Medial Axis Based Routing Has Constant Load Balancing Factor

Jie Gao[1], Mayank Goswami[2]

[1] Department of Computer Science, Stony Brook University
jgao@cs.stonybrook.edu
[2] Max-Planck Institute for Informatics, Saarbrücken, Germany 66123
gmayank@mpi-inf.mpg.de

Abstract. Load balanced routing is a long standing yet challenging problem. Despite many years' of work there is still a gap between theoretical research and practical algorithms. The main contribution in this paper is to bridge the gap and provide rigorous analysis of a practically interesting algorithm for routing in a large scale sensor network of complex shape – routing by using the medial axis of the network. In this algorithm, a skeleton of the network is extracted such that a virtual coordinate system can be developed for greedy routing achieving good load balance in simulations. We show for the first time a constant approximation factor for this algorithm by a highly technical analysis. The analysis explains the performance observed in previous simulations and is also the first known constant approximation algorithm for load balanced routing in a sensor network with non-trivial geometry.

1 Introduction

Load balanced routing is a fundamental and challenging problem. It is of crucial importance in wireless sensor networks since overloaded nodes may deplete their battery prematurely, severely hampering the utility of the network. It is also more challenging in the sensor network setting as only distributed, lightweight algorithms are useful. Despite many years' of research there is still a separation between algorithms with theoretical guarantees and algorithms that are practically useful. In the theoretical direction, load balanced routing is often formulated as minimizing the maximum traffic load of a given set of routing requests on a network with fixed capacities. Approximation algorithms using global optimizations on graphs [13, 14] do not meet the low resource requirement and do not have constant approximation ratio. In practice, a number of algorithms arrive at a good balance between good performance and low requirement on computation and communication. But nothing provable is known. The results in this paper bridge the gap by providing the first constant approximation ratio for a practically interesting algorithm.

Wireless sensor networks differ from other types of networking scenarios in the rich geometric properties. In this setting, n sensors are embedded (positions chosen uniformly randomly) in a geometric region $\Omega \subset \mathbb{R}^2$ providing dense coverage of Ω. A sensor network is a graph on the vertices. We assume that a unit disk graph is used to model wireless communication: two nodes are connected by an edge if the distance is at most 1. The load of a node is the number of paths (out of the total $\binom{n}{2}$ paths) passing

through it under all-pairs communication. One wishes to find paths that minimize the maximum load, that are also easy to store/compute.

Unfortunately, understanding the dependency of load balanced routing on the geometric shape of Ω is still very limited. For sensors uniformly randomly placed inside a disk with all pairs traffic, shortest path routing (i.e., routing paths along straight lines) will create higher traffic load at the center of the disk. But the highest traffic load can be shown to be a constant factor away from the optimal solution (minimizing the maximum traffic) [8]. Nothing is known on the optimal solution in a disk beyond a discrete approximation using simulations [12]. Apart from that, constant approximation solution using greedy routing is known for narrow strips [7] and for simply connected domains admiting a constant stretch area-preserving map to disks [8]. The general question of finding a constant approximation load balanced routing scheme for an *arbitrary* geometric domain is still open.

In this paper we achieve for the first time a constant approximation ratio for load-balanced routing in a network Ω with arbitrarily complex shape. This problem is substantially more challenging than the case of a simply connected domain. In the case of a disk or other simply connected domain, the main challenge is to avoid concentration of routing paths either at the center of the disk or at a reflex vertex (corner) of the domain. But when we move to a domain with holes, a new challenge appears. We need to decide how the routing paths get around the holes and how such traffic is distributed. This decision has to be dependent on the 'width' of the corridors on each side of holes. See Figure 1. Thus the topology of the domain and the topological structure of the routing paths are essential.

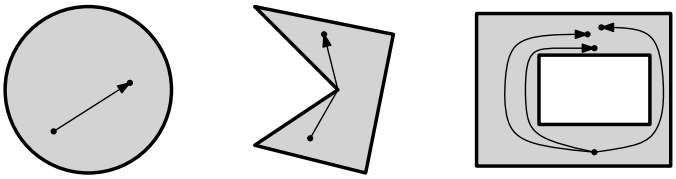

Fig. 1: In simply connected domains, the main challenge is to avoid path concentration (left two figures). In the case of a non-simple domain, we may need to distribute the paths around the holes by how much resource we have available along the 'corridors'.

We show that routing using the medial axis κ of the sensor network field, with some modification, achieves an approximation factor of $O(1)$, when the source and destination are uniformly randomly selected. The medial axis of a 2D domain Ω is the collection of points that have more than one closest points on the boundary of Ω. It is a planar graph homotopic to Ω. Thus as a skeleton of Ω, it has been used in a distributed routing algorithm to guide messages to pass around 'holes' [3]. Basically the medial axis is represented by a compact graph to be disseminated to all nodes in the network. Each node knows its relative position to the medial axis and a greedy routing algorithm directs a message to the destination by first travelling 'horizontally' in parallel with the medial axis and then 'vertically' to the destination. In the same paper it has been demonstrated using simulations that this algorithm has guaranteed delivery and good performance in balancing traffic load, compared to other greedy alternatives. In this paper we build on this algorithm. With an algorithmic enhancement we show that the

maximum traffic load is within a constant factor of the maximum traffic load of optimal routing algorithm.

Specifically, our contributions are the following:

- We enhance the original medial axis based routing algorithm by running a linear program (LP) optimization on the medial axis κ. This LP helps us decide how to distribute routing paths with respect to the holes in the network. The solution to the LP provides a compact, probabilistic routing guidance. This knowledge is distributed to all nodes in the network. Each node s stores $O(|\kappa|)$ information such that for any destination t, a set of paths with probabilities can be extracted to guide how the message should travel with respect to the medial axis κ. The actual routing path is again realized by a local, greedy procedure.
- We show that the above algorithm achieves a constant approximation ratio. The analysis is highly technical. Majority of the proofs can be found in the journal version, and we present the intuition and proof sketch in this extended abstract.

Related Work. In the field of networking algorithms, load balanced routing on a graph with given source destination pairs is a long standing problem. One way to formulate this problem is to select routes that minimize congestion (the maximum number of messages that any node/link carries), termed the *unsplittable flow problem*. Solving this problem optimally is NP-hard even in very simple networks (such as grid). The best approximation algorithm has an approximation factor of $O(\log n/\log\log n)$ [13, 14] in a network of n vertices. It is also shown that getting an approximation within factor $\Omega(\log\log n)$ is NP-hard [1]. Another popular way to formulate the problem is to consider node disjoint or edge disjoint paths that deliver the largest number of given source destination pairs. This is again NP-hard [9] and the best approximation factor known is $O(\sqrt{n})$ [4]. It is NP-hard to approximate within a factor of $\Omega(\log^{1/2-\varepsilon} n)$ [5]. These approximation algorithms are mostly only of theoretical interest. They require global knowledge and are not suitable for distributed settings.

