Load Balanced Short Path Routing in Wireless Networks

Jie Gao, *IEEE member*, Li Zhang

***Abstract*— We study routing algorithms on wireless networks that use only short paths, for minimizing latency, and achieve good load balance, for balancing the energy use. We consider the special case when all the nodes are located in a narrow strip with width at most $\sqrt{3}/2 \approx 0.86$ times the communication radius. We present algorithms that achieve good performance in terms of both measures simultaneously. In particular, the routing path is at most 4 times the shortest path length and the maximum load on any node is at most 3 times that of the most load-balanced algorithm without path length constraint. In addition, our routing algorithms make routing decisions by only local information and in consequence are more adaptive to topology changes due to dynamic node insertions/deletions or due to mobility.**

Keywords: wireless network, load-balanced routing, short path routing

I. Introduction

A mobile ad-hoc network consists of autonomous devices that can directly communicate to their nearby nodes. Nodes that are not within direct communication range use other nodes to relay messages between them. Routing in such an ad-hoc multi-hop network is challenging due to the lack of central control and the high dynamics of the network. Previous work has focused on discovering and maintaining routes that keep the connectivity between the nodes, or furthermore, that minimize the number of hops on a path. One important restriction of a wireless network is that nodes are energy constrained as they are normally powered by batteries. Besides minimizing latency, another good reason for using the shortest path routing is that it is, in some sense, good for overall energy efficiency because the total energy needed to transmit a packet is correlated to the hop length of a path. However, the algorithms that aim to minimize the path length may ignore fairness in routing — for example, the shortest path routing is likely to use the same set of hops to relay packets for the same source and destination pair. This will heavily load those nodes on the path even when there exist other feasible paths. Such an uneven use of the nodes may cause some nodes to die earlier, thus creating holes in the network, or worse, leaving the network disconnected. In addition, unbalanced use of the nodes may discourage them to participate in the routing process.

Jie Gao is affiliated with the Center for the Mathematics of Information, California Institute of Technology, Pasadena, CA 91125. E-mail: jgao@ist.caltech.edu. Work was done when the author was at Department of Computer Science, Stanford University, Stanford, CA 94305. Li Zhang is affiliated with Hewlett-Packard Labs, 1501 Page Mill Road, Palo Alto, CA 94304. E-mail: l.zhang@hp.com.

In a wireless network, the biggest energy drain usually comes from the transmission of packets. In this paper we measure the energy consumption of a node by the total size of packets relayed by the node. In our study, we assume the usual model that each node has a fixed transmission radius. Nodes within each other's transmission range can directly communicate with each other. The energy consumption for the direct transmission between such two nodes is independent of their geometric distance[1]. Then the load-balanced routing can be formulated as to minimize the maximum load on the nodes in the network. The ideal algorithm would be to minimize both the latency and the maximum load simultaneously. However, these two goals are conflicting to some extent: the shortest path routing restricts the resources that can be used, while load-balanced routing aims to use all the available resources to even the load. One can construct an example to show that these two goals are indeed conflicting, i.e. shortest path routing algorithm necessarily creates heavily loaded nodes, and the optimum load-balancing algorithm necessarily uses long paths (See Appendix IX-A for an example). In practice, the nodes are often distributed in special ways such that we may be able to achieve good, though not necessarily the best, performance in terms of both measures simultaneously. In this paper, we consider a special case arising from practice and present algorithms whose performance is within a small constant factor of the optimum solution in terms of both measures.

The case we consider is when the nodes are located in a "narrow" strip with width at most $\sqrt{3}/2 \approx 0.86$ times the communication radius of each node. This model captures the situation when the nodes are on highways or along streets, for examples, when the wireless network is built for inter-vehicle communication [10] or for people walking on streets. In such cases, routing can be done in two phases. In the first phase, the nodes figure out the "meta-path" needed to route a packet with the aid of the position information and the underlying transportation network map, which is normally static and easily available. In the second phase, the actual routing path is realized with the guidance of the meta-path. The problem then reduces to routing for nodes located in a narrow strip.

Even for this restricted scenario, the problem of planning routes that achieves the best load-balancing is still difficult. It is NP-hard to compute the most balanced routes, even in a very simple network. There have been approximate algorithms developed for the problem. But none of the previous algorithms is local, i.e. they all require

[1]Advanced power control techniques are not considered in this paper.

global coordination, and the approximation ratio often has only theoretical interest. We show that when the nodes are located in a narrow strip, there exists an efficient algorithm that approximates the optimum solution within a small constant factor.

What makes routing on a narrow strip easier is that the greedy forwarding guarantees to find a path, if such a path exists. This is obvious when the nodes are aligned on a line. We show that it is also true for a strip with width at most $\sqrt{3}/2$ times the node communication range. However, even when the forwarding direction of a packet is obvious, there is still freedom to choose to which node, among all the nodes in the neighborhood, to relay the packet. If we wish to achieve shortest path routing, it is appropriate to use the greedy method of sending the packet to the furthest reachable node in the correct direction. However, this may create heavily loaded nodes. On the other hand, if we adopt the greedy strategy of forwarding a packet to the node with the lightest load, it may result in extremely long path. In this paper, we combine the greedy strategies for minimizing the path length and for minimizing the load to achieve constant competitive ratios of both measures.

The basic idea of our methods is that we maintain, for each node, a set of edges, called *bridges*, that are guaranteed to make substantial progress measured by the relative distance to the destination. Every time a node chooses the "lightest" bridge to relay a packet. This way, we show that our algorithm has good performance in terms of both path length and maximum load. In addition, we show that the bridges can be dynamically maintained by using only local information. Specifically, we can guarantee the following properties of our algorithm.

1) It uses only short paths: the number of hops of the path used is at most four times as many as the number of hops of the shortest path algorithm. If the nodes are aligned on a line, the length of the routing path is at most twice the shortest path length. These bounds are theoretical worse-case bound. We observed through simulation that the competitive ratios are actually smaller under reasonable traffic patterns.
2) It balances the load: the maximum total size of packets passed on any node is at most three times as much as that of the optimum load balanced routing.
3) It is localized and scales well to large networks: each node only needs information in its local neighborhood to make routing decision; and as a consequence, our algorithm is very adaptive under dynamic changes and mobility as only a node's neighborhood is affected.
4) It is online: the routing decision of a packet depends only on the current state of the network, which is determined by previously routed packets. It does not need to know the traffic pattern in the future.

We start with an important special case when all the nodes are aligned on a line. In this case, we can achieve a better approximation factor of two for path length. In addition, we show that by distributing a collection of binary search trees on the nodes, we can reduce both memory needed on each node and the routing/update cost when a node has many nodes in its neighborhood. We then extend the algorithms to the case when the nodes are within a narrow strip.

In addition to providing rigorous analysis, we have also implemented the algorithm and studied the performance by simulation. We simulate the algorithm under different traffic patterns and compare it to the shortest path routing. The good performance of our algorithm is supported by the simulation results as well. For example, even for random traffic pattern, under which we would expect the shortest path routing works well, the maximum load created by our algorithm is only about 20% of the shortest path routing. We also compare the number of hops in the path produced by our algorithm to that in the shortest path and show that the path length is only increased by a small fraction.

