A Review on Analytical Approach to Designing Message Ferry Routes

Mr. Chetan R Chauhan, Mr. Shailendra K Mishra

Department of Computer Science and Engineering,

PARUL INSTITUTE, VADODARA, INDIA.

Abstract

Real-world mobile ad-hoc networks are often partitioned into multiple connected components separated from each other by large distances — for example, large-scale rescue operations with teams at different sites. A key paradigm for communication in such networks is message ferrying, where mobile agents such as unmanned aircraft traverse the network geography and carry messages from senders to receivers. End-to-end latency is an important metric for routing in mission-critical settings and ferry routes have to be optimized to minimize messaging de-lays. In this work, we formalize the problem of constructing ferry routes that minimize two aggregate measures of delay (average and maximum), and provide lower bounds on the performance of any routing algorithm for these objectives. We provide a simple scheduling algorithm called PIZZAS that comes within a constant factor of the optimum in idealized but tractable scenarios. Existing work into ferry scheduling uses heuristics to construct specific algorithms; in contrast, we examine the e entire class of latency-minimizing algorithms and provide a theoretical framework to understand their performance.

1 Introduction

Sparse mobile ad-hoc networks are characterized by intermit-tent connectivity between nodes — there does not always exits a path between the source of a message and its destination. Such networks form an important real-world subset of Delay Tolerant Networks (DTNs), a networking paradigm for partially connected networks used in diverse settings such as disaster response, military co-ordination, wild-life monitoring and outer-space networks [7, 3].

Many real deployments of DTNs exhibit clustered topologies [15, 16], where each cluster of nodes has internal ad-hoc connectivity but cannot communicate with other clusters; in other words, the networking graph is disconnected and composed of multiple connected components. An example is a disaster recovery scenario in a flooded city with no working communications infrastructure, where rescue workers armed with short-range radios can interact with their own team but cannot contact teams in other parts of the city.

One approach to communication in such networks is to exploit or arrange for contact between nodes in the system (for example, see [21]). However, in real settings node movement is deter-mined more by strategic and tactical considerations than communication efficiency, and this is particularly true for clustered topologies in disaster scenarios — it is unlikely that a firefighter will battle his way through wildfires to deliver data to a remote team, or that chance encounters will occur between members of different teams. A more promising approach is to use dedicated *message ferries* [27, 28] that range over the theater of operations, picking up packets from senders and delivering them to receivers.

In this paper, we examine the use of aircraft to ferry data in a sparsely connected mobile ad-hoc network. The use of aircraft has been proposed previously for sensor data collection [20, 22] and for message ferrying [24]. Inexpensive Unmanned Aerial Vehicles (UAVs) can be used to ferry data reliably in battlefield and rescue settings [4]; each aircraft sweeps the site, collecting data exchange requests and delivering them as needed. The critical metric for any communication scheme in such scenarios is end-to-end message delay; hence, planning aircraft routes to minimize message delivery latencies is of vital importance.

Accordingly, we provide the first theoretical framework for analyzing the latency characteristics of route planning algorithms for message ferries. While specific heuristic-based algorithms for planning ferry routes have been proposed in literature [28], this paper takes the first step in examining the generic class of ferry routing algorithms and provides provable lower bounds on the best achievable end-to-end latency provided by any such algorithm. In addition, we show that a relatively simple scheduling algorithm comes within a constant factor of the optimal solution.

The results in this paper are applicable to many related problems involving mobile transport agents, such as time-critical data fusion in sensor networks [18]. In fact, the relevance of this work extends beyond communication in distribution systems; a more general example is that of a public transport system where a number of vehicles go around a city, picking up and dropping off passengers at certain points.

We provide an empirical validation of the PIZZAS algorithm and show that it works well even in settings which are analytically intractable.

The rest of the paper is organized as follows: in Section 2, we present our model of communication, and various variants of it. In Section 3, we state our algorithm and give statements of our results. Analytical results and proofs are presented in Section 4 followed by empirical validation for sample scenarios in Section 5. We give a survey of related work in Section 6, and conclude and suggest future directions in Section 7.

2 Modeling and problem definition

In this section we precisely model the problem considered in this paper. The idea here is to capture the core of the real world problem by focusing on a few essential parameters in such a way that the resulting simpler problem is interesting as well as analytically tractable.

2.1 The network model

The connectivity graph of a partitioned ad-hoc network is disconnected and comprises of n connected components (denoted by V), also called *super nodes* or simply *nodes*. The super nodes contain a small number of short-range radios in a relatively small geographic area — they are represented by a *stationary* point in the Euclidean space — and cannot contact other super nodes without external infrastructure.

We consider the use of aircraft as message ferries to transfer data between the n super nodes. To this end, we have a collection of $k \geq 1$ aircraft which fly above super nodes to collect data and deliver it to its destination. Since the geographic size of a super node is relatively small compared to distance between them, we assume that the aircraft has to move (exactly) above it to ex-change data with it. The aircraft move at speed s1,s2,….,sk and denote the sum si equal to adding all values from 1 to k. Once above a super node, the super node can upload/download data to/from the aircraft. The aircraft can also exchange data among themselves, when they are in vicinity of each other in the air. Another method for air-craft to exchange data is for an aircraft to leave data with one super node and for the other aircraft to collect it from the same super node. An important realistic restriction in the model is that when a super node generates a packet, no aircraft is informed of this event: some aircraft has to fly over the super node to learn whether that super node has any packet to transmit.

Let G = (V, E, l) denote the graph with vertex set V (the set of all super nodes) and edge set E with length function $l : E \rightarrow R_{\geq 0}$ on them. As mentioned before, super nodes are assumed to be embedded in the Euclidean plane (hence l is a Euclidean metric). We make the following definition to quantify the *spread* of the network.

Definition 1. *For a graph* G = (V, E, l) *and a node* $v \in V$, *let* $N^*(v) = \arg\min_{\{u|(v,u)\in E\}} l((v, u))$ *denote the closest vertex to* v. *Let* $l_{min} = \min_{e\in E} l(e)$ *denote the shortest edge in* G, *i.e., the distance between closest pair of nodes.*

With n super nodes and k aircraft, the goal is to schedule aircraft's routes so as to facilitate communication among super nodes with least possible delivery delays. In the rest of this section we discuss variations of our model based on (1) the traffic pattern, (2) the measure of end-to-end latency used, and (3) the layout of the super nodes in the Euclidean space.