In the wireless sensor network setting, a number of algorithms make use of the geometric property in designing load balanced routing algorithms. Mei and Stefa [11] suggested to wrap a square network into a torus so as to avoid loading the network center. Yu et. al [17] map the network as the skeleton of a convex polytope using the Thurston's embedding. In [15], a network of multiple holes is converted to the covering space to 'remove' the hole boundaries and prevent them from being heavily loaded. All of them are shown by simulations to have reasonable performance, but no theoretical guarantee is known.

2 Our Medial Axis Based Routing Scheme

Before describing our algorithm, we set up the ground work by defining the media axis. Throughout the paper we consider sensor nodes deployed uniformly in a planar domain Ω of complicated shape or topology.

Medial Axis in the continuous setting: The medial axis κ of a geometric domain Ω is defined to be the set of points with more than one closest point to the boundary of Ω. It is known that κ is a planar graph that is homotopic to Ω [2]. κ is composed of

branches (called *medial edges*) joined by junction points (called *medial vertices*), that can have degree 1 (being the endpoint of a branch) or degree at least 3. Each point a on the medial axis has a maximal empty disk inside Ω that touches at least two points on $\partial\Omega$. The line segment connecting a with one of its tangent points is called a *chord* of a.

Medial Axis based coordinates: For each point $p \in \Omega$ that is not on the medial axis, from [3] we know that p lies on a chord xy, with $x \in \kappa$, $y \in \partial\Omega$. And in fact, y is p's closest point on $\partial\Omega$. By this property we can define the projection of p on κ as the point x and denote it as $\kappa(p)$. Furthermore, any point $p \in \Omega \setminus \kappa$ lies on *exactly* one such chord, and is uniquely characterized by the endpoints of this chord (one on κ and one on $\partial\Omega$) and its Euclidean distance from the endpoint on the medial axis. Thus, to every point p one can assign coordinates $(x(p), y(p), r(p))$ where $x(p) \in \kappa$ and $y(p) \in \partial\Omega$ are the endpoints of the unique chord p lies on, and $0 < r(p) = |px(p)|/|x(p)y(p)| < 1$ is the normalized distance from $x(p)$.

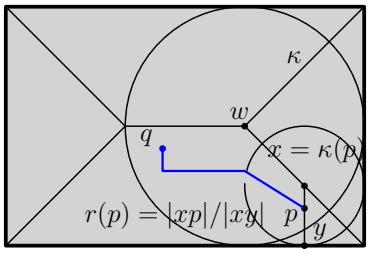

Fig. 2: An example of a medial axis of a domain Ω. Two medial balls centered at x, w are shown. We also show a possible routing path from p to q.

2.1 Summary of the algorithm

1. Extract the medial axis $\hat{\kappa}$ of the discrete sensor network. Assign all the sensors medial axis based coordinates.
2. Run a flow program on the medial axis graph. This returns a routing scheme on the medial axis.
3. Store a compact representation of this routing scheme at the sensors, using a compact representation of the medial axis graph (CMAG).
4. Extend the routing scheme on the medial axis to get a routing scheme on the entire domain, by routing first "in parallel" and then vertically to the medial axis. By routing in parallel, we move in the direction guided by the routing scheme along the medial axis and keeping the normalized distance to the medial axis to be the same (see Figure 2). By routing vertically, we mean the message travels along a chord towards the destination.

Step 1*:* Extracting the medial axis and assigning coordinates proceeds by 1) detecting the boundaries of the sensor domain Ω, 2) flooding to get a discrete medial axis (nodes having equal hop-counts to the boundary) and finally 3) naming each node by a local computation. These steps are described in detail in [3] and we omit the discussion here.

2.2 Step 2: The Flow Program

Let the (discrete) medial axis graph be denoted as $\hat{\kappa} = (V, E)$, where V is the set of medial vertices and E is the set of medial edges. **(We probably should say what the shortest path trees are, either here, or in the summary above.)** We denote by $\hat{r}(v)$ the depth of the shortest path tree rooted at $v \in V$, and by $\hat{n}(v)$ the number of sensors

in the tree(s) rooted at this node. For each node we assign a capacity $c \cdot \hat{r}(v)$, where $c > 0$ is a parameter.

Our flow program will have $k(k-1)$ commodities, where $k = |V|$. We have one commodity for every ordered pair (i, j) of nodes in V. For an ordered pair (i, j) we define two commodities $\vec{ij}$ and $\vec{ji}$. Node i and node j serve as source and sink for commodity $\vec{ij}$, and the roles are reversed for $\vec{ji}$. We set their demands as $d_{\vec{ij}} = d_{\vec{ji}} = \hat{n}(i)\hat{n}(j)$.

Let the flow of commodity $\vec{ij}$ along the edge (u, v) be $f_{\vec{ij}}(u, v)$. Our flow must satisfy the following constraints:

- **Capacity constraints** The load of v is due to three kinds of messages: messages with source v, messages with destination v, and messages with neither source nor destination as v.

$$\sum_{j \neq v} \sum_{u} f_{\vec{vj}}(v, u) + \sum_{j \neq v} \sum_{u} f_{\vec{jv}}(u, v) + \sum_{\vec{ij}: i \neq v, j \neq v} \sum_{u} f_{\vec{ij}}(u, v) \leq c\hat{r}(v)$$

  Note that the first two sums each simplify to $m - \hat{n}(v)$, as there are precisely these many messages with v as source and destination.
- **Flow conservation** $\sum_{w \in V} f_{\vec{ij}}(u, w) = \sum_{w \in V} f_{\vec{ij}}(w, u) \quad \forall u \notin \{i, j\}$.
- **Demand satisfaction** $\forall i, j \in V, \quad \sum_{w \in V} f_{\vec{ij}}(i, w) = \sum_{w \in V} f_{\vec{ij}}(w, j) = d_{\vec{ij}}$.

The program we want to run is to minimize c subject to these conditions. Since the only unknown variables above are the flow quantities, i.e., $f_{\vec{ij}}(u, v)$ for commodity $\vec{ij}$ on edge (u, v), we can do the following: we first fix c large enough (say equal to the sum of all the demands, which is $O(n^3)$, where n is the number of sensors in Ω), and find whether the set of three constraints are *feasible*. This would imply that the chosen capacity is sufficient to route the flows according to the specified demands. We then do a binary search on the optimal value of c; each time we halve the current value of c and check for feasibility. Thus we end up with the optimal c, in a runtime that is $O(\log n)$ times the complexity of checking feasibility, which sums up to $O(k^2 \log n)$. This is pretty efficient; note that k is the number of nodes on the medial axis $\hat{\kappa}$, which is much smaller than n (the number of sensors).

