Network Design and Routing in Peer-to-Peer Networks and Mobile Ad Hoc Networks

A Thesis
Presented to
The Academic Faculty

by

Shashidhar Merugu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
August 2005

Network Design and Routing in Peer-to-Peer Networks and Mobile Ad Hoc Networks

Approved by:

Dr. Ellen Zegura, Advisor
College of Computing
Georgia Institute of Technology

Dr. Constantinos Dovrolis
College of Computing
Georgia Institute of Technology

Dr. Mostafa Ammar
College of Computing
Georgia Institute of Technology

Dr. Jun (Jim) Xu
College of Computing
Georgia Institute of Technology

Dr. Samrat Bhattacharjee
Department of Computer Science
University of Maryland

Date Approved : July 11th, 2005

To my family, for everything.

ACKNOWLEDGEMENTS

I owe a lot to Ellen Zegura, my thesis advisor. She has been my mentor, guide, critic and a friend, all-in-one. She always believed in me and encouraged me, particularly during the hard times that tested my reasons for undertaking this study. Thank you Ellen.

I have been fortunate to work with Mostafa Ammar during the last two years. I cherish my interactions with him, especially his insightful questions that have always inspired me to improve my clarity of thought. I acknowledge his support and advice.

When Bobby Bhattacharjee was a senior graduate student, he took me into his fold and introduced me to the life of a graduate student. I am grateful to Bobby for his advice, friendship and keeping a constant watch on my progress over the years.

It is my pleasure to have interacted with Constantinos Dovrolis and Jun (Jim) Xu. I thank them for their suggestions, particularly during my "work-in-progress" presentations at the group seminar, that have improved my thesis.

Ken Calvert and Milena Mihail have advised me on various topics on several occasions and I thank them for their guidance.

Everyone in the Networking and Telecommunications group at GCATT has helped me in some way or the other over the years and I am grateful for the assistance and friendship. In particular, it is my privilege that I had Sridhar Srinivasan as my critic. Sridhar had to endure me almost every day while patiently listening to my ideas – good and bad – and has been most influential on my work. Thank you so much Sridhar. My thanks also to several past and present members of our group especially Christos Gkantsidis, Richard Liston, Pradnya Karbhari, Abhishek Kumar, Youngsu Chae, Matt Sanders and Wenrui Zhao for the long brain-storming sessions that helped in shaping up my thesis.

Lastly, I thank my family without whose love and sacrifice I would not be what I am and I owe them everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Peer-to-peer networks and mobile ad hoc networks are emerging distributed networks that share several similarities. Fundamental among these similarities is the decentralized role of each participating node to route messages on behalf of other nodes, and thereby, collectively realizing communication between any pair of nodes. Messages are routed on a topology graph that is determined by the peer relationship between nodes. Although routing is fairly straightforward when the topology graph is static, dynamic variations in the peer relationship that often occur in peer-to-peer and mobile ad hoc networks present challenges to routing.

In this thesis, we examine the interplay between routing messages and network topology design in two classes of these networks – unstructured peer-to-peer networks and sparsely-connected mobile ad hoc networks.

In unstructured peer-to-peer networks, we add structure to overlay topologies to support file sharing. Specifically, we investigate the advantages of designing overlay topologies with small-world properties to improve (a) search protocol performance and (b) network utilization. We show, using simulation, that "small-world-like" overlay topologies where every node has *many close neighbors and few random neighbors* exhibit high chances of locating files close to the source of file search query. This improvement in search protocol performance is achieved while decreasing the traffic load on the links in the underlying network.

In the context of sparsely-connected mobile ad hoc networks where nodes provide connectivity via mobility, we present a protocol for routing in space and time where the message forwarding decision involves not only where to forward (space), but also when to forward (time). We introduce space-time routing tables and develop methods to compute these routing tables for those instances of ad hoc networks where node mobility is predictable over

either a finite horizon or indefinitely due to periodicity in node motion. Furthermore, when the node mobility is unpredictable, we investigate several forwarding heuristics to address the scarcity in transmission opportunities in these sparsely-connected ad hoc networks. In particular, we present the advantages of fragmenting messages and augmenting them with erasure codes to improve the end-to-end message delivery performance.

CHAPTER I

INTRODUCTION

Sharing of files – the most dominant application on the Internet today – has led to intense interest in designing and studying peer-to-peer networks [50]. In a peer-to-peer network, a *peer* acts both as a client and a server of the system, unlike the conventional client-server approach where the roles of a client and server are distinguishable. Recent studies on work load of peer-to-peer networks [62, 63] also attest to their immense popularity. Similarly, wireless mobile ad hoc networks are also emerging in a variety of environments with the introduction of low power wireless devices and development of protocols such as Bluetooth [3] that allow short-range and high-speed wireless connectivity. An ad hoc wireless network, or simply an *ad hoc network*, consists of a collection of geographically distributed nodes that communicate with one another over a wireless medium without using any pre-defined infrastructure [52].

Peer-to-peer networks and mobile ad hoc networks share several characteristics. Primarily, nodes in these networks participate in forwarding messages in addition to being a source or a destination. Secondly, the operation of nodes is usually decentralized and nodes self-organize into a network topology graph for communication. The resultant network topologies are often dynamic due to, for example, transient life-times of peers in a peer-to-peer network and node mobility in a mobile ad hoc network. Furthermore, these networks present opportunities to *design* the network topology. For example, a node in a peer-to-peer network has a choice of neighbors enabling design of interesting overlay topologies.

In this thesis, we examine the interplay between routing messages and network topology in two classes of these networks – unstructured peer-to-peer networks and sparsely-connected mobile ad hoc networks. Unstructured peer-to-peer networks (e.g., BearShare, LimeWire based on Gnutella, KaZaa based on FastTrack) are extremely popular and attractive for their simplicity. Simple routing protocols and lack of rigid control on the overlay

topology make these unstructured peer-to-peer networks robust to peer dynamics. Likewise, sparsely-connected ad hoc networks are emerging in several environments such as those for sensor data collection, [1, 8, 64], animal-tracking [35, 67], rural-Internet [51], disaster-relief [84]. Although these ad hoc networks are disconnected when observed over short time-scales, nodes in these networks move around explicitly *carrying* messages to provide connectivity over time.

The contributions of this thesis can be broadly categorized into two complementary goals: *(a)* design network topologies to improve routing performance and *(b)* devise routing protocols suitable for particular network topologies. Specifically, we present the following in this thesis:

- **Adding structure to unstructured peer-to-peer networks.** We focus on routing search queries/replies to support file sharing in unstructured peer-to-peer networks. We seek the advantages of designing topologies with small-world properties to improve (a) search protocol performance, a local gain perceived directly by peer-to-peer network users, and (b) network utilization, a global property that is of interest to network service providers. We show, using simulation, that "small-world-like" overlay topologies where every node has *many close neighbors and few random neighbors* exhibit high chances of locating files close to the source of file search query. This improvement in search protocol performance is achieved while decreasing the traffic load on the links in the underlying network.

- **Routing in sparsely-connected ad hoc networks.** We consider the problem of routing in sparsely-connected ad hoc networks where nodes move around explicitly carrying messages to facilitate communication in an otherwise partitioned (or poorly connected) network. The absence of a path at any instant of time between a source and destination makes the traditional mobile ad hoc routing protocols unsuitable for these networks. We consider two classes of these sparsely-connected ad hoc networks based on the predictability of the dynamic network topology.

  - **Predictable sparsely-connected networks** We present a new paradigm of

routing in space and time for instances of sparsely-connected ad hoc networks where node mobility is predictable over either a finite horizon or indefinitely due to periodicity in node motion. In our protocol for routing in space and time, the message forwarding decision involves not only where to forward (space), but also when to forward (time). Our space-time routing framework includes routing tables that use both destination and time of a message to determine the next hop node to forward the message. Our routing algorithm to compute these space-time routing tables is based on a space-time graph model of the network derived from the mobility of nodes.

– **Unpredictable sparsely-connected networks** When the network dynamics are unknown and unpredictable, most routing protocols advocate that a node replicate and forward all its messages to every neighbor for an eventual delivery to the corresponding destinations. However, the finite communication bandwidth and duration of *contact* opportunities, particularly in sparsely-connected ad hoc networks, limit the number and size of messages that can actually be transmitted. Addressing this limitation, we present forwarding schemes that *spread* the messages through the network to improve the delivery performance (e.g., end-to-end delay). In particular, we present the advantages of fragmenting large messages and transmitting the fragments according to certain heuristics to ensure that the messages spread faster through the network. Furthermore, we augment fragmentation with erasure coding and show the benefits in decreasing the end-to-end message delivery delay while maintaining similar resource consumption.

The rest of the thesis is organized as follows. In Chapter 2, we present background of peer-to-peer and mobile ad hoc networks and specific work related to our focus in this thesis. In Chapter 3, we present a class of small-world-like overlay topologies for improved performance in unstructured peer-to-peer networks. Chapter 4 describes a new space-time routing paradigm for sparsely-connected mobile ad hoc networks. Finally, Chapter 6 lists the contributions of this thesis.

CHAPTER II

BACKGROUND AND RELATED WORK

The thesis encompasses two closely-related areas of networking: peer-to-peer networks and ad hoc networks. We review background and recent research in these two areas that relates to our focus on network topology design and routing.

2.1 Peer-to-peer Networks

In this section, we describe the taxonomy of peer-to-peer networks and a popular search protocol followed by three categories of previous work that are related to our focus on peer-to-peer overlay topologies.

1. Measurement and study of topologies of real-world implementations of unstructured peer-to-peer networks.

2. Algorithms for construction of logical overlay networks that are aware of the underlying network topology by discovering close neighbors.

3. Other application driven factors to consider in topology construction that have impact on search in the peer-to-peer networks.

2.1.1 Taxonomy

Peer-to-peer networks used in file sharing are generally classified into two broad categories based on the file search operation as shown in Figure 1.

Centralized peer-to-peer networks (e.g., Napster [48]) have a central repository that stores an index to all files available in the system. A peer interested in a particular file queries this central repository for information on peers that store the file. The file is requested and downloaded directly from one of the peers that provide this file.

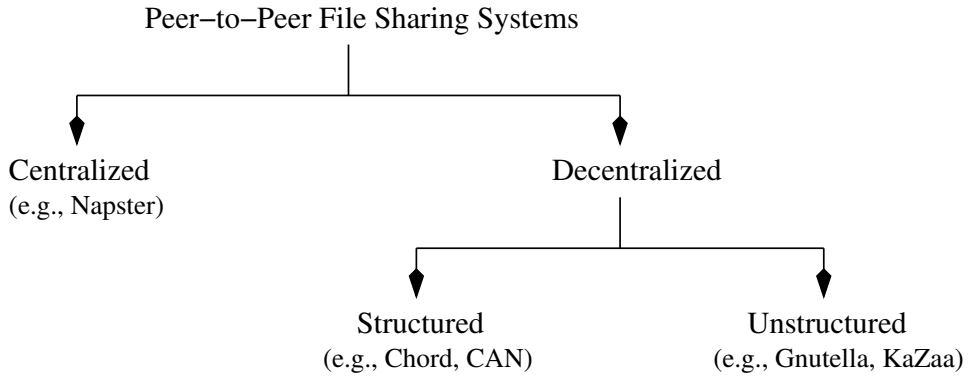

Figure 1: Classification of file sharing peer-to-peer networks

Decentralized peer-to-peer networks do not have such a central index of files, but rather perform the file search operation in a distributed fashion by forwarding the file search query over the peer-to-peer network. In a decentralized peer-to-peer network, each peer maintains a neighbor relationship with a small set of peers and participates in the file sharing protocol (e.g., file search and download). This neighbor relationship, forming a logical link between peers, defines the *topology* of the peer-to-peer network. Messages of the file sharing protocol (e.g., search queries and replies) are exchanged between peers on this overlay topology.

Decentralized peer-to-peer networks can be further divided into two sub-categories: structured and unstructured. Structured peer-to-peer networks (e.g., Chord [70], CAN [54], Tapestry [82], Pastry [61]) use a distributed hash table to lookup files. Given the complete name of a file, these structured peer-to-peer networks are efficient in locating the file. However, searching for files with key words, as is common practice, is known to be complex in these networks [73, 57]. Also, since the overlay topology and placement of files is governed by a rigid structure in these peer-to-peer networks, they may be overwhelmed by reorganization overhead under significant churn of peers. Unstructured peer-to-peer networks (e.g., BearShare, LimeWire based on Gnutella [27], KaZaa based on FastTrack [23]), on the other hand, are extremely popular and attractive for their simplicity. These networks allow complex search queries, in particular, file searching based on key words. Unlike structured peer-to-peer networks, these networks do not impose a tight control on the overlay topology and hence they are robust to peer dynamics.

2.1.2 Search in Unstructured Peer-to-Peer Networks

Unstructured peer-to-peer networks, most notably Gnutella and its family of protocols [28], use a very simple *scoped-flooding search* mechanism to locate files. A search query is propagated to all neighbors from the source node. The query is replicated and forwarded by each intermediate node to all its neighbors, up to a system-defined maximum number of overlay hops from the source. Every intermediate node compares its local contents with the requested file in the search query and responds to the query source on a match. Scoped-flooding is costly when scaled to large number of queries and can only find files within the search radius. However, it is very simple and amenable to a dynamic real-world peer-to-peer network. Unlike structured peer-to-peer networks, unstructured peer-to-peer networks do not have any association between the content and location where it is stored, thereby eliminating the complexity of maintaining such an association in a dynamic scenario [56]. Without an association between content and location, a node does not have any information about which other nodes can best be able to resolve a query. In this case of a "blind" search, scoped-flooding reaches many search candidates as well as limits the distance traveled by a search query.

The success of a scoped-flooding search, both in terms of *quantity* of candidates searched, and the *quality* of results returned, is strongly affected by the topology of the peer-to-peer network. For example, in an overlay topology that is a k-ary tree rooted at the source of the query, a scoped-flooding search up to m hops would examine k^m candidates. On the other hand, if the source and its neighbors form a clique in the overlay topology, many nodes are revisited and hence the number of unique search candidates is lower. The quality of a result can be measured as the distance (e.g., latency, available bandwidth) in the underlying network from the hit node to the query source. A *close* result may be able to better serve the file than a far node. In an overlay topology that is constructed unaware of the underlying network, as is the case with Gnutella topologies [59, 63], the search query hits may be far from the source, even though they are within the scoped-flooding search radius on the overlay topology. The overlay topology also determines the amount of traffic load (search queries/results) on the underlying network. For example, when most of the

overlay links cross Autonomous System (AS) boundaries, there is a proportional amount of inter-AS traffic [63].

2.1.3 Properties of Gnutella Topologies

Ripeanu et al. looked at the projection of the Gnutella logical overlay on the underlying network-layer topology [59]. Their experiment on counting the number of logical edges that stayed within an AS and those that crossed AS boundaries revealed a significant "mismatch" between the logical overlay and the projection on the underlying network. A second study by Saroiu et al., also based on a crawl of Gnutella topology, confirmed that it is highly robust to partitions indicating its similarity to a well-connected random graph [62]. A third study by Jovanović et al. evaluated clustering coefficients and average path lengths of Gnutella virtual topologies [34]. However, this study is based on a crawl performed when Gnutella was in its nascent stages (late 2000) and since there is no known record or a model for growth of Gnutella, it is not clear how to extrapolate the values of clustering and average path lengths to current versions of Gnutella topologies. The large scale and the complexity of current Gnutella servents make it very difficult to capture an accurate picture of Gnutella network today. There is an ongoing effort in the open-software community to crawl and visualize the Gnutella network topology [13].

2.1.4 Discovery of close neighbors

On another front, there have been many efforts on construction of topology-aware overlay networks. Distributed binning is based on the idea that if two nodes perceive similar (in terms of rank or order) distances to well-known Internet landmarks, then they must be near each other [55]. Another proposal called "beaconing" uses an algorithm based on triangulation heuristics to identify nearby nodes by querying a set of "beacons" [65]. This approach is not scalable as every beacon needs to keep track of all nodes in the system along with their respective distances from it. Addressing the scalability issue in beaconing, an alternative based on hierarchical trees called Tiers [4] is proposed. These approaches to find close nodes are complementary to our work. They can be integrated with our algorithm to find the short-link neighbors at a node.

2.1.5 Alternative topology design criteria

Among other considerations for constructing topologies of unstructured peer-to-peer networks, a recent proposal suggests "interest-based shortcuts" to link peers that have similarity in shared content [69]. Search queries are forwarded on the shortcut links before exploring the regular links. Their simulation results show considerable improvement in object location using this simple idea of interest based locality. It is possible to think of a composite metric that combines both distance in underlying topology that we use in our work with the interest based locality to select neighbors.

A recent work [81], very similar in spirit to ours, proposes a new routing table cache replacement algorithm based on small-world properties in Freenet [17]. Information about neighbors is cached in a node's routing table if they contain data items that are "close" in key space to node's own key. Under the invariant condition, each node would point to many neighbors that store "close" data items and few neighbors that store "randomly far" data items. There are two main differences between this work and ours. First, this distance measure is purely over the key space and doesn't consider the distance in underlying network. We expect that this key space distance metric can blend with our distance metric of the underlying network. Second, Freenet design includes the computation of routing tables and query forwarding based on file identifier look-ups. There are no explicit routing tables in Gnutella as their calculation and maintenance is complex due to the large size and dynamics of peers. Hence Gnutella, which relies only on scoped-flooding search, may not benefit from optimizations on the key space.

In addition to the small-world graphs that we considered for an overlay topology of peer-to-peer networks, several potential candidates exist including "scale-free graphs" [6, 7]. In the scale-free graphs, the distribution of vertex degrees follows a power-law, rather than the Poisson distribution of the classical random graph models [10, 29, 22]. Examples of scale-free graphs include the autonomous system topology of the Internet [25], and the web graph [2]. The scale-free graphs are appealing to the context of peer-to-peer networks due to their fast search advantage. Since few nodes have a large degree, one could potentially search the content of many nodes in a single step by visiting these high degree nodes. Another

characteristic of scale-free graphs that could be beneficial to peer-to-peer networks is their resilience to random failures.

2.2 Ad Hoc Networks

In this section, we introduce a classification of ad hoc networks followed by a description of sparsely-connected ad hoc networks that are the focus of our study. Later, we discuss three categories of work related to our proposal on routing in space and time leveraging the predictability of node mobility.

2.2.1 Taxonomy

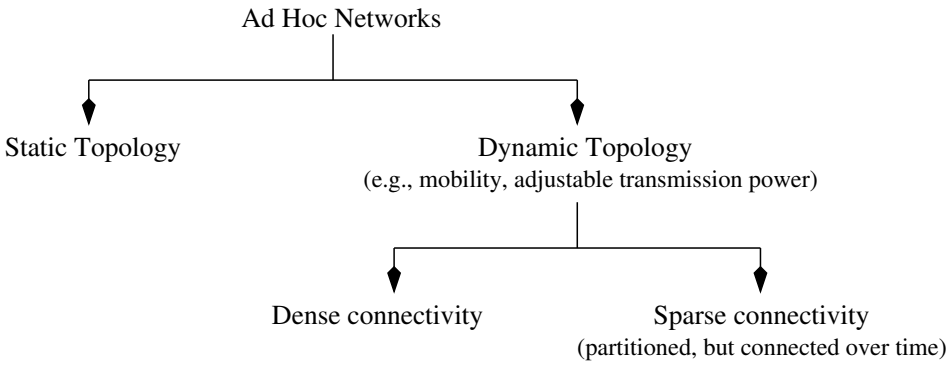

Figure 2: Classification of ad hoc networks

Ad hoc networks can be classified along two dimensions – network topology variation and connectivity of the network, as shown in Figure 2. In a *static* ad hoc network, nodes take a permanent position and the topology graph formed by these nodes is usually fixed. An example of such a network is a collection of sensors deployed in a lake to monitor, say, temperature of water. Unlike static ad hoc networks, the network topology of *dynamic* ad hoc networks varies over time due to, for example, node mobility, transmission power variation. These dynamic ad hoc networks can further be classified based on the connectivity properties of their network topology. A *densely-connected* ad hoc network always has a path from any source to any destination despite the dynamic nature of the network topology. On the other hand, network partitions are a norm rather than an exception in *sparsely-connected* ad hoc networks. As a result, any time snapshot of the topology graph in these

networks is almost always disconnected.

2.2.2 Sparsely-connected ad hoc networks

Sparsely-connected ad hoc networks, are emerging in several environments such as those for sensor data collection [1, 8, 64], animal-tracking [35, 67], rural-Internet [51], disaster-relief [84]. Also, since sparsely-connected ad hoc networks are known to save energy, a scarce resource, they may be the preferred design choice, especially in extreme environments.

Although mobile ad hoc routing protocols are designed to handle network topology dynamics, most of them inherently assume the existence of an end-to-end path between a source and a destination to initiate communication. This premise of path existence may not be valid at any instant of time in these sparsely-connected ad hoc networks, thus making the traditional mobile ad hoc routing protocols unsuitable for these networks. However, the explicit node movements create paths "over time" that include the possibility of a node *carrying* a message before forwarding to another suitable node, resulting in a new *store-carry-forward* paradigm.

A collection of recent projects use this store-carry-forward message paradigm. For example, a project in developing nations uses rural buses to provide Internet connectivity to otherwise isolated and remote villages which do not have any communication infrastructure [51]. Several other examples include robots walking through a sensor farm collecting data from individual sensors [64], hikers monitoring and disseminating weather information in a national park by walking along the park trails [8], and wild-life researchers driving through a forest collecting information about dispersed zebra population [35]. Several other networks such as ad hoc networks of satellites [66] and Inter-planetary Internet [11] share characteristics of store, carry and forward networks.

2.2.3 Age-based space-time routing

Age-based space-time routing (STR) has been proposed recently for mobile ad hoc networks to improve routing in dynamic, but connected, topologies [21]. STR considers the age of a routing entry to a destination when forwarding messages. Our interest in time, by contrast, is how long does a message wait at a node to minimize overall delay to the destination.

STR works with past history of routes, whereas, we look ahead to find new best routes for messages. FRESH, a protocol proposed by the same authors to highlight STR, suggests a new route discovery mechanism where route request propagation is guided by "encounter ages" [20]. A FRESH node forwards the route request to another node that has seen the destination more recently than the former node. The FRESH approach is based on a heuristic that a node with a recent encounter age is more likely to be in the vicinity of (or perhaps closer to) the destination than another node with an old encounter age. We believe that the STR approach (including FRESH) of using history of routes can blend with our proposal to identify routes that are not only stable, based on past history, but also provide best opportunity to deliver message (e.g., in the shortest delay) to the destination.

2.2.4 Predicting node mobility

Our work on space-time routing draws on a precedence of several proposals that leverage predictability to improve routing. Su et al. apply mobility prediction to enhance unicast and multicast routing protocols [71]. They associate an "expiration time" to routes in the ad hoc network and the route selection algorithm prefers routes with longer expiration time. McDonald and Znati propose a framework for computing probabilities of a path and its availability over time in a mobile ad hoc network [45, 44]. They use this probabilistic model to develop an algorithm for clustering nodes such that the chances of existence of a route are high. They also build on this probabilistic model and develop another node proximity model that includes an adaptive learning strategy to estimate the chances that two nodes remain within a given distance threshold [43]. Likewise, Jiang et al. compute the probability that a particular link that is available now would last up to a given time in the future considering movements of nodes [33]. Measurement-based prediction schemes most often use global positioning system (GPS) information to track node location and movement and thereby compute coordinates and longevity of links [72].

2.2.5 Message-oriented forwarding

Unlike flow-oriented forwarding approach used in many traditional on-demand ad hoc routing protocols, message-oriented forwarding alleviates the overhead of route repair under

rapid network dynamics. Forwarding message decisions, even for those messages belonging to the same flow, are independent in message-oriented scheme and hence are fairly robust to link changes in the network topology. Few recent proposals such as Epidemic routing [74] and Probabilistic routing [42] share this message oriented forwarding scheme with our proposal on space-time routing. Epidemic routing is a form of flooding where messages spread through the network by a simple pair-wise exchange protocol where a node forwards only those messages that its neighbor hasn't seen earlier. Probabilistic routing uses a history of node encounters to forward packets along a highly likely path. While these heuristic schemes are applicable in general, our space-time routing goes one step further, especially for those contexts where the network topology evolution, though rapid, is predictable. Mobile ad hoc networks with non-random node motion fall into this category and hence benefit from such efficiency gains of our space-time routing framework.

CHAPTER III

OVERLAY TOPOLOGIES FOR UNSTRUCTURED PEER-TO-PEER NETWORKS

3.1 Introduction

Our work in this thesis focuses on decentralized and unstructured peer-to-peer networks to support file sharing; our goal is to examine the role of overlay topologies on the performance of such networks. In this chapter, we present a class of overlay topologies for these decentralized and unstructured peer-to-peer networks, taking inspiration from the recent work on small-worlds [77, 36].

3.1.1 Overview

Small-world graph topologies are defined by two key properties: *(i)* short paths that are characteristic of random graphs and *(ii)* high *clustering* unlike random graphs, but a typical property of well-ordered graphs. It has been observed that the small-world structure is pervasive and arises "without-design" or "naturally" in many practical systems such as the World Wide Web [78, 2]. Overlays, on the other hand, provide an opportunity to *design structure*. We seek the advantages of designing overlays with a small-world structure. In our model of overlay topologies, a node's links to its neighbors are divided into two categories: *short* links and *long* links. The short links connect close nodes and the long links connect nodes chosen randomly. The fraction of links that are short, called the *proximity factor* (α), is a key design parameter that controls the properties of the resultant overlay topology. When $\alpha = 0$, every link is randomly chosen, and the overlay topology is a random graph. On the other hand when $\alpha = 1$, every link connects to a close neighbor, and the overlay topology looks more well-ordered (e.g., grid-like). Different values of α let us span the spectrum of this class of overlay topologies. We show, using simulation, that overlay topologies with an invariant of *many close neighbors and few random neighbors* at each node exhibit the

following properties: *(a)* the underlying network distance to search candidates increases with the overlay radius as we progress in the scoped flooding search; *(b)* the count of search candidates is fairly high; and *(c)* the system returns close results. We also show that the traffic load on links of the underlying network is better for these overlay topologies than random overlay topologies.

Given our results showing that these overlay topologies perform well, we turn to the practical question of constructing such topologies in a dynamic peer-to-peer environment. We propose a method of topology construction where each node operates independently and in a decentralized manner to select its own neighbors. Neighbors are selected from a pool of candidates using a simple greedy algorithm based on local information. In a practical implementation, we foresee short links connecting nodes belonging to the same AS where possible, while the long links cross AS boundaries. Given the dynamic conditions of a peer-to-peer network, nodes periodically evaluate the distance to their neighbors and replace them if necessary to maintain the invariant ratio of short and long links.

3.1.2 Road-map

The remainder of the chapter is organized as follows. We study the class of overlay topologies and describe properties of peer-to-peer systems that are important for their performance in Section 3.2. In Section 3.3, we present our evaluation methodology and simulation results. We propose an algorithm for overlay topology construction and its evaluation in Section 3.4. We conclude with a summary in Section 3.6.

3.2 Overlay Topologies

In this section, we present a class of overlay topologies and discuss how different instances of this class of topologies can be realized by varying a characteristic parameter. Later in this section, we present dimensions to study the effect of overlay topologies on the performance of peer-to-peer systems.

One of the key purposes of characterizing overlay topologies is to create a simple framework for an empirical analysis. Our choice for a model for the overlay topologies is inspired by the generic lattice network model for small-worlds [37]. In our class of topologies, there

are two kinds of links: *short-links* and *long-links*. Every node in the system selects some *close* nodes to be its short-link neighbors and some nodes *at random* to be its long-link neighbors. An appropriate metric for distance (e.g., latency) in the underlying network defines the *closeness* of neighbors. While the short-link neighbors are carefully chosen to be close to the node, the long-link neighbors, on the other hand, being chosen randomly, are generally far from the node. We make an implicit assumption that the system has a large enough population of nodes that are spread across the underlying network ensuring the existence of close neighbors. Crawls of Gnutella peer-to-peer topologies [59, 62] and live-count of active Gnutella peers [38] support this focus on a large population.

The *proximity factor* (α) of a node, defined as the ratio of the number of short links to the total number of links, is a design parameter that governs the overall structure of the topology. A node with degree d has αd short links and $(1-\alpha)d$ long links. α takes values from 0 to 1, inclusive: $\alpha = 0$ corresponds to all long-links and $\alpha = 1$ corresponds to all short-links. Different values of α let us span the spectrum of this class of overlay topologies.