We should emphasize that in our study, we only consider two high level measures, namely, the path length and the load. In reality, the efficiency of routing in a wireless network depends on many other factors as well. In particular, the various issues at MAC layers such as channel fading, interferences, and collisions can have great impact on the quality of routing. Thus, the MAC layer implementation choices may influence higher level routing decisions. Like most studies in the area, we have chosen to separate the study between the higher level routing decision and lower level MAC implementation. It is expected that in practice the MAC issues be taken into account to achieve good overall performance.

A. Related work

Our work for the nodes in a narrow strip is closely related to the on-line load-balancing problems on related machine model. Azar's paper [2] and Borodin and El-Yaniv's book [3] contain excellent survey of this subject. Load-balancing routing in general can be formulated as the unsplittable flow problem where we aim to minimize the maximum node congestion. This is a well-known NP-hard problem that can be approximated to a factor of $O(\log n/\log\log n)$ [18], [17]. In all the previous work, one either ignores the length of the path or uses only shortest paths for regular networks such as meshes. Another related problem is the on-line virtual circuit routing problem, which has also been studied extensively [16], [2], [3].

There have been extensive study on routing in wireless networks in recent years. Among various metrics used for evaluating the routing quality, the most common one is probably the number of hops on the routing path. The protocols that use shortest path routing include Dynamic Source Routing (DSR) [12], Ad-hoc On-demand Distance Vector routing (AODV) [15] and many others. Please refer to the survey [19] and the references therein.

On the other hand, energy-aware routing algorithms, which try to maximize the network survivability, have attracted considerable interest [10-27]. Energy aware metrics, such as “maximize time to partition” and “minimize maximum node cost”, were first proposed by Singh *et al.* [21]. Chang *et al.* [4], [5] used a flow augmentation algorithm and a flow redirection algorithm to balance the energy consumption on different nodes. Their method, however, requires a full knowledge of traffic demands and does not handle node insertion and deletion. Extensions along this approach were addressed in [22], [28]. Li *et al.* [13] studied the online power-aware routing which minimizes the earliest time when a packet can not be sent. They proved that any online algorithm has unbounded competitive ratio and provided algorithms with zone-based heuristics. Stojmenovic *et al.* [23] and Yu *et al.* [27] proposed methods that use the geographical locations of wireless nodes for energy aware routing. Xu *et al.* [25] proposed an algorithm GAF which is designed to reduce the energy consumption by turning off unnecessary nodes. Energy-unaware routing protocols such as DSR [12], AODV [15], and geographical routing were re-visited to take into account the energy-aware metric [26], [9], [14]. All of the energy-aware protocols mentioned above are heuristics and do not provide any guarantee on the performance.

B. Outline

The paper is organized as follows. In Section II, we introduce some definitions and notations. In Section III, we describe the algorithm for the nodes that are aligned on a line and its efficient implementation. Then, we show that the similar technique can be extended for nodes that are inside a narrow strip in Section IV. In Section V, we discuss some considerations for reducing overhead and for handling dynamic changes. We present the simulation results in Section VI.

II. Models and definitions

Wireless nodes can be modeled as a set of points S in the plane. Let n denote the number of points in S. We assume the communication range of each node is 1. The *communication graph* of S is an unweighted unit-disk graph $U(S) = (S, E)$, where $(p, q) \in E$ if the Euclidean distance between $p, q \in S$ is at most 1. When $(p, q) \in E$, they are also said *visible* to each other. The *length* of a path P, denoted by $|P|$, is the number of hops on the path. For $p, q \in S$, denote by $d(p, q)$ the length of the shortest path between p and q. For any path P between p, q, the *stretch factor* $s(P)$ is defined to be $|P|/d(p, q)$. If $s(P) \le \alpha$, P is called *α-short.*

A routing request has the form $r = (s, t, \ell_r)$ where s, t, ℓ_r represent the source, destination, and packet size, respectively. To route a request r, a path P_r between s and t is allocated to relay the packet. For a set of requests R, a path set $\mathcal{P}$ satisfies R, denote by $\mathcal{P} \models R$, if $\mathcal{P} = \{P_r \mid r \in R\}$ where P_r is used to route r. The stretch factor of $\mathcal{P}$ is defined to be the maximum stretch factor of the paths in $\mathcal{P}$. $\mathcal{P}$ is called α-short if every path in $\mathcal{P}$ is α-short. For example, the shortest path routing algorithm always produces 1-short paths.

For a set of requests R satisfied by $\mathcal{P}$, the *load* $\ell(v)$ incurred to $v \in S$ is the total size of the packets that pass through v, i.e.

$$\ell(v) = \sum_{r \mid v \in P_r} \ell_r \, .$$

The maximum load $\ell(\mathcal{P})$ of $\mathcal{P}$ is defined to be the heaviest load on any node, $\max_{v \in S} \ell(v)$. Denote by $\ell^*(R)$ the load of the most balanced routing, i.e. $\ell^*(R) = \min_{\mathcal{P} \models R} \ell(\mathcal{P})$. The *load-balancing ratio* of $\mathcal{P}$ is then defined to be $\ell(\mathcal{P})/\ell^*(R)$. An algorithm is said *$\beta$-balanced* if for any set of requests R, the load-balancing ratio is at most β. In this paper, our goal is to design wireless routing algorithms with both small stretch factor and small load-balancing ratio.

III. Load-balanced routing on a line

In this section, we focus on the special case when all the nodes are aligned on a line. Later on, we show how the similar technique can be extended to the case when the nodes are in a narrow strip. We first describe an algorithm that achieves both constant stretch factor and constant load-balancing ratio without worrying about the algorithmic issue. We then show that the algorithm can be implemented efficiently and distributedly.

We start with the case when all the requests have unit packet size. In this case, we show a 2-short and 2-balanced routing algorithm. The method for unit packet size fails for variable packet size. By using a different technique, we can achieve the same stretch factor but a slightly worse load-balancing ratio.

A. Hardness of the problem

Before presenting the algorithms, we first show that optimizing the load balancing ratio is difficult even for a simple network, as shown in Figure 1(i). Suppose that each node x_i wishes to send a packet with size ℓ_i to node y_i. They have to choose, from z_1, z_2, a node to relay the packet. The optimum solution is then the most even distribution of the packets on z_1, z_2. This is at least as hard as the subset-sum problem[2], a well-known NP-hard problem [8].

In fact, if there are m nodes $z_1, \cdots, z_m$, inside the intersection of the communication ranges of x_i's and y_i's, then minimizing the maximum load on the m nodes becomes the on-line load balancing problem on m identical machines. Even obtaining an approximation within a ratio of 1.852 has been proven NP-hard [1].[3]

Next, we show that it is impossible to optimize both the stretch factor and the load-balancing ratio. In Figure 1(ii), when x_i sends a packet to y_i, if we insist on using the shortest path, then all the packets have to pass the node z while we may evenly distributed the packets as shown in the figure.