2.2 Communication Pattern

We consider two abstract communication patterns that cover many real-world scenarios.

Centralized: A central point of control (a `headquarters' HQ) exists that coordinates activity in the theater of operations, and traffic flows from super nodes to the headquarters and back.

Any-to-any: super nodes communicate with each other in peer-to-peer fashion without sending data through a central location.

Accordingly, we use two models: one with a central head-quarter that every super node reports to and one without a head-quarter where nodes coordinate with each other directly. Importantly, the headquarters is merely a location with more advanced and expensive communication equipment connecting it to the outside world. In the context of a disaster rescue situation, this could be a truck with a satellite link communicating with an actual command center outside the theater of operations.

In our analysis, we make a crucial assumption about the way traffic is generated in the network. We assume the traffic is *uniform*; data packets are generated at a rate of R packets per time unit. For each packet, source and destination are chosen uniformly at random from the pool of all possible $n(n - 1)$ pairs. This assumption is meant to model a system where a constant aggregate data rate is spread out across different communicating node pairs. Our analysis can be easily modified to support different traffic generation patterns — for example, settings w here certain node pairs communicate at a constant data rate to mirror periodic real-time sensor traffic.

2.3 Objective function

When we schedule aircraft to fly over super nodes, the objective is to make communication efficient. The meaning of efficiency is different depending on the application; we focus on end-to-end message latency, which includes the time that a message waits at its source until an aircraft arrives, the transit time on the aircraft (to reach the destination), and possibly other waiting times at other nodes. There are two natural ways to define low latency.

Minimum worst-case latency (MinMax objective), the objective it to minimize the worst case latency of any message.

Minimum average case latency (MinSum objective), the objective is to minimize the average latency of messages.

The aptness of an objective function for a particular scenario is highly application-specific; hence we consider both of them in our analytic work.

2.4 Super node placement

An important parameter in the problem definition is the structure of the graph G, how super nodes in V are distributed, and what

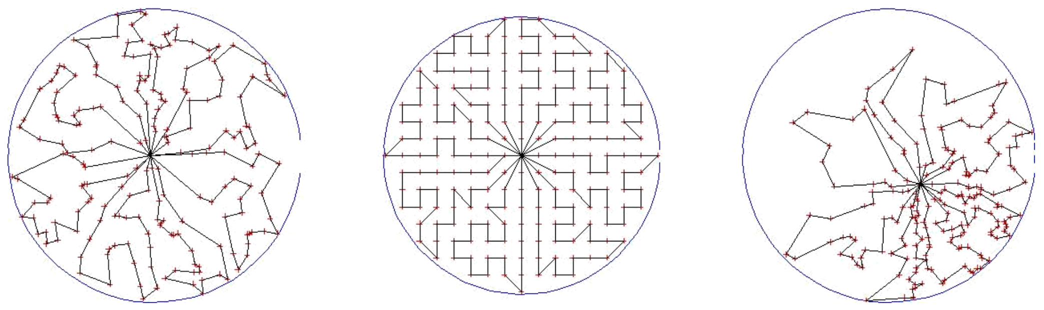

Figure 1: Sample super nodes placements. These are scenarios with headquarters and show trajectories of the PIZZAS algorithm. From left to right: random scenario, regular scenario, and non-uniform scenario.

are their relative distances. We assume that all nodes in graph are distributed in a large circular region of area A. For how they are placed relative to each other, there are two placements that are tractable and natural enough to deserve consideration.

Regular points: the super nodes are placed on lattice points (grid points) in the circle of area A. They are equally spaced in horizontal and vertical direction, see the middle panel of Figure 1.

Random points: the nodes are placed uniformly at random in the circle, see the left panel of Figure 1.

2.5 Eight scenarios

According to the three dimensions of classification described above, we consider 8 models in our analytical work. We use acronyms to refer to them, with a common naming scheme [communication pattern]-[objective function]-[super node placement]. The names of all eight scenarios are then HQ-MinMax-Regular, HQ-MinSum-Regular, NoHQ-MinMax-Regular, and NoHQ-MiniSum-Regular for regular point sets and HQ-MinMax-Random, HQ-MinSum-Random, NoHQ-MinMax-Random, and NoHQ-MinSum-Random for random point sets.

3 Our Results

We start by observing that the problem considered in this paper is strongly NP-hard, assuming a general traffic pattern of communication. This can be seen by a reduction from metric TSP problem and metric k-traveling repairman problem [6]. We omit the proof due to space constraints. Due to the NP-hardness result, we focus our attention on developing approximation algorithms with provable performance guarantees.

As discussed in the previous section, we consider eight models for the analysis: scenarios with and without headquarters, with MinMax and MinSum objective functions, and with super nodes distributed randomly and on a grid. Our main approach is the following. For each model, we first find a lower bound on the performance of any algorithm (in particular the optimum algorithm). Then we prove that a simple algorithm, called PIZZAS (introduced below), achieves the objective function value within a constant factor of the lower bound. This proves that, even using a simple-minded algorithm, we can approximate the objective function within a constant factor of the optimum. We also present a basic empirical validation of the PIZZAS algorithm and compare the results to the theoretical guarantees.

3.1 The PIZZAS algorithm

The PIZZAS[1] algorithm is designed for the scenario with uniform traffic and uniform point distribution, but can be generalized to any setting as shown in Section 5. Given the circular region to be covered with aircraft, PIZZAS algorithm cuts the entire region into k equal sectors of the circle[2], each with inter~~nal~~ angle $\frac{2\pi}{k}$, with one super node common to all areas. It then constructs an approximate minimum TSP (Traveling Salesper-son Problem) tour of the points in each region [2]. See Figure 1 for an example of such routes. The algorithm then sends each aircraft on TSP tour of each sector. Aircraft pick packets from super nodes, deliver them to the destination if destination is on the path of the same aircraft, otherwise, they drop off the packet with the common super node (at the center), from which the appropriate aircraft picks up the packet to deliver it to the destination. In the case of headquarters scenarios, it is assumed that the node common to all regions is the headquarter station.

