NETWORK ROUTING OPTIMIZATION:
THEORY AND PRACTICE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Mihaela Irina Enachescu
June 2008
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

________________________________________
(Ashish Goel)  Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

________________________________________
(Rajeev Motwani)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

________________________________________
(Tim Roughgarden)

Approved for the University Committee on Graduate Studies.

iii
Abstract

Networks such as the internet, corporate networks, and even social ones, are an integral part of today’s society. The utility of the network depends in a fundamental way on the routing protocols that are used to transmit information from a point to another in a network. This thesis focuses on important routing issues in both the wired and the wireless domains. There are multiple objectives, at times conflicting, that one would like to optimize in a routing protocol such as the storage space necessary for the routing scheme, the bandwidth utilization or the amount of information aggregation permitted by the routing scheme.

The first problem we will present is from the wireless domain, in which maximizing information aggregation is of foremost importance. We propose a simple randomized algorithm for routing information on a grid of sensors that allows for a high level data aggregation. Data aggregation is crucial given the severe energy constraints of the sensors. We prove that our routing scheme is a constant factor approximation (in expectation) to the optimum aggregation tree simultaneously for a class of aggregation functions. We also present a deterministic algorithm, which requires additional global information about the number of sensors present, and which provides a slightly improved constant factor approximation to all optimal aggregation trees considered.

Next we study how to use routes that are as short as possible (i.e. have low stretch) while keeping routing tables small in the wired domain. The motivation here is scalability. We derive the specific space-stretch tradeoffs of routing in Bernoulli random graphs, and provide the first proof that stretch less than 3 can be obtained in conjunction with sublinear routing tables in this class of graphs. We also develop an efficient algorithm that finds the near-optimum space-stretch tradeoff for landmark-based schemes in any given network. Our algorithm produces routing tables that use no more than $O(\log n)$ more space than the
optimum landmark-based scheme for achieving stretch $s$ with $L$ landmarks. This is a novel tool for obtaining near-optimum stretch-space tradeoffs for specific graphs.

Finally, we address the placement of relay service agents (routelets) in the Internet to assist multipath transport protocols; these protocols are designed to achieve better network utilization and fairness by exploiting path diversity in the Internet. We identify three different routelet deployment scenarios, provide LP formulations for routelet placement in each of the scenarios, and prove that the placement problem is NP-complete in two of the scenarios. We provide rounding algorithms with provable properties for a subset of the scenarios, and compare their performance through simulations on several BRITE topologies of varying scales.
Acknowledgements

I would like to thank...
# Contents

Abstract iv

Acknowledgements vi

1 Introduction 1

2 Routing in Sensor Networks 4
   2.1 Introduction .................................................. 4
       2.1.1 Our Results ............................................. 6
       2.1.2 Related Work ............................................ 7
   2.2 Problem Definition ........................................... 9
   2.3 Relating Opportunistic Aggregation to Collision Time .......... 11
   2.4 The Probabilistic Distribution Shortest Path Algorithm ........ 12
       2.4.1 Proving the Average Collision Time of the Random Walks .... 13
   2.5 The Hierarchical Decomposition Approach ...................... 19
       2.5.1 The Hierarchical Decomposition Algorithm ............... 19
   2.6 Conclusions and Future Work .................................. 21

3 Compact Routing 22
   3.1 Introduction .................................................. 22
       3.1.1 Our Results ............................................. 23
       3.1.2 Related Work ............................................ 24
   3.2 Model and Definitions ......................................... 25
       3.2.1 Landmark-Based Routing Scheme .......................... 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Stretches Less than 3 in Bernoulli Random Graphs</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Technical Details for Lemmas 3.3.2 and 3.3.3</td>
<td>31</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Proof idea for Lemma 3.3.2</td>
<td>32</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Proof of Lemma 3.3.3 [Modified from [19]]</td>
<td>32</td>
</tr>
<tr>
<td>3.5</td>
<td>Greedy Algorithm for Landmark Selection</td>
<td>36</td>
</tr>
<tr>
<td>3.6</td>
<td>Simulations and Real Network Data Results</td>
<td>38</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Random Graphs</td>
<td>39</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Power-law Graphs</td>
<td>42</td>
</tr>
<tr>
<td>3.6.3</td>
<td>AS Graph from Real Network Data</td>
<td>43</td>
</tr>
<tr>
<td>3.7</td>
<td>Conclusions</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>Routelet Placement for Multipath Routing</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>Related Work</td>
<td>47</td>
</tr>
<tr>
<td>4.3</td>
<td>Problem Formulation</td>
<td>48</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Deployment Scenarios</td>
<td>49</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Problem Statements</td>
<td>50</td>
</tr>
<tr>
<td>4.4</td>
<td>NP hardness Result</td>
<td>51</td>
</tr>
<tr>
<td>4.5</td>
<td>LP Formulation</td>
<td>53</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Scenario with no routelets</td>
<td>53</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Scenario (a)</td>
<td>54</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Scenario (b)</td>
<td>55</td>
</tr>
<tr>
<td>4.5.4</td>
<td>The LP for Scenario (c)</td>
<td>57</td>
</tr>
<tr>
<td>4.6</td>
<td>Rounding Algorithms</td>
<td>58</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Scenario (a)</td>
<td>59</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Scenario (b)</td>
<td>61</td>
</tr>
<tr>
<td>4.7</td>
<td>Experimental Methodology and Evaluation</td>
<td>61</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Results - Scenario b</td>
<td>63</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Results - Scenario a</td>
<td>67</td>
</tr>
<tr>
<td>4.8</td>
<td>Conclusion</td>
<td>71</td>
</tr>
<tr>
<td>A</td>
<td>Chernoff Bounds</td>
<td>72</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Contributions and open problems. ......................... 47
4.2 Topology Characteristics (generated using BRITE) ........ 62
List of Figures

2.1 Sensor correlation example for $k = 4$ ................................................. 9
2.2 Hierarchical algorithm illustration ......................................................... 20

3.1 Random Landmark Selection scheme in random graphs ......................... 40
3.2 Greedy Landmark Selection in random graphs ..................................... 40
3.3 Histogram of node degree and the degree distribution of landmarks chosen
   by Greedy Landmark Selection scheme in a random graph ....................... 41
3.4 Highest-degree Landmark Selection in random graphs ........................... 42
3.5 Highest-degree Landmark Selection on power-law graphs ...................... 43

4.1 Deployment Scenarios ........................................................................... 49
4.2 Reduction from Balanced Complete Bipartite Subgraph ........................... 52
4.3 Aggregate throughput, Scenario b, Topo1 ............................................. 63
4.4 Throughput per SD-pair, Scenario b, Topo 1 ......................................... 64
4.5 Dependence of aggregate throughput on the number of SD-pairs ............ 65
4.6 Number of routelets used, Scenario b, Topo 1 ....................................... 66
4.7 Performance accross various topologies for scenario b ......................... 66
4.8 Aggregate throughput, Scenario a, Topo1 ............................................. 67
4.9 Per SD-pair throughput for the five algorithms, Scenario a, Topo 1 ........... 68
4.10 Dependence of aggregate throughput on the number of SD-pairs, Scenario
     a, Topo1 ......................................................................................... 69
4.11 Number of routelets used, Scenario a, Topo1 ....................................... 69
4.12 Max-edge violation factor, Scenario a, Topo1 ....................................... 70
4.13 Performance accross various topologies for scenario a ......................... 70
Chapter 1

Introduction

Routing is among the most intellectually challenging aspects of internetworking; after thirty years of research networks still exhibit problems [...] 

–Doug Comer, VP Cisco

This thesis studies several optimization problems related to network routing. The objectives we seek to optimize in this work are: data aggregation (for sensor networks), scalability (memory requirement necessary to route on paths that are as short as possible) and bandwidth utilization.

In Chapter 2 we study sensor networks, which are distributed data collection systems, frequently used for monitoring environments in which “nearby” data has a high degree of correlation. This induces opportunities for data aggregation, that are crucial given the severe energy constraints of the sensors. Thus it is very desirable to take advantage of data correlations in order to avoid transmitting redundancy. In our model, we formalize a notion of correlation, that can vary according to a parameter $k$. Then we relate the expected collision time of ”nearby” walks on the grid to the optimum cost of scale-free aggregation.

We also propose a very simple randomized algorithm for routing information on a grid of sensors that satisfies the appropriate collision time condition. Thus, we prove that this simple scheme is a constant factor approximation (in expectation) to the optimum aggregation tree simultaneously for all correlation parameters $k$. The key contribution in our randomized analysis is to bound the average expected collision time of non-homogeneous
random walks on the grid, i.e. the next hop probability depends on the current position. The work in this chapter is based on joint work with Ashish Goel, Ramesh Govindan, and Rajeev Motwani, which has been published as [26].

In Chapter 3, we study compact routing, an area motivated by the importance in communication networks to use routes that are as short as possible (i.e. have low stretch) while keeping routing tables small for scalability reasons. Recent advances in compact routing (for example [5, 20]) show that a stretch of 3 can be achieved while maintaining a sub-linear (in the size of the network) space at each node. It is also known that no routing scheme can achieve stretch less than 3 with sub-linear space for arbitrary networks [53]. In contrast, simulations on the Internet inter-domain graph (and other internet-like topologies) have indicated that stretch less than 3 can indeed be obtained using sub-linear sized routing tables [44]. In this chapter, we further investigate the space-stretch tradeoffs for compact routing by analyzing a specific class of random graphs and by presenting an efficient algorithm that (approximately) finds the optimum space-stretch tradeoff for any given network.

We first study compact routing on a popular model of random graphs, known as Bernoulli (or Erdős-Renyi) random graphs, and prove that stretch less than 3 can be obtained in conjunction with sub-linear routing tables. In particular, stretch 2 can be obtained using routing tables that grow roughly as $n^{3/4}$ where $n$ is the number of nodes in the network.

Compact routing schemes often involve the selection of landmarks. We present a simple greedy scheme for landmark selection that takes a desired stretch $s$ and a budget $L$ on the number of landmarks as input, and produces a set of at most $O(L \log n)$ landmarks that achieve stretch $s$. Our scheme produces routing tables that use no more than $O(\log n)$ more space than the optimum scheme for achieving stretch $s$ with $L$ landmarks. This may be a valuable tool for obtaining near-optimum stretch-space tradeoffs for specific graphs. We simulate this greedy scheme (and other heuristics) on multiple classes of random graphs as well as on Internet like graphs.

The work in this chapter is based on joint work with Mei Wang and Ashish Goel and has been published as [28].

In Chapter 4, we address the placement of relay service agents (routelets) in the Internet to assist multipath transport protocols; these protocols are designed to achieve better
network utilization and fairness by exploiting path diversity in the Internet. We identify three different routelet deployment scenarios, provide LP formulations for routelet placement in each of the scenarios, and prove that the placement problem is NP-hard in two of the scenarios. We provide rounding algorithms with provable properties for a subset of the scenarios, and compare their performance through simulations on several BRITE topologies of varying scales. We observe that our rounding algorithm leads to a 2-5 fold increase in bandwidth utilization compared to the default single path routing. This closely matches the performance of the optimal LP throughput in the several topologies tested.

The work in this chapter is based on joint work with Ravi Kokku, and has been published as [27].
Chapter 2

Routing in Sensor Networks

As mentioned in the introduction, there are many objectives that routing schemes can be optimized for, depending on the application domain. In this chapter, we will focus on maximizing the information aggregation, a crucial objective in wireless sensor network communication.

2.1 Introduction

Consider a network where each node gathers information from its vicinity and sends this information to a centralized processing agent. If the information is geographically correlated, then a large saving in data transmission costs may be obtained by aggregating information from nearby nodes before sending it to the central agent. This is particularly relevant to sensor networks where battery limitations dictate that data transmission be kept to a minimum, and where sensed data is often geographically correlated. In-network aggregation for sensor networks has been extensively studied over the last few years [45, 36, 61]. In this chapter we show that a very simple opportunistic aggregation scheme can result in near-optimum performance under widely varying (and unknown) scales of correlation.

More formally, we consider the idealized setting where sensors are arranged on an $N \times N$ grid, and the centralized processing agent is located at position $(0, 0)$ on the grid. We assume that each sensor can communicate only to its four neighbors on the grid. This idealized setting has been widely used to study broad information processing issues in
sensor networks (see [52], for example). We call an aggregation scheme opportunistic if data from a sensor to the central agent is always sent over a shortest path, i.e., no extra routing penalty is incurred to achieve aggregation.

To model geographic correlations between data collected by nearby sensors, we assume that each sensor can gather information in a $k \times k$ square (or, a circle of radius $k/2$) centered at the sensor. We will refer to $k$ as the correlation parameter. Let $A(x)$ denote the area sensed by sensor $i$. If we aggregate information from a set of sensors $S$ then the size of the resulting compressed information is $I(S) = \left| \bigcup_{x \in S} A(x) \right|$, i.e., the size of the total area covered by the sensors in $S$. The parameter $k$ would usually depend on the type of the information being sensed. One may build for example sensors that can monitor distinct factors such as 1. temperature and 2. birds with RFID tags attached to their feet. There is no reason for these two distinct phenomena to share the same $k$ value. Accordingly, we will assume that the parameter $k$ is not known in advance. In fact, we would like our opportunistic aggregation algorithms to work well simultaneously for all $k$.

There are scenarios where the above model applies directly. For example, the sensors could be cameras which take pictures within a certain radius, or they could be sensing RFID tags on retail items (or on birds which have been tagged for environmental monitoring) within a certain radius. Also, since we want algorithms that work well without any knowledge of $k$, our model applies to scenarios where the likelihood of sensing decreases with distance. For example, consider the case where a sensor can sense an event at distance $r$ only if it has “intensity” $f(r)$ or larger, where $f$ is a non-decreasing function. Then, events of intensity $y$ correspond to information with correlation parameter roughly $f^{-1}(y)$; if these events are spread uniformly across the sensor field then an algorithm which works well for all $k$ will also work well for this case.

Thus, we believe that our model (optimizing simultaneously for all $k$) captures the joint entropy of correlated sets of sensors in a natural way for a large variety of applications, a problem raised by Pattem et al. [52].

For node $(i, j)$, we will refer to nodes $(i - 1, j)$ and $(i, j - 1)$ as its downstream neighbors, and nodes $(i + 1, j)$ and $(i, j + 1)$ as its upstream neighbors. Since we are on a grid, we will also informally say that the neighbors are to the left or down/bottom (for downstream) and right or up/top (for upstream) of the original node $(i, j)$. We would like to construct a
tree over which information flows to the central agent, and gets aggregated along the way. Since we are restricted to routing over shortest paths, each node has just one choice: which downstream node to choose as its parent in the tree. In our algorithm, a node \((i, j)\) waits till both its upstream neighbors have sent their information out\(^1\). Then it aggregates the information it sensed locally with any information it received from its upstream neighbors and sends it on to one of its downstream neighbors. The cost of the tree is defined as the total amount of (compressed) information sent out over links in the tree.

Note that we do not need all sensors at a certain distance to transmit synchronously; we just need to make sure that a node sends its information only after both its upstream nodes have transmitted theirs. This can be enforced asynchronously by each sensor. In any case, Madden et al. [47] have developed protocols to facilitate synchronous sending of information by sensors (depending on the distance from the sink) which we can leverage if needed.

### 2.1.1 Our Results

We propose a very simple randomized algorithm for choosing the next neighbor – node \((i, j)\) chooses its left neighbor with probability \(i/(i+j)\) and its bottom neighbor with probability \(j/(i+j)\). Observe that this scheme results in all shortest paths between \((i, j)\) and \((0, 0)\) being chosen with equal probability \(^2\). We prove that this simple scheme is a constant factor approximation (in expectation) to the optimum aggregation tree simultaneously for all correlation parameters \(k\). While we construct a single tree, the optimum trees for different correlation parameters may be different.

The key idea is to relate the expected collision time of random walks on the grid \(^3\) to scale free aggregation. Consider two neighboring nodes \(X\) and \(Y\) (i.e. nodes which can communicate with one another in our model), and randomly trace a shortest path from each

\(^1\)Of course maybe one, or both, of the upstream nodes may decide not to choose \((i, j)\) as the parent node; however we assume that node \((i, j)\) gets notified anyway when its upstream nodes send information out.