In an instance of the class of topologies with $\alpha = 0$, all the neighbors of a node are randomly chosen, and this results in a *random graph*. It is well-known that random graph topologies have a low diameter and are fairly robust in terms of connectivity [10]. The average distance between neighbors, in this all long-links case, is typically the mean node-to-node distance in the underlying network. As α increases, there is less "randomness" and more "order" in the topology. The "order" in the topology comes from the increasing number of short-links. Short-links "induce" more short-links. Intuitively, if A and B are short-link neighbors of C, then chances are that A and B are also close to each other and could be short-link neighbors themselves. This possibility of a node's neighbors being neighbors themselves introduces "clusters" in the topology. Clustering, a measure of "cliqueness" of a neighborhood, is formally defined as the fraction of allowable edges that occur among the set of neighbors of a node [77]. As α increases, the number of short-links increase and the topology is increasingly clustered. The instance of topology with $\alpha = 1$ corresponds to all short-links and we expect the topology to contain multiple clusters and to be possibly disconnected. However, an instance of the topology with many short-links

but few long-links, i.e., with an α between 0.5 and 1.0, is expected to exhibit both the properties of a random graph and a clustered graph. We call such instances "small-world-like" overlay topologies. These small-world-like topologies contain multiple clusters formed by the many short-links, but their random long-links connect across clusters and keep the graph connected as well as lower its diameter.

3.2.1 Desirable Properties

We focus on two performance criteria of peer-to-peer systems: *(a) search protocol performance*, a local gain perceived directly by a user of the system and *(b) utilization of the network*, a global property that is of interest to network service providers. Search protocol performance is studied along the two dimensions of search space and underlying network distance to search results. Network utilization is measured by the traffic load on the links in the underlying network.

3.2.1.1 Search Space

The size of scoped-flooding search space around a query source node is measured by number of unique candidate nodes visited by the search query. Since scoped-flooding search protocol is ignorant of particular file being searched for, the chances of finding the requested file depend significantly on the size of the search space. The factors that determine the size of search space include search radius and structure of the overlay topology. It is also desirable that the overlay topology be always connected and robust from sudden departures of nodes that are quite common in a dynamic peer-to-peer network. Connectivity of the topology ensures that scoped-flooding search does not terminate early and that all nodes up to the search radius are explored.

3.2.1.2 Underlying Network Distance to Search Results

After completion of the search process when the search results are returned, the query source has a choice of selecting one of the hit nodes to serve the file. This file transfer performance depends on the distance between the query source and the chosen hit node in the underlying network. Though the query source and hit node are relatively close in the

overlay, it is not necessary that these two nodes are close in the underlying network. If the overlay topology is *aware* of the underlying network, the scoped-flooding search mechanism can find close hit nodes. To quantify this awareness, we define *stride* at i^{th} step to be the term $(D_i - D_{i-1})$, where D_i denotes the average distance in the underlying network to all nodes that are exactly i hops away on the overlay topology.

3.2.1.3 Network Traffic

A benefit of an overlay topology being aware of the underlying network is an improvement in the utilization of links in the underlying network. One of the most common definitions of traffic load in overlay networks is the notion of *stress*. Stress [16] of a link in the underlying network is defined as the number of logical links (of the overlay topology) whose mapped paths include the underlying link.

In summary, we desire overlay topologies that satisfy a dual objective of improving the performance of scoped-flooding search protocol and minimizing the traffic load on the links of the underlying network. To realize these goals, we seek "good" structural properties on the overlay to have *(a)* a large number of unique search candidates and *(b)* an awareness of overlay topology with the underlying network. An overlay topology with structural properties similar to that of a random graph gives large number of unique search candidates, however it does not "fit" well with the underlying network. On the other hand, an overlay topology with high clustering conforms well with the underlying network, but has a lower number of unique search candidates. We aim to find a suitable balance in the structural properties of overlay topologies for these two goals of producing a higher count of search candidates and being underlying network-aware.

3.3 Evaluation

In this section, we analyze, by simulation, how the desirable properties described in Section 3.2.1 vary with the proximity factor (α) in our class of overlay topologies. We briefly explain our simulation framework followed by analysis of results from different experiments. These

experiments, designed to study the role of overlay topology on the performance of peer-to-peer networks, evaluate *(1)* success of scoped flooding search queries, *(2)* underlying network distance to search results, and *(3)* stress on links in the underlying network.

3.3.1 Methodology

3.3.1.1 Simulation Parameters

Our simulations are performed using *p-sim*, a flexible tool that can simulate Gnutella peer-to-peer network protocol on representative Internet topologies [47]. A network topology comprising 5152 routers was generated using the Transit-Stub graph model from the GT-ITM topology generator [80]. Transit-stub graphs capture the hierarchical nature and locality of routing domains in the Internet. Network latency is used as a measure of distance in our experiments. The link latencies are assigned values according to the type of the link, using a uniform distribution on the following ranges: [15ms, 25ms] for intra-transit domain links, [3ms, 7ms] for transit-stub links, [1ms, 3ms] for intra-stub links. In each experiment, end-hosts are attached uniformly at random to the stub routers of the underlying network topology. These end-hosts participate in the overlay network. All our experiments have 4096 end-hosts (overlay nodes). All overlay nodes have at least 3 and at most 4 neighbors[1]. In order to simulate scoped-flooding search for files, 100 unique files with varying popularity are introduced into the system. Each file has multiple copies stored at different locations chosen at random. The number of copies of a file are proportional to their popularity. The count of file copies is assumed to follow Zipf distribution with 2000 copies for the most popular file and 20 copies for the least popular file. The queries that search for these files are also initiated at random hosts on the overlay topology. Again the number of queries for a file is assumed to be proportional to its popularity. Each query follows the scoped-flooding search protocol up to a maximum of 4 overlay hops from the query source[2].

[1] Implementations of Gnutella protocol (including Limewire [39], GTK-Gnutella [31], Gnucleus [30]) allow users to configure the number of neighbors. Typical recommended values for node degree are 4 and 5. In addition to simulation experiments with node degree range of $[3, 4]$ presented here, we also ran our experiments with the degree ranges of $[4, 6]$ inclusive and the results are similar to what we present here. One key difference though is in the diameter of the overlay topology that decreases with increasing degree.

[2] Though early versions of Gnutella protocol specification suggested a TTL of 7, most popular Gnutella servents including Limewire [39] use a default TTL value of 4 to limit system resource usage. http:// core.limewire.org/ source/ browse/ core/ com/ limegroup/ gnutella/ SettingsManager.java (v1.245)

We keep track of the number of search candidates seen by each search query and average distance to these search candidates in the underlying network. In addition to measuring quantities related to scoped-flooding search, we also measure the diameter and average path length of the overlay topology and the stress on the links in the underlying network. Each experiment is repeated multiple times with different seeds to remove any bias in random number generation that is used in multiple stages of simulation (e.g., placement of end-hosts on the underlying network, copies and queries of files on end-hosts). Our plots include 95% confidence intervals on the empirical data gathered from the simulation experiments.

3.3.1.2 Overlay Topologies

We constructed four different instances of the class of overlay topologies described in Section 3.2. These four instances of topologies are compared along different dimensions of the desirable properties described in Section 3.2.1. The instances of topologies have proximity factors of $\{0.0, 0.33, 0.66, 1.0\}$. With a minimum degree of 3 at each node, these four values of proximity factor correspond to four combinations of numbers of short and long links: $(0, 3)$, $(1, 2)$, $(2, 1)$ and $(3, 0)$. While the long-link neighbors are randomly chosen from the node population, global information about underlying network distances to all nodes is used in deciding the short-link neighbors[3].

An overlay topology instance with $\alpha = 0.0$ ("all-long-links"), corresponds to the case where neighbors of every node are randomly chosen. Current unstructured peer-to-peer systems use a neighbor selection scheme that is similar to this all-long-links instance of overlay topologies. When a node joins the peer-to-peer system, it contacts a *gateway* that returns a set of randomly picked seed-neighbors from its cache of known nodes. The neighbor relationship is maintained as long as the neighbors are active and have resources. When a neighbor departs, the node finds a random replacement from the candidate set of nodes that it learns during its life-time. There is no evidence of locality on the underlying network

[3]Since we are only interested in studying the various kinds of topologies, we believe that this "oracle" approach of constructing overlays is acceptable in our simulation environment. However, implementing a particular choice of overlay topology in a practical environment would have to work without this assumption of global knowledge of distance to all nodes. We address this practical issue of topology construction later in Section 3.4.

in either choice of initial neighborhood or replacement nodes. Since the neighbor selection is random, the overlay topology of current peer-to-peer systems resembles that of a random graph. Hence this kind of all-long-links topology is representative of the real-world peer-to-peer topologies. At the other end of the spectrum, every node has close neighbors in the "all-short-links" topology with $\alpha = 1.0$. The instance of $\alpha = 0.66$ is a topology where every node has more short links than long links and is similar to a small-world graph. We also refer to this "many-short-links" instance as "small-world-like" overlay topologies.

3.3.2 Simulation Results

3.3.2.1 Success of Search

In this experiment, we studied the success rate of scoped-flooding searches for each type of topology. A query for a file is "successful" when it finds at least one match for the requested file within the search radius. Success rate of a file is defined as the ratio of number of successful queries to the total number of queries issued for that file throughout the system. Though the number of queries we simulated per file is proportional to the popularity of that file, "success rate" is a normalized quantity across all the files. In Figure 3, we plot the success rate of queries for files with different popularity for each instance of overlay topologies[4]. All topologies have a high success rate for more popular files because of the large number of copies for these files. However as the popularity decreases, the number of copies of files in the system decreases and the success rate for all the topologies goes down. One key observation is that the success rate for the all-short-links topology (with $\alpha = 1.0$) drops rapidly, compared to the success rate for the other three values of the proximity factor. We also note that the success rate for small-world-like topology is fairly close to that of the all-long-links topology.

The effects that we observe in Figure 3 follow directly from Figure 4. In Figure 4, we plot the cumulative number of unique nodes that are reachable as we increase the search radius of the scoped-flooding scheme. The all-long-links topology ($\alpha = 0$) reaches more nodes at

[4]To improve clarity of the figure and examine relative trends of each of the curves corresponding to different proximity factors, we plot only 25 points for the 100 files used in simulation. Each point displayed on the curve is an average of four consecutive files.

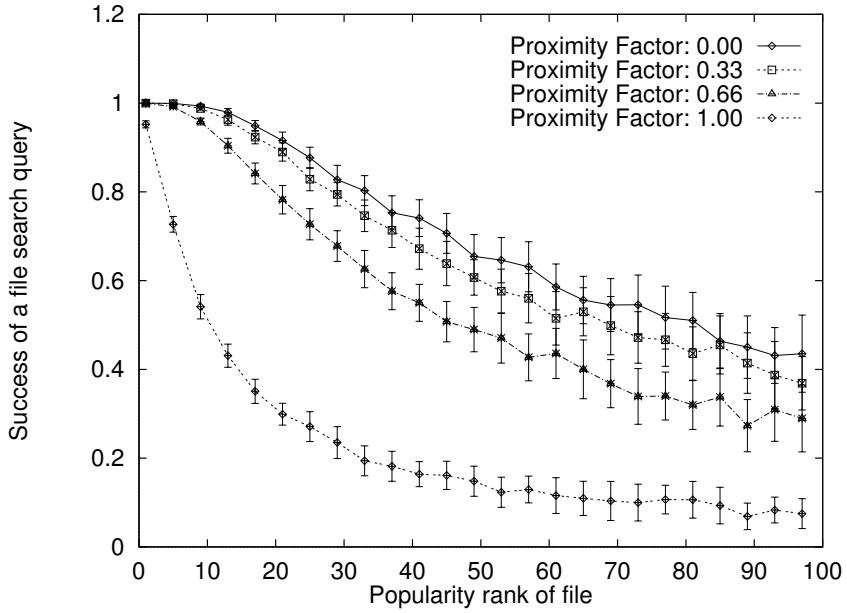

Figure 3: Scoped-Flooding Search Query Success Rate

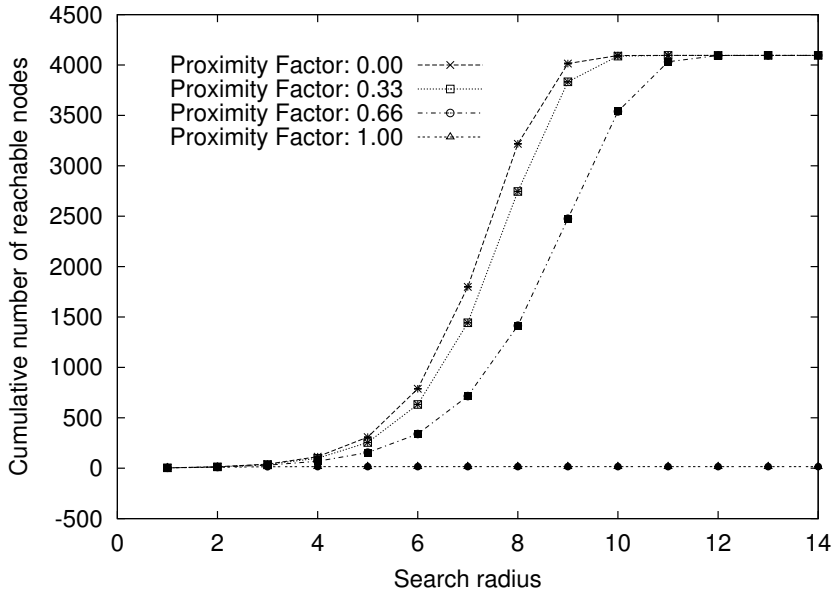

Figure 4: Count of reachable nodes with increasing search radius

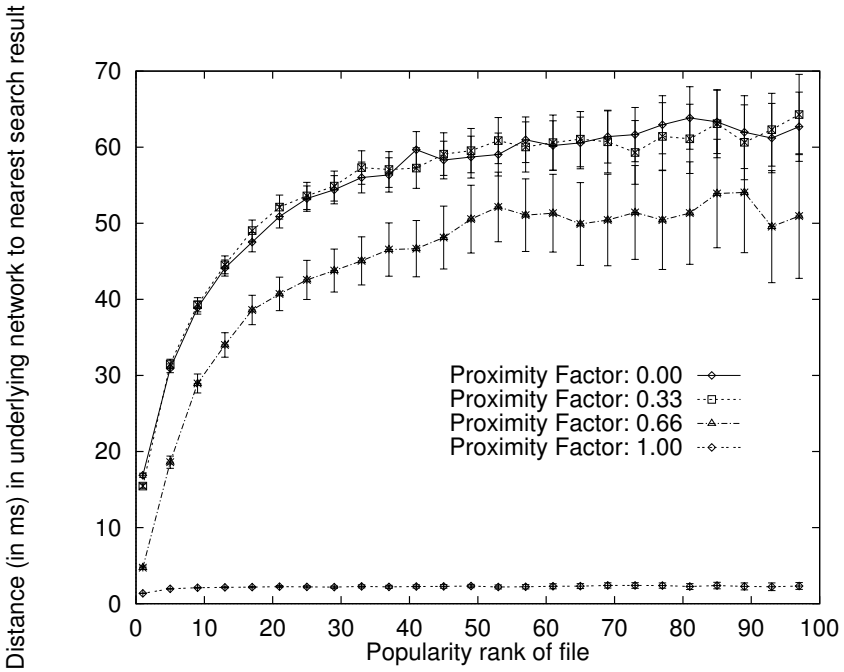

Figure 5: Underlying network distance to nearest search result

each radius than all the other three topologies; but with a maximum search radius of 4 that is used in our simulation, the number of unique nodes reached by these three topologies ($\alpha = \{0, 0.33, 0.66\}$) is very similar. The small-world-like topology can access a fairly large number of nodes because each long link allows the query to reach more nodes outside its local region. The similarity in number of search candidates results in similar success rate that we observed in Figure 3 for these three topologies. Also, the low success rate of the all-short-links topology in Figure 3 is due to the small (almost constant) number of nodes that are accessed at increasing search radii. This topology causes nodes to form clusters and makes the graph disconnected. Table 1 lists the diameter and average path length of the four kinds of overlay topologies. Note that the diameter and the average path length of the small-world-like topology ($\alpha = 0.66$) is comparable to the more random topologies ($\alpha = \{0.0, 0.33\}$). On the other hand, the all-short-links topology is disconnected in all our multiple trials and the values for diameter and average path length that are reported in this table are for the connected components of the topology.

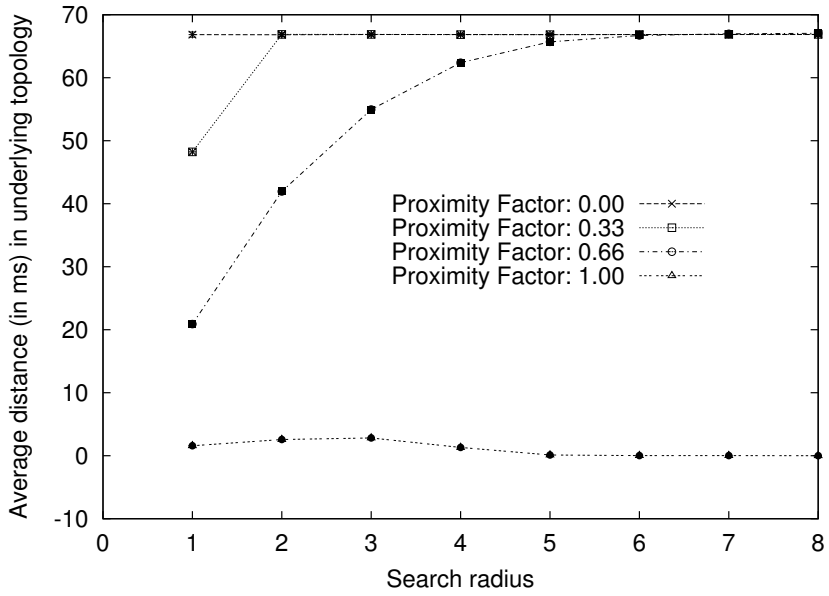

Figure 6: Variation of average underlying network distance to nodes at a given radius

Table 1: Path properties (diameter and average path length) of overlay topology.
Note: $*$ the overlay topology with $\alpha = 1.0$ is disconnected and the values presented here are for connected components.

Proximity Factor	0.0	0.33	0.66	1.0
Overlay Diameter	11	12	14	6*
Avg. Path Length	7	7	8	2*

3.3.2.2 Distance to search results

Of the multiple file locations returned by a scoped-flooding search, the nearest one in the underlying network may be able to better serve the file than other locations. We study the variation of this distance to the nearest search result node for the four kinds of topologies in Figure 5. The underlying network distance to nearest search result is plotted as a function of file popularity. Each point is an average over all the queries issued for that file. We note first that the distance to the nearest search result increases as the file popularity decreases because less popular files have fewer copies and the search query has to travel farther to locate those copies. Second, the nearest search result for small-world-like topologies ($\alpha = 0.66$) is closer than the nearest result for topologies with $\alpha = 0.0$ and $\alpha = 0.33$. Even though the all-short-links topology shows the smallest distance, we have seen in Figure 3 that most of these queries are failures. The reason for these observations follows directly from Figure 6 where we plot the average distance in the underlying network to nodes located at increasing values of the search radius. This metric called "stride", as defined in Section 3.2.1.2, measures "awareness" of an overlay topology with the underlying network. The stride for all-long-links topology ($\alpha = 0$) is flat (zero) as all the links are randomly chosen and hence the average distance to nodes is independent of how far they are on the overlay from the source node. It can also be clearly seen that the neighborhood of nodes in small-world-like topology is closer in the underlying network (a positive stride), leading to the search results being found closer in the underlying network. On the other hand, in the all-short-links topology, all the nodes at increasing radii are very near the source and exhibit a tendency to form clusters. We also notice that, on average, after a small number of overlay hops, the all-short-links topology is disconnected.

3.3.2.3 Stress

To show the effect of traffic load by various overlay topologies on the underlying network, we plot the mean stress of links in the underlying network as the size of the population increases in Figure 7. The all-short-links overlay topology has extremely low mean stress because for every node, its set of neighbors is selected from nodes that are close to it

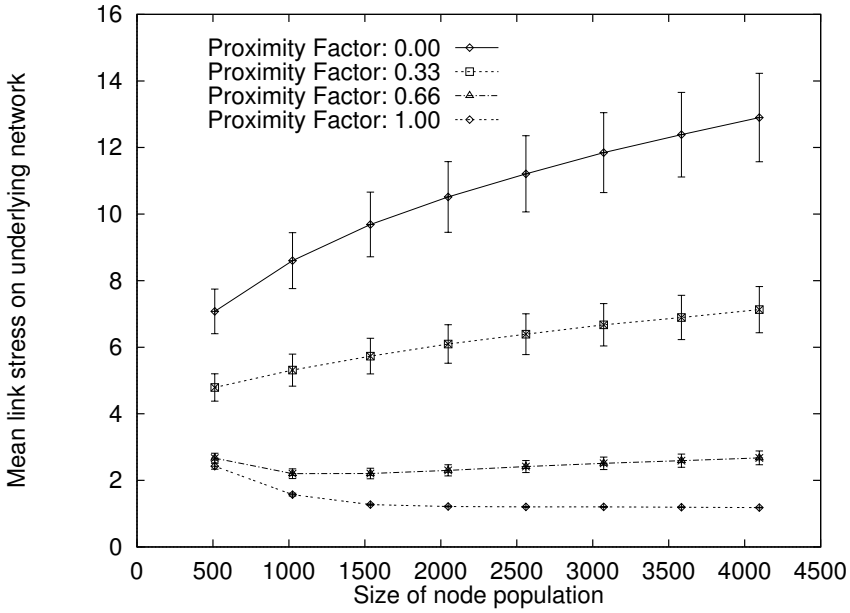

Figure 7: Variation of mean stress on links of the underlying network as the size of node population increases for different kinds of topologies

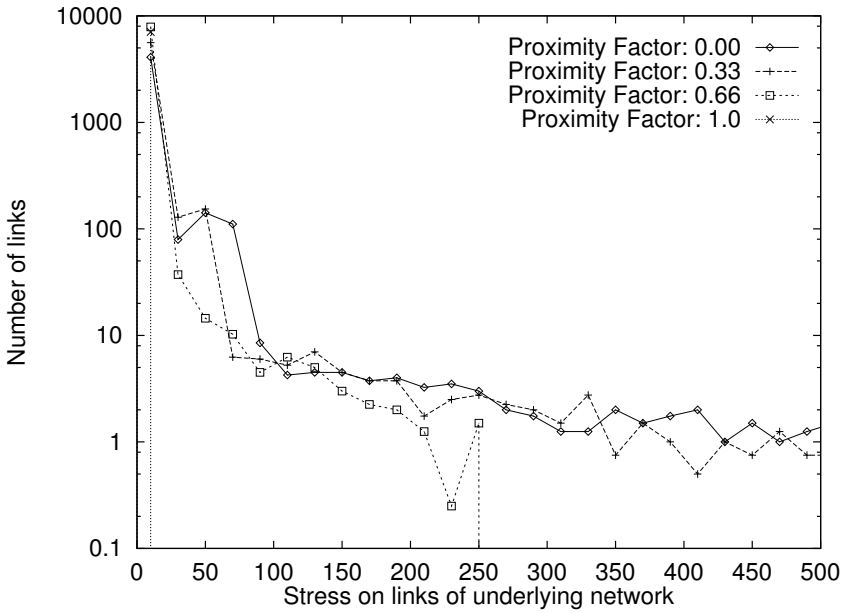

Figure 8: Distribution of stress on links of the underlying network

and the links to those nodes are not shared by any other pair of nodes, resulting in low stress. In the case of all-long-links ($\alpha = 0.0$) topology, a node's neighbors are picked from the entire set of nodes, leading to overlay links that span the underlying network, causing underlying links to carry multiple overlay links. The many-long-links ($\alpha = 0.33$) topology is in between the two extremes. The graph shows that the amount of randomness in choosing the neighbors increases the mean stress of the topology. The small-world-like ($\alpha = 0.66$) topology, by virtue of having many short links, has a low mean stress like that of the all-short-links topology. Some of the disproportionate increase in stress of more random topologies ($\alpha = \{0.0, 0.33\}$) as the population size increases, is potentially due to the use of transit-stub graphs for our simulations. The links connecting stubs to transit nodes could possibly form bottlenecks pushing up the mean stress of a topology. To investigate this, we plot the frequency distribution of the stress on underlying links in Figure 8. It can be clearly seen that random topologies have many more links with high stress than the all-short-links and small-world-like topologies on the same underlying network.

3.3.3 Summary of results

A summary of the observed properties of the four kinds of topologies is given in Table 2. As expected of random graphs, the all-long-links topology has a low diameter, reaches out to many candidates in a scoped-flooding search and is always connected. However, it does not utilize the links of underlying network well and the stride is almost zero for these topologies. On the other end of the spectrum, the all-short-links topology behaves like a well-ordered graph with higher diameter, and better utilization of the links in underlying network. However, the search space is quite small and the topology is disconnected most often. In between these two ends of the spectrum, but closer to the case of all-short-links, we observe that the topologies, with many short links and few long links, have desirable properties. They not only have low diameter, large search space and are always connected like an all-long-links topology, but are also aware of the underlying network and can utilize the links better like an all-short-links topology.

Table 2: A summary of properties of the four kinds of topologies.

Topology kind → Property ↓	All long links	Many long links	Many short links	All short links
Network Diameter	Low	Low	**Low**	High
Connectedness	Connected	Connected	**Connected**	Disconnected
Search Space	Large	Large	**Large**	Small
Stride	Zero	Zero	**Positive**	Positive
Network Utilization	Poor	Poor	**Good**	Good

3.4 Overlay Topology Construction

We have shown in the previous section that overlay topologies with many close neighbors and few random neighbors at each node exhibit desirable properties. Now we consider how to practically construct such overlay topologies in a peer-to-peer environment. In this section, we explain our incremental greedy algorithm for overlay topology construction. Our algorithm design goals include the following:

1. **Scalable** with the number of participant nodes.
2. **Distributed** in execution using only local information available at a node. We do not expect to use either special infrastructure or any cooperation of a central agent for the construction or maintenance of the topology.
3. **Resilient** to dynamic arrivals and departures of nodes in a peer-to-peer network.

We propose an operation called `Adapt` that is executed by nodes in the peer-to-peer system. By executing an `Adapt` operation, a node selects "better" neighbors according to its local view of the system. Each node in the system has the following parameters: *(1)* the number d of neighbors; *(2)* the proximity factor α that determines the neighborhood ratio of short and long links; *(3)* a *adapt-threshold* γ of distance to determine whether another adapt operation is necessary; and *(4)* an *adapt-epoch* δ value to schedule a periodic adapt

operation. The life-time of a node on the peer-to-peer system is divided into two phases: a join phase and a maintenance phase. In the join phase, a new node, say X, adapts repeatedly until its short-link neighbors are at a "satisfactory" distance. Let D_i denote the average distance in underlying network to the short-link neighbors of X after i^{th} invocation of `Adapt`. X adapts until: $|D_i - D_{i-1}| < \gamma$. When this condition is met, say, at time t_0, we say X has "settled" in its neighborhood. Later, in the maintenance phase, X schedules a periodic `Adapt` operation at every adapt-epoch δ time units, i.e., at $\{(t_0 + \delta), (t_0 + 2\delta), \dots\}$. An `Adapt` operation is also triggered when one or more neighbors depart.

A description of the `Adapt` procedure in pseudo-code is given in Figure 9. The three input parameters to this procedure are the node-ID (u), the number of short links (s) and the number of long links (l). s and l are computed according to the node parameters d and α where $s = \alpha d$ and $l = (1 - \alpha)d$. In the first two steps, node u builds a candidate list C of potential neighbors from its current local neighborhood. These candidates include current neighbors N of u and neighbors of current neighbors, i.e., all nodes seen in a "depth-first-traversal" starting from u up to a depth of two hops. Node u measures distances to each of these candidates $v \in C$ from itself. A new set of neighbors N' is formed by selecting the s "closest" candidates according to distance rank and picking l candidates at random from the rest. Thus node u acts in a greedy fashion to improve its local neighborhood.