Once we get the optimal value of c and the corresponding flows $f_{\vec{ij}}(u, v)$ on every edge for every commodity, we can find the paths to take from a particular source i to source j using the "path stripping" algorithm in [14]. This returns a set of paths that realize the optimal flow, and these paths constitute the routing scheme on the medial axis graph $\hat{\kappa}$. We call this routing scheme Γ_κ.

Compact Representation of Γ_κ Once the medial axis is extracted and the coordinates are assigned, we also store a compact representation of the medial axis graph, called CMAG. This graph has the following properties:

1. The number of vertices in the CMAG equals the number of medial vertices.
2. A path (on the medial axis) between two medial vertices that does not go through any other medial vertex, corresponds to an edge between the corresponding pair of vertices on the CMAG. Thus "consecutive" medial vertices have an edge between them.

3. The size of the CMAG is linear in the number of big topological features in Ω; if Ω has h holes (boundaries), then this graph has size $O(h)$.

The proof of the following theorem can be found in the appendix.

Theorem 1. *Let $\hat{\kappa}_c = (V_c, E_c)$ denote the CMAG, with $k = O(h)$ number of vertices and edges. The routing scheme Γ_k can be stored compactly in a way that requires $O(h)$ space for any node on the medial axis, and $O(h2^h)$ space for any medial vertex (a vertex of $\hat{\kappa}_c$).*

2.3 Extending Γ_κ to Γ on Ω

Given a source-destination pair $(s, t) \in \Omega$, we now use Γ_κ to build a path from s to t. Roughly, the path starts from s, follows points at the same normalized distance (same r in terms of the medial axis coordinates in Figure 2) to the medial axis as s until it arrives at the chord containing t, and then follows part of the chord to arrive at t. The homotopy of this path is the same as the homotopy of the path from $\kappa(s)$ to $\kappa(t)$, as determined by Γ_κ.

In the discrete setting chords are replaced by shortest path trees rooted at nodes on the (discrete) medial axis. Unless the message is already in the tree containing t (in which case the message is sent along the tree to arrive at t), a node v forwards the message to a neighboring node with the same (or closest) normalized height as v (height of v divided by the height of the tree containing v). Which direction to send is determined by the path $\Gamma_\kappa(\kappa(s), \kappa(t))$, i.e., by the routing scheme on the medial axis. Due to the discrete nature, a small local detour might be necessary in order to get to a node with the same normalized height; the message tries to return to the original normalized height (that of s) as soon as possible.

At last, we remark that we need to also involve the rotary system as introduced in [3]. Essentially, consider a route along the medial axis, P that goes through a medial vertex q with degree 3. If the canonical pieces for the two medial edges on P right before and after q do not share a common chord of q, the message will travel along a rotary system centered at q by following an arc connecting the two canonical pieces. There are two arcs connecting the two pieces, clockwise or counterclockwise. In [3], the direction is arbitrary. Here we always split with probability half among the two choices so each gets half of the total traffic.

Comparison to the medial axis routing scheme in [3] The idea of routing in parallel to the medial axis is the same as in [3]; the main innovation here is in the flow program in Step 2. In [3] the routing used on the medial axis was simply the shortest path routing. Our routing scheme given by the flow program helps us prove the approximation guarantee. Theorem 1 guarantees that this routing scheme is also compactly represented and lightweight, much like the shortest path scheme.

3 Network Model For Analysis

In this section we describe the network model for our theoretical analysis. We remark that this model is purely for the proofs and is not necessary for the routing algorithm.

We work with a large number of sensors uniformly distributed in a geometric domain Ω with holes. We assume that Ω only has a finite number of arc segments on its boundary $\partial\Omega$, that is, a part of the boundary that coincides with part of a circle. In this case, all but a finite number of points W on the medial axis κ of Ω have only a constant number of chords. In addition, we will punch a point hole at each point of W and recompute the medial axis κ. If there are still points on κ with an infinitely number of chords, we repeat the same procedure until no point on the medial axis has an infinite number of chords. We remark that this procedure will only create a finite number of additional point holes. In the following discussion we also assume there is no degeneracy, i.e., any maximal empty ball is tangent to at most three points on the boundary. Thus all points on κ have at most three chords. The domain Ω can be decomposed into *canonical pieces* by removing the medial axis and the chords on the medial vertices. Each piece is bounded by two chords, a medial edge (or a medial vertex) and a piece of $\partial\Omega$.

The next sentence is confusing. I think what you want to say is we work in the model (with these special sensors) setting? In this paper we work with the discrete medial axis [3] but for the clarity of the proof, we will work with a **(perhaps add special?)** set of sensors S in Ω and the (discrete) medial axis defined on it.

Discrete Sensors S. We assume that sensors have bounded density ε: inside any disk of radius ε inside Ω, there is at least one sensor inside. Further we assume that any two sensors are at distance at least $\varepsilon/2$ apart. The minimum density requirement is typically guaranteed by sensing coverage requirement. The requirement on minimum distance separation can be obtained by using a greedy algorithm to subsample in a dense region. Basically, select any sensor not yet selected, remove all sensors within distance $\varepsilon/2$, and continue. Any two sensors selected are of distance at least $\varepsilon/2$ apart.

To define a discrete sample of Ω, we also assume a uniform set of sensors along the continuous medial axis κ with density ε, such that any point of κ has a sensor within distance ε along κ.

Discrete Medial Axis on S. We define the discrete medial axis on S using the same ideas as in [3]. First, the sample points on κ are connected into a graph as a discrete approximation of κ. In the continuous setting a chord is a line segment connecting a point on the medial axis to a tangent point on the boundary. In the discrete setting, due to the discrete resolution a chord is a tree T_a rooted at a point $a \in S$ on the medial axis. In particular, suppose a lies on a medial edge, and its neighbors are a_1 and a_2 to the left and right of a respectively. And m_1, m_2 are halfway midpoints between a_1 and a, a and a_2 along κ respectively. Suppose the two chords of m_1 and m_2 connect to $p_1, p_2 \in \partial\Omega$ respectively. We build a tree T_a containing all the nodes of S inside the region bounded by four pieces of boundaries: the segment on κ between m_1, m_2, the two chords m_1p_1 and m_2p_2, and the boundary segment between p_1p_2. See Figure 3.