\(^2\)Note that if you multiply the resulting probabilities, as the path approaches the origin the denominators are exactly the same for all the paths; the numerators are also the same (but permuted depending on the specific path).

\(^3\)In the random walks considered here the probability of each move will depend on the current grid position.
of them to the sink. Define the collision time to be the number of hops (starting at say $X$) before the traces first meet. We first show (Sect. 2.3) that if the average expected collision is $O\left(\sqrt{N}\right)$, then we have a constant factor approximation algorithm to the optimal aggregation for all correlation parameters $k$. We then show that the average expected collision time for our randomized algorithm is indeed $O\left(\sqrt{N}\right)$ (Sect. 2.4). This analysis of the average expected collision time is our main technical theorem and may be of independent interest. To achieve this result, we first analyze the expected number of differing steps (where the two paths move in different directions) and then prove that the probability of a step being a differing step is a super-martingale.

We also present (Sect. 2.5) a slightly more involved hierarchical routing algorithm that is deterministic, and has an average collision time of only $O(\log N)$; hence the deterministic algorithm is also a constant factor approximation for all correlation parameters $k$. While this scheme has a slightly better performance, we believe that the simplicity of the randomized algorithm makes it more useful from a practical point of view.

Our results hold only for the total cost, and critically rely on the fact that information is distributed evenly through the sensor field. It is easy to construct pathological cases where our algorithm will not result in good aggregation if information is selectively placed in adversarial chosen locations.

This result shows that, at least for the class of aggregation functions and the grid topology considered in this chapter, schemes that attempt to construct specialized routing structures in order to improve the likelihood of data aggregation [35] are unnecessary. This is convenient, since such specialized routing structures are hard to build without some a priori knowledge about correlations in the data. With this result, simple geographic routing schemes like GPSR [39], or tree-based data gathering protocols are sufficient [36, 47].

### 2.1.2 Related Work

Given the severe energy constraints and high transmission cost in the sensor network setting, data aggregation has been recognized as a crucial operation, which optimizes performance and longevity [34]. In the sensor network literature, aggregation can refer to either a database aggregate operator (min,max,sum etc.) [16, 47, 48], or to general aggregation
functions such as the one we consider in this chapter.

Goel and Estrin [31] studied routing that leads to a simultaneously good solution (a $\log N$ approximation) for a large class of aggregation functions. In their model, the amount of aggregation only depends on the number of nodes involved, independent of location, and the network need not be a grid. In our problem, the amount of aggregation depends on the location of the sensors being aggregated: the closer two sensors are, the more correlated their data is. But it is also easier to aggregate data from nearby nodes. Hence, it seems intuitive that better simultaneous optimization may be possible for our case, an intuition that we have verified in this chapter.

We build on the work of Pattem et al. [52] who study a closely related question, comparing three different classes of compression schemes for sensor networks: routing-driven compression, in which the routes from the nodes to the destination point just follow a shortest path, and in which compression is done opportunistically whenever possible, compression-driven routing which builds up a specialized routing structure, and distributed source coding which leverages a priori information about correlations. After a theoretical analysis of these schemes, they introduce a generalized cluster-based compression scheme in which correlated readings are aggregated at a cluster head, which is studied via simulations. They find that across a wide variety of correlations (roughly parameterized by the joint entropy of two sensors spaced $d$ apart), the cluster-based compression scheme works reasonably well with a relatively fixed cluster size. Our model captures a wider range of joint entropy functions (since we also approximate any linear composition of $k$-correlated information for different values of $k$), one of the open problems they pose. Also, we present a formal proof of simultaneous optimization. It is easy to see that their cluster-based compression scheme does not perform well in our model, in that no single cluster size can be within constant factor of the optimal aggregation tree for all $k$.

Another study of routing schemes for correlated sensors has been performed by Cristescu et al. [21]. They showed that for a two-stage model where the amount of information depends only on whether a node is an internal node or a leaf, finding the optimum aggregation tree is NP-hard; they also present a constant factor approximation for this problem. Their result holds for an arbitrary sensor network (as opposed to just a grid).
2.2 Problem Definition

Recall our setting in which sensors are arranged in a $N \times N$ grid with a centralized processing agent at $(0,0)$. Each sensor can only communicate with its immediate neighbors on the grid (at most four). We can assume that each sensor knows its (x,y)-coordinates. This can be done for example via the fine-grain localization method described by Savvides et al. [60].

The sensor network can sense multiple kinds of data. For a specific type of data, we will refer to the information contained in a $1 \times 1$ grid square as a value. We then define this type of data $k$-correlated data if the following holds:

(i) Each value is sensed by all the sensors in a $k \times k$ grid centered at that location, as in Fig. 2.1. We will assume for simplicity that $k$ is even, so that the notion of centering is well defined.

(ii) Let $A_k(x)$ denote the set of grid squares sensed by sensor $x$. If information from a set $S$ of sensors is aggregated, the resulting information is of size $|\bigcup_{x \in S} A_k(x)|$

We will look at $k$-correlated data for which $k < N/2$, since otherwise we obtain an interesting case in which all information can be captured by a single sensor in the network. As stated in the introduction, we need not assume that the nodes equidistant from the central agents send data synchronously. However, we assume the transmission is partially synchronized, so that a node sends information only after receiving all data from its upstream neighbors and after finishing aggregating that information with its own data. We assume that a sensor cannot withhold information, and needs to send all information it can sense.
Given this setting, we want to find a tree on which to send information from all sensors to \((0, 0)\) so as to minimize the cost, simultaneously, for all values of \(k\). In our cost model we focus on the transmission cost, assuming perfect aggregation, i.e. assuming a value \(v\) is transmitted across an edge \(e\) at most once. Each time such a value is transmitted from some node to its downstream neighbor in the routing tree, the total cost increases by 1. More formally the cost we consider is given by the following equation:

\[
\text{COST} = \sum_e |\{v| \text{value } v \text{ is transmitted across edge } e\}|
\]

**Theorem 2.2.1** Assume that sensors with correlation parameter \(k\) are arranged in a \(N \times N\) grid and are sending information to the centralized processing agent along the optimal tree. The cost of the optimal tree \((\text{OPT}_k)\) is \(\theta(N^3 + (Nk)^2)\).

**Proof:** Look at one individual value, at point \((x, y)\) with \(x, y \geq 0\) and construct the minimum cost routing for it.

The closest node to the origin that senses this value, at coordinates \((x - k/2, y - k/2)\), has to send the value all the way to the origin, so a cost of \(D = x + y - k\) (distance from the point to the origin must be paid). All values incur this cost.

The value must be transmitted by all the nodes that can sense it, each node thus introducing a cost of 1. Thus, for all values for which the sensing \(k \times k\) square is included in the \(N \times N\) grid (it is easy to observe that there are \((N - k)^2\) such values), there is a cost of at least \(k^2\) before the distinct values can be aggregated at a single node. We ignore this contribution to the cost for the other values.

Since we assume different values cannot be aggregated between them, we get a lower bound for the overall cost of at least:

\[
\sum_{\text{values}} D + (N - k)^2 k^2 = 4 \left( \sum_{0 < x < N, 0 < y < N} (x + y - k) \right) + (k(N - k))^2
\]

\[
= N^2(N - 1 - k) + (k(N - k))^2
\]

If we only consider parameters \(k < N/2\) then \(N - 1 - k \leq N/2\) and the above becomes
There may not exist a single tree which is optimal for each value \( v \). This is because from the point of view of a value \( v_1 \) a certain sensor may need to communicate with one downstream neighbor for optimal aggregation, while from the point of view of another value \( v_2 \) that same sensor may need to communicate with the other downstream neighbor. This would lead to an impossible solution for the routing tree in which only one downstream neighbor can be selected. However, our analysis, while not solving for the exact value, does give a lower bound on \( \text{OPT} \).

\[ \theta(N^3 + (Nk)^2) \text{ as desired.} \]

### 2.3 Relating Opportunistic Aggregation to Collision Time

Recall our definition of opportunistic algorithm, i.e. one in which the information from node \( X \) is sent to the processing agent on a shortest path. Note that the paths from any two neighbors \( X \) and \( Y \), having the same destination, will eventually meet at some point \( Z \). We call the distance from \( X \) to \( Z \) the collision time of \( X \) and \( Y \).

**Theorem 2.3.1** An opportunistic algorithm with average expected collision time \( O(\sqrt{N}) \) gives a constant factor approximation to the optimum aggregation cost for all \( k \).

**Proof:**

In a similar fashion as in our proof for the lower bound for the \( \text{OPT} \), look from the point of view of a data value which is shared by \( k \times k \) sensors. Inside the square we pay the same cost as in our lower bound for the \( \text{OPT} \) (i.e. at most \( k^2 \)). Also the lower-left node transmits the value to \((0,0)\) via a shortest path just as in \( \text{OPT} \). So far, the cost is the same.

To analyze the extra cost incurred by the opportunistic algorithm from our hypothesis, consider the left and lower sides of the \( k \times k \) sensing square of a given value. The paths from all sensors inside the square will go through one of the points on these sides. Consider these paths from the sides of the square to \((0,0)\). There is some extra cost equal to the collision time between two adjacent nodes from the left and lower sides of the square, since two instances of the shared value are transmitted, instead of only one instance as would happen in the optimal case. It is easy to see that for each pair of adjacent nodes, there are \( k \) values.
that incur the extra cost due to the collision time, or put another way, $k$ shared values for which these nodes are on the sides. Summing up the OPT cost and the extra cost we obtain the following equation for the total cost of our algorithm:

$$\sum_{\text{values}} D + k^2 N^2 + k \sum_{\text{sensors}} \text{(collision time of the paths from two adjacent sensors)}$$

$$= \theta \left(N^3 + (kN)^2\right) + O\left(kN^{2.5}\right)$$

The first two terms are the same as in the lower bound for $OPT_k$.

If $k < \sqrt{N}$ then the $N^3$ term dominates the $(Nk)^2$ term, as well as the $k \times N^{2.5}$ term, and we get an $O(1)$-approximation.

If $k > \sqrt{N}$ then the $(Nk)^2$ term dominates the other two terms, and we get again an $O(1)$-approximation.

Note that we compare to a lower bound for $OPT$, not $OPT$ itself, which may be hard to compute, so the constant factor may be even less than what we can compute here.

### 2.4 The Probabilistic Distribution Shortest Path Algorithm

We will present a simple randomized opportunistic algorithm for constructing a tree. The path from each node will be a random walk towards the processing agent, but the walks are not independent. The main result is to prove that the average expected collision time of two adjacent paths in the resulting routing tree is $O\left(\sqrt{N}\right)$. The analysis of our random process may well be of independent interest. Applying theorem 2.3.1, we conclude that this algorithm produces a constant factor approximation of the optimal aggregation trees for any value of $k$.

**The Probabilistic Distribution Algorithm:** For every node, if the node is located at position $(x, y)$, choose to include in the MST the left edge with probability $\frac{x}{x+y}$ and the down edge with probability $\frac{y}{x+y}$.

**The Random Walk view:** We can view the above process as a tree constructed from random walks originating from each sensor. At each time step the current node chooses one
of the (at most) two downstream nodes as its parent. Because a node waits for its upstream nodes to transmit we can view the process as a flow in which the data gets closer by one to the origin at each time step. In our model, when two walks meet (passing some step through the same node) they “collapse” into a single walk and lose their independence. The analysis of the expected collision time for this random process is presented below. We believe our analysis is interesting since the random walk is non-homogeneous, thus standard random walk results do not apply.

2.4.1 Proving the Average Collision Time of the Random Walks

Theorem 2.4.1 (Random Walk Theorem) The average expected collision time of two adjacent walks as generated by the randomized probabilistic distribution algorithm is \( O\left(\sqrt{N}\right) \).

Let us first introduce some notation, definitions, and lemmas which would help us prove the above result.

Two neighboring nodes can be either horizontal or vertical neighbors, and one, say the second, must be the upstream neighbor of the other. Thus there is a \( \frac{1}{x+y} \) probability to meet initially. If they do not meet initially, then the upstream node chooses as its parent the other downstream node, which on the grid is at distance 2 from the first node, and at the same distance from the origin.

Let’s assume that the two walks do not meet initially. Thus, we will analyze the collision time of the random walks originating at \((x-1, y)\) and \((x, y-1)\). This new “diagonal” collision time provides a lower bound in the collision time of the initial “horizontal/vertical” neighbors. In fact we will prove the result stated in our Random Walk Theorem for this redefined notion of collision time, which then implies the original theorem.

Note that, because the nodes are at the same distance from the origin we can imagine them moving towards the processing agent “in sync” (this synchronicity assumption is not needed but it helps in thinking about the process). Look at the horizontal difference between the two paths, as a function of time, and let’s denote this by \( \Delta_t(x, y) \). Initially, \( \Delta_0(x, y) = 1 \). Because in general we focus our attention to a specific \((x, y)\) we will drop these parameters from the notation. We want to analyze \( E[t_c] \) where \( t_c \) is such that \( \Delta_t = 0 \) for the first time. Observe that \( t_c \) is precisely the collision time as defined earlier, since the
two walks start from the same distance from the origin, and at every time step we assume the walks move one unit closer, so there is a one-to-one correspondence between time and distance from the initial point to the collision point. Once the horizontal distances become equal, the vertical distances must also be equal and the two paths would meet.

Let $M = x + y - 1$, the initial distance from the origin.

At each time step, $\Delta_t$ can stay the same or become different (increase or decrease by 1). We call a step at which $\Delta_t$ differs from $\Delta_{t-1}$ a differing step. By analyzing these differing steps we will transform the problem from a two dimensional process to a one dimensional process.

We will first analyze the number of differing steps before collision (Lemmas 2.4.2 and 2.4.3), and then bound the probability that a step is a differing step (Lemmas 2.4.4 and 2.4.5). These results together will lead to the proof of the main result.

Let’s denote by $D(x, y)$ the number of differing steps before $\Delta_t$ becomes 0 for the first time.

**Lemma 2.4.2** $E[D(x, y)]$ is $O(\sqrt{\min(x, y)})$.

**Proof:** At time $t$, when the first path is above at, say, point $(x_1, y_1)$ and the second path is below at point $(x_2, y_2)$ we know that $x_1 + y_1 = x_2 + y_2 = M - t$. Initially $x_1 < x_2$, so this above/below relation will continue to hold until $\Delta = x_2 - x_1$ first becomes 0. Also, initially, $x_2 = x$ and $y_1 = y$.

Based on our probabilistic model, and the above/below relation we derive the following for the next time step:

\[
\Pr(\Delta_{t+1} - \Delta_t = 1) = \frac{x_1 y_2}{(x_1 + y_1)^2} \quad \text{and} \quad \Pr(\Delta_{t+1} - \Delta_t = -1) = \frac{x_2 y_1}{(x_1 + y_1)^2}
\]

Using $y_1 = M - t - x_1$ and $y_2 = M - t - x_2$ we obtain the following:

\[
\Pr(\Delta_{t+1} - \Delta_t = 1) + \Pr(\Delta_{t+1} - \Delta_t = -1) = \frac{(M - t)(x_1 + x_2) - 2x_1 x_2}{(M - t)^2}
\]

and $\Pr(\Delta_{t+1} - \Delta_t = 1) - \Pr(\Delta_{t+1} - \Delta_t = -1) = -\frac{\Delta_t}{M - t}$.
Now define \( p_f(t) = \Pr(\Delta_{t+1} - \Delta_t = 1 | \Delta_{t+1} - \Delta_t \neq 0) \) and \( p_r(t) = \Pr(\Delta_{t+1} - \Delta_t = -1 | \Delta_{t+1} - \Delta_t \neq 0) \) to be the conditional (normalized) probabilities of a forward (positive) change in \( \Delta \), and of a reverse (negative) change in \( \Delta \), respectively.