One of the key features of the `Adapt` procedure is its simplicity. `Adapt` is similar in spirit to the simple scoped-flooding search of the unstructured peer-to-peer systems. Both schemes scout their immediate neighborhood, but for different reasons: one for closer neighbors, the other for locating files. `Adapt` is decentralized and does not make use of any central authority or any infrastructure. It is based only on a local view of the node and hence can be easily deployed on current peer-to-peer systems. Since each node scouts a neighborhood that is two hops deep, the number of messages exchanged in each `Adapt` operation are $O(\mathcal{D}^2)$ where $\mathcal{D}$ is the max-degree of a node. When the max-degree $\mathcal{D}$ of nodes in the peer-to-peer system is a constant, the computational complexity of these `Adapt` operations is bounded by a constant. The triggers for an `Adapt` operation, both based on an adapt-epoch and adapt-threshold, make this scheme amenable to dynamic arrivals and departures of nodes

of a peer-to-peer system.

```
Adapt(u, s, l) { // u: node, s short links and l long links
    // get neighbors of u
    N ← {u.nbrList}
    // add current neighbors to candidate list
    C ← N
    // add neighbors of current neighbors to candidate list
    ∀v ∈ N, C ← C ∪ {v.nbrList}
    // compute distance to each candidate
    For each node v in C {
        D[v] ← distance(u, v)
    }
    // sort the distance array
    sort(D)
    // find 's' closest and 'l' random candidates
    N' ← closest(D, s) ∪ random(D, l)
    // assign new neighbors
    u.nbrList ← N'
}
```

Figure 9: Pseudo-code of `Adapt` procedure

3.5 Evaluation of Overlay Topology Construction

We have evaluated the join phase of our proposed method for constructing overlay topologies. We have also examined the question of partial deployment, where only a fraction of the node population observe the invariant ratio of many close neighbors and few random neighbors. We use the same evaluation framework and methodology as described in Section 3.3.1 for these experiments. We present both these results in this section.

3.5.1 Join phase

There are two metrics to evaluate the cost of the join phase: *(1)* the number of `Adapt` operations performed before the termination condition, and *(2)* the variation in distance to neighbors with multiple `Adapt` operations. Fewer `Adapt` operations mean a rapid and short join phase. We also seek a low final distance to neighbors at the end of the join phase. In this experiment, we randomly chose 256 locations in the underlying network for a new node. This new node has three short links and one long link. It starts with seed-neighbors

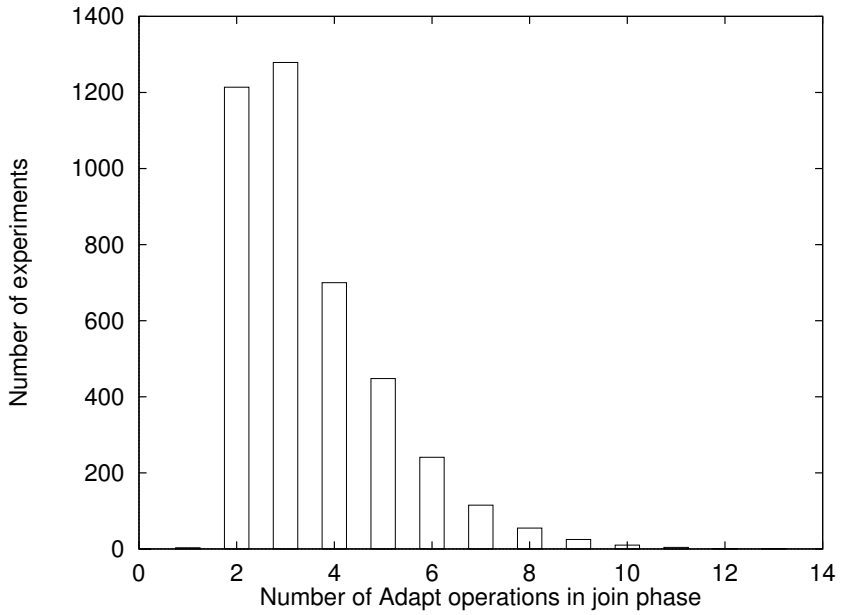

Figure 10: Number of Adapt operations in join phase

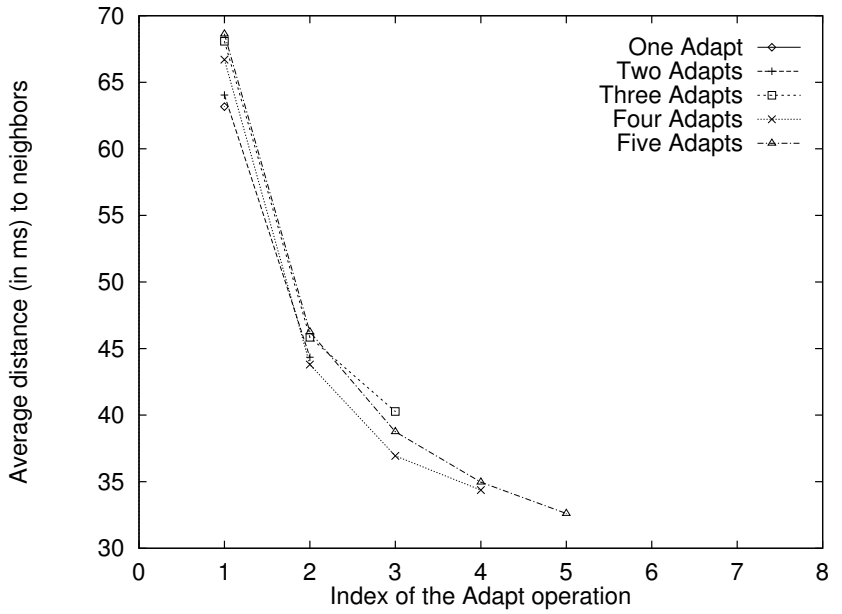

Figure 11: Decrease in average underlying network distance to neighbors with multiple Adapt operations

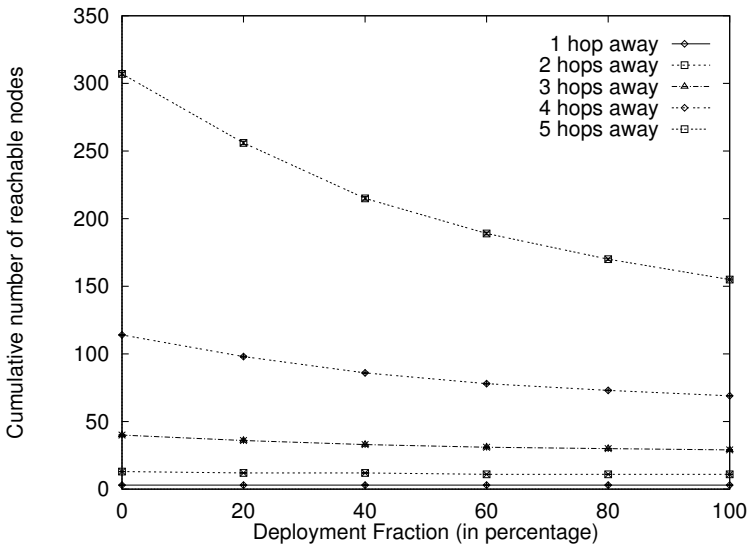

Figure 12: Cumulative count of search candidates under partial deployment

selected at random from the node population. This random choice of seed-neighbors is repeated 16 times. In all, we conducted $256 \times 16 = 4096$ runs of this experiment to evaluate the join phase. A histogram plot of the number of `Adapt` operations is shown in Figure 10. A mean of 3.5 and standard deviation of 1.55 `Adapt` operations are needed in each join phase. Figure 11 shows the reduction in the average distance to neighbors with each `Adapt` operation. Each curve corresponds to the set of runs that terminated after specified numbers of `Adapt` operations. We can note from the figure that the average distance to neighbors decreases with `Adapt` operations. However, the join phase terminates when the final average distance to neighbors is 38.5ms, on average over all the runs, even though closer nodes exist in the system. It is possible to refine our simple greedy algorithm to find such closer nodes. For example, instead of a depth of two, one could think of a deeper traversal so that a node can explore more candidates for closer neighbors. But this improvement comes at a price of increase in measurement of distance to a larger set of candidates, thereby increasing the message complexity of this algorithm.

3.5.2 Partial Deployment

In this experiment, we study the case where a fraction of the node population observe the invariant ratio of many close neighbors and few random neighbors, while the rest of

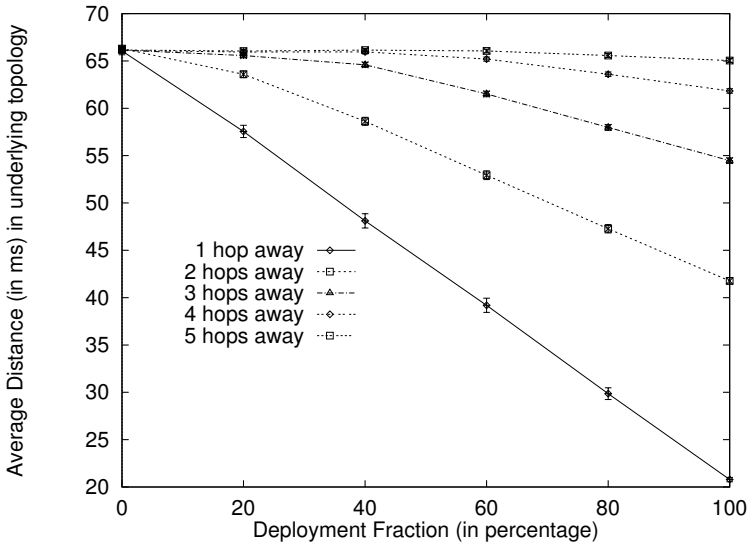

Figure 13: Average distance to nodes under partial deployment

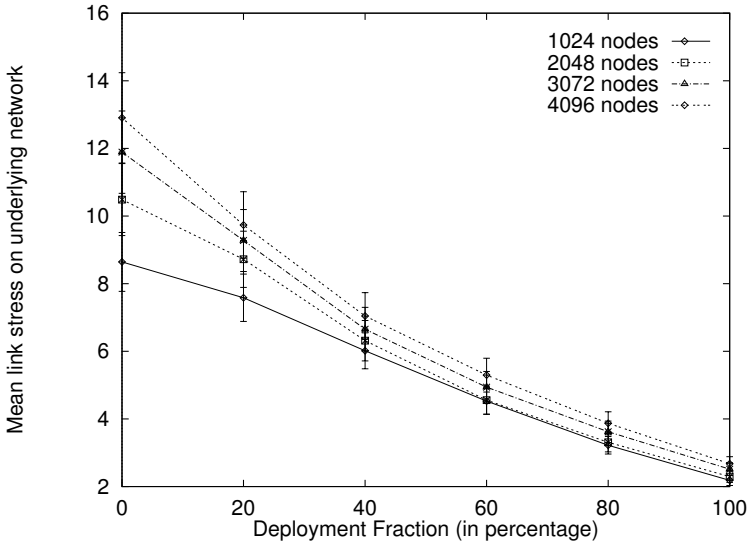

Figure 14: Mean stress on links of the underlying network for partial deployment

the nodes use a random scheme for their neighbors. This experiment is representative of an incremental deployment of our topology construction algorithm. The invariant ratio of many close neighbors and few random neighbors is observed by using a proximity factor of $\alpha = 0.66$. A fraction of node population follow this proximity factor of $\alpha = 0.66$ and their size is varied from 0% to 100% in this experiment in increments of 20%. The rest of the nodes in the system use a proximity factor of $\alpha = 0.0$ corresponding to the random scheme of neighbor selection prevalent in current real-world peer-to-peer networks. On this spectrum of deployment fraction variation, the topology instance with 0% deployment corresponds to the case of all-long-links topology where all the links are chosen randomly. At the other end with 100% deployment, the topology is an instance of the many-short-links topology that has small-world properties. Figure 12 shows the cumulative count of nodes reachable at different radii by a scoped flooding search protocol. In Figure 13, we plot the average distance to nodes in the underlying network at different search radii and Figure 14 shows the variation in mean stress on the links of the underlying network for different sizes of node population. The values at either end points of the deployment fraction spectrum (0% and 100%) confirm our observations reported in Section 3.3 for the many-short-links and all-long-links topology instances. Also, each of these figures show gradual improvement in the size of search space, average distance to search results and mean stress as the deployment fraction is varied from 0% to 100%. This gradual improvement indicates the feasibility of deploying our schemes incrementally in real-world peer-to-peer networks.

3.6 Summary

Overlay topology plays a fundamental role in the performance of an unstructured peer-to-peer network. In this chapter, we have presented a class of overlay topologies and their characteristic parameter called proximity factor. Proximity factor relates to the underlying network distance between a node and its neighbors. Different instances of this class of overlay topologies are realized by varying the proximity factor. These topology instances are examined, by simulation, along two performance-related dimensions: *(a)* search protocol performance and *(b)* utilization of links in the underlying network. Our simulation results

show that in a particular "small-world-like" topology instance of this class where every node has many close neighbors and few random neighbors, *(1)* the chances of locating files are high and *(2)* the nodes where these files are found are, on average, close to the query source. This improvement in search protocol performance is achieved while *(3)* decreasing the traffic load on the links in the underlying network. To demonstrate the feasibility of constructing such "small-world-like" overlay topologies in a practical peer-to-peer environment, we have presented a simple greedy algorithm called `Adapt`. A node executing `Adapt` operates independently and in a decentralized manner to select its neighbors. Our evaluation suggests that `Adapt` finds closer neighbors and our approach is incrementally deployable.

CHAPTER IV

ROUTING IN SPACE AND TIME IN PREDICTABLE SPARSE AD HOC NETWORKS

While the Chapter 3 focussed on the design of a network topology to improve the performance of a routing protocol, this chapter considers a complementary goal of designing a routing protocol for a network that has special topological characteristics – sparse-connectivity and dynamics.

4.1 Introduction

Routing in sparsely-connected networks follows a new paradigm of store, *carry* and forward messaging [83, 51, 64, 8, 35, 66, 11, 68]. In addition to the usual storing and forwarding of data, nodes in these new environments move around explicitly carrying messages to facilitate communication in an otherwise partitioned (or poorly connected) network. Consider an example mobile network of four nodes $\{A, B, C, D\}$ as shown in Figure 15. These nodes move in trajectories as indicated. Each sub-figure is a snapshot of the network during a time interval. A link is established between two nodes when they enter each other's radio coverage area. This link lasts as long as the nodes are visible to each other and is broken when they lose radio connectivity as they move away from each other. Note that in this example network, the graph is not connected at any point in time. However, the graph is connected over time by paths, where messages are carried for a certain duration on their way towards the destination. Consider routing a message from a source A starting at time $t = 0$ towards destination B. There are many possible paths that include waiting at intermediate nodes. The message can take a path P_1: A to D at $t = 0$, wait at D until $t = 2$, and D to C at $t = 2$ and C to B at $t = 3$. The message can also take an alternative path P_2 that includes waiting at the source A itself until $t = 2$ and delivering directly to B at $t = 2$.

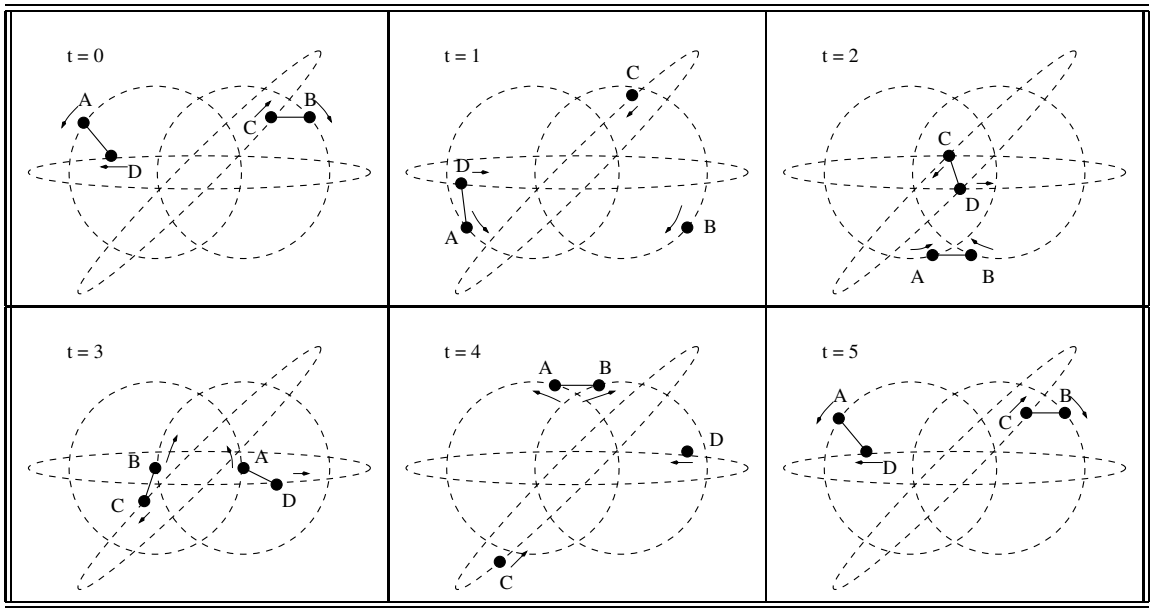

Figure 15: An illustration of a varying topology of an example mobile network of four nodes A, B, C, D. The dashed circles and ellipses represent the trajectories followed by these mobile nodes. Each sub-figure is a snapshot of the network during the indicated interval of time. Solid lines between nodes represent communication links. Arrows represent their directions of movement.

We make the following observations from this example: *(i)* P_1 and P_2 are paths over space and time from A to B. The message travels from one node to another in space and also waits at a node in anticipation of a future link. *(ii)* A path of least end-to-end latency (P_2) need not always be the first available (P_1). The forwarding decision is best taken by looking ahead into the future, as a future link might deliver the message earlier than current available links. *(iii)* Although P_2 has the least latency and uses only a single hop, spatial hop count minimization, as is done in many flavors of ad hoc routing protocols, does not necessarily minimize the end-to-end latency of messages.

In this chapter, we present a *space-time routing* framework for instances of these networks that have predictable motion, either over finite time horizons or infinite time horizons due to periodicity. Specifically, we solve this routing problem: Given a set of nodes and how they move up to a certain time in the future, our goal is to construct space-time routing tables that specify when and to whom a node must forward a message in order to meet some routing objective. Our solution is based on a *space-time graph* model of the network derived from the mobility of nodes. The space-time graph model captures the dynamic evolution

and connectivity over time of the network topology. We devise a routing algorithm using this space-time graph model to select a suitable next hop node from the current as well as the future neighbors. Consequently, the next hop for forwarding a message from a node is a function of both the destination and time, unlike traditional forwarding approaches based only on the destination of a message.

The remainder of the chapter is organized as follows. We start with a model of network in Section 4.2 and introduce our space-time routing framework in Section 4.3. We derive our space-time graph model in Section 4.4 that forms a basis for our algorithm to compute routes in Section 4.5. We present our evaluation methodology and results in Section 4.6 followed by a summary in Section 4.7.

4.2 Model

In this section, we describe our model of a network and how messages are forwarded from source to destination. Table 3 summarizes the notation used in our model.

Table 3: Notation used in our space-time graph model

Variable	Meaning
t	time index
v_i	i^{th} node
$G(t) = (V, E(t))$	time-varying graph representing network topology
$P_i(t)$	geographic position of i^{th} node
γ	distance threshold of radio coverage
$L_{ij}(t)$	time-varying link function between v_i and v_j
τ	granularity of time dimension in space-time graph
$\mathbf{T}$	duration of predictability

4.2.1 Network Connectivity

Consider a mobile network comprising n nodes represented by the set $V = \{v_1, v_2, \ldots, v_n\}$. Since the graph representing the connectivity between these nodes varies with time, we have $G(t) = (V, E(t))$ where V is the set of nodes and $E(t)$ is a time varying set of links between these nodes. The time-varying set of links $E(t)$ can also be represented by link functions of time. $L_{ij}(t)$ is a boolean function that represents whether a communication link is present

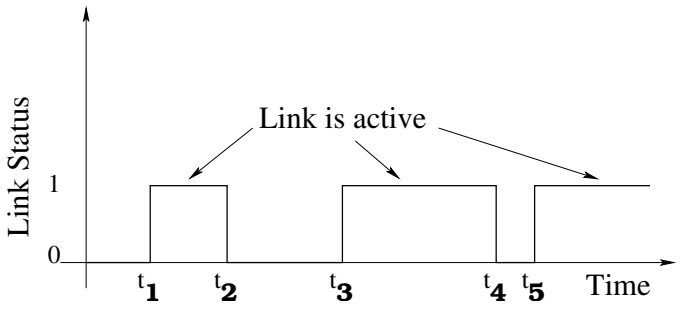

Figure 16: Time-varying boolean function $L_{ij}(t)$ representing a communication link between two nodes v_i and v_j.

or not between nodes v_i and v_j at time t. We assume that two nodes have a communication link when they are geographically close to each other within a distance threshold of γ such that the radio quality of a link between the two nodes satisfies the minimum requirement for a successful data communication. Let $P_i(t)$ denote the geographic position of node v_i at time t. We can define the time-varying link function $L_{ij}(t)$ for every pair of nodes (v_i, v_j) as:

$$L_{ij}(t) = \begin{cases} 1 & \text{if} \quad |P_i(t) - P_j(t)| \leq \gamma \\ 0 & \text{otherwise} \end{cases}$$

Figure 16 illustrates the time-varying nature of the link function. We note that there are $n(n-1)$ such link functions, corresponding to every pair of nodes.

We define duration of predictability $\mathbf{T}$ as the time horizon to foresee the network topology evolution. Specifically, the geographic position function $P_i(t)$ is known from current time (t_c) up to $(t_c + \mathbf{T})$ in the future for all nodes $v_i \in V$. Mobile networks with periodic node movement, as shown in the example network in Section A.1, present a special case for prediction. Periodicity in the individual node motion provides a succinct representation and implicitly gives us an infinite duration of predictability (i.e., $\mathbf{T} = \infty$).

4.2.2 Forwarding Messages

Consider forwarding a message $M = (s, d, t_s, m)$ from a source s to a destination d. M arrives at the source at t_s and takes m time units for transmission from one node to another

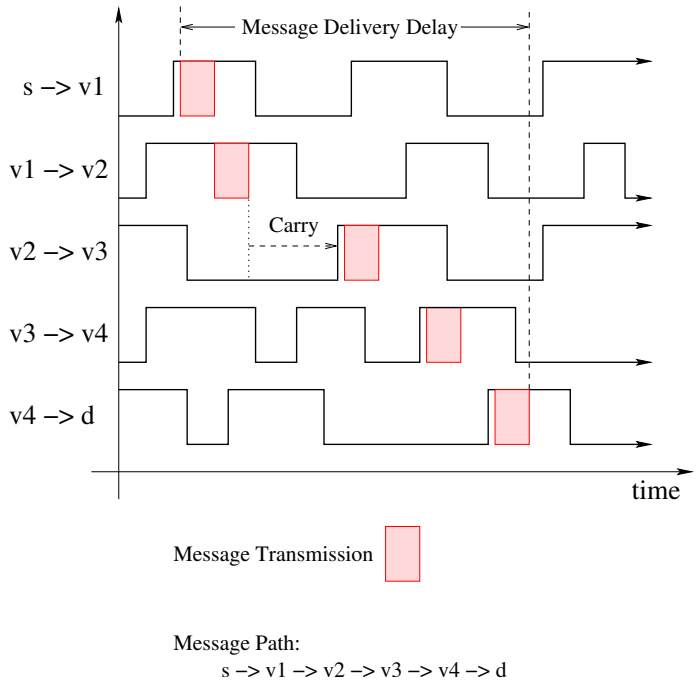

Figure 17: An illustration of forwarding a message from source s to destination d over a sequence of nodes $(s, v_1, v_2, v_3, v_4, d)$. The links between each pair of nodes in this sequence vary with time and the message is forwarded at appropriate opportunities.

[1]. This message M can be transferred from v_i to v_j starting at time t_x only if $L_{ij}(t) = 1, \ \forall t \in \{t_x, t_x+m\}$. In other words, the message transmission interval $[t_x, t_x+m]$ is a subset of an *active* time interval of the link function L_{ij}. Consequently, a message forwarding path is a schedule $P(s, d, t_s, m) = (\pi, \omega)$. Here $\pi = (s, v_1, v_2, \ldots, v_l, d)$ is a sequence of nodes that comprise a path from source to destination and $\omega = (t_0, t_1, \ldots, t_l)$ are the times of start of transmission at respective hops where *(i)* $t_i \geq (t_{i-1} + m)$ – message has to be received completely before forwarding to next hop node and *(ii)* $L_{v_i v_{i+1}}(t) = 1, \ \forall t \in [t_i, \ t_i + m]$ – link is active during the message transmission interval.

Figure 17 illustrates message transmission on a path comprised of dynamically changing links. Intuitively, one can imagine a "band" (of message transmission duration) going from left to right as we travel from source to destination. Occasionally this band has to "slide"

[1]For simplicity, we assume that all communication links have equal transmission speed and that propagation delay is negligible. This assumption ensures that the message takes the same amount of m time units on any communication link. It is straightforward to adapt the formalism for varying link speeds and significant propagation delays.

towards right when there is no link available for transmission immediately to the chosen next hop node. For example, in the figure, after receiving the message from v_1, v_2 could not forward it immediately to v_3 but had to carry for some time until the link $v_2 \rightarrow v_3$ became active.

4.3 Space-time routing framework

4.3.1 Routing Tables

In our space-time routing framework, the message forwarding path $P(s, d, t_s, m) = (\pi, \omega)$ is realized collectively by local forwarding decisions of each intermediate node. Unlike many traditional routing protocols that use only spatial connectivity information, our space-time routing framework considers the time dimension as well as space. Since the network topology changes with time, the "best" outgoing link for a message depends not only on its destination but also on the topology evolution. Therefore, the next hop in our space-time routing framework is a function of both the destination and time. Figure 18 contrasts our space-time routing table with a traditional routing table. A space-time routing table is a matrix of two dimensions, one for destination addresses and the other for instances of time. Entries in this table are actions that might include carrying a message for a certain duration before forwarding to another node. For example, in the figure, a message that is ready for transmission at $(t-2)$ to a destination d_i has to be carried for 2 time units and forwarded on a link to node H. We also note that the next hop node to the same destination d_i could be different at different times due to variation in network topology. Again, in the example figure, next hop node (H) at t is different from that (L) at t'.

4.3.2 Message Transmission Schedule

When a message arrives at an empty node, it looks up the space-time routing table to determine the next hop node and transmission time. Subsequent messages that arrive at the same node look up the routing table to schedule their transmission. However, conflicts can occur as other messages might be queued at the node to use the same transmission opportunity. In that case, we look up the space-time routing table again with a new virtual arrival time that is past the missed transmission opportunity. We may need to repeat this

Destination	Next Hop Node
$\vdots$	$\vdots$
d_i	h_i
$\vdots$	$\vdots$

(a) Traditional Routing Table

Destination Address	Time of Message Forwarding Lookup							
	0	...	$(t-2)$	$(t-1)$	t	...	t'	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
d_i			Carry 2 Forward H	Carry 1 Forward H	Carry 0 Forward H		Carry 0 Forward L	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

(b) Space-Time Routing Table

Figure 18: Comparison of a space-time routing table with a traditional routing table

look up operation several times until we find a feasible transmission time for the message.

4.3.3 Routing Table Construction

We define routing in space and time as the construction of space-time routing tables that aid in forwarding decisions to meet a particular routing performance objective (e.g., to minimize end-to-end delay). A naive solution to the routing problem is to list all feasible paths in space and time from the source to the destination. Among the set of feasible paths, one could search for an optimal path that, say, minimizes overall end-to-end delivery delay of a message. Although this solution is simple, it is computationally expensive; the number of paths can be exponential in the graph size.