Consider a canonical piece, we define a *normalized contour* κ_i of level $i \in [0, 1]$ as the collection of nodes (sensors) w such that w lies on a chord xy with $x \in \kappa$, $y \in \partial\Omega$, and $i = |wx|/|xy|$ (the normalized hop-count distance to κ).

Each node w defines a parent in the tree T_a as the node that lies on one lower-level contour than w. We

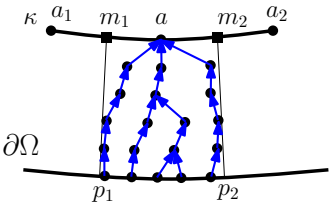

Fig. 3: A discrete chord.

define this tree T_a as the discrete chord of $a \in \kappa$. For each point $p \in S$, we denote by a its projection on κ if T_a contains p. This is denoted by $\hat{\kappa}(p)$, to be differentiated from the continuous projection of p on κ. We also denote by $r(a)$ the length of the chord of a. We denote by $\hat{r}(a)$ the depth of the tree T_a, and $\hat{n}(a)$ the total number of nodes in T_a.

For a node $a \in \kappa_0$ (the medial axis), we first want to understand the relationship between $\hat{r}(a)$ and $\hat{n}(a)$. In the continuous setting, the analog of $\hat{r}(a)$ refers to the length of a chord issued at a, while the analog of $\hat{n}(a)$ refers to the measure of the points on this chord. And these two numbers are actually the same. In this discrete setting, however, these two measures are different. In the extreme setting, imagine a perfect disk of radius r. The medial axis is a single point, the center of the disk, o. The chords are organized in about $O(1/\varepsilon)$ trees, each one containing roughly $O(r^2/\varepsilon)$ nodes. So in this case $\hat{r}(o) = O(r/\varepsilon)$ and $\hat{n}(o) = O(r^2/\varepsilon)$ – a big discrepancy. However, this can only happen when the point o has infinitely many chords, i.e., the boundary segment is part of an arc. Under our assumption, $\hat{n}(a) = O(\hat{r}(a))$ for any $a \in \kappa$. The proof of the following Lemma is in the appendix.

Lemma 2. *Suppose any point on the medial axis κ of domain Ω has only a finite number of tangents. Then $\hat{n}(a) = O(\hat{r}(a))$ for any $a \in \kappa$.*

Route in Parallel to κ. We elaborate how to route a message 'in parallel' to the medial axis. This is the analog of routing along a normalized contour inside a canonical piece with respect to the medial edge. That is, we wish to send the message from a node v to the next node whose projection on κ is along the guidance obtained from the flow algorithm, and the normalized distance to κ is the same as that of v. In the discrete setting, suppose v lies on the chord of a_1 and the next node on κ is a_2. We choose the next node from the chord of a_2. We may not have a node in T_{a_2} with exactly the same normalized distance to κ as v. Instead, we choose the node u whose normalized distance to κ is the closest to the normalized distance from the source s to κ. By our density constraint, u is within distance ε from the normalized contour κ_i, where i is the normalized distance of s to κ.

Route Vertically to κ. When the message gets to a node whose projection to κ is the same as the projection of the destination t, we simply route the message along the tree to the destination.

4 Proof of approximation guarantee

In this section we present the proof of the following theorem:

Theorem 3. *There exists a constant $C \geq 1$ such that the maximum load of the routing scheme Γ obtained above is at most C times the maximum load of the optimal routing scheme Γ^* on Ω.*

Proof Sketch. The proof is highly technical and involves a number of new ideas. The outline of the proof is shown in Figure 4. Let the maximum load of Γ^* be realized at a point $p \in \Omega$, and equal ℓ_p^*, and similarly define q and ℓ_q for the routing scheme Γ.

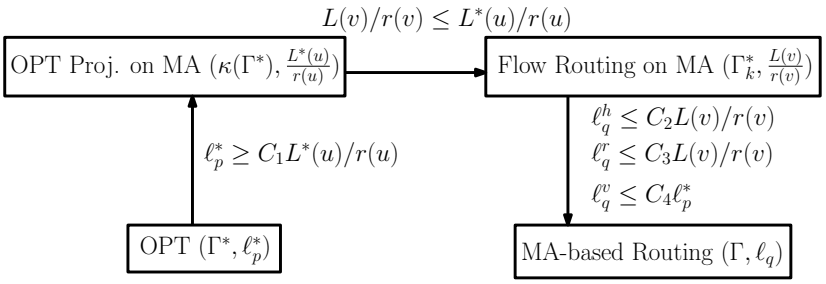

Fig. 4: The outline of the proof for medial-axis based routing.

There are three main steps to relate the maximum traffic load of the optimal load balanced routing algorithm (ℓ_p^* at node p), with the maximum traffic load of the medial axis based routing algorithm (ℓ_q at node q). In the intermediate steps we will also consider the projection of optimal routing scheme on medial axis. Denote by $\kappa(\mathit{\Gamma}^*)$ the "projection" of the optimal routing scheme (for all-pairs) on the medial axis. That is, for each path $\gamma^*(s,t) \in \mathit{\Gamma}^*$, we project each node onto the medial axis, getting a path along the medial axis from $\hat{\kappa}(s)$ to $\hat{\kappa}(t)$. This path might be non-simple. For example, it is possible that multiple nodes on the path $\gamma^*(s,t)$ maps to the same node on $\hat{\kappa}$. It is possible that the projected path revisit a node on $\hat{\kappa}$. When such things happen, we simply remove the redundant segments and only keep a simple path along $\hat{\kappa}$. From now, we measure the traffic load caused by $\kappa(\mathit{\Gamma}^*)$ after this small simplification.

Regarding the traffic model for $\kappa(\mathit{\Gamma}^*)$, it is easy to see that for two nodes i and j on the medial axis, node i on medial axis sends $\hat{n}(i)\hat{n}(j)$ messages to node j (recall that $\hat{n}(i)$ is the size of the subtrees rooted at node i) under $\kappa(\mathit{\Gamma}^*)$, and the same amount goes from node j to node i.