[2]A variant of the subset-sum problem is: in a given set of integers, does any subset sum to exactly half of the total sum?

[3]When the packet size is uniform, we do not know if the problem is still NP-hard.

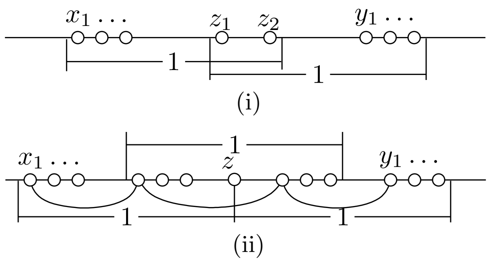


Fig. 1. Load-balanced routing is hard. The problem in (i) is equivalent to the subset sum problem. In (ii), the shortest path from x_i to y_i all pass through z, but one can evenly distribute the load by using the path as shown in the figure.

B. Requests with unit packet size

Suppose that all the nodes lie on a line. For each node $p \in S$, denote by x_p the coordinate of p. Then the communication range of p is the interval $I(p) = [x_p - 1, x_p + 1]$. Define the left (right) communication range of p as $I_l(p) = [x_p - 1, x_p)$ $(I_r(p) = (x_p, x_p + 1])$.

The algorithm GREEDY1 works as follows: Each node p_i keeps track of $\ell(p_i)$, the total number of packets it has relayed so far, and also the maximum load in its left and right communication range (p_i exclusive), denoted by $\ell_l(p_i)$ and $\ell_r(p_i)$, respectively. When a node p_i receives a new request with destination t, it checks if t is within its communication range. If it is, then p_i simply sends the request to t. Otherwise, assume t is to the right of p_i, then p_i sends the request to the furthest node in its right communication range whose load is strictly smaller than $\ell_r(p_i)$, the maximum load in the right communication range $I_r(p)$. In other words, p_i chooses the next hop to be as far as possible without increasing the maximum load in its right communication range. If all the nodes in $I_r(p_i)$ have the same load, i.e., the maximum load $\ell_r(p_i)$, then p_i sends the request to the furthest node in $I_r(p_i)$. In this case, the maximum load in the right communication range, $\ell_r(p_i)$, is increased by 1.

GREEDY2 is obtained by adding one look-ahead to GREEDY1: Whenever a receives a request, it finds the next hop b according to GREEDY1 and then asks b to find the next hop c. If c is in a's communication range, then a shortcuts b and sends the packet directly to c. Otherwise, a sends the request to b.

Theorem 3.1. *GREEDY2 is 2-short.*

Proof: Suppose that P_r is a left to right path produced by GREEDY1. Take any four adjacent nodes, say a, b, c, d from left to right, along P_r. We claim that a and d are not visible to each other. Suppose otherwise, then

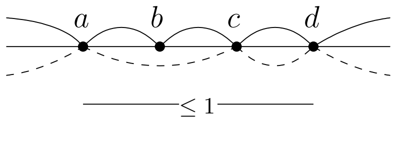


Fig. 2. For any four adjacent nodes a, b, c, d along a path generated by GREEDY1, a and d are not visible to each other.

a, b, c, d are all mutually visible, as shown in Figure 2. Since c, d are both in $I_r(b)$ and b chooses c instead of d to be the next hop, we must have that $\ell(c) < \ell(d)$ by the greedy forwarding strategy. Therefore $\ell(c) < \ell(d) \leq \ell_r(a)$. That is, c is a node further away from a than b and c does not have the highest load in a's right communication range. By GREEDY1, a should have chosen c or a node to the right of c to be the next node, contradicting with the fact that b is the next hop of a on path P_r.

Since GREEDY2 does one-hop look-ahead based on GREEDY1, a direct consequence of the above fact is that for any two non-adjacent nodes a, b on a path produced by GREEDY2, they are not visible to each other. This also explains why one shortcut is sufficient in GREEDY2. Therefore, the total number of nodes used by GREEDY2 is at most twice that of the shortest path routing, since any unit interval has at most two nodes on a path produced by GREEDY2 and at least one node on the shortest path. This proves that the stretch factor of GREEDY2 is no more than 2. □

Theorem 3.2. *GREEDY2 is 2-balanced.*

Proof: Suppose the maximum load of any node in the network is $\ell(\mathcal{P})$, after a set of requests R have been routed. We consider the first time when the maximum load on all the nodes reaches $\ell(\mathcal{P})$. Suppose that node i has the maximum load $\ell(\mathcal{P})$ after i relays the request r. That is, at this point no other node has load equal or more than $\ell(\mathcal{P})$, and $\ell(i) = \ell(\mathcal{P})$. Assume that the forwarding direction of request r is from left to right. Since i relays

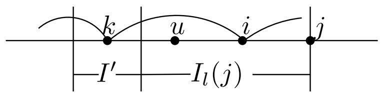


Fig. 3. Load-balancing competitive ratio of GREEDY2.

the request to a node to its right, then there must be at least one node in i's right communication range. Suppose that j is the node to the immediate right of i in S, and k is the previous hop of i on the path P_r (Figure 3). According to GREEDY2, the reason that k chooses i as the next hop and therefore increases the maximum load in k's right communication range, i.e., $\ell_r(k)$, is because i is the furthest node in $I_r(k)$ and all the other nodes in k's right communication range $I_r(k)$ have the same load $\ell(\mathcal{P}) - 1$. Thus k must be outside $I_l(j)$, i.e., k and j cannot be visible to each other — otherwise k would have chosen j, instead of i to relay the packet. Therefore all the nodes in j's left communication range $I_l(j)$ must be inside k's right communication range $I_r(k)$.

Assume there are m (≥ 1) nodes inside j's left communication range $I_l(j)$. Then every node $u \in I_l(j)$, except for i, must have load exactly $\ell(\mathcal{P}) - 1$, as shown earlier. Therefore the total load summed over all the nodes in $I_l(j)$ is $(m - 1)(\ell(\mathcal{P}) - 1) + \ell(\mathcal{P}) = m\ell(\mathcal{P}) - m + 1$. Since a path generated by GREEDY2 has at most two nodes inside any unit interval, according to Theorem 3.1, each request r with packet size ℓ contributes at most 2ℓ to the total load over all the nodes in $I_l(j)$. Therefore the number of requests that pass the interval $I_l(j)$ is at least $(m\ell(\mathcal{P}) - m + 1)/2$. In contrast, for any routing algorithm, each request must use at least one node in $I_l(j)$. Therefore any routing algorithm has to distribute the $(m\ell(\mathcal{P})-m+1)/2$ requests on the nodes in $I_l(j)$. So we have that the optimum maximum load, $\ell^*(R)$, is at least $(m\ell(\mathcal{P})-m+1)/(2m)$. This proves that $\ell(\mathcal{P}) \leq 2\ell^*(R)+1$. $\square$

C. Requests with variable packet sizes

For requests with variable size, GREEDY2 can not guarantee a good load balancing ratio. For example, with variable sized packets, we can force GREEDY2 to alternate between two nodes while it is possible to distribute the loads among nearby nodes (See Appendix IX-B). Hence, we use a different greedy strategy with stretch factor of 2 and load-balancing ratio of 3. This idea will also be used for routing inside strips.