Our goal, by contrast, is to

1. assimilate time-varying link information into a space-time graph model (Section 4.4),

2. apply a routing algorithm on these space-time graphs to identify these optimal paths efficiently (Section 4.5), and

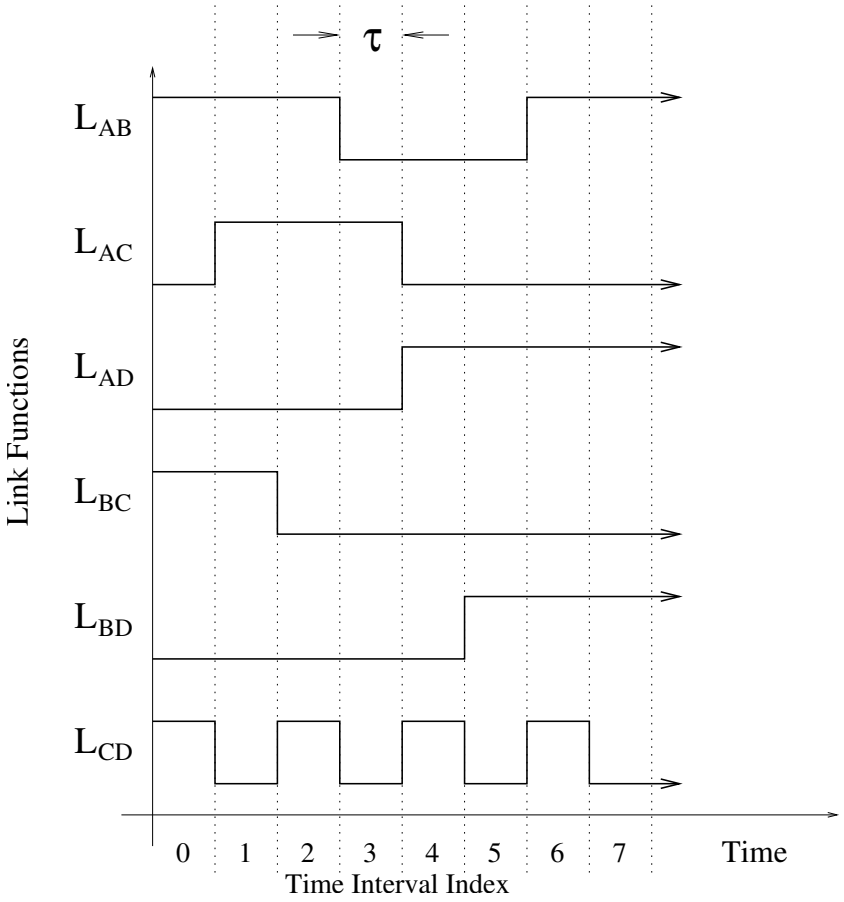

Figure 19: Time-varying link functions of an example mobile network with four nodes $\{A, B, C, D\}$ (different from the one shown in Figure 15).

3. extract forwarding information from the computed paths into space-time routing tables.

4.4 Space-Time Graph

We start with an overview of our space-time graph model followed by details of its construction. For clarity, we illustrate this construction using a hypothetical mobile ad hoc network with four nodes $\{A, B, C, D\}$ whose link functions over time are given in Figure 19. We follow the notation summarized in Table 3.

4.4.1 Overview

Our *space-time graph* model captures both the space and time dimensions of the network topology. The main idea is to construct a layered graph, where each layer corresponds to a discrete time interval (of length τ) in the life of the network. We will return shortly to the issue of discretizing time, but first we describe the layered graph construction.

Starting from the set of nodes V in the network, multiple copies of each node are made and stacked up as layers on each other as illustrated in Figure 20. Each layer of this graph has one copy of every node in the network and a *column* of vertices in this layered graph corresponds to a single node in the network.

We introduce two kinds of links on this layered graph – *temporal* and *spatial* links. A formal description of the spatial and temporal link construction is given in the Appendix. Directed temporal links, shown in Figure 20, connect "time-copies" of the same node between consecutive intervals of time and hence remain within a single column, i.e., the vertical time dimension. Traversing a temporal link denotes "carrying" a message by a node.

Forwarding a message from one to node to another in the network is represented by a traversal of a spatial link. Construction of spatial links on this layered graph follows the link functions L_{ij}. As an example, the spatial links going from node A to node C are illustrated in Figure 21 following the link function L_{AC} drawn in Figure 19. Directed spatial links go from a vertex in one column to a vertex in another column, i.e., span the horizontal space dimension. In order to incorporate message transmission delays, these spatial links are *delayed* so that the destination vertex of each spatial link is as many time layers later than the source vertex as the transmission delay of a message. For example, a transmission of a message of size τ units from node A during interval 1 to node C is represented by the $A_1 \rightarrow C_2$ spatial link. Similarly, a transmission of a message of size 2τ units is denoted by $A_1 \rightarrow C_3$ spatial link, and so on.

We introduce a link coloring scheme to distinguish paths available on the space-time graph for messages of different sizes. Although messages can be arbitrarily long, we consider discrete bins of width τ for message sizes and each bin is assigned a unique color drawn from a set of colors $\{0, 1, 2, \ldots, C_{max}\}$. Temporal links are colored using a special "wild-card"

color 0 that matches all colors. The color of a spatial link is determined by the message transmission delay that the link represents. For example, the spatial links $A_1 \rightarrow C_2$, $A_2 \rightarrow C_3$ and $A_3 \rightarrow C_4$ are all assigned color number 1 as they represent a delay of τ units. Similarly, $A_1 \rightarrow C_3$ and $A_2 \rightarrow C_4$ are colored number 2 as they correspond to message transmission delays of 2τ units.

Routing a message (s, d, t_s, m) of size m from source node s to destination node d starting at time t_s at the source is now equivalent to finding a colored path on this space-time graph. The source s and start time t_s accurately describe the time layer and vertex corresponding to the source. The destination d is represented by the column of vertices spanning all time layers corresponding to node d. The message size m corresponds to a bin of color c where $(c-1)\tau < m \leq c\tau$. A path on the space-time graph using spatial links with color c and wild-card colored temporal links represents a feasible route from source to destination. The shortest path among all these feasible paths corresponds to a route of least end-to-end delay.

A shortest path algorithm can be applied to this space-time graph to find routes. This space-time graph can also be enhanced by assigning appropriate weights to the spatial and temporal links to reflect different routing criteria. For example, one could also assign weights to the spatial links proportional to the cost of a message transmission. A shortest path algorithm on this weighted graph finds routes that use least overall cost of message transmission. Similarly, we could also assign weights to the temporal links to study the effects of queuing and costs of storing messages at intermediate nodes.

4.4.2 Level of Time Granularity

We now consider the issue of discretizing time by choosing an appropriate value of τ. Our choice for the length of this time interval τ is motivated by a need for an accurate, yet succinct representation of the dynamic network topology. In general, a small value of τ implies a large number of layers in the space-time graph whereas a large value of τ would not be able to capture most of the link activity. We describe below a choice of such a representative time unit that is small enough to capture the network topology evolution, but long enough to describe it in less number of layers.

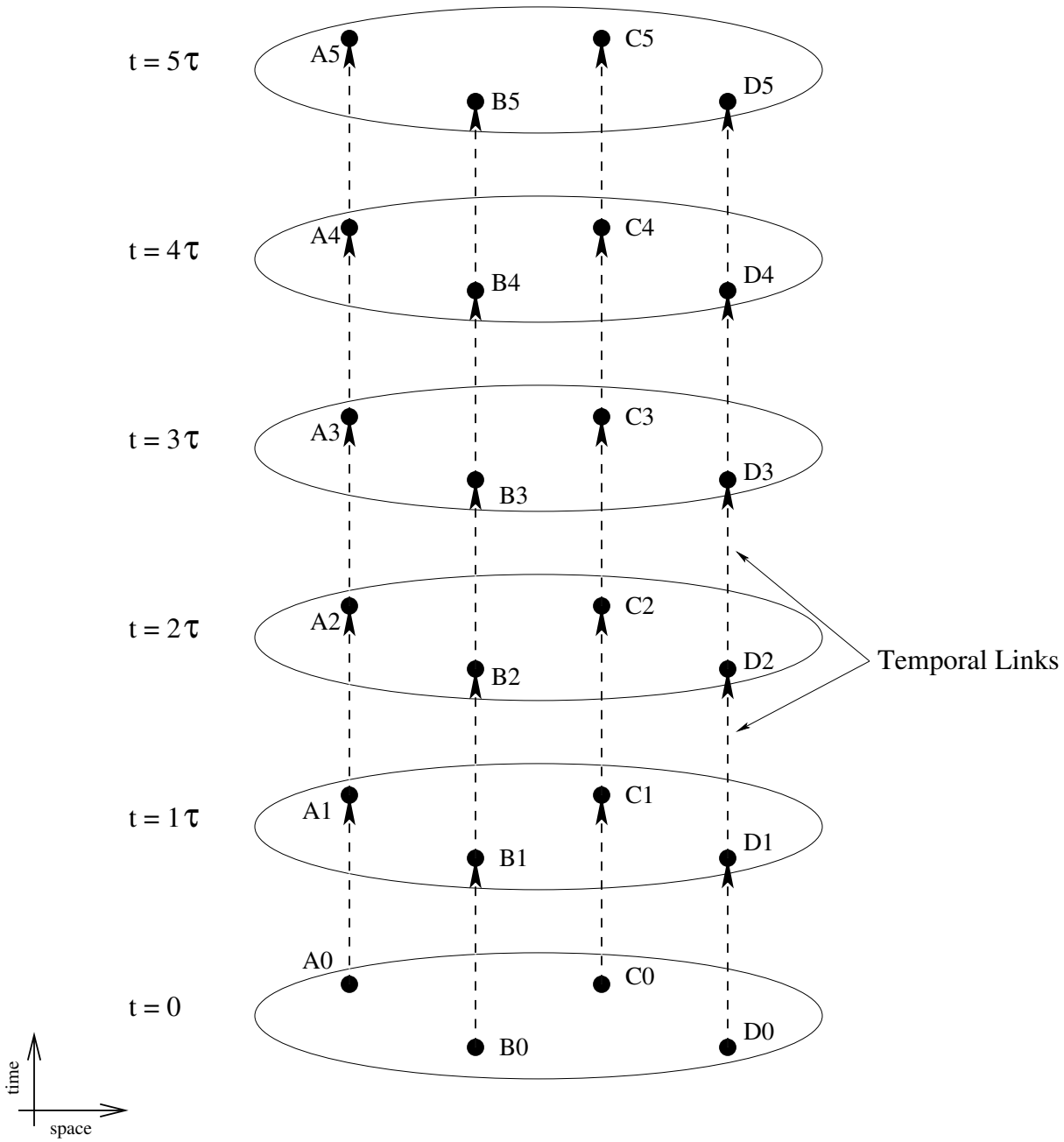

Figure 20: Temporal links of the layered space-time graph. Each layer corresponds to a discrete time interval in the life of the network. Temporal links connect the same node across consecutive layers.

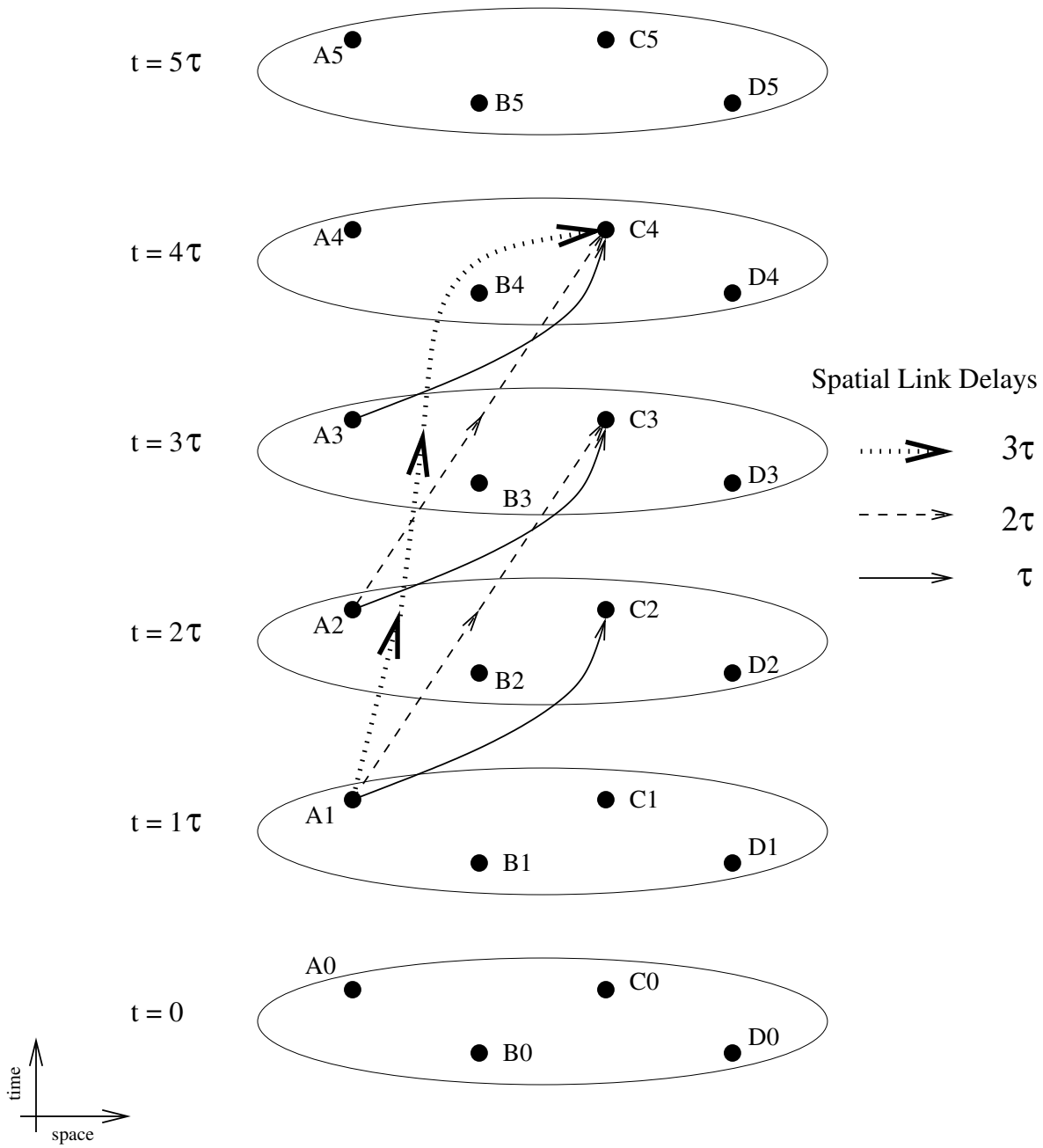

Figure 21: Spatial links of the space-time graph. For simplicity, spatial links that go only from node A to node C following the link function L_{AC} of Figure 19 are shown here. The spatial links are colored appropriately to incorporate message transmission delays as indicated in the legend.

Assuming the network topology is predictable up to a finite horizon we obtain for each pair of vertices, v_i, v_j, all the intervals during which the link between v_i and v_j is active. We then consider the set $\mathcal{T}$ consisting of the lengths of all the active intervals corresponding to all pairs of vertices. Since every link transition in the network is separated by some time length in $\mathcal{T}$, a time unit that is an integral divisor of all the lengths in $\mathcal{T}$ would allow us to capture all the link transition information. Hence, we set the representative time unit τ for the construction of our space-time graph to be the greatest common divisor of all the time lengths in the set $\mathcal{T}$. Figure 19 illustrates the selection of τ based on the greatest common divisor criteria.

4.4.2.1 Link function approximation

Since the link functions of different pairs of nodes are possibly independent of each other and the value of τ depends on the lengths of the active intervals of all pairs of nodes, it is highly likely that the τ obtained using the greatest common divisor criteria might have a very small value. This would in turn result in a large number of layers in the space time graph. To address this problem, we propose the following approximation to the link functions.

We start with a lower bound τ_l that we wish to maintain for a value of τ. A good estimate of τ_l can be derived from the least possible size of messages in the network. Our first approximation ignores any link activity that lasts less than τ_l time units as the link does not last long enough for any message transmission. Next, we align the link function transition points along multiples of τ_l by shortening the link activity interval. For example, with a choice of $\tau_l = 2$, an interval of $[9, 15]$ is approximated to $[10, 14]$ such that the transition points are multiples of τ_l. When the transition points of link functions are thus aligned along the multiples of τ_l, the lengths of inter-transition gaps in the transition point collection would also be multiples of τ_l. Therefore, our representative time unit τ would an integral multiple of τ_l.

Although this approximation loses potential transmission time at the beginning and end of an active time interval, this loss is always less than τ_l time units at either end point. With

a suitable choice of τ_l, the benefits of such an alignment would outweigh the approximation losses incurred.

4.4.3 Space-Time Graph Construction

We describe below the construction of our layered space-time graph $G' = (V', E')$ in a formal way starting from the set of nodes V and the link functions L_{ij} $(\forall i, j)$. We show how the temporal and spatial links are added to the time layers of vertices.

4.4.3.1 Time layers of vertices

For each node in V, we create multiple copies of the node corresponding to each τ-time interval up to $\mathbf{T}$. Hence, the vertex set V' of the space-time graph can be defined as

$$V' = V \times \{0, 1, \ldots, (\mathbf{T}-1)\},$$

$$\text{i.e., } V' = \{v_{it} | 1 \leq i \leq n; 0 \leq t < \mathbf{T}\} \quad (1)$$

where v_{it} denotes the pair (v_i, t), i.e., node v_i at time interval t.

4.4.3.2 Temporal links

For each node in V, we connect the vertices in V corresponding to successive intervals of time by directed links in order to allow for the possibility of messages being carried by the node. Hence, the set of temporal links E_0 of the space-time graph can be defined as

$$E_0 = \{(v_{it}, v_{i(t+1)}) | \quad 1 \leq i \leq n; \quad 0 \leq t < \mathbf{T}\}; \quad (2)$$

When the network topology is periodic, we can also add links that wrap around from time interval $(\mathbf{T}-1)$ to time interval 0. (i.e., $(v_{i(\mathbf{T}-1)}, v_{i0}) \in E_0$).

Since a node can potentially carry messages of arbitrary sizes, we color these temporal links with a wildcard color zero that denotes any color.

4.4.3.3 Spatial links

The spatial links of our space-time graph are derived from the link functions L_{ij} of pairs of nodes in the network. We define different sets E_c of spatial links based on message transmission delays. A spatial link that belongs to E_c has a message transmission delay of

$c\tau$ time units and is colored c. These spatial links are suitable for traversal for messages of size m that fall into the message size bin c.

In order to obtain the set of c colored edges E_c, we consider all the vertex pairs $(v_{it}, v_{j(t+c)})$ such that the link function L_{ij} is active during the interval $(t\tau, (t+c)\tau]$ where τ is the constant time interval between the different layers of the space-time graph, i.e.,

$$E_c = \{(v_{it}, v_{j(t+c)}) |\ 1 \leq i, j \leq n;\ 0 \leq t < \mathbf{T};$$
$$L_{ij}(s) = 1\ \forall s \in (t\tau, (t+c)\tau]\}; \quad (3)$$

Each edge in the set E_c is assumed to be colored c. The complete set of edges E' of the space time graph is obtained as the union of the temporal and all the different colored spatial linkes, i.e., $E' = \bigcup_{c=0}^{c_{max}} E_c$.

4.5 Routing using the Space-Time Graph

In this section, we formulate the problem of routing messages in a dynamic network in terms of finding the shortest paths in the space-time graph described in the previous section. We also propose an algorithm that exploits the structure in the space-time graph in order to efficiently compute the shortest paths between every pair of nodes beginning at any instant of time. Later, we formulate the shortest delay routing problem as a linear integer program for instances of the networks where the traffic load is known in addition to the space-time graph.

4.5.1 Problem Formulation

Of all the paths from source to destination, we are interested in a path that minimizes the end-to-end message delivery delay. Minimizing delay is perhaps the most common routing performance metric. However, we do note that other routing performance metrics, such as number of spatial hops, are potentially of interest, especially when every message transmission is expensive. Although our focus in this section is to minimize delay, the space-time graphs are extensible to incorporate alternative routing performance optimizations.

The space-time graph of a dynamic network topology has the property that the end to end delay for any message is exactly equal to the length of the path traversed in the

space-time graph. Now, routing a message of size m at time t from a source node v_i to a destination v_j corresponds to finding a path of color c (where $(c-1)\tau < m \leq c\tau$) in this space-time graph from vertex v_{it} to any vertex in the "column" set $\{v_{js} |\ 0 \leq s < \mathbf{T}\}$ of the space-time graph. Specifically, minimum delay routing is equivalent to finding the shortest path, i.e, the path with the least overall delay $|s-t|$. This can be formally stated as the optimization problem,

$$\min_{p \in G',\ \mathrm{color}(p)=c,\ v_{it} \xrightarrow{p} v_{js}} |s-t| \quad 0 \leq s < \mathbf{T} \tag{4}$$

where p is a path of color c in the space-time graph G' from the source vertex v_{it} corresponding to the source node v_i at time t and some time copy v_{js} of the destination node v_j.

4.5.2 Routing Algorithm

Our routing algorithm is closely related to the Floyd-Warshall algorithm that computes the shortest paths between all pairs of vertices in a graph [26, 18]. If we only need single-source multiple destination paths, we could techniques similar to the ones described here to design a routing algorithm based on Dijkstra's shortest path algorithm [19, 18].

First, we observe that routing in space time can be formulated as a shortest path problem. Hence, an obvious approach to solve the problem is to apply shortest path algorithms such as the Floyd-Warshall algorithm or the Dijkstra's algorithm. However, since the topology of a mobile network varies with time, a shortest path computed at one time instant may no longer be shortest or even exist at a later time. Hence, neither the Floyd-Warshall nor the Dijkstra's algorithm, in their original form, are directly applicable in computing shortest paths that hold over time. Therefore, we propose techniques to adapt these shortest path algorithms to reflect the notion of "paths across time" on these dynamic network topologies. More specifically, we use our space-time graph and its properties as a basis to compute the shortest paths that are preserved in both space and time.

4.5.2.1 Delay Invariant

The Floyd-Warshall algorithm uses a dynamic programming formulation of the all-pairs shortest path problem to achieve a time complexity of $\Theta(|V|^3)$, where $|V|$ is the total number of vertices in the graph. In a space-time graph G with n nodes and a duration of predictability of $\mathbf{T}$ time units, the total number of vertices is $n\mathbf{T}$ – one vertex for each node at each time unit. A naive adaptation of the Floyd-Warshall algorithm would require $\Theta(n^3\mathbf{T}^3)$ operations. Since $\mathbf{T}$ is usually much larger than n, this would be a very inefficient solution. Fortunately, it is possible to exploit the structure of the space-time graph to reduce the search space and achieve a much better computational complexity of $\Theta(n^3\mathbf{T})$.

In particular, we identify the following *delay invariant* for the shortest paths in our space-time graph.

$$\mathcal{D}(i, j, (t - x), c) \leq \mathcal{D}(i, j, t, c) + x$$

where $\mathcal{D}(i, j, t, c)$ denotes the delay of the shortest path of a specified color c from a source node v_i to a destination node v_j beginning at time t at the source node. Alternatively, $\mathcal{D}(i, j, t, c)$ is the delay corresponding to the shortest path of color c from the vertex v_{it} in the space time graph to any vertex in the column corresponding to v_j (i.e. $\{v_{js} | \ 0 \leq s < \mathbf{T}\}$). The delay invariant states that any message that starts from a source node v_i towards a destination node v_j, but begins earlier than t, say $(t-x)$, will not suffer a delay greater than $(\mathcal{D}(i, j, t, c) + x,)$. To see why the above delay invariant holds, we observe that the delay $(\mathcal{D}(i, j, t, c) + x)$ corresponds to a valid path consisting of temporal links from $v_{i(t-x)}$ to v_{it}, and the shortest path from v_{it} to the space-time nodes corresponding to v_j. Hence, the delay $(\mathcal{D}(i, j, t, c) + x)$ forms an upper-bound on the shortest possible delay $\mathcal{D}(i, j, t - x, c)$ from $v_{i(t-x)}$ to the space-time nodes corresponding to v_j. Note that the delay invariant holds irrespective of the color of the message. We use this delay invariant property to reduce the search space when computing the shortest delay paths.

4.5.2.2 Algorithm Details

Routines 1 and 2 show the pseudocode of our algorithm for computing the shortest paths for all possible messages in the space-time graph. There are two main components in our

algorithm - the initialization phase and the shortest path computation phase. Each of these phases needs to repeated for all possible path colors $c \in \{1, \cdots, C_{max}\}$ in order to obtain the complete routing tables.

The initialization phase computes the shortest path delay between a source vertex v_{it} and any vertex in the set $\{v_{js} |\ 0 \leq s < \mathbf{T}\}$ for all $i, j, t : 1 \leq i, j \leq n,\ 0 \leq t < \mathbf{T}$ based only on paths consisting of at most one spatial link, i.e., not involving any intermediate nodes. This step, however, incorporates all the temporal link information into the delay matrix. In the initialization phase, for each edge $(v_{it}, v_{j(t+c)})$ of a specified color c, we set the $\mathcal{D}^0(i, j, t, c)$ to c. Also, when there is no direct link between v_i and v_j at t, but $\mathcal{D}^0(i, j, (t+1), c)$ has some finite value d, then we set $\mathcal{D}^0(i, j, t, c)$ to $d + 1$. This second operation is based on the the delay invariant property described earlier.

The shortest path computation phase is based on dynamic programming formulation that involves decomposing the problem in the same way as in Floyd-Warshall's original algorithm. More specifically, our algorithm computes for each iteration $k,\ 1 \leq k \leq n$ (outermost loop of the algorithm), the shortest path between a source vertex v_{it} and any vertex in the set $\{v_{js} |\ 0 \leq s < \mathbf{T}\}$ using only intermediate nodes drawn from the subset $\{v_1, \cdots, v_k\}$. Let $\mathcal{D}^k(i, j, t, c)$ denote the shortest path delay computed in k^{th} iteration. The following equation establishes a relationship between the shortest path delays computed in k^{th} iteration with those in the previous $(k-1)^{th}$ iteration. This relationship is also illustrated in Figure 22.

$$\mathcal{D}^{(k)}(i, j, t, c) \leftarrow \min \begin{cases} \mathcal{D}^{(k-1)}(i, j, t, c) \\ \\ \mathcal{D}^{(k-1)}(i, k, t, c) + \\ \mathcal{D}^{(k-1)}(k, j, (t + \mathcal{D}^{(k-1)}(i, k, t)), c) \end{cases}$$

Intuitively, at any time t, for any pair of nodes (v_i, v_j), consider all paths starting from v_i at time t and reaching v_j at some later time, such that the intermediate vertices in these paths is taken from the subset $\{v_1, \cdots, v_k\}$. This operation corresponds to the k^{th} iteration of the outermost loop of our algorithm. Let p be the shortest path among all of them. If

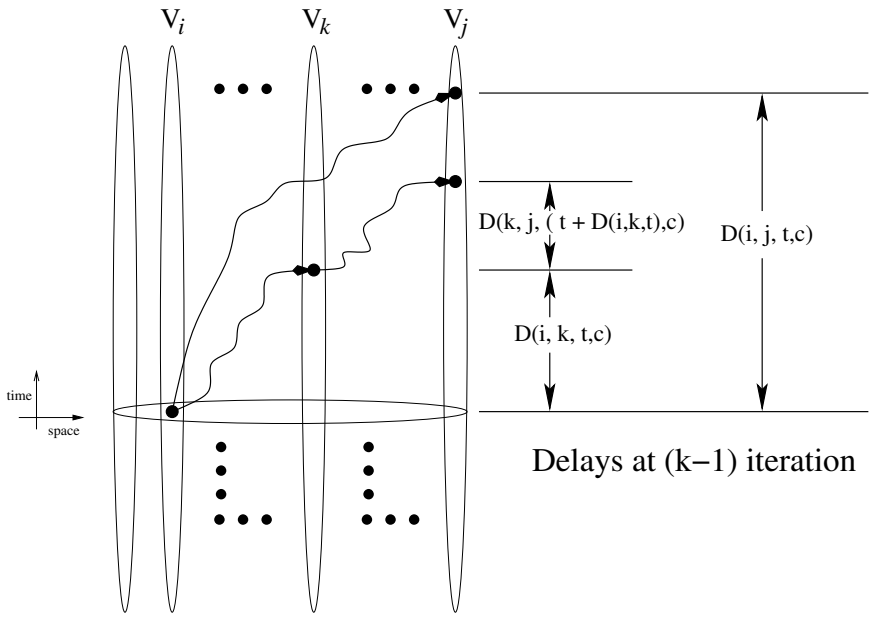

Figure 22: Computing the shortest path delay from source node v_i starting at time t to destination node v_j. At the k^{th} iteration, we compare the new path using node v_k to the shortest delay calculated in previous $(k-1)$ iterations.

p uses only nodes from $\{v_1, \cdots, v_{k-1}\}$, then it would have already been discovered in the $(k-1)^{th}$ iteration. On the other hand, if p uses v_k as an intermediate node, then it can be broken into two sub-paths p_1 and p_2, where p_1 goes from v_i to v_k, starting at time t and taking d_1 time units. The second sub-path p_2 goes from v_k to v_j, starting at time $t+d_1$ and taking d_2 time units. Thus p can be constructed by looking at the shortest path from v_i to v_k starting at t and the shortest path from v_k to v_j starting at $t+d_1$, where both paths use nodes only from the subset $\{v_1, \cdots, v_{k-1}\}$ and hence were found by the $(k-1)^{th}$ iteration. Successively performing this computations till $k = n$ provides the desired shortest delay over all the paths in the space-time graph.