Also define \( \lambda \) as below:

\[
\lambda = p_f(t) - p_r(t) = \frac{\Pr(\Delta_{t+1} - \Delta_t = 1) - \Pr(\Delta_{t+1} - \Delta_t = -1)}{\Pr(\Delta_{t+1} - \Delta_t = 1) + \Pr(\Delta_{t+1} - \Delta_t = -1)} = -\frac{\Delta_t(M-t)}{(M-t)(x_1 + x_2) - 2x_1x_2}
\]

Since \( p_f(t) + p_r(t) = 1 \), we can rewrite \( p_f(t) \) and \( p_r(t) \) as:

\[
p_f(t) = \frac{1}{2} + \frac{\lambda}{2} \quad \text{and} \quad p_r(t) = \frac{1}{2} - \frac{\lambda}{2}
\]

where \( \lambda \) still contains a dependence on \( t \). The convergence to \( \Delta = 0 \) can only be slower if \( \lambda \) is smaller in absolute value. Note that by removing the \( 2x_1x_2 \) term from the denominator of \( \lambda \) we can only decrease the overall absolute value of \( \lambda \). Also, we get the same effect if we replace \( x_1 + x_2 \) by \( 2 \max(x_1, x_2) = 2x_2 \).

Also, the \( M - t \) factor will get simplified so we can replace \( \lambda \) by \(-\frac{\Delta}{2x_2}\) to obtain new forward and reverse probabilities, independent of \( t \) and only dependent on \( \Delta \):

\[
n_f(\Delta) = \frac{1}{2} - \frac{\Delta}{4x_2} \quad \text{and} \quad n_r(\Delta) = \frac{1}{2} + \frac{\Delta}{4x_2}
\]

Now consider an integer random walk in \([0, \max(x_1, x_2) = x_2]\), with an absorbing barrier at 0, and a reflecting one at \( \max(x_1, x_2) = x_2 \).

We analyze the behavior of this one dimensional random walk in lemma 2.4.3. By construction, the expected time for this new random walk to reach 0 starting from 1 is an upper bound to the expected time for \( \Delta \) to reach 0 starting from 1.

We can then conclude that \( \Delta \) reaches 0 in \( O(\sqrt{x_2}) \) by directly applying the result in lemma 2.4.3. By symmetry we can also obtain time \( O(\sqrt{y_1}) \). Since \( x_2 = x \) and \( y_1 = y \) initially, the theorem is proven.

\[\square\]

**Lemma 2.4.3** Consider an integer random walk starting at point 1 on the interval \([0, x]\). Assume that, if we are at position \( j \) the random walk moves right with probability \( n_f(j) \), and left with probability \( n_r(j) \) in the interval \([1, x-1]\) where \( n_f(j) \) and \( n_r(j) \) are as defined

\[\text{Note that } \lambda \text{ is negative.}\]**
in lemma 2.4.2. Assume that the point 0 is absorbing, and that the point x is reflecting (i.e. the walk moves to x − 1 with probability 1 from x). If the walk starts at point 1, then the expected number of time steps necessary for this walk to first reach 0 is $O(\sqrt{x})$.

**Proof:**

Note that at each step we move either in one direction or the other, since, by definition, $n_r + n_f = 1$.

Define $B(j)$ to be the expected number of steps before the point $j - 1$ is first visited, assuming that the random walk starts at point $j$. We are then looking for the value of $B(1)$. We will use the properties of the walk, in particular the values of $n_f(j)$ and $n_r(j)$ to derive a recursive formula for $B(j)$ and then get a bound for $B(1)$.

If we pass exactly $i + 1$ times through point $j$ before reaching point $j - 1$, the expected number of steps is $iB(j + 1) + 1$. The probability of this event is $n_r(j)n_f(j)^i$. Since $i$ can range from 0 to $\infty$ we get the following relation for $B(j)$, where $j \in [1, x - 1]$:

$$B(j) = \sum_{i=0}^{\infty} n_r(j)n_f(j)^i(iB(j + 1) + 1) = n_r(j)\sum_{i=0}^{\infty} n_f(j)^i + n_r(j)B(j + 1)\sum_{i=0}^{\infty} n_f(j)^i$$

$$= \frac{n_r(j)}{1 - n_f(j)} + \frac{n_r(j)n_f(j)}{(1 - n_f(j))^2}B(j + 1) = 1 + \frac{n_f(j)}{n_r(j)}B(j + 1) = \frac{2x - j}{2x + j}B(j + 1) + 1$$

Further note that $B(x) = 1$ because $x$ is a reflecting barrier, so in the next step we move back with probability 1.

We want to solve for $B(1)$, the value of interest.

If we expand $B(1)$ in terms of $B(x)$ we obtain:

$$B(1) = \sum_{i=1}^{2x} \frac{2x - 1}{2x + 1} \times \ldots \times \frac{2x - i}{2x + i}$$

To simplify notation, denote $2x$ by $X$ and $\frac{2x - 1}{2x + 1} \times \ldots \times \frac{2x - i}{2x + i}$ by $T_i$.

Note that the $T_i$’s are decreasing as $i$ increases, since all component factors are less than 1. Now, note that for $i \in \{1, \ldots, 2\sqrt{X}\}$ we have $T_i \leq 1$.

For $i \in \{2\sqrt{X} + 1, \ldots, 3\sqrt{X}\}$ we have $T_i \leq \left(\frac{X - \sqrt{X}}{X + \sqrt{X}}\right)^\sqrt{X}$ since the last $\sqrt{X}$ factors in each of these $T_i$ are all less than $\frac{X - \sqrt{X}}{X + \sqrt{X}}$. 


In general, for any \( m \), if \( i \in \{ m\sqrt{X}+1, \ldots, (m+1)\sqrt{X} \} \) we have \( T_i \leq \left( \frac{X-\sqrt{X}}{X+\sqrt{X}} \right)^{\sqrt{X}^{(m-1)}} \).

Thus \( B(1) \) can be upper bounded by a geometric series with sum \( \frac{X+\sqrt{X}}{2\sqrt{X}} \). Note that the \( \left( \frac{X-\sqrt{X}}{X+\sqrt{X}} \right)^{\sqrt{X}} \) is approximately \( e^2 \), and thus constant, for large enough \( X \), where \( X = 2x \). Thus, the first term (the fraction) of this bound is a constant, and we conclude that \( B(1) \) is \( O \left( \sqrt{x} \right) \).

\[ \]

**Definition 2.4.1** Define \( p_t(x, y) = \Pr[\Delta_t \text{ is differing} \mid \text{two walks have not collided yet}] \).

As before, we will omit the arguments \( x, y \) since they are fixed.

**Lemma 2.4.4** For all \( t \), \( p_{t+1} \geq p_t \)

**Proof:**

Suppose the first walk is at coordinates \((i, j)\) and the second one at coordinates \((i + \Delta_t, j - \Delta_t)\).

**Case 1** (\( \Delta_t \geq 2 \)): Then \( \Delta_{t+1} \geq 1 \), since the difference between \( \Delta_t \) and \( \Delta_{t+1} \) can be at most 1. Thus the random walks would not meet at time \( t + 1 \), so we eliminate the conditioning for \( p_{t+1} \), and we have the following:

\[
\Pr[\Delta_t \text{ is differing}] = f(i, j, \Delta_t) = \frac{i(j - \Delta_t) + j(i + \Delta_t)}{(i + j)^2}
\]

\[
\Pr[\Delta_{t+1} \text{ is differing}] = g(i, j, \Delta_t) = \frac{i(j - \Delta_t)f(i - 1, j, \Delta_t + 1)}{(i + j)^2} + \frac{i(i + \Delta_t)f(i - 1, j, \Delta_t)}{(i + j)^2} + \frac{j(i + \Delta_t)f(i, j - 1, \Delta_t - 1)}{(i + j)^2} + \frac{j(j - \Delta_t)f(i, j - 1, \Delta_t)}{(i + j)^2}
\]

It is easy to verify, using Mathematica for example, that \( f(i, j, \Delta_t) - g(i, j, \Delta_t) = 0 \), and hence, \( p_t = p_{t+1} \).

**Case 2** (\( \Delta_t = 1 \)): In this case the conditioning in the definition of \( p_{t+1} \) implies that one of the cases in the above formula cannot take place. We still obtain that \( p_{t+1} \geq p_t \).
Since we want to maintain $\Delta_{t+1} \geq 1$ (no collision at time $t + 1$), we eliminate the case in which the first walk moves from $(i, j - 1)$ to $(i - 1, j - 1)$ and the second walk moves from $(i - 1, j)$ to the same point as the first walk.

Thus our formula for $p_{t+1}$ becomes:

$$
\Pr[\Delta_{t+1} \text{ is differing | the two walks do not collide}] = \frac{i(j - \Delta_t)f(i - 1, j, \Delta_t + 1)}{(i + j)^2} + \frac{j(i - \Delta_t)f(i - 1, j, \Delta_t)}{(i + j)^2} = g(i, j, \Delta_t) - \frac{j(i + \Delta_t)f(i, j - 1, \Delta_t - 1)}{(i + j)^2}
$$

while the formula for $p_t$ remains

$$
\Pr[\Delta_t \text{ is differing}] = f(i, j, \Delta_t) = \frac{i(j - \Delta_t) + j(i + \Delta_t)}{(i + j)^2}
$$

Taking the difference between the two, and simplifying, using Mathematica for example, we obtain:

$$(p_{t+1} - p_t)(i + j)^2 = i^3 - i^2(j - 2) - i(j - 1)^2 + (j - 1)^2j$$

If $j > i$ the right hand side reduces to $2i^2 + (j - i)[(j - 1)^2] - i^2$ which is positive; if $i > j$ the right hand side reduces to $[i^2 - j^2](i - j) + 2i$ which is again positive; if $i = j$ the right hand side is just $2i^2$, again positive.

Combining this with the fact that $(i + j)^2 \geq 0$ for all $i, j$ we deduce that $p_{t+1} - p_t$ is always positive, which is exactly what we wanted to prove.

Lemma 2.4.5 The expected time before a differing step between two adjacent walks originating at coordinates $(x, y)$ and $(x + 1, y - 1)$ is $O\left(\frac{x + y}{\min(x, y)}\right)$.

Proof:

From lemma 2.4.4 we see that at each time step the probability of a differing step is bounded below by $p_0$ which is the initial probability of having a differing step, given by:

$$
\frac{x(y - 1) + (x + 1)y}{(x + y)^2} = \theta\left(\min(x, y)^2\frac{\max(x, y)}{(x + y)^2}\right) = \theta\left(\frac{\min(x, y)}{x + y}\right)
$$
This implies our lemma. ■

**Proof:** [Random Walk Theorem]

Consider two walks at \((x, y)\) and \((x + 1, y - 1)\).

From Lemma 2.4.5 we bound the probability of a differing step to happen. Combining this with the result from Lemma 2.4.2 which bounds the expected number of differing steps before the two walks meet we obtain:

\[
E[\text{collision time for adjacent walks at } (x, y)] = \Theta \left( \sqrt{\min(x, y)} \frac{x + y}{\min(x, y)} \right) = \Theta \left( \frac{x + y}{\sqrt{\min(x, y)}} \right)
\]

Taking the sum over all \(x, y\) pairs we obtain: \(\sum_{x,y} \left( \frac{x+y}{\sqrt{\min(x,y)}} \right) = \Theta(N^{2.5})\).

Thus, since there are \(O(N^2)\) pairs of adjacent nodes, the average is exactly \(O(\sqrt{N})\), which concludes the proof of the expected average collision time theorem. ■

### 2.5 The Hierarchical Decomposition Approach

We now present a deterministic algorithm for constructing a tree that produces a constant factor approximation for any value of \(k\). This algorithm has better properties (its average collision time is \(O(\log N)\) instead of \(O\left(\sqrt{N}\right)\) for example), but it is more involved. Also the approximation provided is still \(O(1)\).

The solution is based on the idea of dividing the grid into sub-grids, and collecting all the values in a given sub-grid at the sensor closest to the origin before forwarding it onto the next sub-grid.

#### 2.5.1 The Hierarchical Decomposition Algorithm

We present the construction and the proof of correctness in parallel. We need two stages: a top-down stage in which we establish the sub-grids recursively, and a bottom-up stage in which we put the sub-grids together. We will assume for simplicity that \(N\) is a power of 2.
The Top-Down Stage: Divide the first quadrant in four sub-grids of size $N/2 \times N/2$, each of which is further divided in four size-$N/4 \times N/4$ sub-grids, and so on. For each sub-grid we will make sure that the MST converges to the sensor closest to the origin, i.e. if there is choice in what direction to move towards the origin, choose the choice that would not leave the sub-grid. If there is still choice choose arbitrarily.

The Bottom-Up Stage: We will prove by construction the following lemma.

Lemma 2.5.1 If a $2^k \times 2^k$ sub-grid has the property that its average collision time is less than $ck$ for all adjacent node pairs in the sub-grid, then we can construct a $2^{k+1} \times 2^{k+1}$ sub-grid with average collision time of $c(k+1)$ for all adjacent node pairs, where $c$ is some constant greater than 2.

Proof:

We assume the parent node is determined for all nodes inside the $2^k \times 2^k$ sub-grid, and thus we have constructed an MST, rooted at the sensor node closest to the origin, such that the property is true. If we combine four copies of this construction, as in Fig. 2.2 we need to establish the parent node of the three root sensors $B$, $D$, and $C$ representing the upper-left, lower-right, and upper right sub-grids respectively. For the first two the choice is forced (the sensor at $B$ needs to go left, and the one at $D$ needs to go down). For the third (the sensor at $C$) let us route to the left.

Now calculate the new average for the $2^{k+1} \times 2^{k+1}$ sub-grid, assuming the hypothesis holds for the $2^k \times 2^k$ ones.

We have $2(2^k)^2$ pairs included in each of the 4 smaller sub-grids, and thus have average less then $ck$, from the hypothesis. We also have $2^{k+2}$ new pairs (the ones spanning the white lines) that have collision time bounded by $2^{k+2}$. Thus we obtain a new average collision
time of: \( \frac{8ck(2^k)^2 + (2^k+2)^2}{2(2^k+1)^2} \leq ck + 2 \leq c(k + 1) \) as long as \( c > 2 \).

The base case is trivial.

### 2.6 Conclusions and Future Work

In this chapter, we have argued that there exists a routing tree which is a constant factor approximation (in expectation) to the optimum aggregation tree simultaneously for all correlation parameters \( k \). We present two constructions and prove that they obtain a constant approximation factor. Our result has important consequences – it obviates the need for specialized routing structures at least for the class of aggregation functions considered in this chapter. This is convenient, since such specialized routing structures are hard to build without some a priori knowledge about correlations in the data.

There are several possible future research directions that this work leads to. It would be interesting to study the behavior of our randomized algorithm for non-grid topologies (for example on a random graph), or for the grid-topology model with generalized connectivity assumption, in which nodes have a larger number of neighbors. Another research direction would be to extend the aggregation model, either by defining a more general framework, or by analyzing the range of aggregation functions that can be obtained by combining the already defined functions.
Chapter 3

Compact Routing

3.1 Introduction

Routing messages is a central functionality of a network. For the efficient use of network resources (link capacities, etc.) we want to build schemes which route along paths that are as short as possible. This is measured by the stretch factor, which is the maximum ratio between the length of the path traversed by a message and the length of the shortest path between its source and its destination. On the other hand, as networks grow in size it becomes important to reduce the amount of memory that needs to be maintained at every node for routing purposes.

There is an obvious trade-off between the storage requirement and the stretch factor. On the one hand you can have shortest path routing (i.e. with a stretch factor of 1) while maintaining $O(n \log(n))$ tables at each node (one entry for every other node in the network for a total storage of $O(n^2 \log(n))$). In that scenario upon receiving a message, a node can just look up the entry corresponding to the destination and forward along the shortest path. At the other extreme, you can maintain only an $O(\log(n))$ identifier at each node, and send messages by “flooding” the entire network, in a depth-first search manner for example, with a worst-case stretch factor equal to $n$ for general graphs. Neither of these solutions scales well.

For special types of graphs (e.g. trees [65], [58], outerplanar and planar networks [29], [6], growth-bounded networks [29] [7]) there are routing schemes which achieve optimal
or near-optimal stretch with sub-linear memory requirements. Except in the case of special graphs mentioned above, routing schemes with $o(n)$ memory requirement and stretch values less than 3 have been particularly hard to achieve in general graphs. In fact, lower-bound results indicate that universal routing schemes cannot achieve stretch less than 3 with $o(n)$ memory at each node [30].