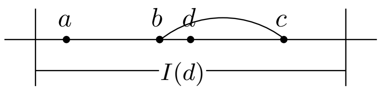


Fig. 4. The bridge bc over d, b is to the left of d, c is to the right of d and b,c are visible to each other.

For each node $d \in S$, a pair of nodes b and c form a *bridge* over d if $b \in I_l(d)$, $c \in I_r(d)$, and b, c are visible to each other (node d itself is a degenerate bridge). See Figure 4 for an example. The load of a bridge $L(bc)$ is defined as $\max(\ell(b), \ell(c))$. The node of a bridge with the heavier load is called a *heavy node* (in case of ties, both are called heavy nodes). The lightest bridge among all the bridges over d is denoted as $\phi(d)$. Intuitively a bridge guarantees that we can make substantial progress within one hop. Thus the lightest bridge is a good candidate for the next hop with consideration of load balancing.

The algorithm GREEDY3 works as follows. Whenever a node a receives a request, say, from its left, we first check if the destination is within a's communication range. If not, a asks the furthest node d in its right communication range and use the lightest bridge $\phi(d) = bc$ to route the request. The node c makes the next routing decision if the destination is not reached yet. Thus the routing path generated by GREEDY3 is composed of bridges.

¿From the description of GREEDY3, c is in the right communication range of node d, the rightmost node in a's communication range. Thus a cannot see the node c. Therefore, for any four adjacent nodes x, y, z, w on the path produced by GREEDY3, x and w are not visible to each other since they must be separated by a bridge. Using the same technique as in the previous subsection, we can add one look-ahead to GREEDY3 to shortcut the path if two non-adjacent node can see each other in the path. Therefore, any two nodes on the path, if not adjacent, must be at least distance one apart. The stretch factor of GREEDY3 is 2. We now argue that GREEDY3 has a load balancing ratio of 3.

Theorem 3.3. *GREEDY3 is* 3*-balanced, i.e.* $\ell(\mathcal{P}) \leq 3\ell^*(R)$.

Proof: The proof is by induction. Denote by R_t the set of the first t requests and $\mathcal{P}_t$ the set of paths used to deliver R_t. The claim is clearly true when $t = 1$. Suppose that after the t-th request is delivered, we have that $\ell(\mathcal{P}_t) \leq 3\ell^*(R_t)$. We now argue that after we deliver the $t+1$-th request, $\ell(\mathcal{P}_{t+1}) \leq 3\ell^*(R_{t+1})$.

We prove the claim by contradiction. Suppose that $\ell(\mathcal{P}_{t+1}) > 3\ell^*(R_{t+1})$. Consider the first time when this condition is violated as we route the $(t+1)$-th request r_{t+1}. Suppose that it is right after the node a receives r_{t+1} and routes it through bc, the lightest bridge over d. That is, the load on one of the two nodes b, c of the bridge now exceeds the value $3\ell^*(R_{t+1})$. Let ℓ denote the packet size of request r_{t+1}. Then, at this point, $L(bc)+\ell > 3\ell^*(R_{t+1})$, i.e.

$$L(bc) > 3\ell^*(R_{t+1}) - \ell\,. \qquad (1)$$

Since bc is the lightest bridge over d, any other bridge xy over d has heavier load, i.e., $L(xy) \geq L(bc)$. Recall that a node x is heavy if there exists a bridge xy or yx over d such that x has the heavier load among x, y. Thus a heavy node x has load $\ell(x) \geq L(bc)$. Denote by Φ the set of heavy nodes in d's visible range $I(d)$. All the nodes in Φ have load at least $L(bc)$. Assuming that $|\Phi| = m$, the total load on nodes in Φ for GREEDY3 is at least $\sum_{x\in\Phi} \ell(x) \geq m\cdot L(bc)$, after the first t requests are routed. In addition, each request uses at most one bridge over d. Therefore, in GREEDY3, a request contributes at most twice of its load to the total load of the nodes in Φ, i.e. the total size of those requests that need to route over d is at least $m\cdot L(bc)/2$. On the other hand, for each such request, any routing algorithm has to use at least one bridge to jump over d. Since each bridge passes at least one heavy node, any routing algorithm has to distribute the requests of total size of $m \cdot L(bc)/2$ on the m nodes in Φ, i.e. the maximum load of the nodes in Φ produced by the optimal algorithm for the first t requests is at least $L(bc)/2$. That is,

$$\begin{aligned} \ell^*(R_t) &\geq L(bc)/2 > (3\ell^*(R_{t+1}) - \ell)/2 \quad \text{by (1)} \\ &\geq \ell^*(R_{t+1}) \quad \text{by } \ell \leq \ell^*(R_{t+1}) \end{aligned}$$

We have reached the contradiction $\ell^*(R_t) > \ell^*(R_{t+1})$. Thus, we must have that $\ell(R_{t+1}) \leq 3\ell^*(R_{t+1})$. By inductive principle, the statement is true for any $t > 0$. $\square$

D. Distributed Implementation

In this section, we present an efficient implementation of the above algorithms. We assume each node knows its location by either GPS or some other localization methods [11], [20], [24]. We also assume that the rough location of the destination is known such that the source node knows whether it should send the packet to its left or right. Denote by $h_1(p)$ the number of nodes inside the communication range of p and by $h_2(p)$ the number of nodes that is at most two hops from p. Our implementation has the following properties:

- A wireless node makes the routing decision by using only local information.
- Each node p only stores $O(\log h_1(p))$ bytes[4].
- For GREEDY2, a node p makes the routing decision in $O(\log h_1(p))$ time. For GREEDY3, a node p makes the routing decision in $O(\log^2 h_2(p))$ time.
- Any dynamic update, including changing load on p, adding or deleting a node p, takes $O(\log h_1(p))$ time.

1) Requests with unit packet sizes: When all the requests have unit packet size, each node p needs to compute p^*, the furthest node in p's right (or left) communication range whose load is not maximum over all the nodes in $I_r(p)$. In the following part of this subsection, we focus on how to find p^* by a memory-efficient mechanism. Once p^* is found, the packet is delivered to p^*. This process is repeated until the destination is reached.

First, if we build a balanced binary search tree on all the nodes in $I_r(p)$, we can clearly compute p^* in time $O(\log |I_r(p)|)$. By this implementation the memory of a node p is $O(|I_r(p)|)$. Here we propose a more efficient implementation which actually distributes the storage and computation to each node. Then each node only needs poly-logarithmic storage. To achieve this, we pay some price for extra communication. A node p finds out the next hop p^* by asking its neighbors to do some computation. We assume that in the procedure of finding the next hop p^*, the size of the control information transferred is very small and thus can be omitted. If this is not the case, i.e., the control information is also taken into account in loading the wireless nodes, we should use the first scheme where a node keeps the locations of all its 1-hop neighbors.