It is easy to see that the initialization phase has a complexity of $\Theta(n^2\mathbf{T})$ and the shortest path computation has a complexity of $\Theta(n^3\mathbf{T})$ based on the `for` loops as the rest of the computations and comparisons are constant time operations.

Routine 1 Shortest_Delay_Path

Input: Space-time graph $G' = (V', E')$ with colored edges, number of vertices n, time period $\mathbf{T}$, message color c

Output: Shortest path length from node v_i at time t to node v_j of color c $\mathcal{D}(i, j, t, c)$, $1 \leq i, j, < n$; $0 \leq t < \mathbf{T}$

Method:

```
D = Initialize_Delay (G', n, T, c)
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            for t = 1 to T do
                d_curr = D(i, j, t, c)
                d_1 = D(i, k, t, c)
                d_2 = D(i, k, (t + d_1), c)
                d_alt = d_1 + d_2
                if (d_alt < d_curr) then
                    D(i, j, t, c) = d_alt
                    intHop[v_i][v_j][t] = v_k
```

Routine 2 Initialize_Delay

Input: Space-time graph $G' = (V', E')$ with colored edges, number of vertices n, time period $\mathbf{T}$

Output: Shortest path length from node v_i at time t to node v_j of color c, $\mathcal{D}(i, j, t, c)$, $1 \leq i, j, < n$; $0 \leq t < \mathbf{T}$ based on temporal links and single spatial hops.

Method:

```
for t = T - 1 downto 1 do
    for i = 1 to n do
        for j = 1 to n do
            if((v_{i,t}, v_{j,t+c}) ∈ E_c) then
                D(i, j, t, c) = c
            elseif (D(i, j, (t + 1), c) < ∞) then
                D(i, j, t, c) = D(i, j, (t + 1), c) + 1
            else
                D(i, j, t, c) = ∞
```

4.5.3 Integer Program Formulation

The integer program formulation assumes the knowledge of message arrivals in addition to the dynamic topology evolution. A solution to this integer program provides a schedule of message transfers that minimizes the average delivery time of messages.

In this analysis, we define message types based only on the (source, destination, color) combinations. For a particular message type $p = (s, d, c)$, let $M(p, t)$ denote the number of messages with length corresponding to the color c that originate at the source node $s = src(p)$ during the time interval $((t-1), t]$ and are destined for the node $d = dst(p)$. For the purpose of integer program formulation of shortest delay routing, we assume that we are given the message traffic arrival functions $M(p, t)$ for all message types p up to a time $\mathbf{T}$ $(0 \leq t \leq \mathbf{T})$.

We start with our colored space-time graph defined in previous section. We enhance this model by defining a capacity function $C : E_c \bigcup E_0 \rightarrow Z^+$ on the spatial links as: $\forall i \neq j$, $C(v_{it}, v_{js})$ is equal to capacity of the link between v_i and v_j in number of messages that can be transmitted in a unit time. For temporal links of the type $(v_{it}, v_{i(t+1)})$, the capacity $C(v_{it}, v_{i(t+1)})$ is defined as the limit on the number of messages that can be stored at the node v_i.

Variables Let $\delta^+(v_{it})$ and $\delta^-(v_{it})$ respectively denote the set of incoming links and the set of the outgoinh links at vertex v_{it}. Let X_a^p be the number of messages of type p carried on a link a and let l_a be the delay associated with the link a. Let $D(p, t)$ be the number of messages of type p that arrive at the destination node $d = dst(p)$ in the interval $((t-1), t]$. For a sufficiently large value of $\mathbf{T}$, the number of messages produced in time period $\mathbf{T}$ is in practice almost equal to the number of messages that reach their destination in the same time period.

Objective Since our objective is to find the optimal schedule corresponding to minimum average delay over a time period $\mathbf{T}$ and the number of packets produced in this time period is constant, we minimize the total delays on each edge in the space time graph, i.e.,

Minimize

$$\frac{\sum_{p \in P} \sum_{a \in E_c \bigcup E_0} X_a^p l_a}{\sum_{t=0}^{T} \sum_{p \in P} M(p,t)} \tag{5}$$

subject to constraints [6], [7], [8], [9], [10], [11], and [12].

Node Conservation Constraints

$\forall i, t, p: \ 1 \leq i \leq n; \ 0 \leq t \leq T; \ p \in P;$

When $v_i \ \neq \text{src}(p); \ v_i \ \neq \text{dst}(p)$,

$$\sum_{a \in \delta^+(v_{it})} X_a^p = \sum_{a \in \delta^-(v_{it})} X_a^p. \tag{6}$$

When $v_i = \text{src}(p)$,

$$\sum_{a \in \delta^+(v_{it})} X_a^p \ + \ M(p,t) = \sum_{a \in \delta^-(v_{it})} X_a^p. \tag{7}$$

When $v_i = \text{dst}(p)$,

$$\sum_{a \in \delta^-(v_{it})} X_a^p \ = \ 0, \tag{8}$$

$$\sum_{a \in \delta^+(v_{it})} X_a^p \ = \ D(p,t). \tag{9}$$

Overall Packet Conservation Constraints

$\forall p : p \in P;$

$$\sum_{t=0}^{T} M(p,t) = \sum_{t=0}^{T} D(p,t). \tag{10}$$

Capacity Constraints

$\forall a: \ a \in E_c \bigcup E_0 (c = color(p))$

$$\sum_{p \in P} X_a^p \leq C(a). \tag{11}$$

Non-negativity Constraints

$\forall a, p: \ a \in E_c \bigcup E_0 (c = color(p)); \ p \in P;$

$$X_a^p \geq 0. \tag{12}$$

4.6 Evaluation

In this section, we evaluate, by simulation, the performance of the routing algorithm described in Section 4.5 by comparing with three other algorithms based on heuristics. We also explain our evaluation methodology and analyze results from our experiments.

4.6.1 Heuristics

Several routing algorithms can be designed using heuristics drawn from routing in traditional networks. We describe three such heuristics below.

1. **Hot Potato Routing** In hot potato routing (HPR), a node forwards a message to the earliest neighbor. When multiple neighbors are available simultaneously, it picks one at random to forward the message. Due to the variation in a node's neighbors over time, hot potato routing, in some sense, "throws the message around" the network until it reaches its final destination. The life-time of a message in the network is limited by a maximum count on the number of hops. Although this local greedy heuristic minimizes the wait time of messages at every node, forwarding loops can occur.

2. **Most Frequent Neighbor Routing** Nodes maintain a set of frequent neighbors in this most frequent neighbor routing (MFN) approach. Neighbor frequencies are computed from observations over a long duration. In most frequent neighbor approach, a node forwards a message to its most frequent neighbor. Like hot potato routing, most frequent neighbor routing is also a greedy heuristic that tries to minimize wait time at a node by seeking the most frequent neighbor as a next hop. A message is no longer forwarded when it reaches a maximum limit on the number of forwarding hops.

3. **Epidemic Routing** In epidemic routing (ER) [74], a node forwards a message to all its neighbors. This forwarding operation is repeated both in space and time. The scope of this flooding operation is limited by an expiration time associated with each message. Unlike HPR and MFN where delivered messages are flushed out of the

network, messages continue propagating in ER in different parts of the network even after a copy of the message might have been delivered to its destination.

4.6.2 Methodology

We have written an event-driven message based simulator for this study. A network of 128 nodes is simulated on a two dimensional plane. We consider a square region of $\{(0,0),(1000m,1000m)\}$ as our simulation area. We use two kinds of node mobility models – one with predictability up to a finite time horizon and the other with periodic node motion that gives infinite predictability. We describe both models here:

- **Finite Time Horizon** When simulation begins, the nodes are populated randomly on the square simulation area. Nodes move in straight lines with a randomly chosen motion vector (*speed, direction and duration*), a variation of the random walk mobility model used in evaluating many ad hoc network routing protocols [14]. Each node changes its motion vector between 1 and 4 times chosen uniformly at random during the entire simulation. Speeds are also sampled uniformly from $[1,5]$ m/s. A node's direction of motion is chosen from the set $\{\pi/4,\pi/2,3\pi/4,\ldots,2\pi\}$ radians uniformly at random. We assume that nodes within 50 distance units of each other can communicate. The finite time horizon extends to 512 simulation units for predictability in node motion.

- **Infinite Time Horizon** Nodes move in circular trajectories[2]. The centers of the circles are chosen uniformly at random from the grid points of the square simulation area. The radii of circles are chosen uniformly from the set $\{100,200,300,400,500\}$. The start location (at $t=0$) of each node is also selected uniformly at random on the circumference of the circle at any multiple of $\pi/8$ radians. The nodes are assumed to move at a constant speed. We chose the angular speeds of motion again uniformly at random from the set $\{\pi,\pi/2,\pi/4,\cdots,\pi/512\}$ radians per time unit. As earlier, we

[2] We have chosen circles for simplicity, but our results extend to other geometries as well. What matters more in the message communication is the likelihood of intersections of node trajectories than the actual geometric shape of these trajectories.

consider a distance threshold of 50 units to allow communication between two nodes. Since the nodes in the network move in a periodic fashion, the network topology has infinite predictability.

We have considered two kinds of message simulation. In the first one, we simulate different forwarding algorithms with only one message in the network. This exercise is to bring out the best case performance of each message forwarding algorithm for a suitable comparison. In the second case, we consider a realistic message traffic model with bursty arrivals of messages. We choose 128 source-destination pairs for exchanging messages. The sources and destinations are selected uniformly at random from the entire node population. The inter-arrival time between bursts is drawn from an exponential distribution with a mean of 10 time units. The length of each burst of messages is sampled uniformly between $[5, 15]$ messages. Our evaluation results are collected from multiple simulation runs, each with different seeds for random number distributions.

The simulation proceeds in increasing order of time-stamps of events. Message arrivals and departures at a node are the main events in our simulation. For each message, we keep track of the nodes visited on its way from source to destination. We also record the arrival and departure times of this message at each of the nodes on the path taken from source to destination. When a message arrives at a node, the forwarding process estimates its departure time and the next hop neighbor from the current node according to the simulated routing algorithm. We have implemented our routing algorithm to identify shortest paths in space and time (SPST) described in Section 4.5 and each of the three heuristics based algorithms described in Section 4.6.1.

4.6.3 Simulation Results

4.6.3.1 Network Topology

Our choice of mobility parameters generates a network that is always disconnected and rapidly varying. For example, in our random-walk motion simulations, a network of 128 nodes has about 61 links on average (standard deviation = 7.8) at any instance of time. About 6.5 links either form or break up every simulation time unit. Clearly, the number of

links are far less to ensure paths between any source destination pair. This poor network connectivity limits the applicability of traditional ad hoc routing protocols. Also, even if a path does exist between a pair of nodes, it is highly likely that it may be disrupted frequently leading to a large overhead in repair and maintenance of routes using traditional ad hoc routing protocols.

4.6.3.2 Success of message delivery

Table 4: Success of message delivery

Routing Algorithm	Delivery success rate (%)
SPST	100.00
HPR	15.7
ER	39.94
MFN	60.08

In this experiment, we studied the success rate of delivery of each of the four routing algorithms. A message is successfully delivered if it reaches its destination before it exceeds its resource limit (e.g., maximum number of hops in hot potato and most frequent neighbor routing). We define success rate of a routing algorithm as the ratio of the number of messages that are successfully delivered to the total number of messages that are forwarded. Table 4 shows the percentage of success rates of the four routing algorithms. Although the network topology is always disconnected at any instance in time, there are paths in space and time that keep the network connected "over time". Our SPST routing algorithm identifies these paths and is clearly able to route almost all messages to their destinations resulting in a very good success rate of message delivery. It is possible, however, that there may not be a space-time path from a source to destination within the time horizon and our algorithm fails to deliver in such a case. Our simulations indicate that such an occurrence is rare and depends on particular node mobility patterns. The other three heuristics based routing approaches perform poorly. They propagate the message around the network until it is either delivered or runs out of resources. Their success of delivery depends on the likelihood of occurrence of the right sequence of links to create a path to the destination and actually using those links

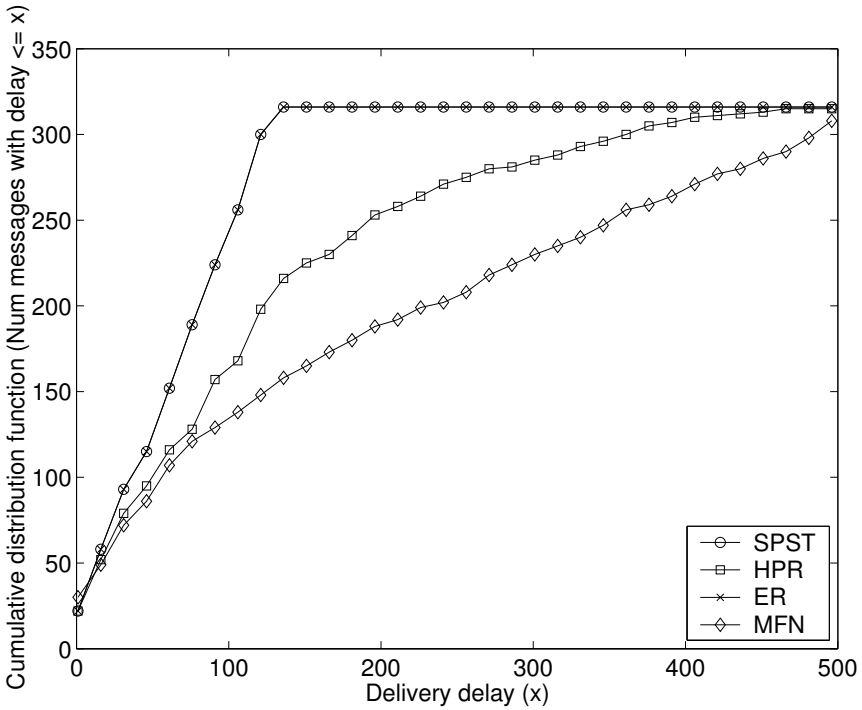

Figure 23: Distribution of message delivery delay from source to destination for various routing algorithms

at the right time. The success rates of HPR, MFN, and ER can be improved by increasing the resource limit but at the cost of increasing resource usage. Other factors also determine the success rates of these heuristics. For example, the structural properties of the network topology, whether it is clustered or not, sparse or dense, determine the number of nodes reached by a flooding algorithm such as ER.

4.6.3.3 End-to-end message delivery delay

In this experiment, we studied the performance of the four routing algorithms with respect to the delay of message delivery. We define the end-to-end message delivery delay as $\Delta = |t_d - t_s|$ where t_s is the message arrival time at the source and t_d is the time of delivery at the destination. The message delivery delay along a route includes the transmission delays of comprising links as well as *opportunity* delays where a message is carried by a node for a certain duration until it meets a suitable node for forwarding.

Figure 23 plots a cumulative distribution function (CDF) of end-to-end message delivery

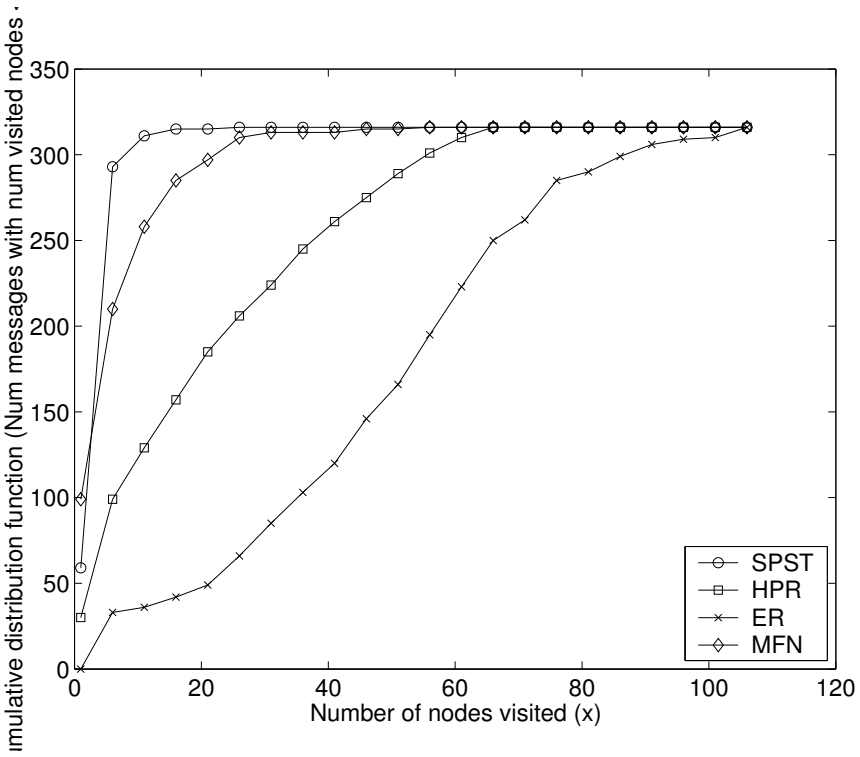

Figure 24: Distribution of number of nodes visited while forwarding a message by various routing algorithms

delays. For a fair comparison, we have included delays of only those cases when a message is successfully delivered by all the four routing algorithms. First, we note that the message delivery delays of SPST and ER are equal and the least. Since ER floods the message along all links in space and time, there is certainly a copy of the message traveling along the shortest delay path identified by our SPST algorithm. Hence, a message reaches its destination after the same time in both these algorithms. Although both HPR and MFN use a greedy heuristic to minimize delay locally at a node, they incur longer delays globally than SPST and ER.

4.6.3.4 Resource usage

We study the cost of message forwarding in this experiment by observing the number of nodes visited by a message on its route from source to destination. This cost is representative of message transmissions as every intermediate node forwards the message. In case of ER, an intermediate node forwards the message more than once to reach its neighbors at different

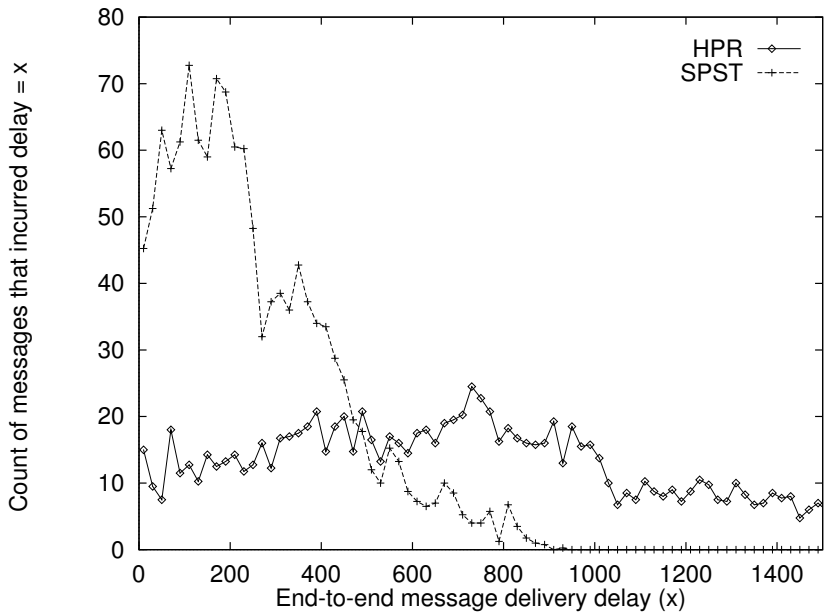

Figure 25: Message delivery delay in bursty traffic model

times.

Figure 24 is a cumulative distribution function of the number of nodes visited by a message as it is forwarded. Similar to the previous plot, this figure also plots values for only those cases when a message is successfully delivered by all the four routing algorithms. Although we have observed that ER has the same message delivery delay as SPST, a message visits many more nodes due to its flood-based forwarding making it an expensive choice for least delay. A message forwarded by SPST visits the least number of nodes as the routes are computed by examining the topology evolution up to the time horizon. HPR and MFN, on the other hand, use many more nodes to reach destination because of their message hand-off nature.

4.6.3.5 Delays with bursty traffic

Our earlier experiments compared the relative merit of each of four forwarding algorithms by examining their best case performance. In this experiment, we evaluate the delays incurred by messages under a realistic message traffic model that includes bursty arrivals. In Figure 25, we plot a distribution of the message delivery delays for our SPST routing algorithm and HPR routing algorithm. The x-axis corresponds to different values of delay

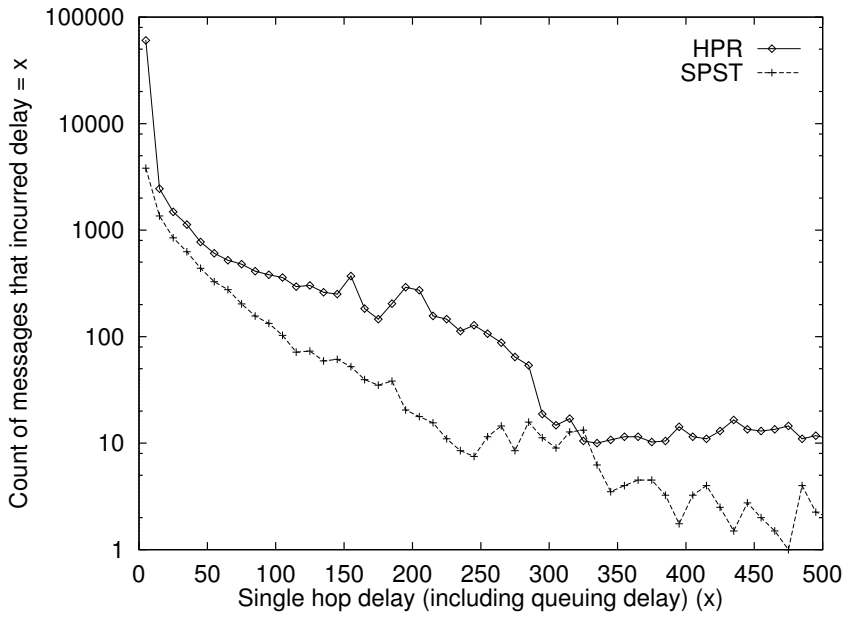

Figure 26: Delay incurred at a single node

and y-axis corresponds to the count of messages that have incurred this delay. We observe that, even under bursty traffic, our SPST routing algorithm does better than hot potato routing in terms of mean end-to-end delay. Figure 26 is a distribution of the delays incurred at a single node. These single hop delays measure the amount of time spent by a message at an intermediate node. The message may wait at a node because either there are other messages ahead of it in the node queue or a link to the appropriate next hop neighbor is not available yet. Since HPR algorithm uses a greedy heuristic, messages incur a lower single hop delay than those that use SPST algorithm.

4.7 Summary

In this chapter, we have considered routing messages in special networks where the topology, due to its disconnectivity, is not amenable to traditional mobile ad hoc routing protocols. In these networks, nodes move around explicitly carrying messages and creating paths over time. Our work explored this idea of routing in both space and time dimensions. Specifically, we took advantage of predictability in node motion available in some of these networks to construct space-time routing tables over a time horizon. These space-time routing tables

consider time in addition to destination to determine the next hop node for forwarding. We derived a space-time graph model that gives a symbolic representation of paths taken by messages as they are routed in space and time. Using these space-time graphs, we presented an adaptation of the Floyd-Warshall algorithm to solve the minimum delay routing problem on these dynamic network topologies that are almost always disconnected at any instant in time. Our evaluation results support our observations that least delay paths in space and time using minimal resources can be identified with very high probability.

CHAPTER V

ROUTING IN UNPREDICTABLE SPARSE AD HOC NETWORKS

Chapter 4 dealt with routing in predictable sparsely-connected ad hoc networks, whereas this chapter concerns routing messages when the network dynamics are unknown and unpredictable, as is often the case in several real-world sparsely-connected ad hoc networks.

5.1 Introduction

Sparsely-connected networks are characterized by the non-existence of paths between most pairs of nodes, due to the lack of enough edges on the topology graph. When the network topology is dynamic, for example, due to node mobility, sparse-connectivity often means that a node has a few and rare contact opportunities with other nodes. When they do occur, these contact opportunities often last for short durations only. The scarcity of contact opportunities presents a challenge in transferring messages, particularly those that are large[1]. Our focus in this chapter is on routing large messages in sparsely-connected networks.

While sparse-connectivity is a key concern in these networks, the lack of knowledge and unpredictability of the network topology further limits the options for a routing protocol. Popular choices for routing with unknown network topology are based on message replication [74, 40]. In these protocols, a node replicates a message and forwards along multiple paths to eventually deliver the message to its destination. The message replication could occur, for example, only at the source, and/or every intermediate node. Also, a node could replicate the message not just once, but multiple times, perhaps until an expiration time associated with the message. Despite the large overhead incurred by these routing protocols

[1] Although the size of a message is a relative measure compared to the duration of contact opportunities, we do believe that several delay-tolerant applications exist where large messages are transferred from source to destination. For example, "Postmanet" proposes transfer of large video files over high-delay postal networks [76].

due to message replication, they are perhaps the best option when the network topology is unknown.

Although message replication based routing protocols were originally proposed to address the lack of knowledge of network topology, the network itself was not necessarily considered to be sparsely-connected. In our work, however, we encounter challenges while adapting a message replication based routing protocol for sparsely-connected networks. These challenges are particularly evident when large messages are fragmented to fit the short contact opportunities and these fragments are replicated and forwarded independent of each other. Table 5 is a simple example to illustrate the challenges of fragmentation and replication considered together as a solution for unpredictable and sparsely-connected networks.

Table 5: A simple example to illustrate the consequence of fragmentation and replication in sparsely-connected networks with limited contact opportunities.

Time	Node Contacts	Choice of fragments	
		Same	Different
t_1	$A \rightarrow B$	m_1	m_1
t_2	$A \rightarrow C$	m_1	m_2
t_3	$B \leftrightarrow C$	?	m_1/m_2
t_4	$B \rightarrow D$	m_1	m_1
t_5	$C \rightarrow D$	?	m_2

Consider a network of four nodes $\{A, B, C, D\}$. Let A be the source of a large message M to the destination D. Let M be fragmented into m_1 and m_2 to fit into short contact opportunities that are typical of a sparsely-connected network. B and C are intermediate nodes that can transfer the fragments of the message M to its destination. The meetings between nodes occur in the increasing order of time-stamps as shown in the table. Let us say that when node A meets B, the contact opportunity lasts for a short time that allowed transmission of only one fragment m_1. Let us also assume that next contact opportunity between A and C also lasts for a short duration allowing transmission of only one fragment. If A forwarded the same fragment m_1 to C, then nodes B and C do not have anything to exchange when they meet later. Also, the destination D could get m_1 from B, whereas

D could not get any new fragments from C. This degradation in performance could be avoided, for example, if A forwarded m_2 instead of m_1 to C. In that case, B and C could exchange either m_1 or m_2 when they meet. Also, D could get m_2 from C, thereby delivering both fragments (and hence the entire message) by time t_5 whereas the previous choice of A sending the same fragment to B and C would incur further delay to deliver the second fragment m_2 to D.

The example highlights an important observation that current transmissions affect the utility of future contact opportunities. In the example, the transmissions $A \stackrel{m_1}{\rightarrow} B$ and $A \stackrel{m_1}{\rightarrow} C$ determine whether any fragment can be exchanged at a future contact opportunity $B \stackrel{?}{\leftrightarrow} C$. Although this example illustrates the effect only on direct neighbors of A, one could imagine this effect at a multiple hops away in the future. Also, this effect need not be limited to fragments of a message between a single source-destination pair, but could be extended to messages from multiple source-destination pairs as well. In addition to the under utilization of the contact opportunities, an important consequence of this effect is the large end-to-end delays in delivering messages. In the case of large messages, the message may not be useful until all the comprising fragments are delivered and the delivery delay of the message could be the delay of delivering the last fragment that completed the message.