Note that this two stage process is important in order to utilize the k aircraft efficiently. Simply sending all k airplanes on the same TSP tour, or having them fly around independently, would incur latency that is more than a constant factor away from the optimum.

It is somewhat surprising that such a simple algorithm achieves delivery latencies that are only a constant factor away from latencies incurred by an optimal schedule. We compute an upper bound on this constant factor in the next section.

3.2 Preview of analytic results

We first prove a theorem that will help us with providing lower bounds for routing scenarios in variety of models (Theorem 4). This theorem uses a very simple idea: if aircraft are not given enough time, they cannot cover enough edges of the network to connect all possible pairs. This idea leads to sufficiently strong lower bounds for a variety of models so as to prove constant approximation factors for the PIZZAS algorithm. We then specialize the theorem to the eight models we study.

Recall that l_{min} denotes the minimum pairwise distance be-tween the super nodes of the network, and S denotes the sum of

[1]PIZZAS stands for **P**oints **I**ndependent **Z**ig **Z**ag **A**ccess **S**ystem.

[2]These k sectors of the circle look as if a pizza were being sliced into k slices, hence the name.

speeds of all aircraft. For the models with regular points in a circle (super nodes placed on grid points), we obtain lower bounds on the latency of any algorithm of the form $\Omega(1) n l_{\min} \sqrt{A}/s$ where the constant depends on the particular model. We obtain upper bounds for PIZZAS algorithm, and for such models they look like $O(1) \frac{\sqrt{n}\sqrt{A}}{S}$. We use the fact that for n grid points, the min minimum distance is $\sim \frac{1}{\sqrt{n}}$, and this way get a constant factor

For models in which n points are placed uniformly at random in the circular region of area A, we obtain lower bounds of the form $\Omega(1) \frac{\sqrt{n}\sqrt{A}}{S}$ and upper bounds of the form (using PIZZAS) $O(1) \frac{\sqrt{n}\sqrt{A}}{S}$. Computing the ratio of upper bound to lower bound gives a constant factor approximation for all models.

The constant factors computed by this approach are not necessarily the best attainable. We choose our techniques due to their relative simplicity rather than with the goal of computing the tightest bounds possible. The main message of this paper should be taken as follows: a relatively simple lower bounding technique combined with a naive algorithm can give a constant factor approximation for a diverse variety of models.

4 Analytic results

This section presents the main technical contribution of the pa-per. Out of the eight models mentioned in previous sections, we focus on HQ-MinMax-Random model and present detailed proofs for this scenario. The proofs for the other seven models have a similar flavor and are described in detail in the full version of this paper [19].

We first present a general theorem in Section 4.1 for proving lower bounds in these models, and then specialize it to the HQ-MinMax-Random scenario. In Section 4.2, we prove (up-per) bound on the min-max latency of PIZZAS algorithm, and use it to prove that PIZZAS is a constant factor approximation algorithm for HQ-MinMax-Random scenario. Finally, in Section 4.3, we present results for the other seven models.

4.1 Lower bound on OPT

4.1.1 General theorem for lower bounds

Here we present the main theorem used for proving various lower bounds in this paper. Let us introduce some notation fir st.

Notation 2. *For a graph* $G = (V, E)$, *let* $\mathrm{MSpT}(G)$ *and* $\mathrm{MStT}(G)$ *denote the length of a minimum spanning tree and minimum Steiner tree of* G, *respectively. Let* $\mathrm{MSpT}(G, c)$ *and* $\mathrm{MStT}(G, c)$ *for* $c \geq 1$ *denote the minimum length of a forest on* G *and Steiner forest on* G, *respectively, with* c *connected components in it. We have* $\mathrm{MSpT}(G) = \mathrm{MSpT}(G, 1)$ *and* $\mathrm{MStT}(G) = \mathrm{MStT}(G, 1)$.

The Steiner tree can use points on the plane that are not super nodes; they are called Steiner vertices, or *meeting points* in our model, since aircraft can meet and possibly exchange data at these points. The set of meeting points for algorithm O during time period $[t_0, t_1]$ is denoted by $MP_O([t_0, t_1])$ The following definition of a coverage graph reflects the trajectories of the aircraft during a specified time period. Note that when we talk about edge (u, v) in the Euclidean graph, we mean the straight line between point's u and v.

Definition 3 (Coverage graph)**.** *For an algorithm* O *for routing aircraft and time period* $[t_0, t_1]$, *the* coverage graph $G_O([t_0, t_1])$ *is the graph defined on vertex set* $V \cup MP_O([t_0, t_1])$ *and with edge set* $E_O([t_0, t_1])$ *defined as follows. An edge* $e = (u, v) \in E_O([t_0, t_1])$ *if some aircraft went from* u *to* v *during time period* $[t_0, t_1]$ *(without visiting any other node in between). Note that aircraft need not move on straight line paths; to span the edge* (u, v), *some aircraft needs to go from* u *to* v *along some path (without visiting any other node on the way). The graph*

$G_O([t_0, t_1]) = (V \cup MP_O([t_0, t_1]), E_O([t_0, t_1]))$ *is called the* coverage graph *for algorithm* O *between time* t_0 *and* t_1.

The following theorem, in essence, states that if the aircraft are not allowed to fly for a long enough time period, then they (collectively) cannot span all super nodes of the network.

Theorem 4 (General Lower Bound Theorem)**.** *Let* O *be any algorithm for routing aircraft over the nodes in the Euclidean graph* $G = (V, E)$. *Let* t_0 *be an arbitrary time instance and*

$$t_1 = t_0 + \frac{\mathrm{MStT}(G)}{\sum_{i=1}^{k} s_i} = t_0 + \frac{\mathrm{MStT}(G)}{S}.$$

Then, for any node $u \in V$, *there exists a node* $v \in V$ *such that if* u *wants to send a message to* v *during time* $[t_0, t_1)$, *then aircraft routed according to* O *will not be able to deliver the message to its destination (i.e.,* v*) before time* t_1.