We construct a virtual forest $\mathcal{F}$ in which every $I_r(p)$ is a union of a constant number of subtrees of $\mathcal{F}$. To build $\mathcal{F}$, we use a binary grouping process where every two mutually visible nodes are grouped. Pick one of two nodes as a (first-level) leader. We then group first-level leaders to create second-level leaders and so on. If a node cannot find a same level leader within its communication range to group, the process simply stops for that node (Figure 5). At the end of the process, each node has a rank which is the highest level it is on.

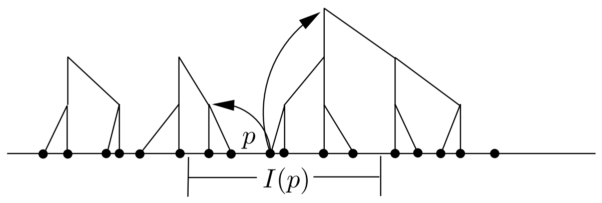


Fig. 5. Algorithm for requests with unit packet size.

The construction of $\mathcal{F}$ can be done by using leader election algorithms such as the randomized algorithm used in [6]. The virtual forest $\mathcal{F}$ is stored in a distributed manner on the nodes in S. If a node u is grouped with a node v on level $i-1$ and u is selected as the level i leader, we call the node v the child of u. The virtual forest $\mathcal{F}$ is stored implicitly such that each node stores the IDs of its parent and children. Now, we consider the communication range $I(p)$ of a node p. There are at most three trees in $\mathcal{F}$ that contain nodes in $I(p)$, which are denoted by a set $\mathcal{T}(p)$. We store at p the set $V(p)$ that contains the highest rank node inside $I(p)$, for each tree $T \in \mathcal{T}(p)$. On each tree $T \in \mathcal{F}$, we also construct a binary search tree structure on the loads on each node distributedly. The storage used at the node p is in the order of $O(\log h_1(p))$.

To compute p^* for p, p asks the nodes in $V(p)$ which node should be p^*. Each of the node in $V(p)$ does a binary search top-down and recursively asks its children to compute p^*. The p^*, once found, is returned to the node p and the packet is delivered from p to p^*. Since each tree in $\mathcal{F}$ is balanced, the computation and any dynamic change of load can be done in time $O(\log h_1(p))$ as well.

2) Requests with variable packet sizes: The algorithm for requests with variable packet size is more complicated. The major task is to find the lightest bridge over a node p. For each node p, define

$$\begin{aligned} f_p(x) &= \min_{x_p \le x_u \le x_p + x} \ell(u)\,, \quad \text{and} \\ g_p(x) &= \min_{x_p + x - 1 \le x_u \le x_p} \ell(u)\,. \end{aligned}$$

Clearly, both $f_p(x), g_p(x)$ are stair-case functions, where f is decreasing, and g is increasing. Since $f_p(0) = g_p(1) = \ell(p)$, there must exist an x^* so that $f_p(x)$ and $g_p(x)$ intersect. We take b^* and c^* to be the nodes such that $f_p(x^*) = \ell(c^*)$ and $g_p(x^*) = \ell(b^*)$, see Figure 6. Then we claim that,

Lemma 3.4. *b^*c^* is the lightest bridge over p.*

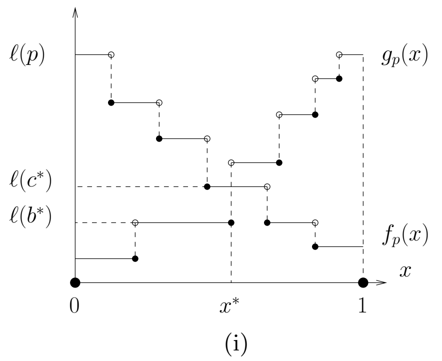
Proof: First since $g_p(x^*) = \ell(b^*)$, $f_p(x^*) = \ell(c^*)$, then $b^* \in [x_p + x^* - 1, x_p]$, $c^* \in [x_p, x_p + x^*]$. So b^*c^* is a valid bridge.

Now we assume that there is another bridge bc with weight smaller than b^*c^*. If $\ell(b^*) \le \ell(c^*)$, as shown in Figure 6 (i), then we know that $\ell(c) < \ell(c^*)$. For the location of c we must have x_c is greater than $x_p + x^*$. Then b's location x_b is inside interval $[x_c - 1, x_p]$. So $b \ge x_c - 1 > x_p + x^* - 1$. The stair-case property of $f_p(x)$ and $g_p(x)$ implies that for all $x > x^*$, $g_p(x) > f_p(x^*) = \ell(c^*)$. So $\ell(b) > \ell(c^*)$. Then the weight of bc is greater than the weight of b^*c^*. This causes a contradiction. The case when $\ell(b^*) > \ell(c^*)$ can be proved in a similar way. $\square$

Computing x^* can be done by using binary search. Again, we can use the method for unit packet size to solve the problem but with one more log factor due to the binary search in the computation and update time.

Remark. When there are dynamic changes such as node insertions and deletions, or load changes, the binary search forest needs to be updated. A change on a node only affects a constant number of trees in the forest. If we use a dynamic balanced binary tree, then each dynamic change

[4]Here we implicitly assumed that the ID of a node is represented in $O(1)$ bytes.



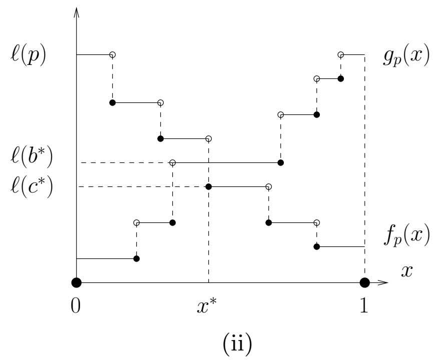


Fig. 6. (i) $g_p(x^*) = \ell(b^*) \leq f_p(x^*) = \ell(c^*)$; (ii) $g_p(x^*) = \ell(b^*) > f_p(x^*) = \ell(c^*)$.

on a node p can be done in $O(\log h_1(p))$ time.

IV. Load-balanced routing for nodes in a narrow strip

The load-balanced routing algorithm can be extended to a strip with width $w \leq \sqrt{3}/2 \approx 0.86$, if the communication radius of each node is 1. In what follows, we assume that a strip is bounded by two horizontal parallel lines where the width is the vertical distance between the two lines. The reason for the restriction on the width will become clear later.

We say a node b is to the left (right) of a, if the x-coordinate of b is smaller (larger) than that of a. Then, for each vertex p, bc is a right (left) bridge if b is inside the communication range of p, c is outside and to the right (left) of p, and b, c are visible to each other (Figure 7(i)). The load of a bridge is then defined as $\max(\ell(b), \ell(c))$. The right (left) lightest bridge is the bridge with the smallest load among all the right (left) bridges. The algorithm GREEDY4 works in a way similar to GREEDY3: when p receives a packet, it first checks if the destination of the packet is inside its communication range. If it is, then the packet is forwarded to the destination. Otherwise, according to which side the destination lies, p chooses the right or left lightest bridge, say bc, sends the packet on the path pb and bc. When the packet arrives on b, it again checks if the destination is in b's communication range, and if it is, forwards the packet to the destination. Then, repeat the above process at the node c until the destination is reached.