### 3.1.1 Our Results

In this chapter, we study how a landmark-based routing approach can lead to achieving stretch values less than 3 with high probability while maintaining $o(n)$ memory requirements for Internet-like graphs and other large distributed networks. We also study the complementary problem: given a maximum stretch value $s$, find out the minimum routing table necessary to achieve this stretch value.

In Section 3.2, we present the details of the routing scheme we consider. It is similar to the Thorup and Zwick [65] stretch-3 universal (i.e. for all graphs) routing scheme. The main idea is to select a set of landmarks that act as a post-office/forwarding center for nearby nodes.

In Section 3.3, we show that with high probability such a routing scheme can achieve stretch less than 3 with $o(n)$ space requirement in Bernoulli random graphs. Our method relies on neighborhood results extended from the work of Chung and Lu [19] for Bernoulli random graphs. We show that stretch $s = 2$ can be achieved with $\tilde{O}(n^{3/4})$ memory at each node in Bernoulli random graphs (as opposed to $\Omega(n)$ memory for general graphs. We hope our results for Bernoulli random graphs can be extended to power-law graphs via a similar neighborhood expansion analysis.

In Section 3.5, we present and analyze a simple compact routing scheme that achieves an approximately optimal stretch-space tradeoff. The core of our scheme is a simple (and efficient) greedy algorithm for landmark selection. Our scheme takes a desired stretch $s$ and a budget $L$ on the number of landmarks as input, and produces a set of at most $O(L \log n)$ landmarks that achieve stretch $s$. Our scheme produces routing tables that use no more total space than the optimum landmark-based scheme for achieving stretch $s$ with $L$ landmarks, ignoring the space used to route to the landmarks. This may be a valuable tool for obtaining
near-optimum stretch-space tradeoffs for specific graphs.

In Section 3.6, we perform simulations of our routing schemes on Bernoulli random graphs, on power law graphs, and on the Internet AS graph. We observe that in practice stretch less than 3 can be achieved with $o(n)$ memory requirement at each node. We also explore a highest-degree landmark selection heuristic as an alternative to the random landmark selection explored in Section 3.3.

### 3.1.2 Related Work

Much of the work in compact routing has been incremental, and some of the initial algorithms and ideas proved to be so flexible as to allow subsequent layers of requirements to be added to the existing framework. One such example is the first $o(n)$ algorithm of stretch 3, proposed by Cowen [20], which required $\tilde{O}(n^{2/3})$ maximum space at every node. This algorithm is based on a very natural idea: to designate a set of “landmarks” which cover all other nodes in the network. The non-landmark nodes are then re-labeled to include the name of the closest landmark. Each node needs to remember routing information regarding its immediate neighbors and the landmarks. Expanding on this idea, Thorup and Zwick [64, 65] created a scheme that achieves $\tilde{O}(\sqrt{n})$ memory for stretch 3, matching the lower bound for general graphs (up to $\log$-factors). Furthermore they provide a generalized scheme that achieves $2k - 1$ stretch while using $\tilde{O}(n^{1/k})$ bits of memory at each node, for any integer $k \geq 2$. They do not explore stretches less than 3, since lower bounds indicate it cannot be achieved in universal compact routing schemes.

To the best of our knowledge there are no lower-bounds for random graphs and power-law graphs known other than those of universal routing schemes for which the following is known: in order to use $o(n)$ memory at each node (or even $o(n^2)$ total memory), we need to route with a stretch factor of at least 3, as shown by Gavoille and Gengler [30]. More generally a girth conjecture of Erdos and others, implies that $\Omega(n^{1+1/k})$ bits of total storage are needed to give distances with stretch strictly less than $2k + 1$, for any integer $k \geq 1$. This conjecture is proved for $k = 1, 2, 3, 5$ as documented in [53]. We should note that these lower bounds are proved under strict naming constraints (names are usually restricted to $\log n$ size for a network with $n$ nodes), while the algorithms we consider allow for larger
name (label) size.

Previous empirical studies document that simple routing scheme have excellent performance on Internet-like graphs [44]. Under certain assumptions regarding the distance distribution in these graphs, Krioukov et. al [44] show an average stretch of 1.1 and table size of 50 for the AS-graph which contains about 1000 nodes for the Thorup and Zwick (TZ) scheme. The authors note that the current degree distribution observed for the AS-graph, as well as the standard deviation of this distribution lead to close to optimum results for average stretch, and make them question the “existence of a certain link between the Internet topology and the analytical structure of the average TZ stretch function.” [44] In contrast to this work our chapter focuses on the study of the maximum, rather than the average stretch value. We also study methods that allow us to find out the minimum routing table necessary to achieve a given stretch value $s$.

We will assume, as in [20], [65], [64], [24] that the designer of the scheme is allowed to assign a poly-logarithmic name to each node (labeled model). Name-independent schemes [13], [14], [7], [4], are inherently harder than labeled schemes, but can many times be derived from the corresponding labeled scheme by using a low-stretch lookup service before routing on the low-stretch path.

### 3.2 Model and Definitions

In this section we formalize our network and routing model, and introduce some of the basic notation that will be used in later sections.

We view a communication network as a symmetric, weighted, finite graph $G = (V, E, \delta)$, $|V| = n$, the nodes representing computers/processors, the edges - bidirectional communication links, and $\delta$ - the edge weight function. Each node $v$ is assigned a unique identifier, and can have arbitrary degree. The weights on the edges induce a distance between any two nodes $u, v$, given by the sum of the edge weights on the shortest path connecting them, and which we will denote by $d_G(u, v)$ or $d(u, v)$. Note that Bernoulli random graphs are un-weighted, and for these, the distance becomes the hop distance between two nodes.

A routing scheme $RS$ is a distributed algorithm for message delivery between any two nodes in the network. A name dependent routing scheme consists of a distributed data
structure, a delivery protocol, and a routing label for every node in the graph. The message is delivered via a sequence of transmission determined uniquely by the distributed data structure (with no centralized control and no randomization).

The length of the path traversed by a message from \( u \) to \( v \) according to the routing scheme \( R \) is denoted by \( d_R(u, v) \). We can now formally define the stretch factor of the scheme \( R \) as \( \max_{u,v} \frac{d_R(u,v)}{d(u,v)} \).

### 3.2.1 Landmark-Based Routing Scheme

The routing scheme we study in this chapter is based on maintaining information about a set of landmarks as well as the neighboring nodes. There are four aspects we need to make explicit: the landmark selection procedure, the storage requirement, the labeling, and the actual routing.

Since the scheme is supposed to bound the necessary memory requirements at all nodes, for any possible graph, a fundamental realization is that neighborhoods need to be defined in terms of volume as well as radius. Thus, we define and denote by \( B_t(v) \) the set of the \( t \) nodes closest to \( v \), breaking ties by increasing node identities. We use \( N_i(v) \) to denote the set of nodes at distance at most \( i \) from \( v \) (in number of hops), and \( \Gamma_i(v) \) to denote those nodes at distance exactly equal to \( i \) from \( v \).

As in the Cowen [20] scheme, we opt for picking the landmark set \( LS \) at random, in a single stage. To ensure that with high probability any node \( v \notin LS \) has a landmark in the set \( B_v(t) \), we recall the following known result:

**Lemma 3.2.1** [Awerbuch et al. [14]] Let \( L \) be a set of landmarks produced by marking a node as a landmark with probability \( (cn/t) \log n \) (where \( c \geq 2 \) is a fixed constant). Then with probability at least \( p = 1 - \frac{1}{ne^{c-1}} \), every node \( v \notin LS \) has a landmark in the set \( B_t(v) \) and also \( |L| \leq \frac{2cn \ln n}{t} \).

We denote by \( l_v \) the landmark assigned to \( v \), and by \( r_v \) the distance (number of hops for the Bernoulli analysis) from \( v \) to its landmark. As in the Cowen scheme, each non-landmark node has label \((v, l_v, p_{l_v})\), where \( p_{l_v} \) is the exit port from the landmark \( l_v \) towards \( v \) along the shortest path. To route from a node \( u \) to another node \( v \), \( u \) first checks if \( v \) is in
its neighborhood/routing table. If so, \( u \) has the next hop info for \( v \) and uses it, otherwise, \( u \) has the next hop info for \( l_v \) and uses that. The scheme ensures that all nodes on the shortest path from \( l_v \) to \( v \) contain \( v \) in their routing table. Note that \( l_v \) does not need to remember \( v \), since the routing information from \( l_v \) is contained in the label.

The storage requirement has three components. First, every node needs to maintain the next hop information for the landmarks. The second routing component, which we will denote by \( RRT_1 \) serves to route from a landmark \( l_v \) to the node \( v \): any node \( u \) that is in the path from \( l_v \) to \( v \) needs to remember the next hop info for \( v \). This component is necessary because the volume based neighborhoods are non-symmetrical. In previous schemes \([20, 65]\) remembering the neighborhood ensures \( RRT_1 \) is covered. We make the assumption that if a node \( w \) is on the shortest path from \( l_v \) to \( v \), then \( w \) needs to remember \( v \).\(^1\) This is what we call a landmark-based routing scheme.

Finally, in order to achieve a given stretch \( s \), every node \( u \) for which \( sd(u, v) < d(u, l_v) + d(l_v, v) \) needs to remember the next-hop info for \( v \). We denote this storage requirement by \( RRT_2 \), and this will be the focus of our space requirement analysis. It is not necessary that all nodes in the path from \( u \) to \( v \) remember the next hop info. If \( u' \) satisfies \( sd(u', v) > d(u', l_v) + d(l_v, v) \), then \( \frac{d(u, u') + d(u', l_v) + d(l_v, v)}{d(u, u') + d(u', v)} < s \), so the resulting path satisfies the stretch requirement.

### 3.3 Stretches Less than 3 in Bernoulli Random Graphs

In this section, we will show that stretches between 1 and 3, can be achieved with high probability in a large family of random graphs. The graphs we are considering here are the Bernoulli random graphs denoted by \( G(n, p) \). The parameter \( n \) represents the number of nodes of a graph in this class. The parameter \( p \) is the probability that an edge exists between any two nodes in the graph (an independent coin toss for each edge).

We will study the radius of the volume based neighborhoods \( R_t(v) = \max\{d(v, u) | u \in B_t(v)\} \), as well as distance based neighborhoods \( N_i(v) = \{u|d(u, v) \leq i\} \) and \( \Gamma_i(v) = \{u|d(u, v) = i\} \). We will show the following:

\(^1\)when multiple shortest paths exist, we break ties consistently, for example by smallest node id from \( v \) to \( l_v \).
**Theorem 3.3.1 (Achieving Stretch 2 in Random Graphs)** Assume \(2c_0 \cdot n^{3/4} \log n\) landmarks are selected, and assume the labels and next-hop info for all \(n^{3/4}\) closest neighbors are remembered at each node. Also suppose \(p > \frac{2 \log n}{n}\) and \(np = o(n^{1/9})\). Then with probability at least \(1 - o(n^{-1})\), we have a stretch of at most 2.

The assumption \(np > 2 \log n\) ensures the average degree is above the connectivity threshold, so that the graph is connected, while the assumption \(np = o(n^{1/9})\) excludes very high density graphs.

To prove our result we will use a stronger version of the results regarding neighborhood size from Chung and Lu [19] which we state below. We provide the proof of the next two lemmas in Section 3.4.

**Lemma 3.3.2 (Upper-Bound Lemma)** Suppose \(p > \frac{c \log n}{n}\) for a constant \(c \leq 2\). Then with probability at least \(1 - o(n^{-2})\), we have

\[
|\Gamma_i(x)| \leq \frac{11}{c} (np)^i \quad \forall 1 \leq i \leq n 
\]

\[
|N_i(x)| \leq \frac{13}{c} (np)^i \quad \forall 1 \leq i \leq n
\]

**Lemma 3.3.3 (Lower-Bound Lemma)** Suppose \(p > \frac{c \log n}{n}\) for a constant \(1 \leq c \leq 2\) and suppose that \(np \leq n^{1/6}\). Then, for each vertex \(x\) and for each \(i\) satisfying \(i_0 \leq i \leq \frac{2n}{3 \log(np)}\), with probability at least \(1 - o(n^{-2})\), we have:

\[
|\Gamma_i(x)| \geq \frac{5}{c} (np)^{i-i_0}
\]

where \(i_0\) satisfies \(i_0 \leq \left\lfloor \frac{2}{c} \right\rfloor + 1\).

**Proof of Theorem 3.3.1:**

To show the desired stretch result, it is enough to bound the ratio \(\frac{r}{R}\) by \(\frac{1}{2}\), where \(r\) is an upper bound of the distance from a node \(x\) to its landmark \(l_x\) (as defined in Section 3.2.1) and \(R\) is a lower-bound on the radius of \(B_{n^{3/4}}(x)\) for any node \(x\), i.e. \(R = \min_x R_{n^{3/4}}(x)\). To obtain a bound on the ratio, we first bound \(r\) and \(R\) individually.
First, by choosing \( c_0 = 3 \) in Lemma 3.2.1, we get that with probability at least \( 1 - o(n^{-2}) \) every node \( x \) has a landmark within its closest \( n^{1/4} \) neighbors. We will now use this to bound \( r_x \), the largest distance from \( x \) to its landmark, using the fact that the sum of all rings at distance \( 1, 2, \ldots, r_x \) from a node \( x \) is at most \( n^{1/4} \) with high probability. This result is true for every node \( x \), thus using the union bound we get that with probability at least \( 1 - o(n^{-1}) \) the following is true for \( r \) (the upper bound on all distances):

\[
n^{1/4} \geq |\Gamma_1(x)| + |\Gamma_2(x)| + \ldots + |\Gamma_r(x)|
\]

Now, since the result in Lemma 3.3.3 holds for a given node \( x \) with probability at least \( 1 - o(n^{-2}) \), then with probability \( o(n^{-1}) \) it will hold for all \( x \), so we can lower bound \( |\Gamma_i(x)| \) as long as \( i_0 \leq i \leq \frac{2}{3} \frac{n}{\log(np)} \). If \( 1 \leq i < i_0 \) we can use the trivial lower bound of 1. [Note: we check in the end that indeed \( r \leq \frac{2}{3} \frac{n}{\log(np)} \) to be able to ensure that the use of the lemma is legal.] Now, using Lemma 3.3.3 we get:

\[
\sum_{i=1}^{r} |\Gamma_i(x)| \geq i_0 + \frac{5}{c} \left( (np)^0 + (np)^1 + \ldots + (np)^{r-i_0} \right)
\]

This result will give us a way to bound \( r \). We have:

\[
n^{1/4} \geq i_0 + \frac{5}{c} \left( \frac{(np)^{r+1-i_0} - 1}{(np) - 1} \right)
\]

Taking the logarithm of both sides, we get:

\[
\log(n^{1/4} - i_0) \geq \log 5 - \log c + \log((np)^{r+1-i_0} - 1) - \log((np) - 1)
\]

Since \( \log \) is an increasing function and \( i_0 \geq 0 \) the above equation implies that:

\[
\frac{1}{4} \log n \geq \log 5 - \log c + \log((np)^{r+1-i_0} - 1) - \log(np)
\]

Since \( \log(x - 1) \geq \log x - c_1 \), for any constant \( c_1 > 1 \) and \( x \) large enough, we get:
\[ \frac{1}{4} \log n - \log 5 + \log c + \log(np) + c_1 > (r + 1 - i_0) \log(np) \]

And thus, \( r < i_0 + \frac{\frac{1}{4} \log n - \log 5 + \log c + c_1}{\log(np)} \) (note the necessary bound on \( r \) mentioned in the beginning of our proof does hold for large enough values of \( n \)).