The main idea behind this theorem is simple. If the airplanes do not have enough time to cover a full Steiner tree, then the resulting coverage graph will be disconnected. It then suffices to find a source-destination pair in disconnected components that needs to communicate.

Proof. Let aircraft 1 through k be at coordinates p_1 through p_k respectively at time t_0. Note that these coordinates might not correspond to any super node in the graph G. Let us consider the time interval $[t_0, t_0 + \varepsilon]$. Aircraft i can travel distance $\varepsilon \cdot s_i$ during this time interval. In a similar manner, the total distance covered by all the aircraft is at most $\sum_{i=1}^{k} \varepsilon s_i = \varepsilon S$. Substituting $\varepsilon = \frac{\mathrm{MStT}(G)}{\sum_{i=1}^{k} s_i}$, we get that the total distance covered by all aircraft in the time interval $[t_0, t_1)$ is strictly less than

$$(t_1 - t_0)S = \sum_{i=1}^{k} \frac{\mathrm{MStT}(G)}{\sum_{i=1}^{k} s_i} s_i = \mathrm{MStT}(G).$$

The Coverage graph $G_O([t_0, t_1)) = (V \cup MP_O([t_0, t_1]), E_O([t_0, t_1)))$ is therefore *disconnected*.

Let C(u) denoted the connected component of $G_O([t_0, t_1))$ containing u. There exists another component $C' \neq C(u)$ in the coverage graph $G_O([t_0, t_1))$, since the graph is disconnected. Let v be a super node in C'. As a result of our choice, there is no path between u to v consisting only of edges in $E_O([t_0, t_1))$.

Therefore a packet generated at u during this interval cannot possibly reach v before the end of this interval, because packets are carried by aircraft, and there is a cut separating u and v through which no aircraft is crossing. This proves the theorem.

4.1.2 Lower bound for HQ-MinMax-Random

In this section, we present some strengthened lower bounds for MinMax scenarios. Although the results here can be applied to all scenarios with MinMax objective function, we focus on the scenario with centralized communication pattern and randomly placed super nodes, i.e., HQ-MinMax-Random scenario. The main idea of the proof appears in Theorem 4, we extend that to take into consideration the probabilistic nature of the communication. We first introduce a notation for stating probabilistic lower bounds.

Notation 5. *By a lower bound of* (L, p) *we mean that lower bound of* L *holds with probability at least* p. *Usual lower bounds then mean* $(L, 1)$.

Theorem 6. *Any algorithm for HQ-Minmax-Random asymptotically (as* $n \to \infty$*) suffers a latency of* $\frac{\sqrt{3}}{4\sqrt{18e}}\frac{\sqrt{nA}}{S}$ *with probability arbitrarily close to* 1.

Proof. Consider any algorithm O and a time instance t_0. We will fix t_1 and consider the graph $G_O([t_0, t_1]) = (V \cup MP_O([t_0, t_1]), E_O([t_0, t_1]))$. Unlike in the proof of Theorem 4, we choose t_1 small enough such that the graph $G_O([t_0, t_1])$ has c components. Since the graph spanned by moving aircraft has at least c components (c to be determined later), its cost is at least MStT(G, c), which can be bound using Theorem 10 (see Section 4.1.3). Since the combined speed of all aircraft is S, if we let $t1 = t0 + \frac{MStT(G,c)}{S}$, then the resulting graph $G_O([t_0, t_1])$ has at least c components. Let us define $t = t_1 - t_0$. Recall that S is the sum of speeds of all aircraft. Let us define the notion of a potentially bad pair.

Definition 7. *Let* $G_O([t_0, t_1])$ *be the coverage graph for interval* $[t_0, t_1]$. *An ordered pair* (u, v) *is called* potentially bad *if* u *and* v *belong to different connected components of* $G_O([t_0, t_1])$. *For* $0 \le f \le 1$, *the ordered pair is called* f *-bad if it is potentially bad and* u *sends a message to* v *during first* f *-fraction of the time interval, i.e., during the time interval* $[t_0, t_0 + f t]$.

If there is a potentially bad pair with respect to $E_O([t_0, t_1])$ which sends a message at time t_0 (or the pair is 0-bad according to the Definition 7), then we get a lower bound of $t = t_1 - t_0$ for maximum latency. But the problem is that there might not be a 0-bad pair.

To circumvent this problem, we use the fact that packets are being generated at a uniform rate of R packets per unit time with uniformly chosen source and destination. The idea behind al-lowing only c components instead of 2 as in the proof of the Theorem 4 was precisely to allow many potentially bad pairs. Let us go into the details now.

We first state a claim lower bounding the number of potentially bad pairs with respect to $E_O([t_0, t_1])$ above, whose proof is straightforward and appears in the full version.

Claim 8. *If there are* n *super nodes and* c *connected components in which they reside, then there must be at least* $(c - 1)(2n - c)$ *potentially bad (ordered) pairs. Also there must be at least* $2(c - 1)$ *potentially bad pairs containing a particular super node.*

Now, coming back to the argument for lower bound. For the scenario with a headquarters, we assume that R is the rate of packet generation in the system. In a similar manner as above, the probability that a single (random) packet has source-destination pair as a potentially bad pair is $\frac{2(c-1)}{2(n-1)}$ (there are $2(c - 1)$ potentially bad pairs and $2(n - 1)$ pairs in all). The probability that none of the $Rf(\Delta t)$ packets generated in the first f -fraction of time period $[t_0, t_1]$ is to be carried between two nodes of a potentially bad pair is

$$\left(1 - \frac{2(c-1)}{2(n-1)}\right)^{Rf\Delta t} \le e^{-\frac{c-1}{n-1}Rf\Delta t},$$

using the inequality $1 - x \le e^{-x}$. The expression above gives us the probability that the lower bound of $(1 - f)\Delta t$ does not hold.

The rather complicated expression above does not seem to give much insight into how small or large the lower bound is and how it depends on various parameters. We will now try to make these bounds more readable.