To show that the above algorithm works correctly, we make the following observation, which also explains the restriction on the strip width.

Lemma 4.1. *Suppose that u, v, w, from left to right, are three nodes in a strip with width at most $\sqrt{3}/2$ and u, w are mutually visible. Then either u or w is visible to v.*

Proof: If neither u nor w is visible to v, then both nodes are outside the communication range of v. In addition, u is to the left of v, and w to the right. It is easy to see that if the width is at most $\sqrt{3}/2$, then u and w cannot be visible to each other (Figure 7(ii)). □

We now claim that

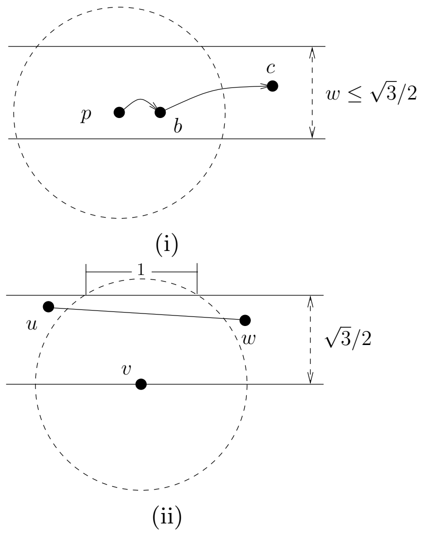


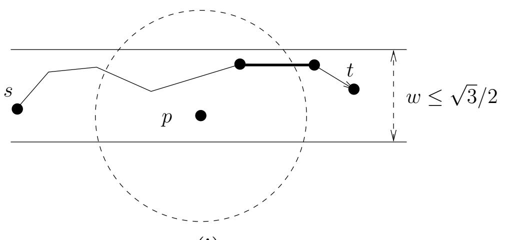
Fig. 7. (i) bc is a right bridge of p. (ii) u, w cannot see each other if they are on different sides of v and outside of v's communication range.

Theorem 4.2. *The stretch factor of GREEDY4 is 4, and the load-balancing ratio is 3.*

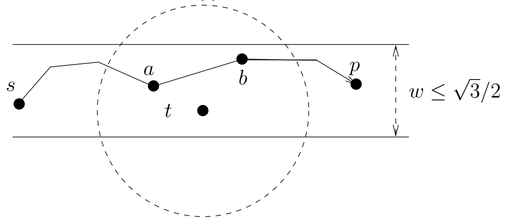
Proof: We first show that the algorithm always succeeds. Suppose that there is a path from a node s to t where t is to the right of s. We argue that GREEDY4 will reach the destination t. Otherwise, assume that the algorithm is stuck at a node p, and p is not visible to t. If p is to the left of t, we argue that there must be a bridge that p can routes to. Since s is to the left of t, any path from s to t has to cross the right boundary of p's communication range. The edge that crosses the boundary then must be a right bridge of p, contradicting with the assumption (Figure 8 (i)).

Now, suppose that p is to the right of t. Consider the pair of adjacent nodes, say a, b, on the path from s to p that sandwich t, i.e., a is to the left of t, b is to the right of t, see Figure 8 (ii). By Lemma 4.1, t is visible to either a or b. Thus, the packet must have been delivered to t by either a or b. Therefore, the algorithm always delivers a packet if there exists a path.

Next we will bound the stretch factor of the algorithm.



(i)



(ii)

Fig. 8. (i) The path from s to t has to cross the right boundary of p's communication range. The thickened edge is a right bridge of p. (ii) The path from s to p has an edge ab that sandwich t.

For a right bridge bc of p, since c is outside the communication range of p, we have that $x_c \geq x_p + 1/2$, where x_c and x_p are the x-coordinates of c and p, respectively. By taking two hops, GREEDY4 is guaranteed to progress at least $1/2$ along the x-coordinate. Clearly, for a packet from s to t, it has to take at least $x_t - x_s$ hops, while GREEDY4 takes at most $2(x_t - x_s)/(1/2) = 4(x_t - x_s)$ hops. Thus, the stretch factor is at most 4.

To bound the load-balancing ratio we use the same techniques as in Theorem 3.3. For a node p that GREEDY4 picked, every path from the source to the destination has to use one right bridge of p, and therefore at least one and at most two heavy nodes. So by the same argument the load-balancing ratio for GREEDY4 is at most 3. □

Furthermore we show that the upper bound $\sqrt{3}/2$ on the width of the strip is indeed necessary. If otherwise, then there could be two paths P_1, P_2 from the source s such that they are not visible to each other except at the source and destination, see Figure 9. So any algorithm based on local information will not be able to find out at the source node whether P_1 or P_2 is more heavily loaded.

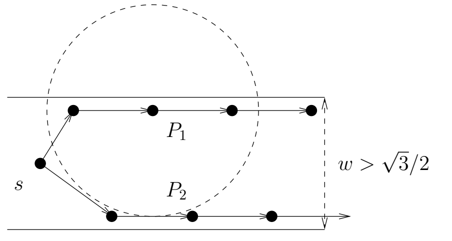


Fig. 9. A strip of width $w > \sqrt{3}/2$, P_1, P_2 are two paths from s.

Clearly, the above algorithm can be implemented such that the cost for both routing decision and update is linear to the number of nodes in the 2-hop neighborhood. The reason that it does not admit an efficient algorithm as in the line case is that the two dimensional dynamic disk range search is much more difficult than the one dimensional case, which can be done by binary search. There are techniques in computational geometry which yield better theoretical bounds, but those methods are impractical and often unnecessary as we expect that in practice each node has only small number of neighbors.

V. Discussion

In this section, we discuss variants of our algorithm for balancing energy, for reducing overhead, and for handling dynamic changes.

a) Routing based on energy: Our previous algorithms try to evenly distribute load, defined as the total size of relayed packets. There are cases, such as in emergency monitoring sensor networks, when it is more important to prolong the network healthy time, the time before any node dies. In those cases, evenly distributing load may not be the best strategy, for example, when the nodes have different initial energy or when the nodes consume different amount of energy to deliver packets. One variant is to run our algorithm against the energy left in each node. The energy can be measured in the best available way for each node, such as by the size of relayed traffic adjusted by a node specific factor or by reading from the battery power meter.

b) Reducing overhead: We have described our algorithms in a per-packet basis. The main purpose is to provide a rigorous analysis. In practice, it may incur too many control messages. To reduce the overhead, we can quantize the packet size or the energy (when using energy based routing) into levels with appropriate scale. This way, a route can be cached, and an update is only needed when the load or energy level of some node changes.

c) Handling dynamic updates and mobility: One challenge for routing in wireless network is to deal with high rate of topology change, either due to the energy constraint or due to node mobility. It would be too expensive to propagate topology changes across network. Since our algorithms only use local information for making routing decision, it is easy to handle dynamic changes — when a node joins or leaves the network, it can only affect its 2-hop neighborhood. Once such a change happens, we only need to update the lightest bridge of the nodes within 2 hops, or we can take the lazy approach to delay the update until a new request comes.