Similarly, we can bound \( R \) using Lemma 3.3.2. First, recall that we only consider neighborhoods up to size \( n^{3/4} \), thus we need to consider at most an \( R \) s.t.:

\[ |\Gamma_1(x)| + |\Gamma_2(x)| + \ldots + |\Gamma_R(x)| \geq n^{3/4} \]

or in other words \( |N_R(x)| \geq n^{3/4} \). From Lemma 3.3.2 with probability at least \( 1 - o(n^{-2}) \):

\[ |N_R(x)| \leq \frac{10}{c} (np)^R \]

for a given \( x \), and using the union bound, this result holds in fact for all \( x \) with probability \( 1 - o(n^{-1}) \). Thus we get the following bound on \( R \):

\[ R \log(np) \geq \frac{3}{4} \log n - \log 10 + \log c \]

We want to show that \( 2r \leq R \). It is enough then to argue (based on the bounds on \( r \) and \( R \)) that:

\[ 2(i_0 \log(np) + \frac{1}{4} \log n - \log 5 + \log c + c_1) \leq \frac{3}{4} \log n - \log 10 + \log c \]

\[ \leq \frac{1}{4} \log n + 2 \log 5 \]

The only contributions on the LHS that does not tend to zero as \( n \to \infty \) is the first term. Using the hypothesis assumption that \( np = o(n^{1/9}) \) and \( c = 2 \) we can bound the \( \frac{2i_0 \log(np)}{\log n} \) term by \( \frac{2}{9} \). Since this is clearly less than \( \frac{1}{4} \), and given that the other terms are negligible, we conclude that the above inequality holds.

It remains to show why this bound on \( \frac{r}{R} \) induces a stretch of at most 2. Let \( x \) be the destination node, and \( y \) the source node. If the routing from \( x \) to \( y \) is done via a
non-direct path then $D = d_G(x, y) > R$ (otherwise this destination node would be in the neighborhood of $x$). The routing cost $d_R(x, y)$ of the non-direct path is bounded by $r + (r + D)$, where $r$ represents the bound from the destination node $y$ to $l_y$, and $r + D$ represents a bound from $x$ to the landmark $l_y$ (using triangle inequality). Thus the stretch cannot exceed $\frac{r + r + D}{D} \leq \frac{R + D}{D} \leq 2$.

Note that the argument regarding the inter-dependence of the stretch $s$ and the ratio $\frac{r}{R}$ exposed in the proof of Theorem 3.3.1 can be expanded. In general, in order to guarantee a stretch value $s$ we need to have $\frac{r}{R} \leq \frac{s-1}{2}$. This ratio gets arbitrarily close to 0 as $s$ approaches the optimum stretch 1.

Since in Lemma 3.3.3 $i_0$ is a constant, $\frac{r}{R}$ is bounded away from 0 as long as $\frac{\log(np)}{\log n}$ is bounded away from 0 and thus we cannot guarantee stretch values arbitrarily close to 1 without imposing strict restrictions on how high the average degree $np$ can be. In general we can show the following tradeoff:

**Theorem 3.3.4** Suppose $np \geq 2 \log n$. If $np = o(n^\tilde{c})$ then we can achieve stretches $s > 1 + 2\tilde{c}$ while storing $\tilde{O}(n^x)$ bits at each node, where $x = \frac{2}{x+1} + \epsilon(\tilde{c})$, where $\epsilon(\tilde{c}) \rightarrow 0$ as $\tilde{c} \rightarrow 0$

The proof is similar with the proof in the previous section. Notice the implications for the memory requirement, which increases steeply as $s \rightarrow 1$, going above $n$ for $s \approx 1.2$ for $n = 100000$. For $s \in [1.5, 3]$, however, the memory requirement is quite scalable. Note also that the upper bound on the average degree must decrease as $s \rightarrow 1$.

### 3.4 Technical Details for Lemmas 3.3.2 and 3.3.3

For our generalized results we need the following lemma (obtained via application of Chernoff bounds) which we quote from Chung and Lu [19]:

**Lemma 3.4.1 (Binomial Distribution)** $B(t, p)$ denotes the binomial distribution with probability $p$ in a space of size $t$.

- Suppose $X$ dominates $B(t, p)$. For $a > 0$, we have $\Pr(X < tp - a) \leq e^{-\frac{a^2}{2tp}}$. 

Suppose $X$ is dominated by $B(t, p)$. For $a > 0$, we have $\Pr(X > tp + a) \leq e^{-\frac{a^2}{2tp} + \frac{a^3}{(tp)^2}}$

### 3.4.1 Proof idea for Lemma 3.3.2

Since we closely follow the approach in [19] for this proof we only mention the change in a parameter needed to show the stronger probability bound necessary for proving Theorem 3.3.1.

In the original proof the authors define a parameter $\lambda$ which they chose to be equal to $\sqrt{3 \log n}$ to obtain the probability of $1 - o(n^{-1})$. We pick instead $\lambda' = \sqrt{7 \log n}$ we can get a probability of $1 - o(n^{-2})$, and modify the constants in the statement of Lemma 3.3.2 appropriately for the results to hold.

### 3.4.2 Proof of Lemma 3.3.3 [Modified from [19]]

For proving Lemma 3.3.3 more changes need to me made to the original proof in [19], so we present the complete proof here.

**Proof:** We first prove the following statement:

**Claim:** With probability at least $1 - o(n^{-2})$ there exists a $i_0 \leq \left\lfloor \frac{2}{c} \right\rfloor + 1$ satisfying $|\Gamma_{i_0}(x)| \geq d$, where $d = \frac{20}{c}$.

Let $k = \left\lfloor \frac{2}{c} \right\rfloor$. Since $x$ is in the giant component, $|\Gamma_k(x)| \geq 1$. There exists a path $xx_1 \ldots x_k$ satisfying $x_j \in \Gamma_j(x)$ for $1 \leq j \leq k$. We write $x_0 = x$. Let $f(x_j)$ denote the number of vertices $y$, for which $x_jy$ forms an edge but $y$ is not one of those vertices $x_0, x_1, \ldots, x_k$. We compute the probability that $f(x_j) \leq d$ as follows:

$$
\Pr[f(x_j) \leq d] = \sum_{i=0}^{d} C_{n-k-1}^{d-i} (1-p)^{n-i} \\
\leq \sum_{i=0}^{d} \frac{(np)^i}{i!} e^{-(n-l-k-1)p} \\
\leq (np)^d e^{-(n-d-k-1)p} \sum_{i=0}^{d} \frac{1}{i!} \\
\leq (c \log n)^d e^{-c(1-\frac{d+k+1}{n}) \log n} \\
= o(n^{-c+\epsilon})
$$
Here, \( f(x_j) \) are independent random variables. The probability that \( f(x_j) \leq d \) for all \( j \leq k \) is at most \( o((n^{-\epsilon})^{k+1}) = o(n^{-2}) \) if \( \epsilon \) is small enough. Therefore, with probability at least \( 1 - o(n^{-2}) \), there is an index \( 1 \leq i_0 \leq k + 1 \) satisfying \( f(x_{i_0+1}) \geq d \). Hence, \( |\Gamma_{i_0}| \geq d \) (we can be shown by contradiction that the nodes represented by \( f(x_{i_0+1}) \) cannot have appeared in layers other than \( \Gamma_{i_0}, \Gamma_{i_0+1} \) or \( \Gamma_{i_0-1} \)).

From Lemma 3.3.2 it follows that \( |N_i(x)| \leq n^{\frac{2}{3}} \) with probability \( 1 - o(n^{-2}) \) for all \( 1 \leq i \leq \frac{\log n}{\log(np)} \).

Next, we need to prove the following claim for \( i = i_0 + 1 \):

**Claim:** Let us denote by \( I \) the inequality \( |\Gamma_i(x)| \leq \frac{1}{8}|\Gamma_{i_0}(x)|(n - |N_{i_0}(x)|)p \). Assuming \( pn \leq n^{1/6} \) the following holds:

\[
Pr[I] = e^{-|\Gamma_{i_0}(x)|(n - |N_{i_0}(x)|)p/8} \\
\leq e^{-dc(1-n^{-1/4}) \log n/8} \\
= o(n^{-dc/9}) = o(n^{-2})
\]

The first inequality holds using Chernoff bounds with \( \beta = 1/2 \), and noting that the probability a node not in \( N_{i_0} \) becomes attached to a node in \( \Gamma_{i_0} \) is at most \( 1/2|\Gamma_{i_0}|p \). Using \( d = \frac{20}{c} \) we obtain a probability at most \( o(n^{-2}) \) that \( |\Gamma_i(x)| \leq \frac{1}{8}|\Gamma_{i_0}(x)|(n - |N_{i_0}(x)|)p \) [or with \( d \geq \frac{10}{c} \)] that \( |\Gamma_i(x)| \leq \frac{1}{8}|\Gamma_{i_0}(x)|(n - |N_{i_0}(x)|)p \).

Combining our two claims, with probability at least \( 1 - o(n^{-2}) \) we have: \( |\Gamma_{i_0+1}(x)| \geq \frac{1}{2}|\Gamma_{i_0}(x)|(n - |N_{i_0}(x)|)p \geq \frac{1}{3}dnp. \) The \( \frac{1}{3} \) becomes \( \frac{1}{3} \) to compensate for the small difference that is introduced by the \( o(n) \) contribution of \( N_{i_0}(x) \).

A similar argument can be made for \( i = i_0 + 2 \). In [19] the proof is based on the claim that \( |\Gamma_i(x)| \) dominates a random variable with distribution \( B(t,p) \) where \( t = |\Gamma_{i_0+1}(x)|(n - |N_{i_0+1}(x)|) \), which in general is not true. \( |\Gamma_i(x)| \) may be smaller due to some vertices being linked to the previous layer by more than one edge. Thus \( B(t,p) \) will double count some vertices. If we add a term that generally will be larger than the double-counting error, we can fix the above statement. Let \( D_i \) represent the contribution of extra vertices. By definition \( |\Gamma_i(x)| + D_i \) dominates a random variable with distribution \( B(t,p) \).

Note that for a vertex to be double-counted two edges from \( \Gamma_{i_0+1} \) need to connect
to the same vertex outside of $N_{i_0+1}(x)$. There are $C_{|\Gamma_{i_0+1}(x)|}^2$ possible ways to pick the conflicting edges, and the probability of a conflict is $p^2$. Thus, the expected value of the number of double-counted vertices is $\leq C_{|\Gamma_{i_0+1}(x)|}^2 p^2 (n - N_{i_0+1}(x))$. Using a Chernoff bound, we can claim that w.h.p. that the total contribution of the double-counted vertices $D_i \leq 2C_{|\Gamma_{i_0+1}(x)|}^2 p^2 (n - N_{i_0+1}(x))$. More specifically, with probability at least $1 - o(1)$ we have:

$$
D_i \leq 2C_{|\Gamma_i(x)|}^2 p^2 (n - N_i(x))
\leq \left(\frac{|\Gamma_i(x)|^2}{2}\right) 2p^2 n
\leq |\Gamma_i(x)| p^2 n \left(\frac{11}{c}\right) (np)^i
\leq |\Gamma_i(x)| \frac{(np)^2}{n} \left(\frac{11}{c}\right) n^{2/3}
= \frac{11}{c} |\Gamma_i(x)| (np)^2 n^{-1/3}
$$

(3.4)

For the fourth inequality above we use the fact that $i \leq \frac{2}{3} \log n / \log(np)$ and thus $(np)^i \leq n^{2/3}$.

Therefore, with high probability we have: $Pr(|\Gamma_i(x)| + D_i < tp - \lambda \sqrt{tp}) < e^{-\frac{\lambda^2}{2}}$.

More specifically, with probability $1 - o(n^{-2}) - e^{-\frac{\lambda^2}{2}}$

$$
|\Gamma_{i_0+2}| \geq tp - \lambda \sqrt{tp} - D_{i_0+1}
\geq |\Gamma_{i_0+1}|(n - n^{3/4})p - \lambda \sqrt{|\Gamma_{i_0+1}| np} - D_{i_0+1}
\geq |\Gamma_{i_0+1}|(np)(1 - n^{-\frac{1}{4}} - \frac{11}{c} n^{-\frac{1}{6}} np - \frac{\lambda}{\sqrt{|\Gamma_{i_0+1}| np}})
\geq \frac{1}{3} d(np)^2 (1 - n^{-\frac{1}{4}} - \frac{11}{c} n^{-\frac{1}{6}} np - \frac{3\lambda}{\sqrt{(np)^2}})
\geq \frac{1}{3} d(np)^2 (1 - n^{-\frac{1}{4}} - n^{-\frac{1}{6}} - \frac{3\lambda}{\sqrt{(np)^2}})
$$


In the above we use the upper bound lemma for $|\Gamma_{i_0+1}(x)|$ and we omitted the parameter $x$ since it is understood from context.

By induction on $i \geq i_0 + 2$, we can show that with probability at least $1 - o(n^{-1}) - 2ie^{-\frac{\lambda^2}{2}}$ [the 2 factor in front of the exponential term accounts also for the probability error introduced by the $D_i$ term] the following holds:

$$|\Gamma_i(x)| \geq \frac{d}{3} (np)^{i-i_0} \prod_{j=2}^{i-i_0} \left(1 - n^{-1/4} - n^{-1/7} - \frac{3\lambda}{\sqrt{(np)^2}}\right)$$

The base case $i = i_0 + 2$ is already done. For the inductive hypothesis, assume the statement holds for $|\Gamma_i|$. We use again the corrected argument that $|\Gamma_{i+1}| + D_i$ dominates $B(t,p)$ for $i_0 + 2 \leq i \leq \frac{2}{3} \log n / \log(np)$. We get, following the same reasoning as in the $i = i_0 + 2$ case above:

$$|\Gamma_{i+1}| \geq tp - \lambda \sqrt{tp} - D_i$$

$$\geq |\Gamma_i|(n - n^{3/4})p - \lambda \sqrt{|\Gamma_i|np} - D_i$$

$$\geq |\Gamma_i|(np - n^{3/4}p - \lambda \sqrt{|\Gamma_i|np}) - D_i$$

$$\geq |\Gamma_i|np(1 - n^{-1/4} - n^{-1/7} - \frac{\lambda}{\sqrt{|\Gamma_i|np}})$$

$$\geq |\Gamma_i|np(1 - n^{-1/4} - an^{-3/4} - \frac{3\lambda}{\sqrt{(np)^2}})$$

The last inequality holds by applying the induction hypothesis, bounding $|\Gamma_i(x)|$ for large enough $n$ and we omitted $x$ for brevity.

Once the induction step is completed, we can conclude the proof by picking an appropriate value $\lambda$.

Take $\lambda' = \sqrt{5 \log n}$ combined with Lemma 3.3.2. Since $i \leq \log n$, we have:

$$1 - o(n^{-2}) - (i - i_0)e^{-\lambda'^2/2} \geq 1 - o(n^{-2}) - in^{-2.5} = 1 - o(n^{-2})$$

Note that $\prod_{j=2}^{i-i_0} \left(1 - n^{-1/4} - n^{-1/7} - \frac{3\lambda}{\sqrt{(np)^2}}\right) \geq 1 - in^{-1/4} - in^{-1/7} - \sum_{j=2}^{i-i_0} \frac{3\lambda}{\sqrt{(np)^2}}$. 
Thus, we probability $1 - o(n^{-2})$ we have:

$$|\Gamma_i(x)| \geq \frac{d}{3} (np)^{i-o} (1 - in^{-1/4} - in^{-1/7} - \frac{\sum_{j=2}^{i-o} 3\lambda}{\sqrt{(np)^j}})$$

$$\geq \frac{d}{3} (np)^{i-o} (1 - in^{-1/4} - in^{-1/7} - \frac{3\lambda}{np 1 - (np)^{-1/2}})$$

$$\geq \frac{d}{3} (np)^{i-o} (1 - \frac{1}{\sqrt{\log n}})$$

$$\geq \frac{d}{4} (np)^{i-o} = \frac{5}{c} (np)^{i-o}$$

3.5 Greedy Algorithm for Landmark Selection

In this section we present a simple greedy scheme for landmark selection that takes a desired stretch $s$ and a budget $L$ on the number of landmarks as input, and produces a set of at most $O(L \log n)$ landmarks that achieve stretch $s$. Our scheme produces routing tables that use no more total space than the optimum landmark-based scheme (as defined in Section 3.2.1) for achieving stretch $s$ with $L$ landmarks. This may be a valuable tool for obtaining near-optimum stretch-space tradeoffs for specific graphs. Note that the analysis does not depend on any assumptions regarding the graph, so the algorithm can be applied to any general network topology.

Let us denote by $M_{v,l}$ the number of nodes $w$ that would need to keep $v$ in either their $RRT_1$ (i.e. $w$ is on the shortest path from $v$ to $l_v$) or $RRT_2$ (i.e. for which the stretch is above $s$) given that $l_v = l$ (i.e. $l$ is $v$’s designated landmark). Let $T_i$ be the total memory requirement at step $i$. Let $v_L$ be the best landmark of node $v$ from the set $\mathcal{L}$, i.e. the landmark that is the argument of $\min_{l \in \mathcal{L}} M_{v,l}$.