We cannot hope for the bounds to hold with probability one, since there is very small probability that packets are not generated in the system. In such a case, no positive lower bound holds. So, we want lower bounds to hold with high probability, or we want them to be violated with very small probability, say p (p will typically be 0.001). Using this bound on the probability of violation of bounds, we get a relation between f and c in expressions above.

For the headquarters scenario, equating the probability of bound being not satisfied with p, taking log on both sides, and solving for f , we get

$$f = \frac{\log(1/p)(n-1)}{(c-1)R\Delta t}$$

Plugging in this value in the expression for the lower bound, we get

$$(1 - f)\Delta t = \Delta t - \frac{\log(1/p)(n-1)}{(c-1)R}$$

$$= \frac{MStT(G, c)}{S} - \frac{\log(1/p)(n-1)}{(c-1)R}.$$

This lower bound holds with probability $1 - p$.

Lower bound for HQ-MinMax-Random For this case, the length of MStT(G, c) can be bounded using Theorem 11. Plugging in $MStT(G, c) \ge \frac{\sqrt{A}}{\sqrt{18}\sqrt{n}}\left(\frac{n}{\sqrt{e}} - c\right)\frac{\sqrt{3}}{2}$ in the expression for lower bound above gives (we should take c to maximize the lower bound, but we simply use $c = \frac{n}{2\sqrt{e}}$)

$$\frac{\sqrt{3}}{4\sqrt{18e}}\frac{\sqrt{nA}}{S} - 2\sqrt{e}\,\frac{\log(\frac{1}{p})}{R} \approx \frac{1}{16.15}\frac{\sqrt{nA}}{S} - 3.29\,\frac{\log(\frac{1}{p})}{R},$$

which holds with probability at least $1 - p$. We take p to be any small constant (say 0.001 so that the bound holds with probability 99.9%), and ignore the lower order term (since it is small in the limit $n \to \infty$) to get the lower bound of $\frac{1}{16.15}\frac{\sqrt{nA}}{s}$. □

Note that we are really not worrying about constant factors here. For example, just by choosing $c = n^{\varrho}$ (for small $\varrho > 0$) above, we would have gotten an approximation factor equal to half of what we got above, but we are not optimizing the performance guarantees.

4.1.3 Some bounds on MSpT(G, c) and MStT(G, c)

In this section, we prove some bounds on lengths of minimum spanning trees and minimum Steiner trees for n-point sets. The proofs of these results are omitted for space constraints and appear in the full version of the paper [19].

Lemma 9. *Let* $G = (V, E)$ *be a graph and* $E' \subseteq E$ *such that* (V, E') *has* c *components (*E' *could contain cycles). Let* $S \subseteq V$ *be a subset of vertices with* $|S| \geq c$*. Then the number of edges of* E' *with at least one end point in* S *is at least* $|S| - c$*. More precisely,*

$$|\{e = (u, v) \in E' : \{u, v\} \cap S \neq \emptyset\}| \geq |S| - c.$$

Now we derive a lower bound on the length of forests on n point-sets.

Theorem 10 (General lower bound on MStT(G, c))**.** *Let* $G = (V, E)$ *be a graph, and* S *be a subset of* V *with* $|S| \geq c$*. Then, the length of any Steiner forest with* c *components is at least*

$$\max_{\text{MStT}(G,c) \geq S: |S| \geq c} \frac{\sqrt{3}}{2}\left(\min_{v \in S} l(v, N^*(v))\right)(|S| - c),$$

where $N^*(v)$ *denotes the distance of* v *to its closest neighbor.*

The proof is an easy consequence of Lemma 9. It involves first proving a similar result for *spanning* forests (rather than Steiner forests) without a factor of $\frac{\sqrt{3}}{2}$. Going from spanning forest to Steiner forest can decrease the lower bound by at most a factor of $\frac{\sqrt{3}}{2}$ [5].

We now specialize this result to the case of uniformly distributed point-set.

Theorem 11 (Lower bound on MStT(G) for uniformly distributed points)**.** *Let* n *points be distributed uniformly at random in a circle of area* A*, and let the resulting graph be called* G*. Then for any* $c \leq \frac{n}{\sqrt{e}}$*,*

$$\text{MStT}(G, c) \geq \frac{\sqrt{A}}{\sqrt{18}\sqrt{n}}\left(\frac{n}{\sqrt{e}} - c\right)\frac{\sqrt{3}}{2}.$$

The proof involves estimating the distance of a point to its nearest neighbor, using concentration of measure inequalities, and using Theorem 10; see [19] for details.

4.2 Upper bound on PIZZAS and performance guarantee

4.2.1 A general upper bound theorem

In this section, we present a loose upper bound on the length of a TSP tour on "slices of pizza."

Theorem 12 (Length of TSP tour)**.** *Let* n *points be placed (possibly adversarially) in a sector of a disc with internal angle* θ*, inner radius* r_1 *and outer radius* r_2*. Then the length of a tour of all its points is upper bounded by*

$$2((r_2 - r_1)\theta r_2 n)^{1/2} + 2\theta r_2 + (\theta r_2) + (r_2 - r_1).$$

The proof of this theorem is an easy adaptation of correspond-ing result for unit squares in [10]. We omit the detail here. As an application of this bound, we will derive an upper bound on the TSP tour length on a sector of a circle.

Corollary 13. *Let* n/k *points be placed in a sector with internal angle* $\frac{2\pi}{k}$ *of circle of area* A*. The length of TSP tour in the sector is bounded above by* $2\sqrt{2}\frac{\sqrt{n}\sqrt{A}}{k}$.

Proof. Take $r_2 = \sqrt{\frac{A}{\pi}}$, $r_1 = 0$, $\theta = \frac{2\pi}{k}$ and $n = n/k$ in the statement of Theorem 12, and ignore the lower order terms, to get the desired result. □

The constant factor in the above corollary can be improved to $\sqrt{6} = \sqrt{2}\sqrt{3} < 2\sqrt{2}$ (and possibly to even smaller factor) by making a little sophisticated tour, which will improve our re-sults. We do not present this improvement here, since it make the analysis more cumbersome.