Handling mobility is harder. One general approach is to discretize time and treat the problem the same as dynamic problems where the topology changes are updated at each discrete time. One problem with this approach is to choose an appropriate sampling rate of the time axis. With a high sampling rate the computation cost increases. With a low sampling rate critical topology changes may be missed. The maximum speed of any node usually gates this rate for the entire system. Another approach is to take advantage of continuity of motion and to track the nodes inside each node's neighborhood, for example, by using the proximity maintenance method in [7]. The method in [7] can track the neighborhood of each nodes efficiently in a distributed and event-driven fashion. Our routing algorithms can be coupled with that algorithm, and an update of data structure is triggered when there is a change of nodes in the neighborhood.

VI. Simulations

The previous sections give rigorous analysis on the worst-case competitive ratios of our algorithms compared with the optimal algorithms. While the worst case analysis is valuable to provide theoretical guarantees, it is not rare that the worst case analysis does not reflect the reality. For example, the load balancing ratio of the shortest path routing in the worst case can be as bad as $\Omega(n)$, but such a scenario may not happen in practical setups. In this section, we evaluate the performances of our load-balanced routing algorithm, GREEDY3 in particular, under more realistic traffic pattern by simulation and comparison to the shortest path routing algorithm.

We should note that our simulation is only on the algorithmic level with the purpose of exploring the competitive ratios under reasonable node distributions and traffic patterns. We do not intend to conduct a complete network level simulation. One reason is that there are non-trivial challenges on cross-layer design, in particular, on the interaction of the MAC layer protocols with the networking layer. Such issues are beyond the scope of this paper.

A. Simulation setup

In our experiments, the wireless nodes are distributed randomly along a line. Specifically, we distribute 1000 nodes randomly in the interval $[0, 100]$. We vary the radius of the communication range of the wireless nodes from 1 to 10. In one set of experiments, we assume that the nodes can handle as many traffic as possible, and we evaluate the maximum packets that one node relays. In the other set, we put a limit on the number of packets a node is able to relay and evaluate the number of packets the network delivers before any node dies. Only the data packets are taken into account in the simulation results, for the reason that the control packets largely depend on how the underlying MAC protocol is designed and implemented.

For the traffic patterns, we consider two cases: random traffic pattern and aligned traffic pattern. In the random traffic pattern, a packet is generated by choosing the source and destination of a packet uniformly randomly among all the nodes. In the aligned traffic pattern, a packet is originated from the left end and destined to the right end, i.e. each packet has to be relayed from the left to the right. In both cases, our experimental results suggest that the load-balanced routing works much better than the shortest path routing in terms of balancing the load. In addition, we measure the length of the paths produced by our algorithm and compare it with the shortest path routing. From time to time, our algorithm produces path noticeably longer than the shortest path. However, on average, the path length is only slightly longer. This indicates that the load-balancing is achieved at the price of increasing the path length but only by a small fraction.

B. Results

We now present detailed simulation results under different scenarios.

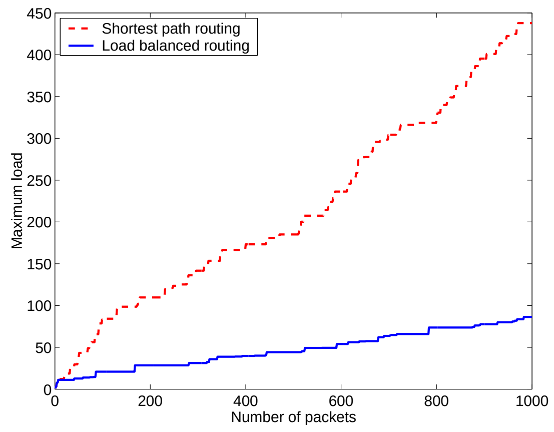
d) Unlimited energy and random traffic: We generate 1000 random packets, each with size randomly chosen between 1 and 10. In Figure 10 (i), we plot, for both the load balancing routing and the shortest path routing, the maximum load in terms of the number of packets delivered by the network, if the communication range has radius 5. According to the data, the ratio of the maximum load of the shortest path routing to that of the load-balanced routing is about 5. We also compare the length (the number of hops) of the paths produced by our algorithm to the shortest path length, both in the worst case and on average. Figure 10 (ii) shows the worst case and average ratio of the length of the paths produced by our algorithm to the shortest path length, as well as the ratio of the maximum load under shortest path routing and load-balanced routing under different communication ranges. The observation from Figure 10 (ii) is that we achieve substantially in terms of the maximum load ratio by paying a little higher price on the maximum and average delay.

e) Unlimited energy and aligned traffic: Under the aligned traffic pattern, a packet is originated from a random node in the interval $[0, 10]$ and destined to a random node in $[90, 100]$. Figure 11 shows the experimental result. The data shows that the ratio of the maximum load of the shortest path routing to that of the load-balanced routing is about 10.3, much higher than the random traffic pattern, if the communication range has radius 5.

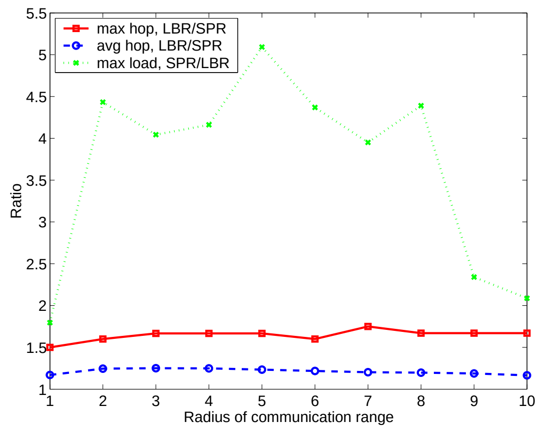
f) Limited energy and random traffic: In this case we assume the nodes have a maximum energy, measured by the maximum number of packets it is able to relay. A node dies if the packets it relays exceed the given limit. We vary the maximum energy from 0 to 90. Correspondingly we show the number of packets delivered before any node dies in Figure 12 (i). Compared to the shortest path routing, the load-balanced routing algorithm delivers about twice as many packets before any node dies for all the energy levels.

g) Limited energy and aligned traffic: We also try aligned traffic under the constrained energy model. We vary the maximum energy from 0 to 150. As shown in Figure 12 (ii), the result for load-balanced routing is better than the case for random traffic.

In summary, we find through simulation that our load-balanced routing algorithms perform consistently better than the shortest path routing in terms of the maximum node load and only slightly worse in terms of the path length. Furthermore, the implementation of the load-balanced routing is fairly simple. We would expect the algorithm to find practical use in real applications.