The three steps in Figure 4 relate the traffic load in the four routing algorithms, two on the original domain Ω and two on the medial axis:

1. Denote by $L^*(i)$ the number of messages passing through node $i \in \hat{\kappa}$ under the projection $\kappa(\mathit{\Gamma}^*)$. Let $u \in \hat{\kappa}$ be the node where the maximum of the quantity $L^*(i)/\hat{r}(i)$ occurs. We first show that this quantity is bounded; that is, $L^*(u)/\hat{r}(u) \le 4\ell_p^*$.
2. On the medial axis, we run an optimization algorithm to find the routing scheme minimizing $L(x)/\hat{r}(x)$ for any node x on the medial axis, for the given traffic pattern. Minimizing max load is an NP-hard problem when the problem is integral, i.e., only a single path is taken by each node. But in our case, we use a non-integral solution which can be interpreted as a family of paths each with a probability measure. Following the probabilistic routing scheme we can minimize the expected maximum traffic load. This routing paths are denoted by the family $\mathit{\Gamma}_\kappa$. Let $v \in \hat{\kappa}$ be the node where the quantity $L(x)/\hat{r}(x)$ achieves its maximum value ($L(x)$ is the number of messages passing through x under $\mathit{\Gamma}_\kappa$). Optimality of the flow program then implies $L(v)/\hat{r}(v) \le L^*(u)/\hat{r}(u)$.
3. In the last step, we take the routing paths along the medial axis and convert them to routing paths in the original network from the source to destination. Specifically, for a source s and t in the domain, we first project them to the medial axis along the chord at $\hat{\kappa}(s), \hat{\kappa}(t)$ respectively. Then we use the routing algorithm guided by

Γ_κ computed in the previous step to compute a path $\gamma(\hat{\kappa}(s), \hat{\kappa}(t))$ along the medial axis. This path is then converted to a path in the original domain – specifically, the message first travels on a path parallel along the medial axis until it arrives at the chord of t; then it follows the chord to arrive at t. It may also use the rotary system around a medial vertex. Thus the traffic at q (the node with max load) in our routing scheme is divided naturally into horizontal (denoted by ℓ_q^h), rotary (denoted by ℓ_q^r) and vertical (denoted by ℓ_q^v) loads. For the horizontal and rotary traffics, we show that there exist constants C_2 and C_3 such that $\ell_q^h \leq C_2 L(v)/\hat{r}(v)$, and $\ell_q^r \leq C_3 L(v)/\hat{r}(v)$. For the vertical traffic, we show that it is bounded by the maximum load of *any* routing scheme on Ω, including that of the optimum Γ^*. Thus we show that $\ell_q^v = O(\ell_p^*)$.

Collecting the inequalities, we have that for some constant C, $\ell_q \leq C\ell_p^*$. Thus the medial axis based routing achieves an $O(1)$ approximation in minimizing the maximum traffic load.

For the complete proof, we urge the reader to read Section 6.3 in Appendix first. In the following we only present the lemma analyzing the horizontal and rotary loads of our proposed routing scheme Γ on Ω. The vertical load is in Section 6.4 in appendix.

Let q be the node with the maximum load under the routing scheme Γ with load ℓ_q. Because of the nature of our routing scheme (Manhattan and sometimes-rotary in the medial axis coordinates), we partition ℓ_q into three parts; we write $\ell_q = \ell_q^h + \ell_q^r + \ell_q^v$ where ℓ_q^h denotes the number of messages passing through q "horizontally" (entered from and was forwarded to a node with same normalized height along the tree as q), ℓ_q^r denotes the number of messages passing through q in the rotary system, and ℓ_q^v denotes the number of messages passing through q "vertically" (were forwarded by q to some node in the same tree as q).

Lemma 4. *Let $v \in \hat{\kappa}$ be the node where the quantity $L(x)/\hat{r}(x)$ achieves its maximum value ($L(x)$ is the number of messages passing through x under Γ_κ), and let q be the node with the maximum load under the routing scheme Γ with load ℓ_q. Then there exist constants C_2 and C_3 such that $\ell_q^h \leq C_2 L(v)/\hat{r}(v)$, and $\ell_q^r \leq C_3 L(v)/\hat{r}(v)$.*

Proof. **Proof for ℓ_q^h:** We first note that by definition of $L(v)/\hat{r}(v)$ as the maximum of the optimal flow, $L(\hat{\kappa}(q))/\hat{r}(\hat{\kappa}(q)) \leq L(v)/\hat{r}(v)$. Hence it suffices to show that there exists C_3 such that $\ell_q^h \leq C_3 L(\hat{\kappa}(q))/\hat{r}(\hat{\kappa}(q))$. For simplicity, set $r := \hat{r}(\hat{\kappa}(q))$. In fact we will prove that for any node v on the chord of $\hat{\kappa}(q)$, the traffic load ℓ_v^h is bounded as required.

The total horizontal traffic load at $\hat{\kappa}(q)$ is actually shared by all the nodes in the chord of $\hat{\kappa}(q)$. If the nodes in the chord share the traffic uniformly, then the claim is trivially true. Note that the flow program, and hence our extension, does not distinguish between nodes on the same chord when it comes to determining homotopy of paths[3].

Therefore, the traffic carried by a node is determined by how many sources stay on the same horizontal level, which can differ. For any node v at normalized height β, denote by H_β the normalized contour formed by all nodes at normalized height β. Now,

[3] For a given destination t, the distribution over the homotopy types of paths from s_1 to t or s_2 to t is the same, if s_1 and s_2 are on the same chord.

only the nodes with normalized height within $[\beta - \varepsilon, \beta + \varepsilon]$ may possibly arrive at v, by our definition. By the bounded density condition, the number of sensor nodes with normalized height within $[\beta - \varepsilon, \beta + \varepsilon]$ is proportional to the total area occupied by the points with normalized height in the same range, which is then proportional to the length of H_β. Thus we have that

Observation 1: The ratio of the number of messages passing horizontally through a node v (at height β) and the total number of messages passing through the chord containing v, is proportional to the length of H_β.

Now we need to examine the length of the contours H_β for different normalized distance $\beta \in [0, 1]$. The heaviest traffic load happens at the node whose depth coincides with the longest contour. The worst case is that the contour at one particular normalized height is long, while all the others are very short. The following observation says that this cannot happen (proof in appendix section 6.5).

Observation 2: Suppose H_i is the longest contour, $i \in [0, 1]$. There is a constant δ such that one of the two cases is true: (i) the contours H_j with $j \in [i, i + \delta]$ have length $\Omega(\text{Length}(H_i))$; (ii) the contours H_j with $j \in [i - \delta, i]$ have length $\Omega(\text{Length}(H_i))$.

Within the chord of $\hat{\kappa}(q)$, $O(\delta r)$ nodes have normalized depth within the range $[i, i + \delta]$ or $[i - \delta, i]$. Even if all other contours have length 0 and the rest of the nodes in the chord share no traffic within $L(\hat{\kappa}(q))$, the maximum node will still receive traffic load at most $L(\hat{\kappa}(q))/(\delta r)$. Since δ is a constant, the claim is true.