4.2.2 Analysis of PIZZAS for minmax objectives

We are now ready to analyze the performance of PIZZAS algorithm and prove that it comes within a constant factor of the optimum. We make an (additional) assumption that the speeds of all aircraft are equal, and they are all equal to s. Therefore, the sum of all speeds S is ks.

Recall that the PIZZAS algorithm chops off the entire region into k equal areas (with one point common to all areas), and constructs an approximately minimum TSP (Traveling Salesper-son Problem) tour of the points in each region (see Figure 1). It sends each aircraft on one TSP tour. Aircraft pick packets from super nodes, deliver them to the destination if destination is on the path of the same aircraft, otherwise, they drop off the packet with the common super node, from which the appropriate aircraft picks up the packet to deliver it to the destination. The algorithm is really simple to implement, and surprisingly gives a constant-factor performance guarantee.

Let us consider the minmax scenario with headquarters. In the worst case, it might take two rounds of an aircraft for a particular packet to be delivered. This might happen if the super node is just after the headquarters on the path of the aircraft, and it generates the packet just after the aircraft has left it. In this case, it will take one round for the aircraft to collect the packet, and another round to deliver the packet to the headquarters. The length of a TSP tour on such region (sector of a circle), divided by common

1
0
8

MinMax	LB ($*\frac{\sqrt{nA}}{s}$)	UB ($*\frac{\sqrt{nA}}{s}$)	Factor
HQ-Random	$\frac{\sqrt{3}}{4\sqrt{18e}}$	$2\sqrt{8}$	91.38
HQ-Regular	$\frac{\sqrt{3}}{4}$	$2\sqrt{8}$	13.06
NoHQ-Random	$\frac{\sqrt{3}}{4\sqrt{18e}}$	$4\sqrt{8}$	182.76
NoHQ-Regular	$\frac{\sqrt{3}}{4}$	$4\sqrt{8}$	26.13

Table 1: Summary of results for MinMax scenarios. All lower and upper bounds are multiplied by $\frac{\sqrt{nA}}{s}$

speed of aircraft, is at most $\sqrt{8}\frac{\sqrt{nA}}{s}$, as proved in Corollary 13. Therefore, the worst case bound we get for maximum latency is $2\sqrt{8}$.

We prove the performance of PIZZAS algorithm in the following theorem.

Theorem 14. *Let* R *be the rate of message generation between headquarters and other nodes. If other points are uniformly distributed, then PIZZAS is a* $2\sqrt{8}\,\frac{4\sqrt{18e}}{\sqrt{3}}$*-approximation algorithm for the HQ-MinMax-Random scenario.*

Proof. For this scenario, we got a lower bound on any algorithm of $\frac{\sqrt{3}}{4\sqrt{18e}}$ and PIZZAS achieves $2\sqrt{8}$. Diving upper bound by the lower bound gives the desired performance guarantee. ☐

This finishes our discussion of performance guarantee of PIZ - ZAS for the scenario with no headquarters, minimizing the maximum delay with random placement of nodes.

4.3 Results for other scenarios

4.3.1 MinMax scenarios

We briefly state the results for other three models which cons ider the goal of minimizing the maximum latency of any packet (MinMax models). The results are obtained in a manner very similar to the last section, with minor modification in estimation of l_{min} for regular points. In Table 1, the first column specifies name of the scenario, second and third columns contain lower and up-per bound respectively, followed by the performance guarantee in the fourth column, which is equal to the upper bound divided by the lower bound.

4.3.2 Minsum objective function

In this section, we mention results for the four models with min-sum objective function. Recall that the goal in this model is to minimize the average latency of packets, not the maximum latency. The proofs are mostly similar to the last section (for minmax objective function), and they can be found in the full version.

The one main difference we would like to point out between minmax scenarios and minsum scenarios is the following: in the minmax scenarios, even one packet taking long causes the algorithm to take long, so proving lower bounds is comparatively easier. If we want to prove lower bounds for min-sum objective, then we need to prove that there are many packets that take long time to be delivered. This calls for another machinery that is commonly used in randomized computation: Chernoff bounds.

MinSum	LB ($*\frac{\sqrt{nA}}{s}$)	UB ($*\frac{\sqrt{nA}}{s}$)	Factor
HQ-Random	$\frac{(1-\delta)\sqrt{3}}{32\sqrt{18e}}$	$\sqrt{8}$	602.65
HQ-Regular	$\frac{(1-\delta)\sqrt{3}}{32}$	$\sqrt{8}$	52.26
NoHQ-Random	$\frac{(1-\delta)\sqrt{3}}{32\sqrt{18e}}(2-\frac{1}{2\sqrt{e}})$	$2\sqrt{8}$	710.36
NoHQ-Regular	$\frac{(1-\delta)\sqrt{3}}{64}$	$2\sqrt{8}$	69.67

Table 2: Summary of results for MinSum scenarios. All upper and lower bounds should be multiplied by a factor of $\frac{\sqrt{nA}}{s}$.

We use Chernoff bounds to prove that a large fraction of packets take long to reach their destination, and prove lower bounds based on them.

Without going into any further details, Table 2 present results for the minsum scenarios. Column 1 shows the name of scenarios, column 2 and 3 show lower and upper bounds, followed by column 4 which states the performance guarantee (or the approximation factor), which is just the quotient of upper bound and the lower bound. The lower bounds in the table contain the factor of $(1-\delta)$, which is there for high-probability results (while applying Chernoff bounds). The number δ can be made as small as required.

With this we conclude our theoretical results. We would like to remind the reader of the discussion about the magnitude of constant factors at the end of Section 3.2. The constant factor algorithms are important from theoretical point of view, but the performance of an algorithm should also be evaluated using simulations. As a first step in this direction, we provide a simplified empirical verification of PIZZAS in the next section.

5 Empirical Validation

As a proof of concept of the proposed PIZZAS algorithm, we implemented it and tested its performance in a highly simplified simulation, yet one that allows us to assess the relevance of the theoretical analysis. The main aim of this empirical validation is to compare the obtained results with the theoretical bounds de-rived in Section 4, as well as to test the performance of PIZZAS in a more realistic setting which is less amenable to analysis.