In this chapter, we address the problem of delivering large messages in an unpredictable sparsely-connected ad hoc network. Our solution comprises two components. First, we use an adaptive scheme that determines the transmission order of fragments at an intermediate node. The adaptive transmission order ensures that all fragments *spread* through the network, leading to a diversity of message composition at nodes. Consequently, nodes can improve the utilization of contact opportunities. Furthermore, such simultaneous propagation of all fragments not only maximizes the probability of message delivery but also lowers the overall end-to-end delay of all the fragments of a message. The second component of our solution is to *erasure-code* the original message at the source in addition to using adaptive transmission. In a typical erasure coding scheme, an original message of n fragments is encoded into $(n+k)$ fragments. Although the number of fragments have increased, the destination node needs to collect only n out of $(n+k)$ encoded fragments to decode the original

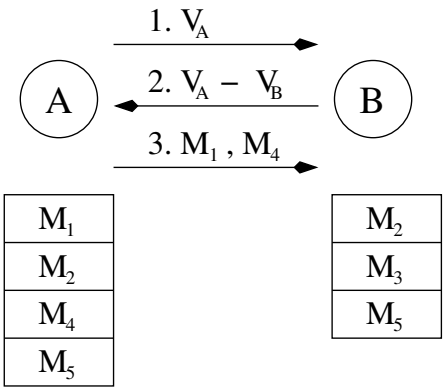

Figure 27: Typical operations at an intermediate node in replication based routing protocol for sparsely-connected networks.

message. The larger number of fragments improves the chances that the destination node finds new fragments with every contact opportunity as opposed to the non-erasure-coding scheme. The increased chances of discovering new fragments can also lead to a decrease in the overall end-to-end delivery delay.

The rest of the chapter is organized as follows. A model of the network, the routing protocol and our assumptions about fragmentation are stated in Section 5.2. In Section 5.3, we describe our adaptive transmission scheduling scheme and our message encoding approach at the source following erasure codes. We present our evaluation methodology and results in Section 5.4 followed by a summary in Section 5.5.

5.2 Model

We follow the model of a sparsely-connected ad hoc network as described in Section 4.2. The model is consistent with the descriptions used elsewhere in the literature [24, 32, 74]. In our model, a node meets other nodes while moving along during its life-time and interacts for a short duration. It is during these intermittent contact opportunities, that nodes can exchange messages. Following is a brief description of a replication-based routing protocol that is perhaps the most popular among different routing protocols proposed for unpredictable ad hoc networks. Later, we describe our assumptions about message transmission in sparsely-connected networks.

5.2.1 Replication-based routing protocol

The underlying theme of most of the routing protocols for ad hoc networks where the network topology evolution is unknown apriori is to follow heuristics. A popular heuristic is to replicate and forward every message to different neighbors assuming that one of the different paths would eventually lead to the destination. To illustrate how messages are transferred from a source to destination under these replication based forwarding schemes, consider the operations performed at an intermediate node as shown in the Figure 27. When two nodes A and B meet, they exchange their summary vectors V_A and V_B that denote the message composition of their current buffers. Subsequently, they use the communication link to exchange the messages in the summary vector difference (i.e., the messages that a node hasn't seen earlier, but are available at the other node). The figure denotes the three steps involved in transferring messages M_1 and M_4 from A to B. These operations are repeated at every node (including the source) until the message reaches its destination. Since this message replication can potentially exhaust all communication and storage resources, these protocols usually include an expiration time-stamp and/or a limit on the number of hops that a message can travel. A message is discarded when it either expires or exceeds the number of "hops-to-live".

5.2.2 Fragmentation

Fragmentation is a common solution when the message size exceeds the length that can be fit in a transmission opportunity. Traditionally, fragmentation has been associated with discrepancies in the underlying media technologies that carry the data. In such cases, protocols typically find a maximum transmission unit (MTU) that meets all the underlying media technologies in the path from source to destination. In our context, however, it is not just the media technologies, but rather external factors (e.g., node mobility) that affect how long a communication link can be active. Thereby limiting how much data can be transmitted, given finite bandwidth of the communication link. Additional factors such as interference, physical obstacles exacerbate this condition of limited transmission opportunities.

Based on when a message is fragmented, fragmentation schemes can be of two types – proactive and reactive. A proactive fragmentation scheme usually involves a MTU discovery and the source breaks the large message into fragments of size less than the MTU. All other nodes simply forward the fragments and do not need to divide the fragments further. In a reactive fragmentation scheme, however, a node attempts to transmit a large message. But if the transmission opportunity ends before the entire message could be transmitted, the node considers the termination point as a fragment boundary – one fragment is successfully transmitted whereas another fragment is not. In this chapter, we assume a hybrid fragmentation scheme to address the lack of knowledge of contact opportunity duration in unpredictable sparsely-connected networks. For simplicity, we assume a small-time unit τ to be our fragment size[2]. The source of a large message divides it into multiple fragments, each of size τ units. Every node tries to forward as many fragments as possible in a sequence until the end of the transmission opportunity. When the transmission opportunity terminates abruptly while a fragment is being transmitted, only that particular fragment is lost. Our simplicity allows a boolean condition that every fragment is either successfully transmitted or not. The choice of τ as fragment size is determined based on the maximum length of transmission that we can tolerate as being lost.

5.2.3 Message vs. Fragment

Throughout this chapter, the term "message" refers to an application data unit with a particular length that is transferred from a source to one or more destinations. Each message, in its entirety, is meaningful to the application at either source or destination. On the other hand, the term "fragment" refers to a piece of the message. All fragments of a message must be received by the destination so that the application can use the message. Therefore, the collective delivery performance (e.g., end-to-end delay) of all the fragments together is more important than the individual delivery performance of each fragment considered separately.

[2]The actual fragment size (e.g., in bytes) is computed by multiplying the time unit τ by the link communication bandwidth.

5.3 Solution Approach

In this section, we present our solution to the problem of transferring large messages in an unpredictable sparsely-connected network. The solution is described in steps, building one on top of the other.

5.3.1 First principles

The first principles approach is a straightforward application of the replicate-and-forward routing protocol without any message fragmentation. Nodes attempt to transmit messages at every contact opportunity; but the transmissions are successful only when the contact opportunity lasts longer than the time necessary to transmit the large message. Most contact opportunities in sparsely-connected networks, however, are rare and short. Consequently, nodes typically have to wait very long until they find a suitably long contact opportunity to transmit the message. Therefore the messages typically propagate slowly leading to long end-to-end delays from source to destination. Although these messages may not travel many hops, they could perhaps expire even before they reach their destinations due to the long waiting periods.

5.3.2 Fragmentation and Scheduling Transmissions

We now consider the case of fragmenting the large message at its source. The fragments are injected into the network by the source and they propagate towards the destination using the replicate-and-forward routing protocol. Every intermediate node attempts to replicate-and-forward as many fragments as possible in a contact opportunity.

In a typical setting, when multiple source destination pairs communicate, the message storage buffer at an intermediate node resembles that of Figure 28. The node may contain fragments belonging to different messages as indicated by shaded regions in the figure. Also, the node may not contain all the fragments of a single message and they need not be in a sequential order too. Each fragment may have arrived at this intermediate node traversing different paths, so the resource consumption limit (TTL) may vary.

Since the current transmissions affect future transmissions as we noted in the example in

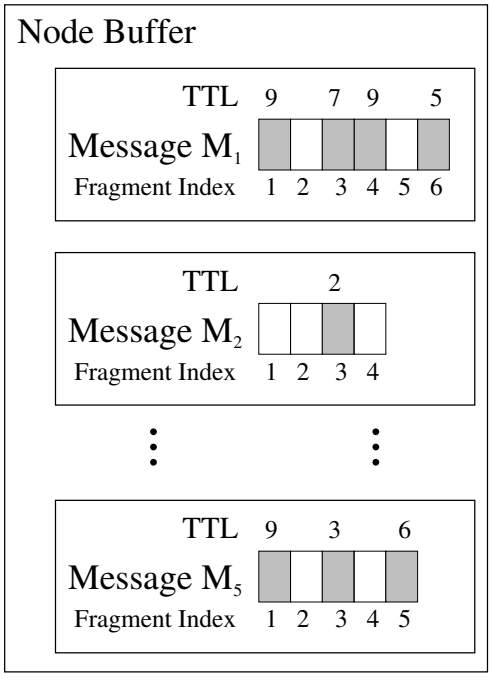

Figure 28: An illustration of fragments of multiple messages at a typical node in the network.

Section 5.1, a critical operation that we consider here is the construction of the transmission schedule. Although the transmission schedule determines the order of fragment transmissions, the success of these transmissions depends on the particular contact opportunity. For example, errors during the contact opportunity and its duration limit the fragment transmissions and these factors are assumed to be unknown and unpredictable in our context of sparsely-connected networks. However, we assume a best-effort strategy of following the transmission schedule. Three heuristics to construct the transmission schedule follow and their operations are illustrated in Figures 29, 30, 31. In each of these figures node u has seven fragments, possibly belonging to one or more messages. The node u is replicating and forwarding these fragments in its contact opportunities with other nodes v_1, v_2 and v_3. These three contact opportunities of u follow each other as indicated by the high-level along time. The fragments transmitted within each opportunity are marked within these timeslots. Although these figures illustrate a special case when all the fragments are of equal size, they need not necessarily be so. Our heuristics on transmission schedule construction can be easily adapted to variable fragment sizes.

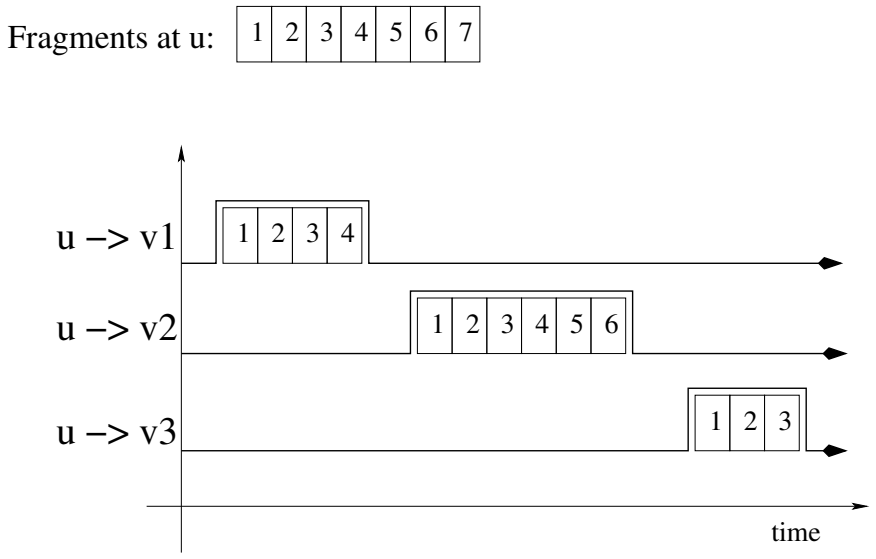

Figure 29: A transmission schedule of fragments that always starts from the beginning for every contact opportunity.

5.3.2.1 Start from the beginning

Consider an orderly sequence of fragments sorted in an ascending order with a key composed of their fragment IDs within a message and their message IDs. In this scheme, a node always transmits from the beginning of the sequence for every contact opportunity. This simple scheme is independent of the outcome of previous transmissions and the particular neighbor providing the contact opportunity. Although it is simple, the bias towards transmitting the first few (higher rank) fragments in every contact opportunity tends to *starve* the fragments that are later (lower rank) in the sequence. Consequently, the highly ranked fragments propagate faster through the network and reach the destination earlier than the lower ranked fragments. In some cases, the lower ranked fragments could expire before they reach the destination. The skew in propagation affects the overall end-to-end delay particularly when all the fragments – both higher and lower rank – are necessary to complete the message at the destination.

5.3.2.2 Continue and wrap-around

In this scheme, a node keeps a "marker" to remember the last fragment that was successfully transmitted in the previous contact opportunity. Starting from the position of the marker,

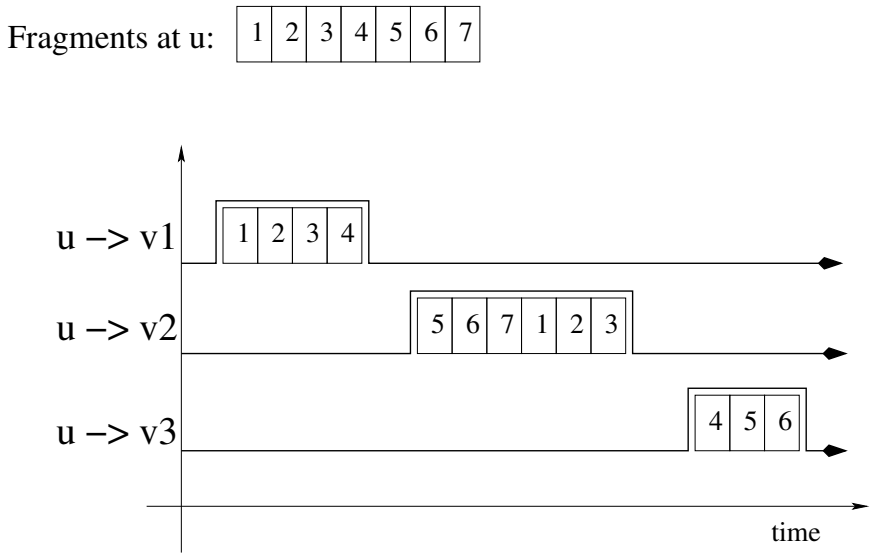

Figure 30: An alternative transmission schedule of fragments where the node continues to transmit from where it left-off from the previous contact opportunity. The sequence order of fragments is still preserved, but it wraps-around once it reaches the end.

the node continues to transmit the fragments in the same orderly sequence as above, but with a wrap-around to cover all the fragments. With a simple marker, this scheme promotes diversity in message composition at nodes in the network as each node forwards different fragments to different neighbors. A variation of this scheme includes maintaining such "markers" per every neighbor, so that consecutive contact opportunities with the same neighbor may be used to transmit different fragments.

5.3.2.3 Random shuffle

As the name "random shuffle" suggests, the node constructs a random permutation of the fragment sequence for every contact opportunity. Fragments are transmitted according this random permutation. The random shuffle scheme also promotes diversity in message composition as the previous scheme as the fragments transmitted at a contact opportunity may be different from those transmitted at earlier contact opportunities. However, in this randomization, all fragments in the node's storage buffer are equally likely to be transmitted, whereas the "continue-and-wrap-around" scheme gives precedence to fragments that were not transmitted in previous contact opportunity.

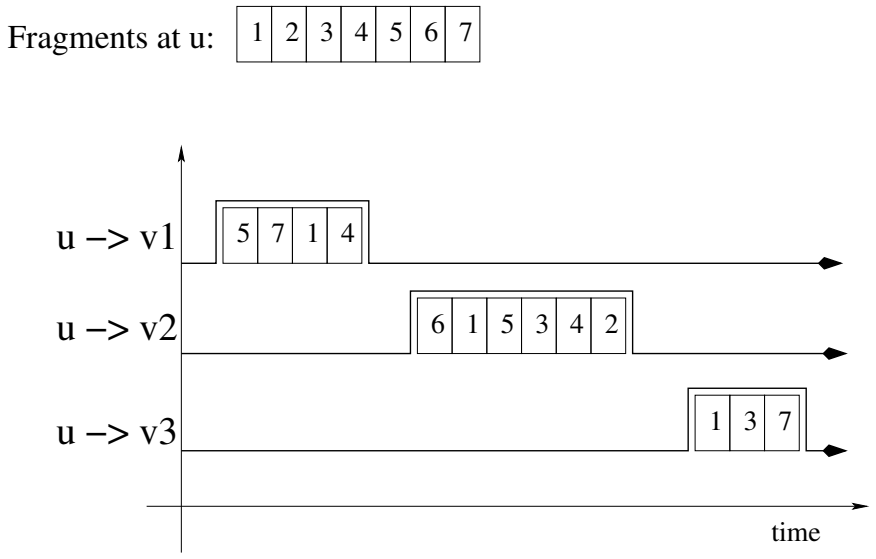

Figure 31: A transmission schedule obtained by shuffling the fragments randomly. The node constructs this random permutation on every contact opportunity and transmits as many fragments as possible.

5.3.2.4 Other scheduling factors

Although, the three schemes described above suggested sorting based on the fragment and message IDs, there are several other factors that could be considered while constructing a transmission schedule. For example, one could consider fragments that are about to expire soon (or perhaps similar resource constraints such as hops-to-live) to be transmitted before other fragments. Also, if the neighbor in a contact opportunity is the destination of some of the fragments, the node may prefer to transmit those fragments ahead of other fragments in its storage buffer.

5.3.3 Encoding at Source

We complement our heuristics on scheduling transmission with a message-encoding scheme at the source using erasure codes. Erasure codes are a class of error-correcting codes[9, 41, 53, 75]. Typical applications in networking and telecommunications have been for forward error correction, for example, to improve reliability of message transfers [60, 12].

The erasure codes can encode n fragments of an original message at the source into $(n + k)$ encoded fragments. The source injects these $(n + k)$ encoded fragments into the

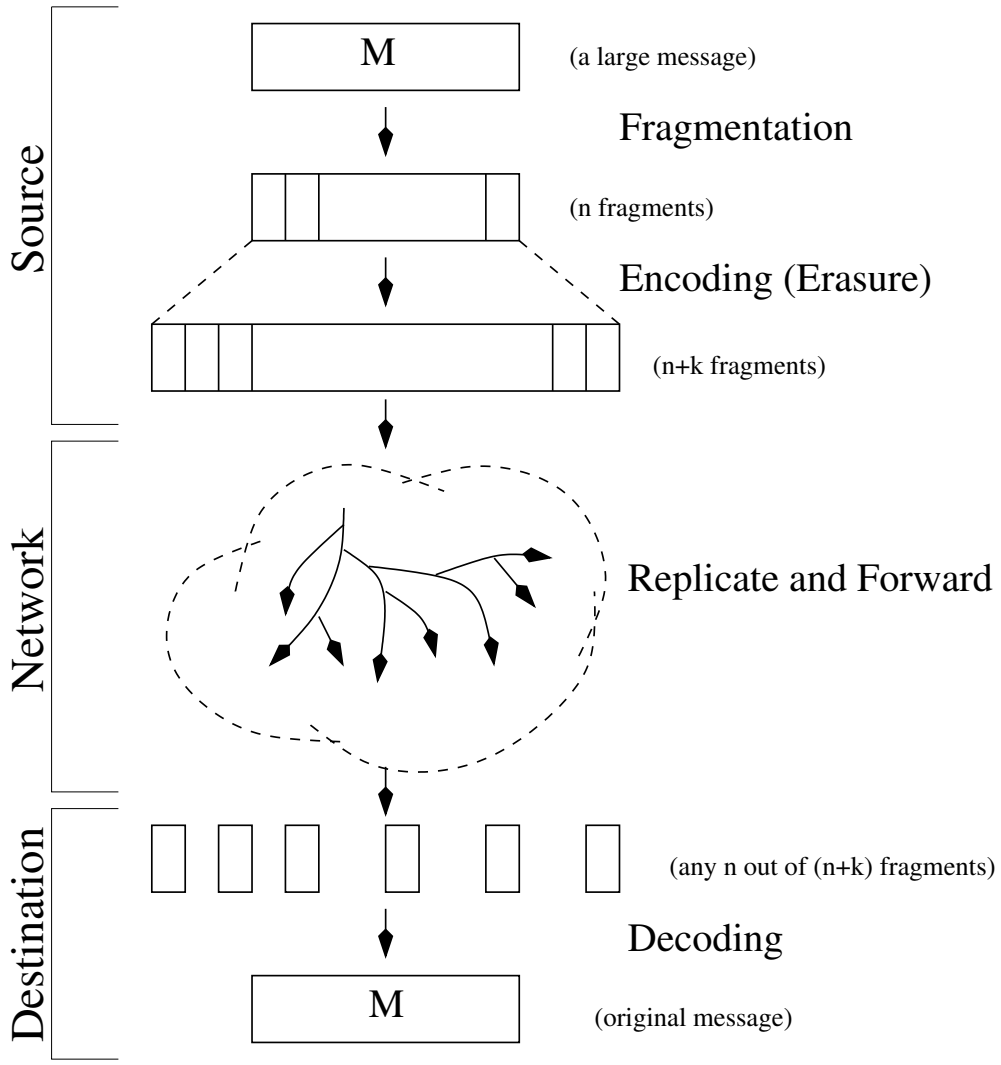

Figure 32: An illustration of a typical sequence of operations in our solution approach to forward a large message from source to destination.

network and they propagate according to the replicate and forward routing protocol. A key property of the erasure codes is that the destination node needs to receive any n out of $(n + k)$ encoded fragments to decode the original message. Figure 32 is an illustration of the sequence of operations in this enhanced approach of fragmentation with erasure coding.

Enhancing the fragment transmission scheduling schemes with erasure coding has two key advantages. First, the increase in the number of fragments (i.e., from n to $(n+k)$) provides a greater choice for nodes to forward fragments within limited contact opportunities.

Also, it decreases the chances that two nodes have exactly similar content in their storage buffers that may lead to under utilization of the contact opportunities. Secondly, the property of "any n out of $(n+k)$" mitigates a problem that we refer to as the "last-fragment-delay" problem. When fragments are replicated and forwarded, the fragment inter-arrival times at the destination node typically increase with the number of fragments. The increasing trend in the inter-arrival times at the destination node is due to the decreasing chance of finding a new fragment that the destination node hasn't seen earlier. This delay (inter-arrival-time) tends to be quite high particularly for receiving the last (n^{th}) fragment as the destination node may have to meet several other nodes before discovering the last fragment. In our enhanced scheme with erasure coding, however, since there are many more $(n+k)$ fragments in the network, we expect that the last fragment delay to receive the n^{th} fragment may be lower. Furthermore, the error correcting property of the erasure codes to improve the overall reliability of message delivery is beneficial to several sparsely-connected networks that suffer losses in transmission.

5.4 Evaluation

In this section, we evaluate, by simulation, the performance of our transmission scheduling heuristics enhanced with erasure coding in sparsely-connected ad hoc networks. Our evaluation also includes a comparison to the direct application of the replicate-and-forward algorithm to these sparsely-connected networks. We begin with our simulation methodology followed by an analysis of results from the experiments.

5.4.1 Simulation Methodology

We wrote a discrete event driven simulator exclusively to evaluate performance of routing protocols on sparsely-connected ad hoc networks. The simulator provides several parameters to tune the network behavior. For example, decreasing the node density and radio coverage range leads to a sparse network that has few and rare contact opportunities between nodes. The simulator also allows adjustment of node speeds and directions that affect the durations of contact opportunities with other nodes. In this study, we use a mobility model that is a variant of the popular random walk mobility used in evaluating many ad hoc network routing

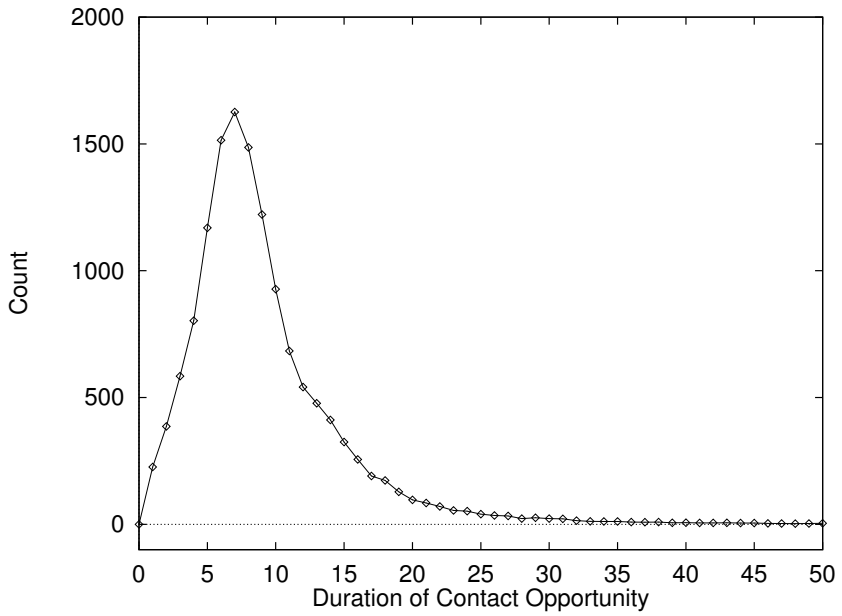

Figure 33: Distribution of durations of contact opportunities in the simulated network.

protocols [14]. In this model, 256 nodes are populated randomly on a square geographical region of $1000m \times 1000m$. Nodes move in straight lines with a randomly chosen motion vector (*speed, direction and duration*). Each node changes its motion vector between 4 and 8 times chosen uniformly at random during the entire simulation. Node speeds are sampled uniformly from $[1, 5]$ m/s. A node's direction of motion is chosen from the set $\{\pi/4, \pi/2, 3\pi/4, \ldots, 2\pi\}$ radians uniformly at random. We assume that nodes within $25m$ of distance of each other can communicate.

Our choice of parameters for the simulation generates a network that is sparsely-connected with a very small number of contact opportunities. For example, the network of 256 nodes has only 62.7 communication links (on an average, with a standard deviation of 7.7) at any instance of time. About 13.3 communication links (on an average) either form or break up every simulation time unit. Also, whenever a communication link does form between a pair of nodes – a contact opportunity – it lasts only for a short duration. A distribution of these durations of contact opportunities is plotted in Figure 33. Clearly, the small number of contact opportunities and their short durations indicate the need to be judicious about transmitting the right set of messages.

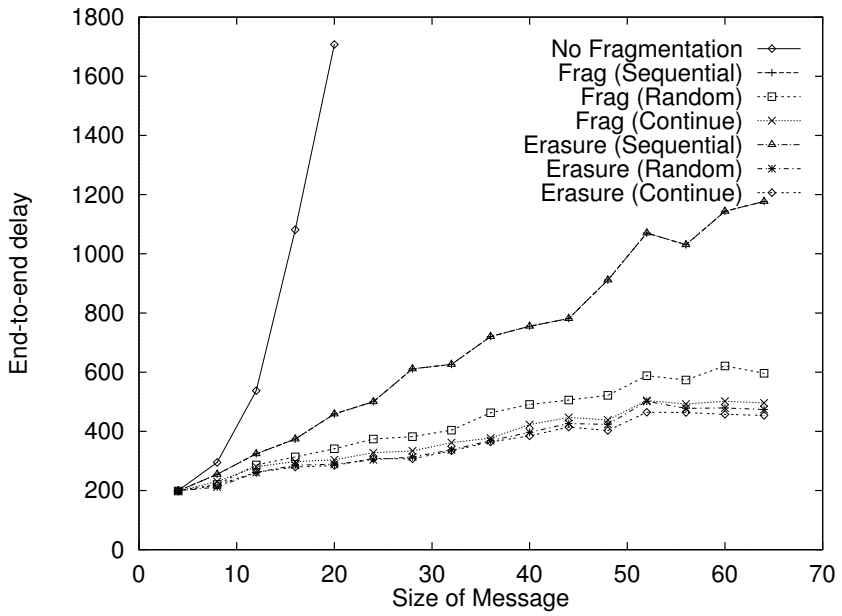

Figure 34: Comparison of end-to-end message delivery delays under different schemes.

5.4.2 Experimental Results

5.4.2.1 End-to-end delivery delay

The end-to-end delivery delay of a message is defined as the time elapsed from the message creation at the source to its arrival at the destination. The replicate and forward routing protocol on a well-connected network forwards a message on most (possibly all) paths from source to the destination. In such networks, the set of paths is often quite large and includes the shortest delay path to the destination. However, in the case of a sparsely-connected ad hoc network, due to limited contact opportunities and finite communication bandwidth, the simple replicate and forward strategy may be able to forward the message only on a small set of paths from source to the destination that may not necessarily include the low delay paths. Although the techniques proposed in this chapter do not explicitly seek the shortest delay paths as a new optimized routing protocol would, our goal is to lower the end-to-end delays within the framework of the replicate and forward routing protocol. We constrain ourselves to the replicate and forward routing protocol primarily for its suitability to unpredictable networks. The particular techniques we consider are fragmentation, the schedule of fragment transmission, and encoding at the source. We evaluate the routing

protocol performance under these variations in this experiment.