Proof of ℓ_q^r: Some nodes will carry both horizontal traffic and the rotary traffic. The proof for the rotary traffic being not very high is in fact the same as the argument on horizontal traffic – by analyzing the length of contours at different normalized depth. Thus we omit the discussion here.

5 Discussions and open problems

Randomized to deterministic: Note that the paths found by the path stripping algorithm are probabilistic; between a given source-destination pair (s, t) the algorithm returns a probability distribution over the set of all possible paths between s and t. One could use randomized rounding as in [14] to get deterministic paths between every pair; however, this will increase our approximation factor from $O(1)$ to a factor that is slightly sublogarithmic in n (the number of sensors on the medial axis).

Other traffic distributions: In this paper we showed how to get a constant factor approximation for uniform traffic load. However, our algorithm can easily be generalized to any arbitrary traffic distribution Π on Ω, although we cannot prove the approximation yet. Let $\Pi(s, t)$ denote the probability of communication between source-destination nodes s and t respectively. Let $u = \kappa(s)$ and $v = \kappa(t)$ be their projections on the medial axis. Denote by C_{iu} and C_{jv} denote the set of nodes at depths i and j in the trees rooted at nodes u and v, respectively, and the cardinalities of these sets by n_{iu} and n_{iv}, respectively.

When we set up the multicommodity flow program, instead of assigning a demand of $r_u r_v$ between nodes u and v on the medial axis, we now assign a demand

$$d_{u,v} = \sum_{i=1}^{r(u)} \sum_{j=1}^{r(v)} \sum_{s \in C_{iu}} \sum_{t \in C_{jv}} \frac{\Pi(s, t)}{n_{iu} n_{jv}},$$

and run the flow algorithm. The paths generated by the flow algorithm can be extended to get a routing scheme on Ω that satisfies the traffic distribution Π.

Thus, the two open questions that remain are 1) does the algorithm of this paper for uniformly deployed nodes and arbitrary traffic patterns provide an approximation guarantee?, and 2) can one remove the uniformly deployed nodes condition, even for the uniform traffic distribution?

References

1. M. Andrews and L. Zhang. Hardness of the undirected congestion minimization problem. *SIAM Journal on Computing*, 37(1):112–131, 2007.
2. D. Attali, J.-D. Boissonnat, and H. Edelsbrunner. Stability and computation of the medial axis — a state-of-the-art report. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Springer-Verlag, 2004.
3. J. Bruck, J. Gao, and A. Jiang. MAP: Medial axis based geometric routing in sensor networks. *Wireless Networks*, 13(6):835–853, 2007.
4. C. Chekuri, S. Khanna, and F. B. Shepherd. An $o(\sqrt{n})$-approximation for EDP in undirected graphs and directed acyclic graphs. *Theory of Computing*, 2:137–146, 2006.
5. J. Chuzhoy and S. Khanna. New hardness results for undirected edge-disjoint paths. Manuscript, 2005.
6. J. Gao and L. Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 483–492, New York, NY, USA, 2003. ACM.
7. J. Gao and L. Zhang. Load balanced short path routing in wireless networks. In *IEEE InfoCom*, volume 23, pages 1099–1108, March 2004.
8. M. Goswami, C.-C. Ni, X. Ban, J. Gao, D. X. Gu, and V. Pingali. Load balanced short path routing in large-scale wireless networks using area-preserving maps. In *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc14)*, pages 63–72, August 2014.
9. R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
10. R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.
11. A. Mei and J. Stefa. Routing in outer space: fair traffic load in multi-hop wireless networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 23–32, New York, NY, USA, 2008. ACM.
12. L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica. Balancing traffic load in wireless networks with curveball routing. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 170–179, New York, NY, USA, 2007. ACM.
13. P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comp. and System Sciences*, pages 130–143, 1988.
14. P. Raghavan and C. D. Thompson. Provably good routing in graphs: regular arrays. In *Proceedings of the 17th annual ACM Symposium on Theory of Computing*, pages 79–87, 1985.
15. R. Sarkar, W. Zeng, J. Gao, and X. D. Gu. Covering space for in-network sensor data storage. In *Proc. of the 9th International Symposium on Information Processing in Sensor Networks (IPSN'10)*, pages 232–243, April 2010.

16. P. J. Vermeer. *Medial Axis Transform to Boundary Representation Conversion*. PhD thesis, West Lafayette, IN, USA, 1994. UMI Order No. GAX95-01783.
17. X. Yu, X. Ban, R. Sarkar, W. Zeng, X. D. Gu, and J. Gao. Spherical representation and polyhedron routing for load balancing in wireless sensor networks. In *Proc. of 30th Annual IEEE Conference on Computer Communications (INFOCOM'11)*, pages 612–615, April 2011.

6 Appendix

6.1 Proof of Compact Representation

Theorem 5. *Let $\kappa_c = (V_c, E_c)$ denote the (compacted) Medial Axis Graph, with $O(h)$ number of vertices and edges. The routing scheme Γ_k can be stored compactly in a way that requires $O(h)$ space for any node on the medial axis, and $O(h2^h)$ space for any medial vertex (a vertex of κ_g).*

Proof. For simplicity we let h denote the number of vertices in κ_c (note that this equals the set of medial vertices, and is of the same order as the number of holes in Ω). There are at most 2^h simple paths in κ_c (once we arrive at a medial vertex, there are only two paths we can use to exit). We consider only simple paths because:

- the routing scheme does not return non-simple paths; loops only end up increasing the load, and do nothing to help the flow, and
- the flow does not "turn back" while traveling along an edge of κ_g, or in other words, a path of degree two nodes on the medial axis.

Each medial vertex stores the probability distribution over routes to every other medial vertex. There are $h-1$ other medial vertices and $O(2^h)$ possible simple paths to each, thus giving us a space of $O(h2^h)$ at these h medial vertices.

To get a space of $O(h)$ at *all other* sensor nodes, we make a couple of observations. Let e_1 and e_2 be two edges in κ_c corresponding to two (medial vertex-free) paths γ_1 and γ_2 in κ. Denote the end points of γ_i as $\gamma_{i\ell}$ and γ_{ir} ($i = 1, 2$).

First, consider source-destination pairs (s_1, t_1) and (s_2, t_2) with $s_i \in \gamma_1$ and $t_i \in \gamma_2$, such that both routing paths exit γ_1 through the same medial vertex (either $\gamma_{1\ell}$ or γ_{1r}) and enter γ_2 via the same medial vertex. Assume that these vertices are $\gamma_{1\ell}$ and $\gamma_{2\ell}$; then the portion of the route between the medial vertices is the same (the same distribution over paths from $\gamma_{1\ell}$ to $\gamma_{2\ell}$) for both (s_1, t_1) and (s_2, t_2).