We consider three scenarios in the validation: random and regular as in the analytical sections, plus non uniform scenario which we describe later. All scenarios have experiments both with the headquarters-centered communication pattern (HQ) and any-to-any pattern. The HQ communication means that messages are going either from a super node to the HQ, or the other way around, while the any-to-any pattern is such that supernodes send messages to each other. But even in the any-to-any case we assume existence of a node that is able to store and pass mes-sages as the aircraft pass over. For simplicity of the text, we call this node an HQ. In the random scenario, super nodes are positioned uniformly at random in a circular area (and with the HQ in the center), and data to be sent is generated with the same rate at every node. In the regular scenario, super nodes are positioned on a grid in a circular area. Finally, in the non-uniform scenario, the super nodes are scattered around according to a normal distribution (rather than uniform), and the data generation rate varies from super node to super node. We hope that this scenario more closely resembles a potential real rescue situation. Details about the scenarios and values used are given below.

We record the delivery delays experienced by individual messages, and compute the average and maximum delays for each scenario. We then compare these results with the theoretical bounds.

5.1 Scenarios

For both uniform and non-uniform scenarios we used parameter values that, to the best of our knowledge, would be reasonable in a possible real world large scale rescue situation. There are 200 super nodes (corresponding to 200 rescue teams) that are spread in a circle with radius of 10km. In the random case, these super nodes are placed uniformly at random in the circle, in the regular scenario the nodes are placed on a grid that fills the circular region, and the non-uniform scenario draws each coordinate from a two dimensional normal distribution with the mean off the center closer to the edge. All three cases are depicted in Figure 1. There are 8 aircraft to serve the area, and their speed is 35m/s (approximately 80 miles per hour).

Traffic is generated at each super node using a Poisson process, with uniform rate of one message per 10 minutes on each node in the uniform and regular scenario, and varying rates for the non-uniform scenario. The messages are either sent to a randomly chosen destination (any-to-any communication), or all of them are sent to the HQ. The HQ station can then further relay the messages over more powerful wireless or satellite link to the true headquarters of the rescue operation, which can be off-site. Additionally, the HQ replies to each packet it receives, with a 5 minute delay. The overall time being simulated is 10 hours, and approximately 24, 000 messages are being delivered in that time period. Sufficient time at the end is allowed to deliver messages in transit.

5.2 PIZZAS Implementation

Here we discuss the particular implementation of the algorithm introduced in section 3.1. First there is a preparation phase to discover the routes for the aircraft:

1. Find a center of the mass of all the super nodes, weighted by their rate of data generation. This will be the position of the HQ station.
2. Find k (number of aircraft) sections of the field, centered at the HQ position, such that each section contains approximately the same weight of the super nodes. Each of these sections will be serviced by one aircraft.
3. For every such section, find an approximation to the TSP tour of the super nodes contained in the section, also going through the HQ. In our implementation, this is done using package tsp solve [8]. The cycles are the sought-for scheduled routes for corresponding aircraft.

Then the actual algorithm proceeds as follows:

4. The aircraft all start from the HQ location, each following its path (loop) in one section.
5. When an aircraft arrives at a super node, it off-loads all messages it carries for that super node, and receives all messages the super node wants to send out.
6. At the HQ, the aircraft off-loads all messages that are to be de-liver outside its section, and receives from the HQ messages that are to be delivered in its section.

5.3 Comparison of theoretical and empirical results

Finally, we compare the analytical performance bounds from Section 4 with the empirical results. The comparison confirms that analytic lower bounds and upper bounds correctly predict the behavior of PIZZAS algorithm, as well as the variation among various models.

We plug in the parameters used in the empirical study into the theoretical bounds summarized in Tables 1 and 2 and compare the numbers with the empirical results. The parameter values are: n = 200, s = 35m/s, k = 8, and $A = \pi \cdot (10000)^2 m^2$.

Random points	LB	PIZZAS	UB
HQ-MinMax	55.41	2186.08	5064.14
NoHQ-MinMax	55.41	3773.28	10128.28

Regular points	LB	PIZZAS	UB
HQ-MinMax	387.64	1997.63	5064.14
NoHQ-MinMax	387.64	3883.62	10128.28

Random points	LB	PIZZAS	UB
HQ-MinSum	4.20	956.77	2532.07
NoHQ-MinSum	7.12	1663.98	5064.14

Regular points	LB	PIZZAS	UB
HQ-MinSum	48.45	1097.48	2532.07
NoHQ-MinSum	72.68	1866.33	5064.14

In the non-uniform scenario, the average delays with and without HQ are HQ-MinSum=762.29 and NoHQ-MinSum=1365.51, and maximal experienced delays are HQ-MinMax=2113.38 and NoHQ-MinMax=3506.47. We have no theoretical bounds for this scenario, but these numbers compare very well with the other scenarios, suggesting that PIZZAS is applicable to a wider range of settings than those listed in this paper.

Note that the empirically obtained numbers all fall more or less in the middle of the theoretical bounds. In addition, when the bounds double, as is mostly the case when comparing scenarios with and without HQ, the empirically observed latencies also roughly double. This suggests that although the rigorous bounds may not be tight enough to predict exactly what the latency will be, they capture very well how the PIZZAS algorithm *scales* when going from one setting to another.

6 Related work

Delay Tolerant Networks (DTNs) were introduced in [7], and routing within such networks was considered in [9]. The concept of mobile routing nodes was proposed in [26]; in this work, the authors proved that designing an optimal ferry route is NP-hard in the case where nodes are stationary. In subsequent work [28], the authors considered the case of multiple ferries and stationary nodes, and used heuristics to design good routes. Also, [29] introduced proactive moment into the model, where nodes or ferries could alter routes to facilitate a rendezvous. In relation to this body of work, we consider the case of multiple ferries and stationary super nodes (multi-node connected components) and provide strong analytical results on performance for an entire class of algorithms as opposed to heuristics.

Alternatives to message ferrying in partitioned mobile ad-hoc networks involve exploiting chance encounters between nodes to exchange data, as in epidemic routing [21], or modifying node routes to allow for communication [12]. Voila [17] examines set-tings very similar to the ones we consider, with multiple components isolated from one another; communication is undertaken by handing off data to nodes that are likely to travel between components. In VADD [25], traffic patterns and road layouts are used to predict node movement for communication in vehicular networks. Other examples of exploiting existing mobility include [11] and [23].