We consider the memory requirement for storing landmark information separately. We will study the remaining total memory requirement (i.e. for $RRT_1$ and $RRT_2$) of the following greedy algorithm for choosing landmarks:
**Input:** Graph G, initial landmark $l_1$, parameter $L$

**Output:** Landmark set $L$ containing $L \log n$ elements

$L \leftarrow l_1$;

$T_1 = \sum_{v \neq l_1} M_{v,l_1}$;

for $i = 2$ to $2L \log n$ do

\[ T_i = T_{i-1}; \]

foreach $x \in V - L$ do

\[ N = 0; \]

foreach $v \in V - L$ and $v \neq x$ do

\[ N = N + \min(M_{v,x}, M_{v,v_L}) \]

end

if $N < T_i$ then

\[ T_i = N; \]

\[ l_i = x; \]

end

end

$L \leftarrow l_i$;

Algorithm 1: Greedy Landmark Selection

Note: $l_1$ can also be chosen via the same Greedy method assuming we start with 0 landmarks and $T_0 = n^2$.

**Theorem 3.5.1** The above Greedy algorithm is guaranteed to produce a landmark set with memory requirement less than or equal to the total memory requirement of an optimal set of size $L$. The Greedy will select $2L \log n$ landmarks (by construction).

**Proof:** [Greedy Performance] In the following proof we will track the memory requirements of an optimal choice of $L$ landmarks (the OPT) compared to the memory requirement of our algorithm (GREEDY). In particular we will compare the number of ”entries” utilized by these solutions.

Each pair of nodes $(w, v)$ represents a potential memory ”entry”, i.e. $w$ need to remember $v$ in reverse routing table (either $RRT_1$ or $RRT_2$) unless either $v$ is a landmark or $v$’s landmark $l$ is such that $d(w, l) + d(l, v) \leq sd(w, v)$. If the entry is not needed, we will say
that a landmark covers the entry.

Lemma 3.5.2 The number of covered entries only increases during GREEDY.

Proof: Although the specific entries may get covered/uncovered, this lemma is true because landmarks, once added are never deleted (thus entries covered by the first case remain covered), and a node changes landmarks only if this improves its number of covered entries (thus entries covered by the second case are exchanged with a larger set of covered entries that were not previously covered).

Now let us analyze what we can say about the magnitude of the decrease during consecutive stages of GREEDY. Let \( D_i = |T_i - \text{OPT}| \) if \( T_i > \text{OPT} \) (\( D_i = 0 \) otherwise). There are at least \( D_i \) entries that could be covered by OPT but remain uncovered by GREEDY after \( i \) landmarks are added to GREEDY. Since OPT covers these entries using \( L \) landmarks, there exists one landmark in OPT that is not currently used by GREEDY and which could cover at least \( \frac{1}{L} D_i \) new elements in GREEDY. By construction, GREEDY will pick a new landmark that covers at least this many elements in the next stage, so that \( |T_{i+1} - \text{OPT}| \leq (1 - \frac{1}{L}) D_i \).

This argument holds for any of the \( L \log n \) stages. Also initially \( D_1 = |T_1 - \text{OPT}| \leq n^2 \), thus at the end of the algorithm:

\[
D_{2L \log n} \leq \left(1 - \frac{1}{L}\right)^{2L \log n} D_1 < 1
\]

Since the difference can only be a non-negative integer, the above shows that after \( 2L \log n \) landmark are added GREEDY covers at least as many elements as OPT.

Note that although the above greedy routine ensures the \( RRT_1 \) and \( RRT_2 \) requirements are no more than those of the landmark-based optimal scheme, we will be using \( O(\log n) \) more landmarks.

### 3.6 Simulations and Real Network Data Results

In this section, we present simulation results to verify the theoretical studies shown in previous sections and extensions to power-law graphs and real network graphs. We focus
on the scalability of routing table size ($RRT_2$ in particular) as the total number of nodes increases while keeping the maximum stretch to less than or equal to 2.

### 3.6.1 Random Graphs

Random graphs are generated based on [15]. With a total of $n$ nodes, each pair of nodes has a probability of $p$ that these two nodes are directly connected by an edge. We explored several schemes of landmark construction based on random graphs.

#### Scheme 1: Random Landmark Selection

In this scheme, landmarks are randomly picked among the nodes as in Section 3.3. The simulation constitutes the following steps.

1. Randomly select $n^{2/3}$ nodes as landmarks in a random graph with a total of $n$ nodes;
2. For each node $u$, choose the landmark that is the closest to this node $L_u$;
3. For each source node $u$, route to all destination nodes $v$ by first going to $v$’s landmark node $L_v$ and then from $L_v$ to $v$. If $d(u, L_v) + d(L_v, v) > 2d(u, v)$, i.e., stretch between $u$ and $v$ is greater than 2, $v$ is placed in $u$’s routing table;
4. Find the maximum routing table size among all nodes.

Figure 3.1 shows that when the number of landmarks increases as $n^{2/3}$, the maximum $RRT_2$ size also increases as $n^{2/3}$, consistent with the theoretical studies laid out in Section 3.3.

#### Scheme 2: Greedy Landmark Selection

The Greedy Landmark Selection algorithm iteratively picks each landmark that minimizes the RRT size. When applied to random graphs, we found that the RRT size is much reduced compared to random selection of landmarks, $n^{1/2}$ rather than $n^{2/3}$. The results on Greedy Landmark Selection are summarized in Figure 3.2.

The smaller RRT size from Greedy Landmark Selection demonstrates that Greedy algorithm is more efficient in selecting landmarks. Analysis of the landmarks chosen from

---

2In this section RRT refers to $RRT_2$.
Figure 3.1: Random Landmark Selection scheme on random graphs: max RRT scales as \( n^{2/3} \) using random selection of \( n^{2/3} \) landmarks.

Figure 3.2: Greedy Landmark Selection on random graphs: ave RRT scales as \( n^{1/2} \) using greedy selection of \( n^{2/3} \) landmarks.
Figure 3.3: Histogram of node degree and the degree distribution of landmarks chosen by Greedy Landmark Selection scheme on a random graph with 4000 nodes total and 252 landmarks.

This scheme show that most of the landmarks chosen are high degree nodes, as seen in Figure 3.3. This result suggests that an efficient choice of landmarks is by picking those nodes with higher degrees than others.

**Scheme 3: Highest-degree Landmark Selection**

The simulation results based on Greedy Landmark Selection prompted us to experiment with a heuristic approach – picking the landmarks based on the degree of the nodes:

1. Rank the degree of all $n$ nodes and pick $n^{2/3}$ of the nodes with the highest degrees as landmarks.

Steps 2-4 are the same as those in the Random Landmark Selection. This scheme is intuitive and easy to implement. Our simulation results, shown in Figure 3.4, show that the average RRT scale as $n^{1/2}$, almost identical to the results based on Greedy Landmark Selection.
CHAPTER 3. COMPACT ROUTING

3.6.2 Power-law Graphs

Previous studies have shown that real networks behave more like power-law graphs [44] (otherwise known as scale-free graphs) rather than random graphs. The degrees of the nodes follow power-law distributions with a small fraction of the nodes having a high degree of connections with others. It is important to analyze the scaling behavior of our technique on these networks.

The power-law graphs are generated according to [11] where several seed nodes are all connected to each other. With the addition of each new node, it is connected to one or a few of the existing nodes with a probability that is proportional to the degree of the existing nodes.

Because of the heavy-tailed nature of the power-law graphs, it is natural to apply the scheme with Highest-degree Landmark Selection, which has been shown above to generate near-optimum landmarks for random graphs. The simulation results plotted in Figure 3.5 show that the scaling in power-law graphs is much more favorable than in random graphs.
graphs. Even when the number of landmarks is reduced to $n^{1/2}$ (compared to $n^{2/3}$ in random graphs), the maximum RRT size is also limited to $n^{1/2}$. Note that stretch is also limited to 2 in these simulations on power-law graphs.

### 3.6.3 AS Graph from Real Network Data

To verify our results on real network topologies, we carried out simulations on the current AS graph acquired from CAIDA [2] with a total of 20906 number of nodes. By applying the Highest-degree Landmark Selection to the AS graph, we found that results match very well with our simulations on power-law graphs (see Figure 3.5).

![Figure 3.5: Highest-degree Landmark Selection on power-law graphs and AS graph from real network: max RRT scales as $n^{1/2}$ using $n^{1/2}$ of highest-degree nodes as landmarks.](image)

These simulation results on the AS graph are quite significant. The existing routing scheme requires that each node stores the routing information to all other nodes in the network. By switching to compact routing, each node only needs to store the routing information to a very small fraction of the nodes: the landmarks and RRTs. In this particular case with the AS graph, we found that the memory required is less than 2% of what is needed
currently. This large gain in memory is accomplished by relaxing the stretch, which is restricted to 2 with our approach. The simulation results on power-law graphs demonstrate that such a scheme is scalable to much larger network sizes.

3.7 Conclusions

In this chapter we explore ways to lower the bound of the maximum stretch to values below 3, while maintaining sublinear average and maximum routing table size. We use the stretch value 2 as an illustrative example throughout the chapter, but the results can be generalized to achieve any stretch less than 3.

Through theoretical proofs and simulations, we demonstrate the bounds for stretch and routing table size, using three landmark selection schemes on three types of graphs: random graphs, power-law graphs, and AS graph from real network.

Our results show: 1). Stretch less than 3 can be achieved with high probability in a large family of random graphs while maintaining sublinear memory size on each node. We presented the memory vs. stretch and average-degree vs. stretch tradeoffs specific to this family of graphs in Lemma 3.3.4. 2). Selecting high degree nodes as landmarks is efficient for reducing memory consumption while keeping stretch bounded by a stretch value $s < 3$. 3). Compact routing scheme applying to AS graph from real network can reduce the maximum routing table size at each node to 1.5% (0.7% on average) of the total number of nodes on the entire network and achieve maximum stretch bounded by 2.

These results are encouraging for reducing routing table size and increasing scalability of the network.
Chapter 4

Routelet Placement for Multipath Routing

4.1 Introduction

Several multipath routing and congestion control protocols (mTCP [67], Han et al. [32] Kelly et. al. [40], Key et al. [41] and Harp [43]) have been proposed recently for better network resource allocation and utilization. The key idea of these proposals is to send packets from a source to a destination through multiple Internet paths; the sending rate on each path is determined by a congestion control algorithm that ensures better network-wide bandwidth utilization and fairness of allocation. Commercial products such as Asankya [1] enable high-quality real-time content using the above idea of multiple paths for packet transport. All of these works are motivated by the observations that (1) Internet today has significant path diversity [9], i.e. given any two communicating end-hosts in the Internet, there exist potentially many independent or partially overlapping paths between them, (2) today’s routing infrastructure often limits the traversal of packets between a source and a destination to a single path, and (3) the routing underlay adapts too slowly to fluctuating network conditions such as failures and bandwidth availability, thereby causing load imbalance [10] on several paths in the Internet that makes applications run at suboptimal performance.

A key commonality in the multipath protocol proposals is the presumption that some
nodes in the Internet (such as overlay routers [54] or diversified routers [66] or stepping-stone routers [41]) can provide relay services for packets thereby allowing packets to take many alternate paths; the agents providing relay service on each node are henceforth called routelets [17]. However, few works address the problem of where in the Internet to place the routelets to be most effective for bandwidth utilization. In this chapter, we address this exact problem:

Given a graph $G$ that represents the routers and links in the Internet, a subset of the routers that are capable of hosting the routelets, and a set of sender-receiver pairs that transmit and receive data, where should we place $k$ routelets to maximize bandwidth utilization?.

A handful of works address a similar relay placement problem for providing resilience to path failures between a source and a destination [33, 57, 63], or for improving end-throughput by breaking an end-to-end TCP connection into a series of shorter-loop connections [46, 57]. The key idea in these works is to have an alternate path through a relay node that shares minimum number of links possible with the default underlay path. However, these solutions do not directly apply to bandwidth maximization because for this purpose, the overlay paths can share as many links as required, as long as the shared links do not become bottlenecks. This difference makes the placement problem for bandwidth maximization more complex.

In this chapter, we consider the routelet placement problem for bandwidth maximization under three deployment scenarios. We make three contributions. Firstly, we formalize the three scenarios of the problem, and formulate them in a constraint optimization framework. We prove that the placement problem in two of the scenarios is NP-hard. Second, we develop rounding algorithms for two scenarios and prove for one scenario that the objective (of bandwidth maximization) remains within a $\log n$ factor of the optimal fractional solution, while using no more than a factor $O(\log n)$ more routelets than the fractional solution. Finally, we show through simulations with several BRITE [50] topologies of varying node populations, node degree, and link bandwidths that our rounding algorithms perform very close to the optimal solutions. We summarize our contributions and some interesting open problems in Table 4.1.

The rest of the chapter is organized as follows. Section 4.3 describes the deployment scenarios and defines the problem statements. Section 4.5 presents the LP formulations for
the deployment scenarios. In Section 4.6, we present rounding algorithms for two scenarios and prove certain desirable properties. In Section 4.7, we present simulation results to demonstrate the efficacy of our rounding algorithms on several network topologies. Section 4.2 discusses the related work, and Section 4.8 concludes. We prove the hardness of scenario b and give the idea behind the hardness result for scenario c in Appendix.

### 4.2 Related Work

Multipath protocols have been well researched since Maxemchuk’s seminal work on dispersity routing [49] to improve throughput and resilience to path failures or packet losses [59, 22]. Striping or inverse multiplexing [8, 23, 56, 42] provides link level mechanisms for splitting input flows among multiple links to increase throughput. More recently, multipath techniques have been proposed in the context of overlay routing or multihomed clients [12, 3, 62]. Katabi et al. [38] and Elwalid et al. [25], propose splitting aggregate traffic flows along multiple paths to achieve load balancing and stability in the context of intradomain traffic engineering. Harp [43] uses multiple paths for scheduling transfers with heterogeneous requirements. Multipath TCP [32] and KV [40] are based on a utility-theoretic framework for network-wide resource allocation, and systematically address fairness. These controllers maintain a nontrivial sending rate on each available path so as to maximize utilization and global fairness.

A key requirement of all these works is the support for letting end-hosts utilize multiple paths in the Internet for transmitting packets. This can be achieved by employing “packet relayers” that are realizable in several ways. A handful of works address a similar node placement problem in the context of providing resilience to path failures between a source

<table>
<thead>
<tr>
<th>Scenario</th>
<th>LP</th>
<th>Hardness</th>
<th>Rounding</th>
<th>Rounding Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>✓</td>
<td>Open</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>b</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Open</td>
</tr>
<tr>
<td>c</td>
<td>✓</td>
<td>Open</td>
<td>Open</td>
<td>Open</td>
</tr>
</tbody>
</table>

Table 4.1: Contributions and open problems.
and a destination [33, 63, 57], and for improving end-throughput by breaking an end-to-end TCP connection into a series of shorter-loop connections [46, 57]. However, to the best of our knowledge, ours is the first work to address the placement problem in the context of maximizing network utilization.

Node placement has also been studied in other contexts such as Web server and cache placement [37, 55]. Qiu et al. [55] study a web server replica placement problem to minimize the cost for clients to access data. Jamin et al. [37] study mirror placement problem, in which mirrors can be placed only at a restricted set of locations, much like our problem constraint. Also, they show a diminishing return of placing more and more mirrors, similar to our observations.

As also argued by Key et al [41], we believe that multipath routing forms a powerful building block for a Robust Internet Architecture of the future. In this chapter, we discuss three scenarios in which such an architecture may be realized.

## 4.3 Problem Formulation

We envision that multipath routing and congestion control protocols will be realized in the Internet in three steps (and timescales). First, a subset of nodes in the Internet called relay nodes are deployed for hosting relay services. Second, for a given set of source and destination (SD) pairs, a certain number of relay service agents or routelets are placed on a subset of the relay nodes to let the SD pairs use alternate paths in the Internet. Finally, using the set of paths enabled by the routelets, multipath routing protocols determine the appropriate packet sending rates on each path between a given SD pair. While node placement is a coarse timescale operation (say days to months), routelet placement can happen at a granularity of minutes to hours, and multipath routing and rate control work at a fine granularity of round-trip times between a source and a destination.

