disruption tolerant networks," UMass, Tech. Rep. 05-57, 2005.

[108] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Efficient routing in intermittently connected mobile networks: the multiple-copy case," *IEEE/ACM Trans. Netw.*, 16:77–90, 2008.

[109] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, 2006.

[110] A. Agrawal and R. E. Barlow, "A survey of network reliability and domination theory," *Operations Research*, 32:478–492, 1984.

[111] L. Zosin and S. Khuller, "On directed steiner trees," in *ACM SODA02*.

[112] 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.

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

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

[115] H. C. Joksch, "The shortest route problem with constraints," *J. Math. Anal. Appl.*, 14:191–197, 1966.

[116] F. A. Kuipers, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Communications Magazine*, 50–56, 2002.

[117] D. S. Reeves and H. F. Salama, "A distributed algorithm for delay-constrained unicast routing," *IEEE/ACM Trans. Netw.*, 8:239–250, 2000.

[118] F. Kuipers, T. Korkmaz, M. Krunz, , and P. V. Mieghem, "Performance evaluation of constraint-based path selection algorithms," *IEEE NETWORK*, 18:16–23, 2004.

[119] 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.

[120] C. Liu and J. Wu, "Routing in a cyclic mobispace," in *ACM MobiHoc08*.

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

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

[123] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," *Comput. Netw.*, 51:2867–2891, 2007.