Separately, Data MULEs [18] were proposed as mobile collection agents in sensor networks. Space-filling curves have been proposed as candidate routes for efficiently traversing sensor net-works [14]. Much of this work deals with different applications (data collection) and different metrics (ferry mileage), but our results could be used in sparse sensor networks where the timeliness of data aggregation is critical.

The use of aircraft as mobile aggregators in sensor networks was considered in [22] and [20]. Also, [1] proposes a combination of high-power ground nodes called gateways and high-altitude aircraft or satellites to provide communication for partitioned networks — messages are first routed within a connected component to the gateway, and then relayed via the aircraft or satellite to other components.

7 Conclusion

Message ferrying is a key routing technology for partitioned mobile ad-hoc networks in large-scale settings such as disaster response, and can be realized through the use of unmanned lightweight aircraft. End-to-end message delays are of critical importance in such settings and depend entirely on the routes of the ferry aircraft. In this paper, we provide the first theoretical study of latency-minimizing ferry routing algorithms and provide rigorous bounds on the achievable latencies of any such algorithm. We also propose a routing algorithm called PIZZAS that comes within a constant factor of the optimal latency and empirically show that it works well in a variety of scenarios. We consider this work to be the first step in providing a rigorous framework for ferry routing, allowing for a theoretical understanding of such algorithms that can be extended to different metrics and to more specific models.

References

[1] M. Ahmed, S. V. Krishnamurthy, R. H. Katz, and S. Dao. Trajectory control of mobile gateways for range extension in ad hoc networks. *Computer Networks Journal (COMNET)*, August 2002.

[2] Sanjeev Arora. Polynomial time approximation schemes for Euclidean Traveling Salesman and other geometric problems. *J. ACM*, 45(5), 1998.

[3] V. Cerf et al. RFC 4838: Delay-tolerant networking architecture, 2007.

[4] C.M. Cheng, P.H. Hsiao, HT Kung, and D. Vlah. Maximizing Throughput of UAV-Relaying Networks with the Load-Carry-and-Deliver Paradigm. *IEEE Wireless Communications and Networking Conference (WCNC)*, 2007.

[5] D. Z. Du and F. K. Hwang. The Steiner ratio conjecture of Gilbert and Pollak is true. *Proc Natl Acad Sci*, 87, 1990.

[6] Jittat Fakchaoenphol, Chris Harrelson, and Satish Rao. The k-travelling repairman problem. In *SODA*, 2003.

[7] K. Fall. A delay-tolerant network architecture for challenged internets. *Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003.

[8] Chad Hurwitz. Library for solving TSP problems http://www.cs.sunysb.edu/~algorith/implement/-tsp/distrib/tsp solve/.

[9] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *SIGCOMM Comput. Commun. Rev.*, volume 34:4, 2004.

[10] E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Khan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinato-rial Optimization*. John Wiley and Sons, 1985.

[11] J. LeBrun, Chen-Nee Chuah, D. Ghosal, and M. Zhang. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. In *Vehic-ular Technology Conference, 2005 VTC 2005-Spring IEEE*, 2005.

[12] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. *Proc. of the 6th annual international conference on Mobile computing and networking*, 2000.

[13] Gabor Lugosi. Concentration-of-measure inequalities. www.econ.upf.es/ lugosi/anu.ps.

[14] S. Patil, SR Das, and A. Nasipuri. Serial data fusion using space-filling curves in wireless sensor networks. *IEEE SECON*, 2004.

[15] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *IEEE Communi-cations Magazine*, 44(11), 2006.

[16] R. Shah and N.C. Hutchinson. Delivering Messages in Disconnected Mo-bile Ad Hoc Networks. *ADHOC-NOW*, 2003.

[17] R. Shah, N.C. Hutchinson, and W.S. Evans. Voil'a: delivering messages across partitioned ad-hoc networks. *Proc. of the 29th Annual IEEE Inter-national Conference on Local Computer Networks (LCN)*, 2004.

[18] R.C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks*, 1(2-3), 2003.

[19] Yogeshwer Sharma, Lukas Kroc, Mahesh Balakrish-nan, and Ken Birman. An Analytical Approach to De-signing Message Ferry Routes (with complete proofs). http://www.cs.cornell.edu/~mahesh/pizzas-full.pdf.

[20] L. Tong, Q. Zhao, and S. Adireddy. Sensor networks with mobile agents. *Proc. of 2003 Military Communications Intl Symp*, 2003.

[21] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. *Duke University*, 2000.

[22] P. Venkitasubramaniam, S. Adireddy, and L. Tong. Sensor networks with mobile access: optimal random access and coding. *IEEE Journal on Se-lected Areas in Communications*, 22(6), 2004.

[23] H. Wu, R. M. Fujimoto, R. Guensler, and M. Hunter. Mddv: a mobility-centric data dissemination algorithm for vehicular networks. In *MOBICOM Workshop*, 2004.

[24] J. Yang, Y. Chen, M. Ammar, and C. Lee. Ferry replacement protocols in sparse MANET message ferrying systems. *Wireless Communications and Networking Conference, IEEE*, 4, 2005.

[25] J. Zhao and G. Cao. VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks. *IEEE INFOCOM*, 2006.

[26] W. Zhao and M. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *Proc. IEEE Workshop on Future Trends in Distributed Computing Systems*, May 2003.

[27] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. *Proc. of the 5th ACM interna-tional symposium on Mobile ad hoc networking and computing*, 2004.

[28] W. Zhao, M. Ammar, and E. Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. *IEEE INFOCOM*, 2005.

[29] Wenrui Zhao, Mostafa H. Ammar, and Ellen W. Zegura. A message fer-rying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc*, 2004.

Author's Profile.

Mr. Chetan R Chauhan is pursuing M.E in PARUL INSTITUTE, VADODARA(INDIA).
Mr. Shailendra K Mishra is serving as a Professor in PARUL INSTITUTE, VADODARA(INDIA).