In this chapter, our focus is on the second step: our objective is to determine the placement of routelets in the Internet such that any two communicating end-hosts can exploit path diversity by exchanging packets through the routelets. In the rest of this section, we discuss the deployment scenarios and formally define the problem statements.
4.3.1 Deployment Scenarios

We explore the routelet placement problem under three deployment scenarios (see Figure 4.1).

1. **Scenario (a)** In the scenario shown in Figure 4.1(a), both the source and the destination nodes are modified to choose alternate paths through the routelets for routing packets. In particular the source node maintains a list of routelets to which it can send packets. Packets are then forwarded to the destination by the routelets. Since using different paths for packets can lead to packet reordering, the destination performs the necessary reordering.

   In this scenario, the routelets do not maintain any state for routing packets. Packets are encapsulated by the source nodes, hence the actual destination address is present in each packet. The routelets just remove the encapsulation and forward the packets towards the destination. As a result, the functionality of the routelets is simple. The disadvantage of this approach, however, is twofold. First, the sources and destinations need to be modified, which makes the solution much harder to deploy compared to a solution that does not need modifications. Second, even though the senders are modified to include the functionality of making the packets take different routes, the actual set of routelets that are to be used between a source and a destination can keep changing with time due to traffic fluctuations. The frequency of updating each sender with the set of good routelets constantly strikes a tradeoff between inefficient use of relay resources, and the generation of unnecessary network traffic for updates.

2. **Scenario (b)** Figure 4.1(b) shows a deployment scenario that addresses one of the
major drawbacks of Scenario (a). In this scenario, the functionality of choosing different paths is offloaded to an entry routelet, thereby leaving the sender unmodified. The entry routelet can act so for several senders and hence allows for sharing information about available relay routelets. Such sharing reduces the amount of traffic generated to keep the set of good relay routelets updated. The destination, however, still requires modifications to either avoid or tolerate the effects of out-of-order packets for some applications.

3. Scenario (c) The scenario in Figure 4.1(c) obviates the need to change either the source or the destination nodes, thereby making the solution readily deployable. This scenario employs an exit routelet to which packets are sent from the relay routelets; the exit routelet reorders the packets before forwarding to the destination, thereby masking off the effects of utilizing multiple paths.

Scenarios (b) and (c) require extra functionality and also maintain state in the entry and exit routelets than Scenario (a). However, they are more attractive for deployment since they require minimum modification to the sources and destinations.

We make the following assumptions in this chapter.

- The underlay routing gives only a single default path for any pair of source and destination nodes in the network.
- Each path has a single bottleneck link.
- All the SD-pairs are known and active.

We can also extend the results to the case when a good model for the percentage of time that they are active is given, but we do not address this extension in this chapter.

4.3.2 Problem Statements

Here we will formalize the problem definitions for all of the scenarios studied, with bandwidth utilization between the SD-pairs as the optimization criterion.
1. **Scenario a** Given a graph $G$ that represents the nodes and links in the Internet, a subset of the nodes that are capable of hosting the routelets, and a set of SD-pairs, where should we place at most $k$ routelets to maximize the bandwidth utilization between the SD-pairs, assuming the paths an SD-pair can utilize to send packets are either a default path or a Source-Relay-Destination (S-R-D) path?

2. **Scenario b** The same problem as in (a), but now the paths can either be the default path or Source-Entry-Relay-Destination (S-E-R-D) paths, with the additional constraint that the entry routelet needs to be on the default path.

3. **Scenario c** The same problem as in (a), but now the paths can either be the default or Source-Entry-Relay-Exit-Destination (S-E-R-E-D) paths, with the constraint that the entry routelet needs to be on the default path, and the default path can only be used if the exit routelet is located on it.

4. **No routelets** Given a graph $G$ that represents the nodes and links in the Internet and a set of SD-pairs, what is the maximum bandwidth that can be achieved using just the default paths? (the MAX-FLOW problem)

The motivation for selecting only a specific number of routelets is due to the costs of installing and maintaining the necessary forwarding software to support our multipath scheme.

### 4.4 NP hardness Result

The problem of placing a given number of relay nodes in an optimum manner for maximizing network utilization is NP-hard for Scenarios b and c. We will explain in detail a reduction from Balanced Complete Bipartite Subgraph (BCBS) to Scenario (b).

BCBS is the problem of deciding, given a bipartite graph $G$ with vertex set $S \cup T$, edges $(s, t)$ such that $s \in S$, $t \in T$ and a parameter $b$ whether there exist a subset of nodes $S' \subseteq S$ and $T' \subseteq T$ such that $|S'| = |T'| = b$ and all such edges are edges in $G$.

We reduce this problem to the decision version of our problem, which is: *Given a graph $G$ that represents the routers and links in the Internet, a subset of the routers that*
are capable of providing the relay service, and a set of SD-pairs, can we place at most $k$ relay services as to increase the bandwidth utilization between the SD-pairs by an amount equal to $a$ (assuming the paths an SD-pair can utilize to send info is either a default path or a S-E-R-D path as in Scenario b)?

We obtain the reduction from BCBS to the above problem as follows:

Create the network by copying $G$ and adding a destination node $D$. We connect all nodes in $S$ with the destination (the dotted lines in Figure 4.2) and also all nodes in $T$ are connected with the destination (the plain lines in Figure 4.2). Every node in $S$ is a source, and the direct links represent the default SD paths. The cloud represents the edges from $G$. All links have capacity 1. Routelets can be placed at either $S$ or $T$. We ask: is it possible to place $2b$ nodes and get a $b^2$ increase in the bandwidth (so $k = 2b$ and $a = b^2$)?

Now we will show that solving the node placement question is equivalent to solving the BCBS problem. Any node assignment that distributes the nodes between the two sides, say $r$ nodes in $S$ and $l$ nodes in $T$, can add at most $rl$ capacity in terms of bandwidth. This product is maximized when $r = l = k/2 = b$, since the sum $r + l = k$ is constant, and the maximum value is $b^2$. No unbalanced distribution can lead to an extra capacity of $b^2$ or more (because the product $rl$ is strictly less than that in the unbalanced case). The balanced distribution only leads to an extra capacity of $b^2$ if the nodes selected form a complete bipartite clique. Thus, to be able to detect whether nodes can be placed to achieve this
improvement in bandwidth or not is to be able to solve the BCBS problem.

Since solving the BCBS problem is NP-hard, determining the answer to the decision version of Problem b is also NP-hard.

The reduction to Scenario (c) is similar, the only difference is that in scenario (c) we need an exit routelet on the default path. Since the default paths are a single hop, that routelet needs to be placed at the destination node \( D \) in order to allow more flow than the default. Placing the exit routelet at the source node creates no additional paths beyond the already existing default path.

4.5 LP Formulation

In this section, we formulate the routelet placement problem in each scenario as a set of linear constraints and objectives. We begin by formulating the base case: a scenario in which there are no routelets, and the total bandwidth between sources and destinations is just governed by the underlying routing. This formulation will serve as a baseline for subsequent formulations of the routelet placement problems, and also for comparing the efficacy of employing routelets for maximizing bandwidth utilization.

4.5.1 Scenario with no routelets

Let \( f_{sd} \) represent the flow (or bandwidth\(^1\)) between a source \( s \) and a destination \( d \) on the default route (chosen by the underlay routing scheme). Our objective for this scenario is to represent the total flow between all pairs of sources and destinations. Hence the total flow is represented as

\[
\sum_{sd} f_{sd} \quad \text{(4.1)}
\]

\[
\sum_{sd} f_{sd} \quad \text{(4.2)}
\]

\(^1\)In the rest of the section, we use the word flow to represent bandwidth.
Observe that the flow between \( s \) and \( d \) is restricted by the bottleneck link capacity on the default route between them. This constraint is represented as

\[
\sum_{e \in P_{sd}} f_{sd} \leq c_e, \forall e \in E
\]  (4.3)

where \( e \) represents an edge and \( P_{sd} \) represents the set of edges on the default route between \( s \) and \( d \).

Finally, the flow between any \( sd \) can not be negative, which is represented as

\[
f_{sd} \geq 0
\]  (4.4)

### 4.5.2 Scenario (a)

Our idea in this scenario is to place a relay routelet to enable packets to take alternate paths, in addition to the direct routes (or paths) chosen by the default routing scheme. We introduce the following variables.

1. \( S \) is the set of all \( sd \) pairs.
2. \( f_{sd} \) is the flow from \( s \) to \( d \) on the default route.
3. \( f_{sd,m} \) is the flow from \( s \) to \( d \) via relay routelet \( m \).
4. \( P_{sd,m} \) is the path (i.e., a set of edges) corresponding to the flow \( f_{sd,m} \).
5. \( M_k \in \{0, 1\} \) is an indicator variable which represents whether node \( k \) is selected to host a relay routelet or not.
6. \( B_{sd,k,m} \) denotes the capacity of the bottleneck link on the path \( P_{sd,k,m} \).
7. \( K \) is the number of routelets to be deployed.

The LP formulation is represented as:
The objective function in (4.5) represents the sum of bandwidths achieved on default paths as well as paths through relay routelets between all source-destination pairs. Constraint (4.6) bounds the number of routelets placed in the network. Constraint (4.7) bounds the total flow through a routelet on node \( m \) to be at most the bottleneck bandwidth on the path through \( m \), whereas constraint (4.8) restricts the total flow passing through a routelet on node \( m \) and through a given edge \( e \) to be at most the edge capacity. We multiply both of these by an indicator of whether a routelet is actually present on node \( m \). Finally (4.9) restricts the sum of flows passing through each edge to be at most the edge capacity.

### 4.5.3 Scenario (b)

In Scenario (b), some of the total routelets will be used as entry routelets. Hence, the formulation requires the following notation.

1. \( S \) is the set of all SD pairs (given).

2. \( f_{sd} \) is the flow from \( s \) to \( d \) on the default route.

3. \( f_{sd,k,m} \) is the flow from \( s \) to \( d \) redirected by the entry routelet on node \( k \) via relay routelet on node \( m \).
4. $P_{sd,k,m}$ is the path (i.e., a set of edges) corresponding to the flow $f_{sd,k,m}$.

5. $M_k \in \{0, 1\}$ is an indicator variable which represents whether node $k$ is selected to host a relay routelet or not.

6. $B_{sd,k,m}$ denotes the capacity of the bottleneck link on the path $P_{sd,k,m}$.

7. $\mathcal{K}$ is the number of routelets to be deployed.

8. $\mathcal{I}(e \in P_{sd,k,m})$ is an indicator (known apriori), which is set to 1 if edge $e$ is contained in $P_{sd,k,m}$.

9. $C_e$ represents the total capacity of edge $e$.

10. $D_{sd,m}$ is only defined if the routelet on node $m$ is on the default path from $s$ to $d$. This variable indicates if this routelet acts as “entry” point for this SD-pair.

The LP formulation for Scenario (b) is as follows.

\[
\begin{align*}
\text{max} & \quad \sum_{sd,m,k} f_{sd,m,k} + f_{sd} \\
\text{s.t.} & \quad \sum_k M_k \leq \mathcal{K} \\
& \quad \forall sd \sum_m D_{sd,m} \leq 1 \\
& \quad \forall sd, m \quad D_{sd,m} \leq M_m \\
& \quad \forall sd, k \quad f_{sd,m,k} \leq D_{sd,m} \times B_{sd,m,k} \\
& \quad \forall sd, k \quad f_{sd,m,k} \leq M_k \times B_{sd,m,k} \\
& \quad \forall e \in E \sum_{sd,m,k} \mathcal{I}(e \in P_{sd,m,k}) f_{sd,m,k} \leq C_e \\
& \quad f_{sd,m,k} \geq 0 \\
& \quad M_k \geq 0 \\
& \quad D_{sd,m} \geq 0
\end{align*}
\]
The constraint (4.13) bounds the number of routelets used. The constraints (4.16) and (4.17) allow flows to pass only via eligible paths (in which the first routelet is a valid entry routelet, and the second is a valid relay routelet). Constraint (4.18) is the edge capacity constraint. The constraint (4.15 allows for at most one entry point for the flow from $s$ to $d$ (there could be none).

4.5.4 The LP for Scenario (c)

In this scenario, besides entry routelets there are also exit routelets for non-default paths. In addition to this, the default paths may not be used if an SD-pair has a valid entry routelet and a valid exit routelet, but the exit routelet in not on the default path.

We have the following difference in notation:

1. $f_{sd,m,k,o}$ is the flow from $s$ to $d$ redirected by the entry routelet on node $m$ via relay routelet on node $k$ towards the exit routelet on node $o$ (the flow value is the variable we want to determine)

2. $P_{sd,k,m,o}$ is the path corresponding to the above flow (i.e. a known set of edges)

3. $B_{f_{sd,m,k,o}}$ is a constant which denotes the capacity of the bottleneck link on the path $P_{sd,m,k,o}$.

4. $E_{sd,m}$ indicates whether the exit routelet on node $m$ is on the default path from $s$ to $d$. 
The LP formulation for this case is as follows:

\[
\begin{align*}
\min & \sum_{sd,m,k} f_{sd,m,k} + f_{sd} \\
\text{s.t.} & \sum_k M_k \leq B \quad \forall u \\
& D_{sd,m} \leq M_m \quad \forall sd, m \\
& \sum_m E_{sd,m} \leq 1 \quad \forall sd \\
& \sum_k D_{sd,k} \leq \sum_m E_{sd,m} \quad \forall sd \\
& f_{sd,m,k,o} \leq M_k \times B_{f_{sd,m,k,o}} \quad \forall sd, k, m, o \\
& f_{sd,m,k,o} \leq E_{sd,o} \times B_{f_{sd,m,k,o}} \quad \forall sd, k, m, o \\
& f_{sd,m,k,o} \leq D_{sd,m} \times B_{f_{sd,m,k,o}} \quad \forall sd, k, m, o \\
& \sum_{e \in P_{sd,m,k,o}} f_{sd,m,k,o} \leq 1_e \quad \forall e \in E \\
& f_{sd,m,k,o}, M_k, D_{sd,m}, E_{sd,o} \geq 0
\end{align*}
\] (4.22)

In the ILP version, variables \( M_k, D_{sd,m}, E_{sd,o} \) take integer values.

We prove in the Appendix that the placement problems in Scenarios b and c are NP-hard.

### 4.6 Rounding Algorithms

In this section we will provide a method to transform the fractional LP solution into an integer one. Our method is slightly different for the different scenarios. For scenario a, the algorithm presented offers provable guarantees, as stated in Theorem 4.6.1. For scenario b, the algorithm needs to be adapted, for reasons explained in section 4.6.2. While providing theoretical bounds for the performance of this modified algorithms remains an open problem, experimentally we observe that in scenario (b) we can match the optimal throughput, with no edge constraints violations, while using at most a \( \log n \) factor more routelets.
4.6.1 Scenario (a)

Let $M_k, f_{sd,m}, f_{sd}$ be the fractional values obtained after running a standard LP solver. Let $\tilde{M}_k, \tilde{f}_{s-d,m}, \tilde{f}_{sd}$ be the corresponding (integer) rounded values obtained as follows:

- Set $\tilde{M}_k = 1$ with probability $M_k$, otherwise set $\tilde{M}_k = 0$.
- If $M_k = 0$ set $\tilde{f}_{s-d,k} = 0$. [Note that if $M_k = 0$ then $f_{s-d,k}$ is also 0 due to constraint 4.7]
- If $M_k > 0$, set, for all SD-pairs $\tilde{f}_{s-d,k} = \frac{\tilde{M}_k f_{s-d,k}}{M_k}$.
- For all SD-pairs, let $\tilde{f}_{sd} = f_{sd}$.

Note that by construction $E[\tilde{M}_k] = M_k$.

In scenarios b and c we also have entry (and exit) routelet variables (such as $D_{s-d,m}$ and $E_{s-d,m}$). The rounding for these more complex scenarios requires further research.