Figure 34 is a plot of end-to-end delivery delay as the size of the message is varied from 4 to 64. We consider transfer of one message from source to destination under different schemes using the replicate and forward routing protocol. The plot is an average value of experiments run with multiple src-dst pairs and multiple trials to avoid bias in random number generation. The size of the message is translated to time units to have a direct correspondence with the duration of contact opportunities that are used for message transmissions. The first curve is for transmitting a message "as-is" without any fragmentation in the network. As expected, the end-to-end delays increase rapidly with message size as nodes have to wait for long durations to find the right-sized contact opportunity to transmit these long messages. Beyond a certain message size, the delays are so long that the message expires even before it is delivered to its destination. Therefore, the curve does not show any points for messages longer than 24 units.

The rest of the curves use fragmentation and propagate the fragments through the network using the replicate and forward routing protocol. The end-to-end delivery delay in these cases with fragmentation counts the arrival time of the last fragment at the destination as all the fragments are necessary to make up the message. The curves are for the cases of fragmentation considered alone and together with erasure coding. All these curves have lower delays than the first case of forwarding without fragmentation. In all these cases, the nodes could transmit as many fragments as possible in each contact opportunity and thereby the fragments were propagating faster (and reached the destination earlier) than the original message could.

Two of these six curves that use a sequential order of transmitting fragments overlap with each other. In this case of sequential order, nodes construct a transmission schedule that always starts from the beginning for every contact opportunity. Two more curves are for the "continue-and-wrap-around" scheduling scheme that resumes from the last fragment forwarded in the previous contact opportunity and wraps around the order of fragments. The last two curves use a random order of fragment transmissions. Also, the case of fragmentation enhanced with erasure coding of fragments at the source exhibits lower delay.

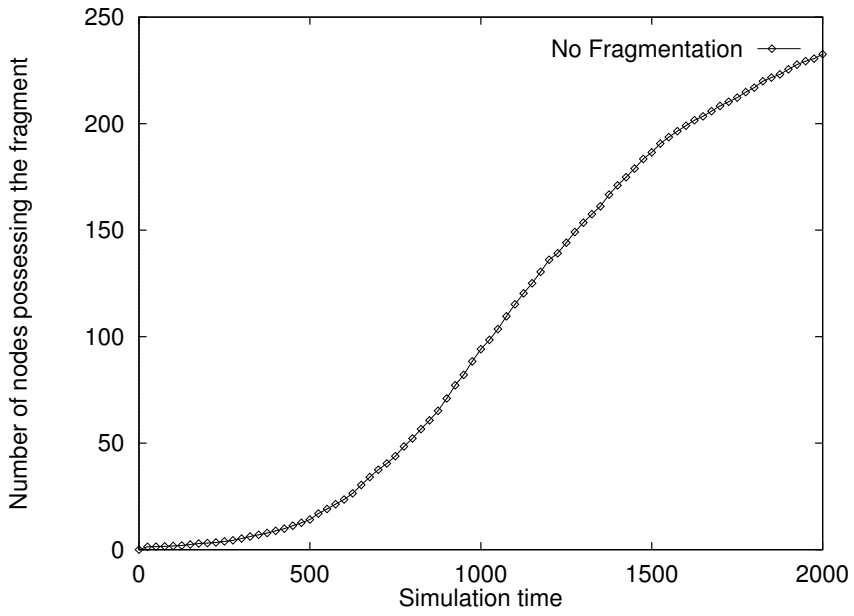

Figure 35: Propagation of a message of size 16 when it is not fragmented.

The gain in delays is prominent as the message sizes are large. The relative order of these end-to-end delays under different variations of fragmentation follows the propagation of the fragments in the network as analyzed in the next section.

5.4.2.2 Fragment Propagation

Figures 35 through 41 depict the propagation of a message of size 16 units through a network of 256 nodes. The x-axis denotes the progress of the simulation, whereas the y-axis indicates the number of nodes in the network that contain the particular fragment (or the entire message in the case of no fragmentation).

When the message is not fragmented, the message propagates very slowly as shown in Figure 35. It takes more than 2000 simulation time units to reach all the nodes. This slow propagation is primarily because nodes have to wait for long durations before forwarding the large message in a suitable contact opportunity. Since the propagation is very slow, messages may expire before they reach their destinations, thus leading to a low success rate of delivery.

In all the other cases of fragmentation, the fragments propagate faster through the network. For example, fragment 0 in Figure 36 takes only about 300 time units to reach

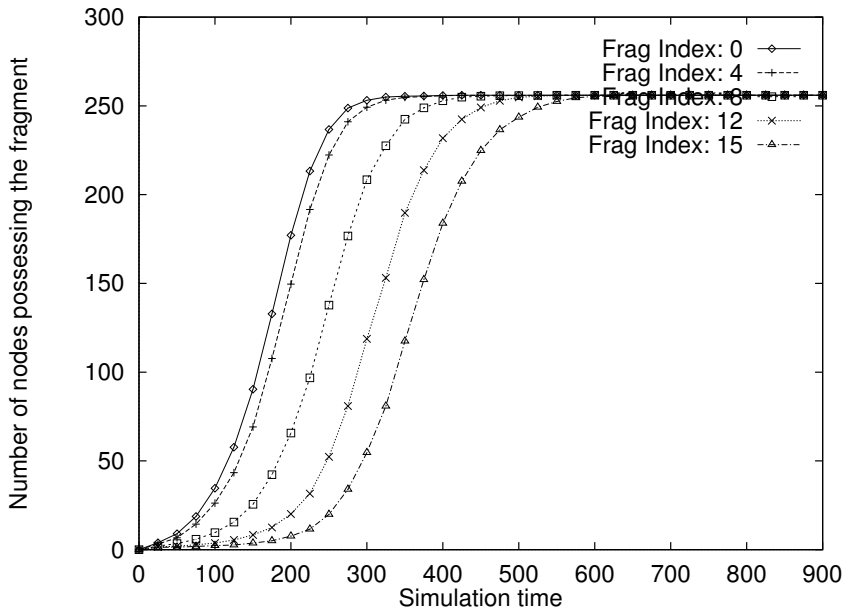

Figure 36: Propagation of fragments of a message of size 16 units when the fragments are forwarded in a sequential order that always starts from the beginning.

every node in the network. The fast propagation of fragments is due to their small size and possibility of transmission in almost every contact opportunity.

However, a key issue with fragmentation is that the destination node must collect multiple number of fragments to make up the original message. And the propagation of these fragments can *interfere* with one another. For example, in the sequential order transmission schedules that always begin with the first fragment as shown in Figures 36 and 37, the higher rank fragments (with lower indices) dominate the fragment propagation and reach all the nodes faster. The lower rank fragments can penetrate the network only after all nodes have the higher rank fragments and do not need to exchange them any more. This starvation of lower rank fragments by the higher rank fragments leads to an increase in the end-to-end delay because the destination node has to receive all the fragments, including the lower rank ones to complete the message.

In the "continue-and-wrap-around" scheme, however, the fragment transmission resumes from the last fragment that was forwarded in the previous contact opportunity. Figures 38 and 39 show the plots for the propagation of fragments in this "continue-and-wrap-around" scheme. In this scheme, due to the preference for the fragments that were not forwarded

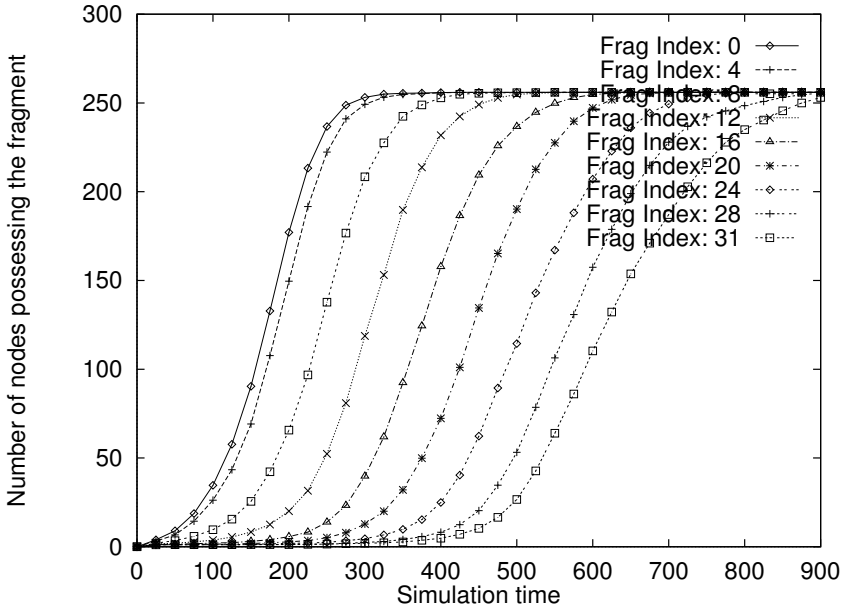

Figure 37: Propagation of fragments of a message of size 16 units that is erasure coded to $(16 + 16)$ units and the fragments are forwarded in a sequential order that always starts from the beginning.

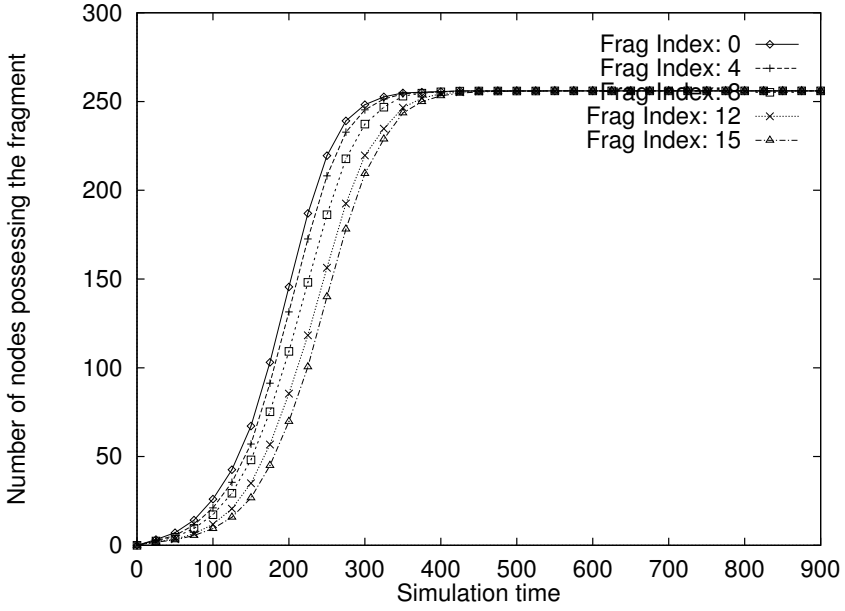

Figure 38: Propagation of fragments of a message of size 16 units when the fragment transmission schedule at a contact opportunity continues from the last fragment forwarded in the previous contact opportunity and wraps-around the fragment order.

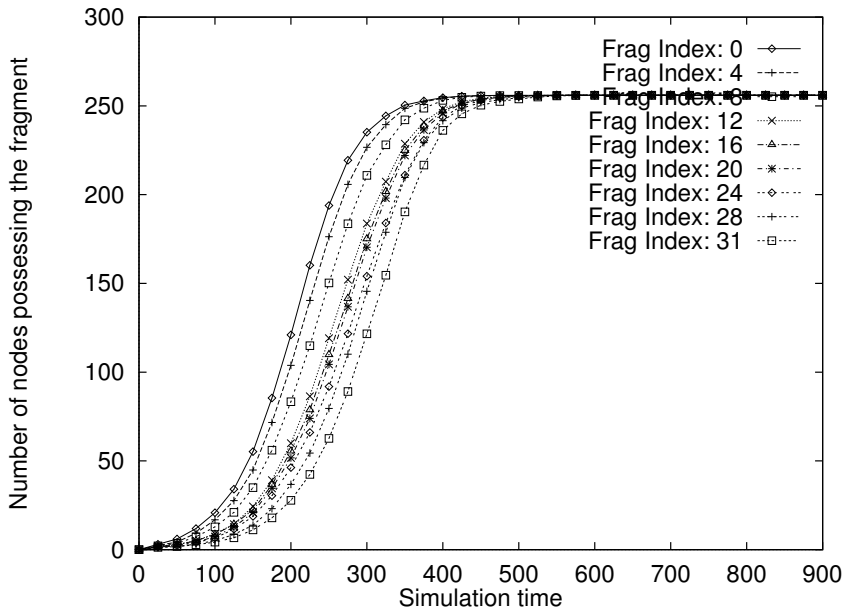

Figure 39: Propagation of fragments of a message of size 16 units that is erasure coded to $(16 + 16)$ units and the fragment transmission schedule at a contact opportunity continues from the last fragment forwarded in the previous contact opportunity and wraps-around the fragment order.

earlier, the later fragments not starved as much as in the "start-from-beginning" scheme. Therefore the time difference between the penetration of the first and last fragments is lower. Consequently, the destination node receives all the fragments earlier.

An alternative schedule of fragment transmission that considers a random permutation of the fragments mitigates this interference between fragments. Since nodes pick fragments at random for transmission at any contact opportunity, all the fragments are equally likely and propagate in almost a similar way as shown in Figures 40 and 41.

Comparing Figures 36 and 40 reveals that although both schemes have the same number of fragments to begin with at the source, the random transmission scheduling scheme propagates all the fragments faster, within about 400 time units, than the sequential transmission scheduling scheme that took about 550 time units to propagate all the fragments to all the nodes. This gain is primarily due to the diversity in the message composition at the nodes. Recall that in a replicate-and-forward routing protocol, nodes exchange only those

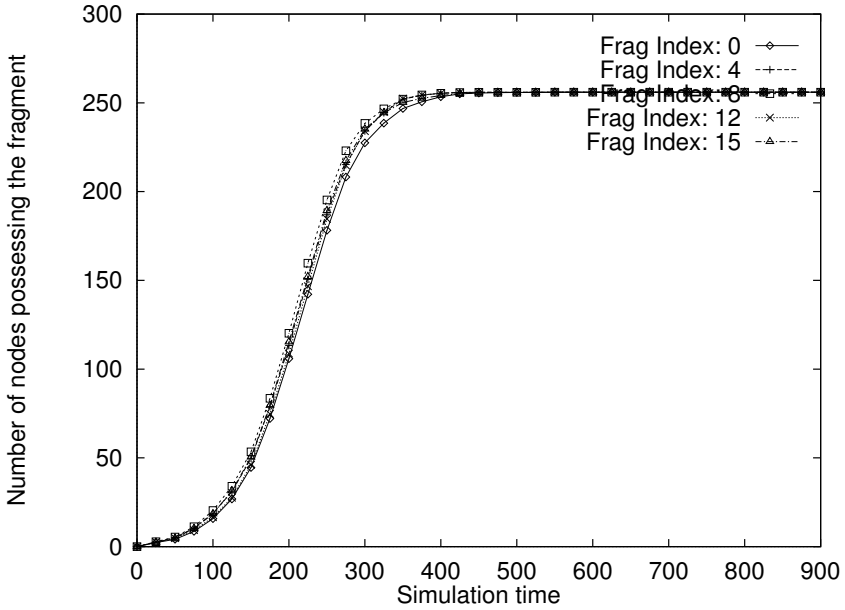

Figure 40: Propagation of fragments of a message of size 16 units when the fragments are forwarded in a random order.

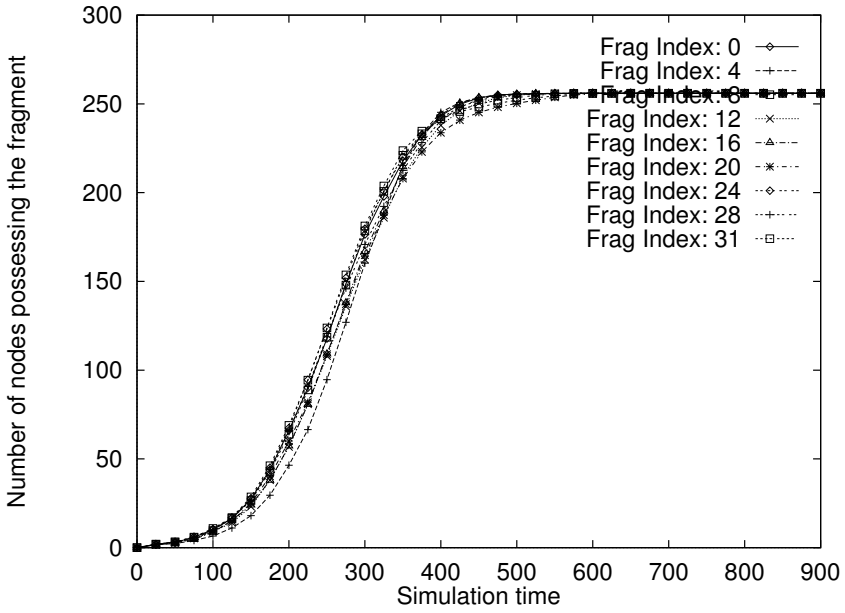

Figure 41: Propagation of fragments of a message of size 16 units that is erasure coded to $(16 + 16)$ units and the fragments are forwarded in a random order.

messages that the other node doesn't have. In the case of sequential transmission schedule, because only few fragments penetrate the network at a time, most nodes have almost similar fragments and therefore do not necessarily have any fragments to exchange when they meet. On the other hand, the randomness in fragment transmission schedule increases the chances that two nodes have different set of fragments when they meet and therefore can exchange those fragments in the difference. These exchanges further improve the speed of fragment propagation. The diversity in message composition can also be achieved by a deterministic scheme that continues and wraps around instead of always transmitting from the first fragment at every contact opportunity. Therefore we see similar gains in 38 as well.

Figure 37 indicates that although erasure coding of the message at the source has introduced more fragments to increase their number to 32, their sequential order of transmission scheduling limits their transmission and the first 16 fragments follow exactly the same propagation pattern as the case without erasure coding in Figure 36. In the case of random transmission scheduling, however, the increase in number of fragments with erasure coding in Figure 41 slightly slows down the rate of propagation as compared to the previous case without erasure coding in Figure 40. However, since the destination node needs to collect any 16 fragments to make up the message, the overall end-to-end delay need not be longer. In fact, the end-to-end delay is lower as confirmed by the fragment arrivals at the destination node examined in the next section.

5.4.2.3 Fragment Arrivals at Destination

Figure 42 denotes the arrivals of fragments of a message at its destination node. The x-axis is the fragment index in its arrival order. Note that the arrival order need not be in the original sequence of fragments as different fragments take different paths to reach the destination. The y-axis denotes the arrival time in the simulation.

As described earlier in the Section 5.4.2.1, the fragment arrivals of both the cases of sequential transmission schedule overlap with each other due to the similar fragment propagation patterns. Whereas, the fragments arrive earlier at the destination node when we use random transmission scheduling in our replicate-and-forward routing protocol.

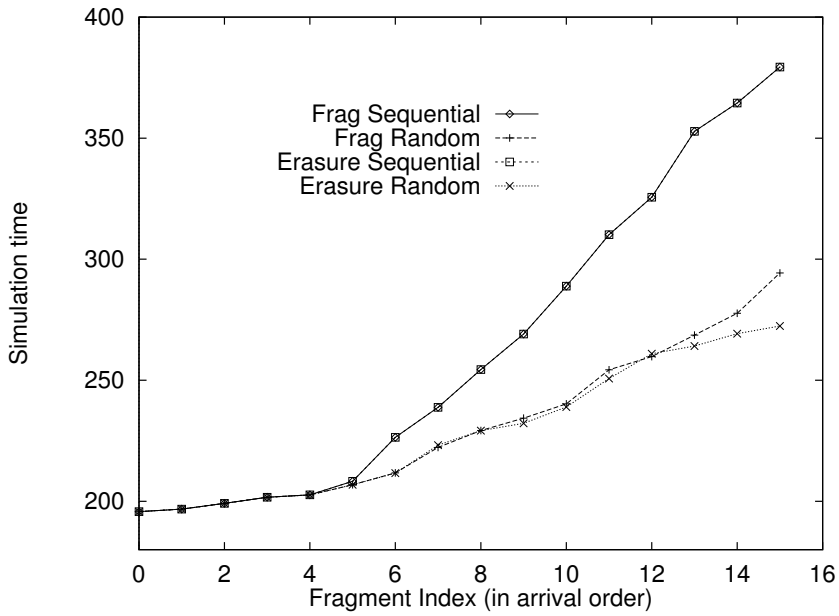

Figure 42: Arrivals of fragments of a message of size 16 at the destination node under various schemes.

Among the two curves for the random scheduling, we observe that the arrivals of fragments are increasingly delayed as we approach the last fragment. This is a property that we referred to earlier in Section 5.3.3 as the "last-fragment-delay". Note that the delays for the erasure-coded scheme are lower primarily because of the availability of a larger number of fragments in the network. Also, because the destination node needs any one of the remaining fragments to count as its last fragment, it has a better chance of finding a fragment earlier. Consequently, the end-to-end delays are the lowest for the case with erasure coding and random scheduling.

5.4.2.4 Delays with Transmission Losses

In this section, we consider the practical case of forwarding fragments with losses in transmission that are often prevalent in several sparsely-connected wireless networks. Figure 43 is a plot of variation of the end-to-end delay under different schemes as the probability of transmission loss is varied. On the x-axis are different values of fragment transmission loss probability. The y-axis is the end-to-end delay. The figure represents only the case with fragmentation as we are interested in observing the effects of transmission scheduling and

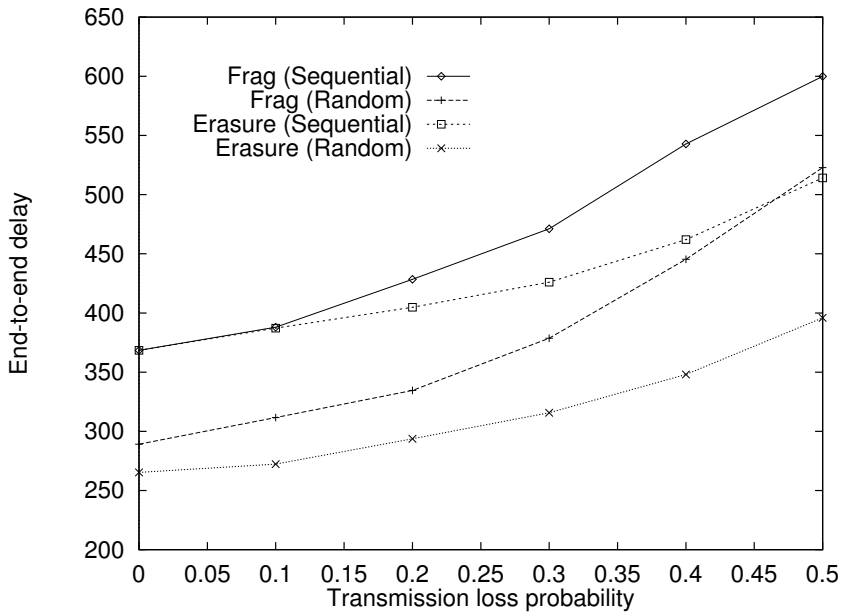

Figure 43: End-to-end message delivery delays with varying transmission loss probabilities.

erasure coding.

As described earlier, when the loss probability is zero, both schemes under sequential transmission scheduling, one with erasure coding and other without, overlap. However, as the losses increase, the scheme with erasure coding gives lower delays due to the error correcting property of erasure codes that can decode the original message even with certain number of losses. The advantage of erasure codes to provide reliability under losses is preserved even under the two cases of random transmission schedules.

5.4.2.5 Multiple Messages

The experiments described so far looked at the propagation of a single message to understand the effects of fragmentation and their interference. However, in this experiment with results shown in Figure 44, we consider the simultaneous propagation of multiple messages originating from different sources and traveling towards different destinations. This experiment is to study the effect of interaction between fragments of different messages at an intermediate node and how it translates into the corresponding end-to-end delivery delays of each of the messages. Compared to the Figure 34, the end-to-end delivery delays of

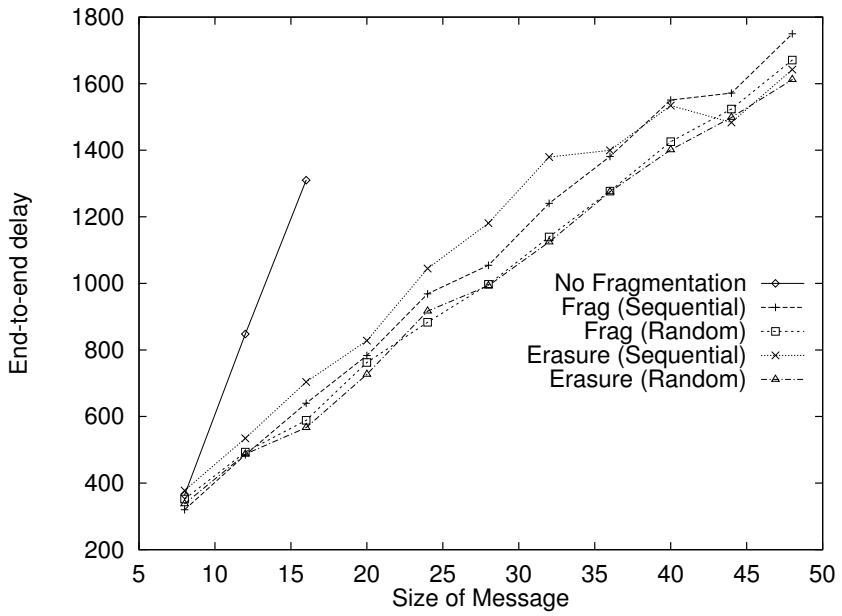

Figure 44: End-to-end message delivery delays when multiple messages are simultaneously forwarded in the network.

messages are longer due to the interaction between multiple messages. The advantages of fragment transmission scheduling that we saw earlier apply in this case of multiple messages as well. But curves for erasure coding incur a penalty as the intermediate nodes do not distinguish between fragments. In the erasure coding scheme that introduced higher number of fragments into the network, the intermediate nodes may forward more fragments than necessary to their neighbors, thus consuming the communication bandwidth that could otherwise be used to transmit additional messages. The penalty could be avoided by an intelligent fragment transmission scheme that recognizes that a message is erasure-coded and does not forward excess fragments to neighbors.

5.5 Summary

In this chapter, we have considered the problem of routing large messages in an ad hoc network that is both unpredictable and sparsely-connected. The replicate and forward protocol that was originally proposed to route messages when a network is unpredictable is perhaps the best choice for routing in these networks too. However, the sparse-connectivity of these networks limits a direct applicability of the replicate and forward routing protocol.

Therefore, within the framework of the routing protocol, we consider two techniques to route large messages while addressing the sparse-connectivity: (*i*) fragmentation and (*ii*) erasure coding.

In a sparsely-connected network with rare and short contact opportunities for transmission, our simulation results indicate that fragmenting large messages improves the chances of delivering the message with shorter end-to-end delays than without fragmentation. We also note that a random schedule of fragment transmissions at intermediate nodes mitigates the interference between fragment propagation and achieves better delivery rates as well as lower end-to-end delays. Furthermore, enhancing the fragmentation approaches with an erasure coding of the large message at the source provides additional benefits. For example, the larger number of fragments in the erasure coding scheme provides a diversity of fragments in the network that improves the rate of propagation of fragments as well as provide greater reliability against transmission losses.

CHAPTER VI

CONTRIBUTIONS OF THE THESIS

Several emerging networks such as peer-to-peer networks and mobile ad hoc networks have special topological characteristics (e.g., dynamic, sparsely-connected) that present challenges to routing. In this thesis, we have examined the interplay between routing and network topology design in these special networks. The contributions of the thesis are in these broad areas:

- **Overlay topology design** We presented a class of overlay topologies for unstructured peer-to-peer networks and their characteristic parameter called proximity factor. Proximity factor relates to the underlying network distance between a node and its neighbors. Different instances of this class of overlay topologies are realized by varying the proximity factor. These topology instances are examined, by simulation, along two performance-related dimensions: *(a)* search protocol performance and *(b)* utilization of links in the underlying network. Our simulation results showed that in a particular "small-world-like" topology instance of this class where every node has many close neighbors and few random neighbors, *(1)* the chances of locating files are high and *(2)* the nodes where these files are found are, on average, close to the query source. This improvement in search protocol performance is achieved while *(3)* decreasing the traffic load on the links in the underlying network.