Essentially all that s_1 needs to know to route to t_1 is which endpoint of γ_1 to exit, and which endpoint of γ_2 to enter, since the information about the rest of the path is stored at the medial vertices. We show how to store this information next.

We claim that there are two "switching" vertices v_1 and v_2 on γ_1 and γ_2, respectively, such that the following holds:

- If the source s is on the path from $\gamma_{1\ell}$ to v_1 and the destination is on the path from $\gamma_{2\ell}$ to v_2, then the medial vertices on the route are $\gamma_{1\ell}$ and $\gamma_{2\ell}$.
- If the source s is on the path from $\gamma_{1\ell}$ to v_1 and the destination is on the path from v_2 to γ_{2r}, then the medial vertices on the route are $\gamma_{1\ell}$ and γ_{2r}.

The other two cases are similar. This can be proved by using a swapping argument, and by the properties of the path stripping algorithm that realizes the optimal flow. Consider $s \in \gamma_1$ and three destinations t_1, t_2, t_3 on γ_2, in order from left to right. Then it cannot be that $\vec{st}_1$ and $\vec{st}_3$ enter γ_2 from the left, while $\vec{st}_2$ enters it from the right. By a simple exchange argument one arrives at a contradiction. Thus for every vertex $s \in \gamma_1$, there is a switching vertex on γ_2. Similarly, one can prove that this switching

vertex does not vary as we move s along γ_1 until a certain point (which is v_1) and then the path starts entering from the right.

In addition to storing κ_c, every node stores which side of the switching vertex it is on for a given edge $e_i \in \kappa_c$. This amounts to storing a bit-vector of length $O(h)$, thus completing the proof.

6.2 Proof of Lemma 2

Proof. We first prove the claim for a non-vertex point $a \in \kappa$. We show that the area of the region S_a is $O(r(a))$, where $r(a)$ is the chord length at a.

Suppose a stays on a medial edge C and we parameterize the edge by length s. Then $a = C(s)$ for some s. $m_1 = C(s_1), s_1 = s - \varepsilon/2$, $m_2 = C(s_2), s_2 = s + \varepsilon/2$. By [16], the chord length at any medial axis point between m_1 and m_2 are actually bounded by $r(a) + O(\varepsilon)$. To show that the area of the region S_a is $O(r(a))$, we only need to show that the boundary segment between p_1 and p_2 is of length at most $O(\varepsilon)$.

We consider the boundary segment B whose projection is C, also parameterized by s. Thus $C(s), B(s) : [0,1] \to \mathbb{R}^2$. Consider the vector from a to the corresponding boundary point $p \in B$, $\kappa(p) = a$. This vector is the normal vector at p. Denote by $N(s)$ the unit length vector along the normal; and $r(s)$ the length of this vector (i.e., the chord length at a), all parameterized by s. Then $B(s) = C(s) + N(s)r(s)$. The length of the boundary segment between p_1 and p_2 is bounded by

$$\begin{aligned}\int_{s_1}^{s_2} ||B'(s)||ds &= \int_{s_1}^{s_2} ||C'(s) + N'(s)r(s) + N(s)r'(s)||ds \\ &\leq \int_{s_1}^{s_2} ||C'(s)||ds + \int_{s_1}^{s_2} ||N'(s)r(s)||ds + \int_{s_1}^{s_2} ||N(s)r'(s)||ds\end{aligned}$$

Each of the three terms on the right hand side is bounded by ε. Specifically,

$$\int_{s_1}^{s_2} ||C'(s)||ds = \int_{s_1}^{s_2} 1ds = s_2 - s_1 = \varepsilon.$$

For the second term, $r(s)$ is the radius of an empty circle tangent at B, thus $r(s)$ is no greater than the radius of osculating circle at $B(s)$. This means

$$r(s) = 1/\operatorname{curv}(B(s)) = 1/|T'(B(s))| = 1/|N'(s)|,$$

where $\operatorname{curv}(B(s))$ is the curvature at $B(s)$ and $T'(B(s))$ is the tangent vector at $B(s)$. Thus the second terms is also bounded by ε. For the third term, recall that $N(s)$ is unit length vector. Thus

$$\int_{s_1}^{s_2} ||N(s)r'(s)||ds \leq \int_{s_1}^{s_2} |r'(s)|ds \leq \int_{s_1}^{s_2} 1ds = s_2 - s_1 = \varepsilon.$$

The last inequality follows from the observation $|r^{'}(s)| \leq 1$ in the proof of Lemma 2.3.1 on page 25 of [16].

The case when a is a medial vertex can be handled in the same way, since a has only a finite number of continuous chords.

6.3 Proofs of First Two Steps in Theorem 3

Lemma 6. *Denote by $L^*(i)$ the number of messages passing through node $i \in \hat{\kappa}$ under the projection $\kappa(\Gamma^*)$. Let $u \in \hat{\kappa}$ be the node where the maximum of the quantity $L^*(i)/\hat{r}(i)$ occurs. Then $\ell_p^* \geq C_1 L^*(u)/\hat{r}(u)$, for some constant C_1.*

Proof. Consider the discrete chord at u, which is a tree T_u with $n(u)$ nodes, each has traffic at most ℓ_p^* (the maximum traffic load). Thus the traffic at u by the projection on $\hat{\kappa}$ is $L^*(u) \leq \hat{n}(u)\ell_p^*$. The claim follows from the fact that $\hat{n}(u) = O(\hat{r}(u))$ for any node $u \in \hat{\kappa}$ (Lemma 2).

Next, let $L(x)$ be the number of messages passing through x under Γ_κ (the routing scheme returned by the multicommodity flow program output), and let $v \in \hat{\kappa}$ be the node where the quantity $L(x)/\hat{r}(x)$ achieves its maximum value. Optimality of the flow program then implies

Lemma 7. $L(v)/\hat{r}(v) \leq L^*(u)/\hat{r}(u)$.

6.4 Proof of Vertical Load

Here we bound the vertical load of our routing scheme.

Lemma 8. *The vertical load ℓ_q^v at q (the node with max load under Γ) is no more than the maximum load of any routing scheme on Ω. In particular, $\ell_q^v \leq \ell_p^*$, where p is the node with maximum load in the optimal routing scheme Γ^*.*

Proof. First we estimate the vertical load introduced by our routing scheme. Recall that a message passes through q vertically iff the destination is in the subtree(s) rooted at $\kappa(q)$. Thus the number of messages passing vertically is $O(mn(\kappa(q)))$, since the source could be any of the m sensors, while the destination must be in the subtree, that has size $O(n(\kappa(q)))$.