**Theorem 4.6.1** The above rounding technique for scenario a produces a total throughput which matches the optimal throughput (in expectation). Furthermore, with high probability the constraints 4.6, 4.9 are not violated by more than a $\log K$ factor, where $K$ is the total number of potential routelet locations. Constraints 4.7 and 4.8 are satisfied by construction.

**Proof:**

We start by proving the first part of the theorem.

When $M_k > 0$, since $f_{s-d,k}, M_k$ are constants determined by the LP, and $E[\tilde{M}_k] = M_k$, we have that $E[\tilde{f}_{s-d,k}] = \frac{E[M_k] f_{s-d,k}}{M_k} = f_{s-d,k}$. $E[\tilde{f}_{s-d,k}] = f_{s-d,k}$ is true also when $M_k = 0$ (both flows equal 0 in that case). Thus, after rounding, by linearity of expectation we have that the rounded objective function (the total throughput) matches the objective function of the LP (in expectation):

$$E \left[ \sum_{sd,m} \tilde{f}_{sd,m} + \tilde{f}_{sd} \right] = \sum_{sd,m} f_{sd,m} + f_{sd}$$
Constraint 4.6 becomes, after the rounding, a sum of $K$ Bernoulli random variables (since $\tilde{M}_k \in \{0, 1\}$) with mean $\mu = E[\sum_k \tilde{M}_k] = \sum_k M_k \leq K$. We can apply a standard Chernoff bound [18] to obtain that with high probability (i.e. with probability at least $1/K$ (as long as $K, \mu \geq 3$) the sum will not exceed the expected value by more than a $\log K$ factor. The Chernoff bound is just the first inequality in the desired result:

$$Pr \left[ \sum_k \tilde{M}_k \geq (1 + \log K)K \right] \leq e^{-(\log K)^2/2} \leq \frac{1}{K}$$

Finally, for every edge $e$, we want to bound the probability that the rounded flow on an edge (denoted by $\tilde{S}_e$ the summation term in (4.31)) exceeds the edge capacity by more than a $\log K$ factor, i.e. we want to bound:

$$Pr \left[ \left( \sum_{sd, k} I(e \in P_{sd,k}) \times \tilde{f}_{sd,k} \right) \geq (1 + \log K)C_e \right]$$

(4.31)

For a fixed $k$ such that $M_k \neq 0$ let $\tilde{F}_{k,e} = \sum sd I(e \in P_{sd,k}) \times \tilde{f}_{sd,k}$ (and define $F_{k,e}$ similarly). These represent the sum of flows via relay $k$ passing through edge $e$ after the rounding (and before the rounding). Note that $\tilde{F}_{k,e} = F_{k,e} \times \tilde{M}_k/M_k$, since $\tilde{M}_k/M_k$ is the transformation factor between the corresponding rounding and LP flow values.

After finding an LP solution, $F_{k,e} = \gamma_{k,e} \times C_e$ for some constant $\gamma_{k,e} \leq 1$ (by constraint 4.8) and such that $\sum_{k:M_k \neq 0} \gamma_{k,e} \leq 1$ (by constraint 4.9).

Using this new notation, first note that $\tilde{S}_e = \sum_{k:M_k \neq 0} \tilde{F}_{k,e}$, which can be seen by summing first over $k$ and then over $sd$ in (4.31), and by noting that the contribution of flows for which $M_k = 0$ is zero both before and after the rounding.

Thus, bounding $\tilde{S}_e/C_e$ is equivalent to bounding $\sum_{k:M_k \neq 0} \tilde{F}_{k,e}/C_e = \sum_{k:M_k \neq 0} F_{k,e}/C_e = \sum_k \gamma_{k,e} \times \tilde{M}_k/M_k = \sum_k \tilde{\gamma}_{k,e}$. Note that $\tilde{\gamma}_{k,e}$ is a random variable $\in [0, 1]$ with $E[\tilde{\gamma}_{k,e}] = \gamma_{k,e}$. A generalized version of the Chernoff bound applies to the sum of these variables, with cummulative expected value equal to $\sum_{k:M_k \neq 0} \gamma_{k,e} \leq 1$, leading the desired result for $K > 5$:

$$Pr \left[ \tilde{S}_e/C_e \geq (1 + \log K) \right] < \frac{1}{K}$$

In our simulations the capacity constraints bound is never violated by more than $\log K$ for $K > 5$. 


4.6.2 Scenario (b)

The difficulty in selecting the routelet and determining the flow values based on the LP fractional solution in scenario (b) is due mainly to the fact that flows now depend on two routelets being present: the entry routelet as well as the relay (deflection) routelet corresponding to a flow.

If we restrict our attention to a single SD-pair, the cumulative LP value for the potential entry routelets is at most one. If we would round as in scenario (a), we could very well leave the SD-pair with no potential entry routelet. Since we would like to have a high probability (say $1 - 1/n$) that the SD-pair have an entry routelet after the rounding, we should round the variables with probability boosted by some factor. We choose the factor to be $\log n$, corresponding to the high probability $1 - 1/n$.

Rounding the flows in this situation is another open problem, and our approach is to run the linear program with the choice of routelets now fixed (i.e. a max-flow problem) to obtain the maximum possible throughput that can be obtained given the random choice of routelets.

Experiments show that this approach leads to good results in practice.

4.7 Experimental Methodology and Evaluation

Our goal in this section is to compare through simulations on several BRITE topologies of varying node population, link, bandwidth and node degree. Table 4.7 shows the characteristics of the topologies we consider. We compare the performance of the following five algorithms:

- **LP**: note the objective may be higher than achievable with integer constraints;
- **ILP**: where possible; may be NP-hard to solve in general;
- **Rounding**: using the different rounding schemes for scenario a and scenario b presented in Section ??;
- **Greedy**: In scenario b, first pick entry routelets for all sd-pairs (first potential routelet encountered is designated the entry routelet for a given sd-pair). Step 2 (the only step for scenario a): compute the sum of bottleneck values for the paths introduced by the remaining
routelets. Select the routelet with the highest bottleneck. Repeat step 2 until the desired number of routelets is selected;

**Random**: First step: pick the desired number of routelets randomly. For scenario b, for each SD-pair also pick an entry points randomly (if available). Since the entry point is subject to constraint 4.15) many SD-pairs do not have an available routelet to select as entry point after the first step.

In order to construct the linear program common information regarding the topology, such as the nodes, the edges, the bandwidth on the edges, and also how the nodes are

<table>
<thead>
<tr>
<th>Topo</th>
<th># nodes</th>
<th>Bandwidth distribution</th>
<th>B/w range (Mbps)</th>
<th>Node degree</th>
<th># potential overlays</th>
<th># SD pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50</td>
<td>HeavyTailed</td>
<td>10-1024</td>
<td>2</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>HeavyTailed</td>
<td>10-1024</td>
<td>2</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>HeavyTailed</td>
<td>10-1024</td>
<td>2</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
<td>HeavyTailed</td>
<td>10-1024</td>
<td>2</td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>400</td>
<td>HeavyTailed</td>
<td>10-1024</td>
<td>2</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>600</td>
<td>HeavyTailed</td>
<td>10-1024</td>
<td>2</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>Uniform</td>
<td>10-1024</td>
<td>2</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>Exponential</td>
<td>10-1024</td>
<td>2</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>HeavyTailed</td>
<td>50-5024</td>
<td>2</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>HeavyTailed</td>
<td>10-1024</td>
<td>5</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 4.2: Topology Characteristics (generated using BRITE)
connected to each other needs to be gathered, besides the information regarding the SD-pairs, $K$, and eligible transit node locations.

In practice, tools such as `traceroute` and `pathchar` can be used to obtain the connectivity information, underlay routing, and bottleneck bandwidth. In our experiments, we use `ns` to collect this information.

We next present the results for the scenario b, followed by the scenario a results.

### 4.7.1 Results - Scenario b

Experiments show that the rounding algorithm achieves the maximum throughput while using no more than $\log r$ extra routelets.

Figure 4.3 illustrates the behaviour of our five algorithms as we increase the bound $K$ on the number of routelets we can place. The tests are run on Topo1. Note how close the rounding output is to the LP output, exceeding the feasible ILP value. This is due in part to the fact that the rounding may exceed the number of routelet (by no more than $\log K$),
allowing the rounding to discover the optimum value before the LP does (at $K = 9$). $K$ is an upper bound (an approximative upper bound for the rounding algorithm), and once the LP/ILP/rounding reach a stable optimum solution, around $K = 20$, no more than this number of routelets is used. The performance of random is an average over 4 runs. The greedy solution is surprisingly good at extracting a high percentage of the flow achieved by the LP/ILP solutions. For comparison, we include the aggregate throughput obtained with default single-path routing.

Figure 4.4 captures the throughput per SD-pair for Topology 1, with $K = 20$ for the 16 SD-pairs with non-zero throughput for one or more algorithm. Note the default bar, which, when not present indicates zero default flow between the specific SD-pair. By careful placement of routelets, LP, ILP and the rounding algorithms are able to maximize the throughput achieved for each SD-pair. Random misses some key entry routelets and thus is often unable to improve the throughput. Greedy correctly selects the entry routelets, but does initially miss some less significant deflection routelets which participate in the optimum solution.
In Figure 4.5 we study the aggregate throughput for Topology 1, as the number of SD-pairs we want to create multiple path for increases. The target total number of routelets $K$ is 8. Rounding is outperforming the LP/ILP solutions since it is allowed to use a few more routelets than the LP/ILP solutions.

We plot the number of routelets placed by the rounding algorithm for this same example in Figure 4.6. The more SD-pairs, the more complex the solution is. As the complexity increases, more nodes are usually necessary to achieve a better solution. While the LP uses fractional values for the routelets to achieve the maximum throughput, the rounding can either select or totally discharge a potential routelet. By referring to Figure 4.5 we note that the range where the rounding algorithm selects more than twice the number of routelets of the LP, corresponds to the range where the rounding algorithm outperforms the LP solution. The total number of routelets used remains below the theoretical bound of $K \log K = 24$.

In Figure 4.7 we present the performance of our various algorithms across the ten topologies. The aggregate throughput values are normalized by dividing the throughput of each algorithm by the LP throughput. The actual LP throughput for each topology is

![Algorithm comparison (n=8) - scenario (b)](image-url)
CHAPTER 4. ROUTELET PLACEMENT FOR MULTIPATH ROUTING

Figure 4.6: Number of routelets used as we vary the complexity of the problem (sd-pairs). Scenario b, Topo1, target number of routelets = 8.

Figure 4.7: Performance across various topologies for scenario b. Target number of routelets = 20.
Figure 4.8: Aggregate throughput as the total number of routelets is increased for the five algorithms compared. Scenario a, Topo1.

listed in parentheses (in the x axis names). Using the rounding technique a multipath protocol can achieve a 2-5 fold increase in bandwidth utilization over the default single path routing.

4.7.2 Results - Scenario a

Recall that the rounding algorithm is different for the two different scenario, thus the results in these section are slightly different than those in Section 4.7.1.

In Figure 4.8 the rounding throughput is computed as the average over 4 distinct trials. The results and conclusions are the similar to the ones for Figure 4.3. The main difference is a higher variability in the rounding results compared to scenario (b) due to the absence of the boosting factor present for scenario (b).

In Figure 4.9 note that the rounding throughput can significantly exceed the LP value for a given SD-pair. More than 30% higher for pair 76-31 for example. This is different
Figure 4.9: Throughput per SD-pair for the five algorithms. Scenario a, Topo 1, target number of routelets = 20.

from scenario (b) and indicates a probable edge violation. It would be interesting to plot the average behaviour as opposed to one individual rounding example.

The results in Figure 4.10 are similar to the scenario (b) Figure 4.5.

We obtain different results in Figure 4.11 than in Figure 4.6. The number of routelets selected matches much better the desired number of routlets in scenario (a) than in scenario (b).

Figure 4.12 illustrates the edge-violations for scenario (a) rounding. Note there is no bound for \( r < 3 \) and in this regime we observe the largest values of edge-violation. For the remaining part of the range, the maximum edge-violation remains under the theoretical bound.

Finally Figure 4.13 illustrated the performance accross the various topologies we tested. Similar to Figure 4.7.
Figure 4.10: Aggregate throughput as the total number of SD-pairs is increased. Bound on the number of routelets is 8. Scenario a, Topo1.

Figure 4.11: Number of routelets used as we vary the complexity of the problem (SD-pairs). Scenario a, Topo1, target number of routelets = 8.
Figure 4.12: Actual max-edge violation factor as we vary the target number of routelets versus the theoretical upper bound (available for $r > 3$ only).

Figure 4.13: Performance across various topologies for scenario a. Target number of routelets = 20.
4.8 Conclusion

In this chapter, we address the routelet placement problem to assist multipath transport protocols that are designed to achieve better network utilization and fairness. We identify three different deployment scenarios of routelet placement, provide LP formulations for placement in each of the scenarios, prove hardness results. We provide rounding algorithms with provable properties for a subset of the scenarios, and compare their performance through simulations on several BRITE topologies of varying scales.
Appendix A

Chernoff Bounds

State here Chernoff, Markov, and prove generalized Chernoff.

Theorem A.0.1 (Chernoff Bound) [51] Let $X_1, X_2, \ldots, X_n$ be independent $\{0,1\}$ trials such that $Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Then for $X = \sum_{i=1}^{n} X_i$, $\mu = \sum_{i=1}^{n} p_i$ and any $\delta > 0$,

$$Pr[X > (1 + \delta)\mu] < \left[ \frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right]^\mu$$

Theorem A.0.2 (Markov Inequality) [51] Let $Y$ be a random variable assuming only non-negative values. Then for all $t \in \mathbb{R}^+$

$$Pr[Y \geq tE[Y]] \leq \frac{1}{t}$$

Theorem A.0.3 (Generalized Chernoff Bound) Let $X_1, X_2, \ldots, X_n$ be independent random variables with $X_i \in \{0, r_i\}$ such that $Pr[X_i = r_i] = q_i$, where $0 < r_i, q_i < 1$. Then for $X = \sum_{i=1}^{n} X_i$, $\mu = \sum_{i=1}^{n} r_i q_i$ and any $\delta > 0$,

$$Pr[X > (1 + \delta)\mu] < \left[ \frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right]^\mu$$

Proof: Similar to the proof for of Theorem A.0.1 in [51], since the exponential function is monotonically increasing for positive reals, for any $t \in \mathbb{R}^+$,
\[ Pr[X > (1 + \delta)\mu] = Pr[e^{tX} > e^{t(1+\delta)\mu}] \]

Applying Markov’s inequality to the right hand side of the above equation we get:

\[ Pr[X > (1 + \delta)\mu] < \frac{E[e^{tX}]}{e^{t(1+\delta)\mu}} \quad (A.1) \]

By the definition of \( X \): \( E[e^{tX}] = E \left[ \prod_{i=1}^{n} e^{tX_i} \right] \), and since the \( X_i \) are independent we observe that \( E[e^{tX}] = \prod_{i=1}^{n} E[e^{tX_i}] \).

The random variable \( e^{tX_i} \) assumes the value \( e^{tr_i} \) with probability \( q_i \) and the value 1 with probability \( 1 - q_i \). It is enough to show:

\[ q_i e^{tr_i} + (1 - q_i) \leq p_i q_i e^t + (1 - r_i q_i) \quad (A.2) \]

in order to be able to use the inequalities in the proof of Theorem A.0.1 in [51].

Equation A.2 is equivalent to \( q_i e^{tr_i} - q_i \leq r_i q_i e^t - r_i q_i \). Canceling the \( q_i \) term we get \( 0 \leq 1 - e^{tr_i} + r_i e^t - r_i = f(r_i) \). Viewing the right-hand side as an equation in a single variable, \( r_i \), and differentiating, we obtain a negative derivative for \( 0 < r_i < 1 \), which means the function is decreasing in the interval, with the minimum value attained at \( r_i = 1 \). Since \( f(1) = 0 \) we obtain the desired inequality.

Let \( p_i = r_i q_i \). Using the above result we are able to obtain the same bound as in [51] for the left-hand side of Equation A.1, namely that:

\[ Pr[X > (1 + \delta)\mu] \leq \frac{\prod_{i=1}^{n} [p_i e^t + 1 - p_i]}{e^{t(1+\delta)\mu}} \]

This implies the desired result via algebraic manipulations (explicit in [51]), for any positive \( t \).
Bibliography