- **Routing in space and time** We introduced routing in both space and time dimensions for sparsely-connected ad hoc networks where the topology, due to its disconnectivity, is not amenable to traditional mobile ad hoc routing protocols. Specifically, we took advantage of predictability in node motion available in some of these networks to construct space-time routing tables over a time horizon. These space-time routing tables consider time in addition to destination to determine the next hop node for

forwarding. We derived a space-time graph model that gives a symbolic representation of paths taken by messages as they are routed in space and time. Using these space-time graphs, we presented an adaptation of the Floyd-Warshall algorithm to solve the minimum delay routing problem on these dynamic network topologies. Our evaluation results support our observations that least delay paths in space and time using minimal resources can be identified with very high probability.

- **Scheduling transmissions unpredictable and sparsely-connected ad hoc networks** Within the framework of a replicate and forward routing protocol that is amenable to unpredictable networks, we have presented techniques to forward large messages while addressing sparse connectivity. Specifically, when messages are fragmented to fit the short contact opportunities, we have explored (*i*) different heuristics for transmission schedules and (*ii*) encoding of the message at the source using erasure codes. Our evaluation, by simulation, confirms the advantages of fragmenting the messages to improve their chances of delivery as well as lower their end-to-end delays. Also, our results support that the order of transmission of fragments at intermediate nodes affects their propagation. Particularly, we show that a random schedule of fragment transmissions has lower end-to-end delays than a sequential scheduling scheme. Furthermore, the fragmentation scheme enhanced with erasure coding at the source increases the number of fragments and therefore provides greater reliability against losses.

APPENDIX A

P-SIM: A SIMULATOR FOR PEER-TO-PEER NETWORKS

A.1 Introduction

Many initial measurement studies on deployed peer-to-peer networks attempted to understand the performance of peer-to-peer networks [59, 62, 63]. However, the dynamics in peer lifetimes and complexity of these networks make it difficult to obtain a precise comprehensive snapshot. Simulation of peer-to-peer networks provides a methodical evaluation and analysis of their performance.

Current network simulators such as *ns* [49], however, are not well suited for the task of simulating peer-to-peer networks. In peer-to-peer networks, we are interested in studying the macroscopic properties of the network as events happen, rather than the fine-grain packet-level dynamics. The events of interest in peer-to-peer network studies occur at medium time-scale and often include node arrivals/departures, search queries for locating files, and robustness/recovery from failures. Most traditional network studies, on the other hand, concern issues at the network and transport layer, for example, routing protocols, queuing disciplines, congestion avoidance and control, and throughput improvements. Another property of peer-to-peer networks that distinguishes them from regular network studies is the variability in topology. The overlay graph formed by the peers is constantly changing and this behavior cannot be easily modeled by current network simulators that assume a static topology. A further characteristic of peer-to-peer networks is their large size (on the order of tens of thousands of nodes [38]) and packet-level simulators such as *ns* are known to have concerns about scalability [58]. Thus these differences present a new set of criteria for simulating peer-to-peer networks.

In this appendix, we present *p-sim*: a tool that can simulate peer-to-peer networks on

top of representative Internet topologies. *p-sim* allows comparative study of various peer-to-peer network protocols under common network conditions. *p-sim* attempts to provide an accurate representation of real-world peer-to-peer networks. The following is a list of capabilities included in *p-sim*:

- **Peer Dynamics:** The population of nodes in a peer-to-peer network is extremely dynamic with nodes joining and leaving the network. The time a particular node is connected to the network also varies widely. *p-sim* can simulate these variations in node lifetimes and arrival and departure patterns.

- **File Sharing and Searching:** Most peer-to-peer networks are used for content location and employ several different mechanisms for performing searches. *p-sim* includes implementations of multiple search protocols.

- **Evaluation:** Metrics on the peer-to-peer network are crucial to study the performance of its protocols and *p-sim* has mechanisms to evaluate metrics on both the overlay network as well as the underlying network.

- **Scalability:** Recent studies on work load of peer-to-peer networks [62, 63] attest to their immense popularity. *p-sim* has performance enhancements to handle the dynamics of thousands of nodes.

- **Extensibility:** *p-sim* is designed to be modular with clean interfaces so that it can be easily extended to implement new peer-to-peer protocols and search mechanisms.

The remainder of the appendix is organized as follows. In Section A.2, we give an overview of *p-sim* and how it works. The User Interface to the simulator is described in Section A.3. We discuss enhancements to *p-sim* to increase its performance in Section A.4 followed by a summary in A.5.

A.2 Overview

p-sim is designed to be generic and captures most of the key components of a peer-to-peer network. In this section, we highlight the sequence of operations in a typical simulation

and describe the different components of *p-sim* that fit together to model a peer-to-peer network.

A.2.1 Simulation Framework

p-sim is based on the popular event-driven simulation technique [5]. In this approach, each event is associated with a *time-stamp*. Events have a description of an *action* and an *identifier* of the corresponding peer where the action is to be performed. Some of the events that are used in our simulator are:

- **Peer Arrival/Departure:** The arrival and departure events denote when a peer has become active (on-line) or inactive (off-line). These events are associated with individual peers and trigger re-organization of the overlay topology.
- **Search Query:** The file search query events are initiated at a peer and try to locate the specified file following the search protocol.
- **Evaluation:** The evaluation events serve two purposes: to monitor the simulation progress as well as to measure properties of the peer-to-peer network.

The list of events is ordered according to increasing time-stamps. We implement a priority min-heap to store the event list. The event list is initialized with certain known events and their associated times. The simulation engine takes the top event (with least time-stamp) and executes the action associated with the event. The virtual clock of the simulation "jumps" to the time-stamp of this event. This execution might add more future events to the event list. The simulation engine repeats this process until the event list is exhausted. The following subsections describe the events corresponding to different components in a peer-to-peer network.

A.2.2 Typical Peer-to-Peer Network Simulation

The specific details of simulation of a peer-to-peer network depend on the input configuration parameters, but we can broadly think of a simulation to proceed along the following sequence of operations:

1. Construction of the underlying physical topology at the router-level using an Internet Topology model (e.g., GT-ITM [80]). This operation also involves attaching end-hosts to the leaf routers to realize a model of the Internet and the computation of a routing table (both distance and next hop information) between every pair of end-hosts on this underlying topology[1].

2. Identification of hosts on the underlying topology that would participate in the peer-to-peer network and form an overlay topology. Initialization of the data structures (e.g., neighbor information, file storage) required for each peer.

3. Distribution of the files among all the peers in the system according the input parameters for file replication. This step includes distribution of the origins of queries for these files as per the query placement parameters.

4. Estimation of the time of arrivals and departures of peers based on the input configuration parameters. We explain in Section A.2.4 how the dynamics of peers are realized by arrival and departure events.

5. Initialization of the list of events with arrival/departure events, query search events, metric evaluation events, etc. and invocation of the simulation engine that executes these events in the increasing order of their time-stamps.

6. Analysis of output log of the simulation. This output log records information about execution of events and also includes the values of metrics as determined by evaluation events.

A.2.3 Topology

A representation of the Internet topology provides a structural framework for simulating networks. Several models of the Internet topology and their synthetic topology generators exist (e.g., [46, 79, 80]). *p-sim* uses the Transit-Stub graph model[2] of the GT-ITM topology

[1] We assume shortest-path routing, but it could be extended to other routing algorithms (e.g., policy-based) as well.

[2] We have extended our simulator to work with Power-law random graph (PLRG) model [46, 79] as well. However these extensions are preliminary and need further support to scale to large topologies.

generator [80]. The Transit-Stub graph model is a representation of the hierarchical router-level topology of the Internet. We extend the router-level topology generated by the Transit-Stub model by "attaching" end-hosts to the leaf routers. The extended topology, thus serves as an underlying physical network. The number of hosts attached per leaf router is determined by a random number distribution as specified by user. Latency values can be assigned to the links of the extended topology according to the type of the link. There are four types of links depending on the endpoints of a link: transit-transit, transit-stub, stub-stub and stub-host. The latency values for each link type can be sampled from a random number distribution, also an input configuration parameter. Of all the end-hosts in the underlying physical network, a fraction of them are selected to participate in the peer-to-peer network. The number of peers in the peer-to-peer network and the choice of their arrangement on the underlying physical network are input configuration parameters.

A.2.4 Peer Dynamics

The population of peers is dynamic with random arrivals and departures. The activity of each peer is modeled as a two state Markov chain. The active state refers to the time when a peer is participating in the peer-to-peer network, while the inactive state refers to the time when a peer is "off-line". A peer arrival event corresponds to a transition from the inactive state to the active state. Similarly, a peer departure event corresponds to a transition from the active state to the inactive state. A *session* comprises an arrival event and its corresponding departure event. The number of sessions per peer during the entire simulation is an input configuration parameter. Users can also specify a random number distribution to sample the lengths of both the active and inactive periods. The peer arrival and departure events are pre-computed and added to the event list during initialization. Any operations by a peer are performed only when it is "active".

A.2.5 File Search Protocol

p-sim implements two file searching protocols: *(1)* a *scoped-flooding* search protocol that is used by Gnutella and its family of protocols [28] and *(2)* a *random-walk* search protocol proposed recently for unstructured peer-to-peer networks [15]. Users can specify either

search protocol for simulation. The input configuration parameters to simulator include the number of files in the system and a distribution that models the file replication (e.g. Zipf). These files are distributed among all the peers during initialization. We also create queries for these files and "spread" their origins over the peer-to-peer network according to user-specified input parameters. The search query process is simulated according to the search protocol by examining the file storage at each of the peers within search radius from query source.

A.2.6 Evaluation Metrics

We define some metrics below that are used to evaluate various aspects of a peer-to-peer network. These definitions are translated into actions that produce output log of the state of the network. The evaluation events are scheduled periodically at a frequency specified by the user. The following metrics are implemented in *p-sim*:

1. **File Success Rate:** A query for a file is *successful* when it finds at least one match for the requested file. *Success rate* of a file is defined as the ratio of number of successful queries to the total number of queries issued for that file throughout the system.

2. **Nearest Search Result:** Of the multiple file locations returned by a search protocol, the *nearest* one in the underlying network may be able to better serve the file than other locations. This distance to the nearest search result is an important metric when evaluating performance of a search protocol.

3. **Connectivity:** Since the peer-to-peer network topology is a variable graph, we check whether it is connected or not at every measurement instance. As part of this test for connectivity, we compute the diameter of the overlay topology among active peers.

4. **Stress:** *Stress* [16] of a link in underlying topology is defined as the number of logical links (between peers) whose mapped paths include the underlying link. Intuitively, stress measures global effect of the overlay topology of the peer-to-peer network on the underlying network.

A.3 Simulator User Interface

We designed a simple user interface language to write the input configuration parameters to *p-sim*.

Figure 45 shows an example of the input configuration written in the user interface language to *p-sim*.

The input configuration has five parts: peers, files, search, topology and evaluation. Within each part, the parameters are specified as: (**keyword** `value`)[3]. Depending on the context, the `value` could either be a constant (e.g., (**count** 5000)) or description of a random number distribution to sample values from (e.g., (**activity** `EXPONENTIAL` 100) to extract samples from an exponential distribution with a mean of 100).

The **peers** part defines the characteristics of individual peers in the simulation. A total of **count** peers are created for the simulation. Currently, the number of **neighbors** maintained by a peer is a `RANGE` with specified minimum and maximum, but could easily be extended to define classes of peers each with a different range of neighbors. Similarly, the storage of **files** at a peer could be a `CONSTANT` or variable to incorporate heterogeneity among peers. The dynamics of peer population are controlled by the **sessions** parameters. Each peer is scheduled to have **numSessions** where the periods of **activity** and **inActivity** are sampled from the specified random number distributions.

The **files** and **search** parts configure the file storage and discovery of these files by search queries. A total of **numFiles** are distributed among all peers of the network. We have a provision to allow multiple copies of these files according to a random number distribution specified by **replication** parameter. This "spread" of files is based on a random number distribution specified according to **placement**. A `UNIFORM` placement picks locations at random uniformly among all peers to assign files. We could easily extend the file storage scheme to include other random number distributions to reflect "locality" in storage of files. The search queries for files are propagated according to the **protocol** up to a maximum of **radius** hops on the overlay topology. For example, (**protocol** SCOPED_FLOOD) and

[3]Pre-defined keywords of our user interface language are shown in boldface in this section. Similarly, pre-defined values are in typewriter typeface.

(**radius** 7) uses the scoped-flooding search protocol with a search radius of 7 hops. We can specify whether we want to simulate queries for `ALL` files or only a fraction of the files in the system. Duplicate queries can be generated according to the random number distribution specified by **duplicateQueries** parameter. The source of these queries can also be decided by sampling a random number distribution given by **queryOriginPlacement**.

The **topology** part describes the configuration of the underlying physical network topology. The topology is constructed by the transit-stub model of the **GT-ITM** topology generator using the configuration found at **location**. The router-level topology thus constructed is extended by adding hosts to the leaf routers. The numbers of hosts added to these leaf routers are sampled from **hosts** random number distribution. Of these end-hosts on the extended topology, we pick **count** number of hosts as participants in the peer-to-peer network. The arrangement of these participant hosts is dictated by the random number distribution given by **peerArrangement**. For example, (**peerArrangement** UNIFORM) picks end-hosts uniformly at random to participate in the peer-to-peer network. Latencies are also assigned to links in the underlying network according to random number distributions in the **linkLatency** parameters.

The final part of the input configuration has **evaluation** metrics. These parameters describe the evaluation metric and the frequency at which is it evaluated. For example, (**stress** 100) refers to a measurement of stress on the links in the underlying network every 100 ticks of simulation. This frequency of measurement is used in scheduling the evaluate events that are executed periodically.

A.4 Performance Improvements

In this section, we describe some of our efforts to decrease the run-time of execution of the simulator, especially when scaled to a large number of peers.

A.4.1 Hierarchical Routing in GT-ITM

We have implemented a piecewise route computation that greatly improves the efficiency of all-to-all route computation in GT-ITM topologies. The scheme computes routes independently within each stub domain and in a graph consisting of the border router for each

stub and all the transit nodes. A complete end-to-end route is then constructed by piecing together routes in each stub domain with the route from origin domain border router to destination domain border router. The efficiency gains are considerable; for example, an all-pairs computation which took hours to run now takes tens of minutes.

A.4.2 Delta Metric Evaluation

The evaluate events that monitor the state of the peer-to-peer network are scheduled periodically. Computing these evaluations, such as stress on the links of underlying network, as defined in Section A.2.6 requires significant amount of time. However, we can reduce this computation time by observing that the number of changes that take place since the last evaluation may be far less compared to the total size of the system. Specifically, the number of logical links that are added or deleted since the last call to stress computation event may be less than the total number of links. Under these circumstances, it is more efficient to remember the data structures from the last invocation and apply the changes that happen in most recent period to obtain a new value of the metric. In addition to stress, this "delta metric evaluation" technique is applicable for other metrics as well. The savings in computation time using this technique are significant. On a peer-to-peer network with 512 peers, periodic stress computation reduces from roughly 700 seconds to about 10 seconds.

A.5 Summary

We have presented *p-sim*, a tool that can simulate the behavior of a large-scale peer-to-peer network. Some of the capabilities of *p-sim* include peer dynamics and file sharing/searching. *p-sim* includes several metrics for evaluating performance of peer-to-peer network protocols. It has a modular design to support the inclusion of future peer-to-peer network protocols as well.

```
(configuration "sample" ; This script is a sample input configuration
    (peers
        (count 5000)
        (neighbors RANGE 3  5)
        (files CONSTANT 8)
        (sessions
            (count 4)
            (activity EXPONENTIAL 100)
            (inActivity EXPONENTIAL 500)
        )
    )
    (files
        ; 100 files whose popularity follows Zipf distribution are placed uniformly at random
        (count 100)
        (replication ZIPF 2048)
        (placement UNIFORM)
    )
    (search
        ; use scoped-flooding protocol to discover files
        (protocol SCOPED_FLOOD)
        (radius 7)
        (query ALL)
        (duplicateQueries ZIPF 1024)
        (queryOriginPlacement UNIFORM)
    )
    (topology
        ; use the GT-ITM model for underlying physical network
        (GT-ITM
            (location "/path/to/topology/config/file")
            (hosts NORMAL 14.0 4.0)
            (peerArrangement UNIFORM)
            (linkLatency ; assign latency to links
                ; the following use the Uniform distribution
                (t-t UNIFORM 15.0 25.0)
                (t-s UNIFORM 3.0 7.0)
                (s-s UNIFORM 1.0 3.0)
            )
        )
    )
    (evaluation
        ; evaluate stress on links every 100ms of simulation
        (stress 100)
        (fileSuccessRate 1000)
    )
)
```

Figure 45: An example of input configuration to a peer-to-peer network simulation, interleaved with comments to express meanings of parameters. Keywords are shown in bold face; comments are italicized and guarded by semi-colons.

REFERENCES

[1] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E., "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] Albert, R., Jeong, H., and Barabási, A. L., "The Diameter of the World-Wide Web," *Nature*, 1999.

[3] "Bluetooth." http://www.bluetooth.com.

[4] Banerjee, S., Komareddy, C., and Bhattacharjee, S., "Scalable Peer Finding on the Internet," in *Global Internet Symposium, Globecom*, 2002.

[5] Banks, J., Carson, J. S., and Nelson, B. L., *Discrete Event System Simulation.* Prentice-Hall, Englewood Cliffs, NJ, 1996.

[6] Barabási, A. L. and Albert, R., "Emergence of Scaling in Random Networks," *Science*, vol. 286, pp. 509–512, Oct. 1999.

[7] Barabási, A. L., Albert, R., and Jeong, H., "Scale-free characteristics of Random Networks: the Topology of the World-Wide Web," *Physica*, vol. A 281, pp. 69–77, 2000.

[8] Beaufour, A., Leopold, M., and Bonnet, P., "Smart-Tag Based Data Dissemination," in *First ACM workshop on Wireless Sensor Network and Applications*, October 2002.

[9] Blahut, R. E., *Theory and Practice of Error Control Codes.* Addison Wesley, MA, 1984.

[10] Bollobás, B., ed., *Random Graphs.* Cambridge University Press, 2nd ed., 2001.

[11] Burleigh, S., Cerf, V., Durst, R., Fall, K., Hooke, A., Scott, K., and Weiss, H., "The Interplanetary Internet: A Communications Infrastructure for Mars Exploration," in 53^{rd} *International Astronautical Congress, The World Space Congress*, 2002.

[12] Byers, J. W., Luby, M., Mitzenmacher, M., and Rege, A., "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *ACM Sigcomm*, 1998.

[13] "Crawler project: http://crawler.limewire.org."

[14] Camp, T., Boleng, J., and Davies, V., "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.

[15] Chawathe, Y., Ratnaswamy, S., Breslau, L., Lanham, N., and Shenker, S., "Making Gnutella-like P2P Systems Scalable," in *ACM SIGCOMM*, 2003.

[16] Chu, Y., Rao, S. G., Seshan, S., and Zhang, H., "A Case for End System Multicast," *IEEE Journal on Selected Areas in Communication, Special Issue on Networking Support for Multicast*, 2002.

[17] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W., "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *Lecture Notes in Computer Science*, 2001.

[18] Cormen, T. H., Leiserson, C. E., and Rivest, R. L., *Introduction to Algorithms*. MIT Press, Cambridge, MA, U.S.A., 1990.

[19] Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[20] Dubois-Ferriere, H., Grossglauser, M., and Vetterli, M., "Age Matters: Efficient Route Discovery for Mobile Ad Hoc Networks UsingEncounter Ages," in *Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2003.

[21] Dubois-Ferriere, H., Grossglauser, M., and Vetterli, M., "Space-Time Routing in Ad Hoc Networks," in *Second Intl. Conference on Ad Hoc Networks and Wireless*, 2003.

[22] Erdős, P. and Rényi, A., "On Random Graphs," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[23] "http://en.wikipedia.org/wiki/fasttrack."

[24] Fall, K., "A Delay-Tolerant Architecture for Challenged Internets," in *ACM Sigcomm*, 2003.

[25] Faloutsos, M., Faloutsos, P., and Faloutsos, C., "On Power-Law Relationships of the Internet Topology," in *ACM Sigcomm*, pp. 251–262, 1999.

[26] Floyd, R. W., "Algorithm 97 (SHORTEST PATH)," *Communications of the ACM*, vol. 5, no. 6, 1962.

[27] "http://gnutella.wego.com."

[28] "http://rfc-gnutella.sourceforge.net."

[29] Gilbert, E. N., "Random Graphs," *Annals of Mathematical Statistics*, vol. 30, pp. 1141–1144, 1959.

[30] "http://www.gnucleus.com."

[31] "http://gtk-gnutella.sourceforge.net."

[32] Jain, S., Fall, K., and Patra, R., "Routing in a Delay-Tolerant Network," in *ACM Sigcomm*, 2004.

[33] Jiang, S. M., He, D. J., and Rao, J. Q., "A Prediction-Based Link Availability Estimation for Mobile Ad Hoc Networks," in *IEEE Infocom*, Apr. 2001.

[34] Jovanović, M. A., Annexstein, F. S., and Berman, K. A., "Modeling peer-to-peer network topologies through "small-world" models and power laws," *IX Telecommunications Forum, TELFOR*, 2001.

[35] Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L. S., and Rubenstein, D., "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet," in *The Tenth International Conference on Architectural Support for Programming Languages and Operating Sytems (ASPLOS-X 2002)*, October 2002.

[36] Kleinberg, J., "Navigation in a small world," in *Nature*, 2000.

[37] Kleinberg, J., "The Small-World Phenomenon:An Algorithmic Perspective," in *ACM Symposium on Theory of Computing*, 2000.

[38] "Gnutella network hosts: http://www.limewire.com/current_size.html."

[39] "http://www.limewire.com."

[40] Li, L., Halpern, J., and Haas, Z., "Gossip-based Ad Hoc Routing," in *IEEE Infocom*, 2002.

[41] Lin, S. and Costello, D. J., *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983.

[42] Lindgren, A., Doria, A., and Schelen, O., "Poster: Probabilistic Routing in Intermittently Connected Networks," in *Fourth ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[43] McDonald, A. B. and Znati, T., "Predicting Node Proximity in Ad Hoc Networks: A Least Overhead Adaptive Model for Selecting Stable Routes," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2000.

[44] McDonald, A. B. and Znati, T. F., "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks," *IEEE JSAC*, vol. 17, no. 8, pp. 1466–1487, 1999.

[45] McDonald, A. B. and Znati, T. F., "A Path Availability Model for Wireless Ad Hoc Networks," in *IEEE WCNC*, pp. 35–40, Sep. 1999.

[46] Medina, A., Lakhina, A., Matta, I., and Byers, J., "BRITE: An Approach to Universal Topology Generation," in *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS*, 2001.

[47] Merugu, S., Srinivasan, S., and Zegura, E., "p-sim: A Simulator for Peer-to-Peer Networks," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS)*, 2003.

[48] "http://www.napster.com."

[49] "The Network Simulator (ns)." http://www.isi.edu/nsnam/ns/.

[50] Oram, A., ed., *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[51] Pentland, A., Fletcher, R., and Hasson, A. A., "A Road to Universal Broadband Connectivity," in *Second International Conference on Open Collaborative Design for Sustainable Innovation; Development by Design (dyd02)*, December 2002.

[52] Perkins, C., *Ad Hoc Networking*. Addison Wesley, 2001.

[53] Pless, V., *Introduction to Error Correcting Codes*. Wiley, 1989.

[54] Ratnaswamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S., "A Scalable Content Addressable Network," in *ACM SIGCOMM*, 2001.

[55] Ratnaswamy, S., Handley, M., Karp, R., and Shenker, S., "Topologically-Aware Overlay Construction and Server Selection," in *Infocom*, 2002.

[56] Ratnaswamy, S., Shenker, S., and Stoica, I., "Routing Algorithms for DHTs: Some Open Questions," in *First International Workshop on Peer-to-Peer Systems*, March 2002.

[57] Reynolds, P. and Vahdat, A., "Efficient Peer-to-Peer Keyword Searching," in *ACM/IFIP/USENIX International Middleware Conference*, 2003.

[58] Riley, G. F., Fujimoto, R. M., and Ammar, M. H., "A Generic Framework for Parallelization of Network Simulations," in *International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, (MASCOTS)*, 1999.

[59] Ripeanu, M., Foster, I., and Iamnitchi, A., "Mapping the Gnutella Network: Properties of Large Scale Peer-to-Peer Systems and Implications for System Design," *IEEE Journal on Internet Computing, Special Issue on Peer-to-peer Networking*, 2002.

[60] Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols," *ACM Computer Communication Review*, vol. 27, pp. 24–36, Apr. 1997.

[61] Rowstron, A. and Druschel, P., "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[62] Saroiu, S., Gummadi, K., and Gribble, S., "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Multimedia Conferencing and Networking*, Jan 2002.

[63] Sen, S. and Wang, J., "Analyzing peer-to-peer traffic across large networks," in *Internet Measurement Workshop*, 2002.

[64] Shah, R. C., Roy, S., Jain, S., and Brunette, W., "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks," in *IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, May 2003.

[65] Shankar, N., Komareddy, C., and Bhattacharjee, S., "Finding Close Friends over the Internet," in *International Conference on Network Protocols*, 2001.

[66] Shen, C., Borkar, G., Rajagopalan, S., and Jaikaeo, C., "Interrogation-based relay routing for ad hoc satellite networks," in *IEEE Globecom*, (Taipei, Taiwan), November 17-21 2002.

[67] Small, T. and Haas, Z., "The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way)," in *The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 233–244, June 2003.

[68] Small, T. and Haas, Z., "The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way)," in *ACM MobiHoc*, Jun. 2003.

[69] Sripanidkulchai, K., Maggs, B., and Zhang, H., "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," in *Infocom*, 2003.

[70] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H., "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *ACM SIGCOMM*, 2001.

[71] Su, W., Lee, S.-J., and Gerla, M., "Mobility Prediction and Routing in Ad Hoc Wireless Networks," *International Journal of Network Management*, 2000.

[72] Su, W., Lee, S.-J., and Gerla, M., "Mobility Prediction in Wireless Networks," in *IEEE Military Communications Conference*, Oct. 2000.

[73] Tang, C., Xu, Z., and Mahalingam, M., "pSearch: Information Retrieval in Structured Overlays," in *ACM HotNets-I*, October 2002.

[74] Vahdat, A. and Becker, D., "Epidemic Routing for Partially Connected Ad Hoc Networks," Tech. Rep. CS-200006, Duke University, 2000.

[75] van Lint, J. H., *Introduction to Coding Theory*. Springer-Verlag, 1992.

[76] Wang, R., Sobti, S., Garg, N., Ziskind, E., Lai, J., and Krishnamurthy, A., "Turning the Postal System into a Generic Digital Communication Mechanism," in *ACM SIGCOMM*, 2004.

[77] Watts, D. J., ed., *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, 1999.

[78] Watts, D. J. and Strogatz, S. H., "Collective Dynamics of Small-World Networks," *Nature*, 1998.

[79] Winich, J. and Jamin, S., "Inet-3.0: Internet Topology Generator," Tech. Rep. CSE-TR-456-02, University of Michigan, 2002.

[80] Zegura, E., Calvert, K., and Donahoo, M. J., "A Quantitative Comparison of Graph-based Models for Internet Topology," *IEEE/ACM Transactions on Networking*, vol. 5, Dec 1997.

[81] Zhang, H., Goel, A., and Govindan, R., "Using the Small-World Model to Improve Freenet Performance," in *Infocom*, 2002.

[82] Zhao, B. Y., Kubiatowicz, J. D., and Joseph, A. D., "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Tech. Rep. CSD-01-1141, University of California, Berkeley, 2001.

[83] Zhao, W., Ammar, M., and Zegura, E., "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," in *ACM MobicHoc*, 2004.

[84] Zussman, G. and Segall, A., "Energy efficient routing in ad hoc disaster recovery networks," in *IEEE INFOCOM*, 2003.

VITA

Shashidhar Merugu was born in Hanamkonda, India. He graduated from the Indian Institute of Technology, Madras with a Bachelor of Technology degree in Computer Science and Engineering in 1997. He received both Master of Science and Doctor of Philosophy degrees in Computer Science from the Georgia Institute of Technology in 2000 and 2005 respectively.

Shashidhar is married to Shanthi Ganesan and enjoys spending time with their wonderful son Siddharth.