The proof proceeds by comparing this vertical load to the maximum load of *any* routing scheme. We claim that for any routing scheme Γ on the network, the maximum load of Γ is $\Omega(m\sqrt{m})$. This is particularly simple if the network communication graph is assumed to be planar (e.g., the Delaunay trianguation of the uniformly distributed sensors) by using the planar separator theorem [10]. The theorem implies that there exists a cut S of size $O(\sqrt{m})$ such that the removal of this set partitions the graph into two subgraphs A and B, with no edges from A to B, and both the sizes of A and B are roughly $m/3$. This means that during all-pairs communication, there are $\Omega(m^2)$ messages with source in A and destination in B, and by definition of S all these messages must pass through a node in S. This implies that the average load of a node in S is $\Omega(m^2/\sqrt{m}) = \Omega(m\sqrt{m})$, which is clearly no greater than the maximum load over S. The proof can be generalized to the case when the graph is a *unit-disk-graph*, using arguments similar to those presented in [6]. We omit the technical proof for UDGs here, and emphasize that the same idea applied using the separators for UDGs in [6] works.

Now we show that $\ell^v(q)$ is smaller than the maximum load of any routing scheme. By arguments above, $\ell^v(q) = O(mn(\kappa(q)))$ and by Lemma 3, $n(\kappa(q)) = O(r(\kappa(q))$.

Thus if we prove that $r(\kappa(q)) = O(\sqrt{m})$, we are done, as the max load of any routing scheme is $\Omega(m\sqrt{m})$. This is a simple area argument; the maximal ball centered at $\kappa(q)$ and of radius $r(\kappa(q))$ is by definition empty and completely contained inside Ω, since it just touches the boundary at finitely many points. The area of this ball is $\pi r^2(\kappa(q)))$, and because of the uniform density assumption, this area must be $O(m)$. Thus $r = O(\sqrt{m})$ and we are done.

6.5 Proof of Lemma 4

We prove the following observation used in Lemma 4

Observation 2: Suppose H_i is the longest contour, $i \in [0, 1]$. There is a constant δ such that one of the two cases is true: (i) the contours H_j with $j \in [i, i + \delta]$ have length $\Omega(\text{Length}(H_i))$; (ii) the contours H_j with $j \in [i - \delta, i]$ have length $\Omega(\text{Length}(H_i))$.

There are three cases. Recall that i is the normalized height of the maximum length contour.

Case 1: i is bounded below by a (fixed) constant. Let $j = i - \delta$. We have

$$\begin{aligned} H_i(s) &= C(s) + iN(s)r(s) \\ H_i^{'}(s) &= C^{'}(s) + i(N(s)r(s))^{'} \\ (N(s)r(s))^{'} &= \frac{H_i^{'}(s) - C^{'}(s)}{i} \end{aligned}$$

Further,

$$\begin{aligned} H_i(s) &= H_j(s) + \delta N(s)r(s) \\ H_i^{'}(s) &= H_j^{'}(s) + \delta(N(s)r(s))^{'} \\ &= H_j^{'}(s) + \delta\left(\frac{H_i^{'}(s) - C^{'}(s)}{i}\right) \\ (1 - \delta/i)H_i^{'}(s) &= H_j^{'}(s) - \frac{\delta}{i}C^{'}(s) \\ ||(1 - \delta/i)H_i^{'}(s)|| &\leq ||H_j^{'}(s)|| + ||\frac{\delta}{i}C^{'}(s)|| \\ \text{Integrating, } (1 - \delta/i)\text{Length}(H_i) &\leq \text{Length}(H_j) + \frac{\delta}{i}\text{Length}(H_0) \\ (1 - \delta/i)\text{Length}(H_i) &\leq \text{Length}(H_j) + \frac{\delta}{i}\text{Length}(H_i) \\ \text{Length}(H_j) &\geq (1 - \frac{2\delta}{j})\text{Length}(H_i) \end{aligned}$$

Now choose $\delta = i/4$, we get $j = 3i/4$, and that $\text{Length}(H_j) \geq \frac{1}{3}\text{Length}(H_i)$, and we are done (note that $\delta = i/4$ is bounded below by a constant too, this is why we required i to be bounded below by a constant).

Case 2: $i > 0$, and is not bounded below by a (fixed) constant.

In this case, we take $j = i + \delta$.

$$
\begin{aligned}
H_i(s) &= H_j(s) - \delta N(s) r(s) \\
H_i^{'}(s) &= H_j^{'}(s) - \delta (N(s) r(s))^{'} \\
&= H_j^{'}(s) - \delta \left(\frac{H_i^{'}(s) - C^{'}(s)}{i} \right) \\
(1 + \delta/i) H_i^{'}(s) &= H_j^{'}(s) + \frac{\delta}{i} C^{'}(s) \\
||(1 + \delta/i) H_i^{'}(s)|| &\leq ||H_j^{'}(s)|| + ||\frac{\delta}{i} C^{'}(s)|| \\
\text{Integrating, } (1 + \delta/i) \text{Length}(H_i) &\leq \text{Length}(H_j) + \frac{\delta}{i} \text{Length}(H_0)
\end{aligned}
$$

If Length$(H_0) <$ Length(H_i) then the last inequality becomes strict and we get Length$(H_i) <$ Length(H_j), which is a contradiction. Hence Length$(H_0) =$ Length(H_i), in which case the last inequality becomes Length$(H_i) \leq$ Length(H_j), which by definition of H_i being longest implies Length$(H_j) =$ Length(H_i).

Case 3: $i = 0$, i.e. H_0 is the longest contour.

$$
\begin{aligned}
H_j(s) &= H_0(s) + jN(s) r(s) \\
H_0(s) &= H_j(s) - jN(s) r(s) \\
H_0^{'}(s) &= H_j^{'}(s) - j(N(s) r(s))^{'} \\
||H_0^{'}(s)|| &\leq ||H_j^{'}(s)|| + ||j(N(s) r(s))^{'}||
\end{aligned}
$$

In the proof of Lemma 2, we showed that $||(N(s) r(s))^{'}|| \leq 2$. The last inequality then reads

$$
\begin{aligned}
||H_0^{'}(s)|| &\leq ||H_j^{'}(s)|| + 2j \\
\text{Integrating, Length}(H_0) &\leq \text{Length}(H_j) + 2j \text{Length}(H_0) \\
\text{Length}(H_j) &\geq (1 - 2j) \text{Length}(H_0)
\end{aligned}
$$

Thus $\forall j \leq 1/4$, we have that Length$(H_j) \geq \frac{1}{2}$Length(H_0), and we are done.