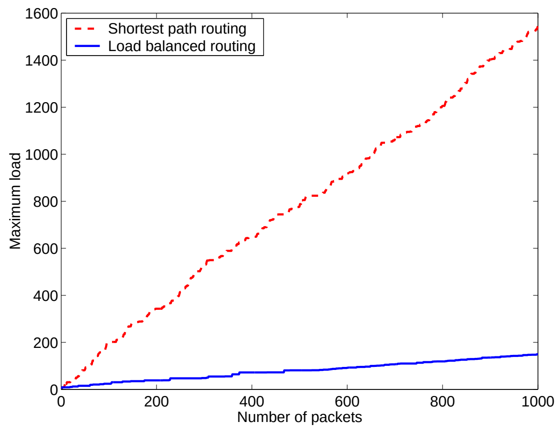


(i)

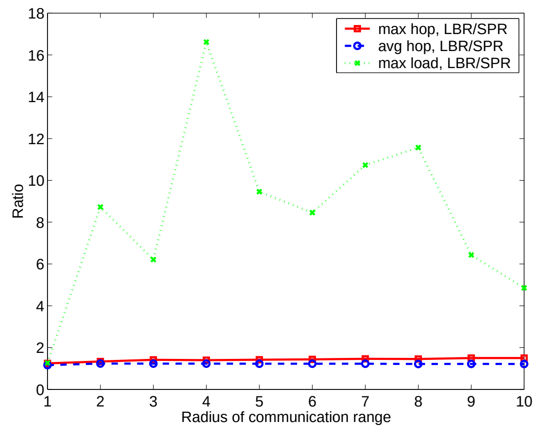


(ii)

Fig. 10. (i) The maximum node load in terms of the number of packets delivered under the random traffic pattern. The dashed line curve is for the shortest path routing, and the solid line curve for the load-balanced routing. The communication range has radius 5; (ii) The worst case and average ratio of the length of the paths produced by our algorithm to the shortest path length under different communication ranges, under the random traffic pattern. We also show the ratio of the maximum load under the shortest path routing to that under the load-balanced routing for different communication ranges.

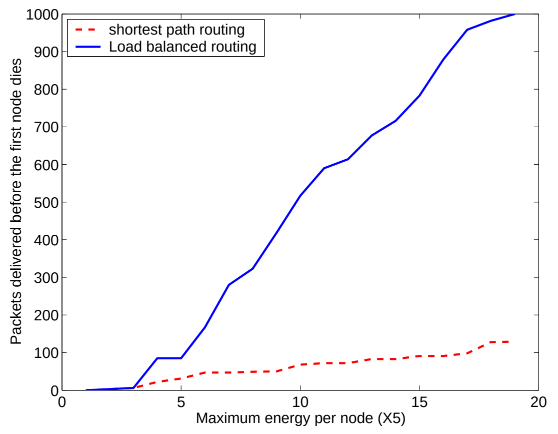


(i)

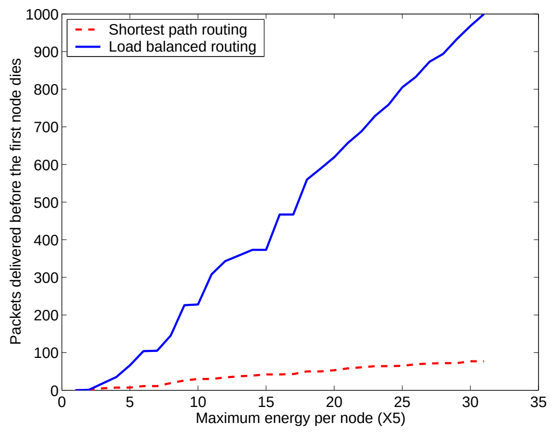


(ii)

Fig. 11. (i) The maximum node load in terms of the number of packets delivered under the aligned traffic pattern. The communication range has radius 5; (ii) The worst case and average ratio of the length of the paths produced by our algorithm to the shortest path length under different communication ranges, under the aligned traffic pattern. We also show the ratio of the maximum load under the shortest path routing to that under the load-balanced routing for different communication ranges.



(i)



(ii)

Fig. 12. (i) The number of packets delivered when the first node dies in terms of the maximum energy of each node under the random traffic pattern. Again, dashed line curve is the shortest path routing, and sold line curve the load-balanced routing; (ii) The number of packets delivered when the first node dies in terms of the maximum energy of each node under the aligned traffic pattern.

VII. Conclusion

In this paper, we initiate the study of wireless network routing with the aim of achieving good performance in terms of both stretch factor and load-balancing ratio. We present algorithms which can achieve constant competitive factors for both measures when all the nodes are located in a narrow strip. In addition, our algorithm is local and deals with dynamic change efficiently.

This work raises a number of open questions. For example, it is not clear yet whether the optimal load balanced routing on a unit disk graph of points on a line is NP-hard, when the packets have equal sizes. It is also not clear whether there exist algorithms with better load balancing ratios and stretch factors. The current work is on a restricted node distribution. It would be interesting to extend the study to other node distributions. For such an extension, we may have to make assumption on the traffic patterns, as load-balanced routing is difficult even for nodes that form a regular grid. Last, the current work focused on the algorithmic design and ignored many of the important MAC issues that can be critical in practice. It would be interesting future work to evaluate the performance of our algorithm under proper MAC and physical layer models.

VIII. Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] S. Albers. Better bounds for online scheduling. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 130–139, 1997.

[2] Y. Azar. On-line load balancing. In A. Fiat and G. Woeginger, editors, *On-line Algorithms: The State of the Art*, pages 178–195. LNCS 1442, Springer, 1998.

[3] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambrdige University Press, 1998.

[4] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proc. INFOCOM*, pages 22–31, 2000.

[5] J.-H. Chang and L. Tassiulas. Fast approximation algorithms for maximum lifetime routing in wireless ad-hoc networks. In *Networking, LNCS 1815*, pages 702–713, 2000.

[6] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. *Discrete and Computational Geometry*, 30(1):45–65, 2003.

[7] J. Gao, L. J. Guibas, and A. Nguyen. Distributed proximity maintenance in ad hoc mobile networks. In *Proc. IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York, NY, 1979.

[9] N. Gupta and S. R. Das. Energy-aware on-demand routing for mobile ad hoc networks. In *Proc. IWDC*, pages 164–173, 2002.

[10] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer. Position-aware ad hoc wireless networks for inter-vehicle communications: the fleetnet project. In *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01')*, pages 259–262, 2001.

[11] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–667, August 2001.

[12] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[13] Q. Li, J. Aslam, and D. Rus. Online power-aware routing in wireless ad-hoc networks. In *Proc. Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 97–107, 2001.

[14] T. Melodia, D. Pompili, and I. F. Akyildiz. Optimal local topology knowledge for energy efficient geographical routing in sensor networks. In *IEEE INFOCOM 2004 - The Conference on Computer Communications*, pages 1706–1717, 2004.

[15] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[16] S. A. Plotkin. Competitive routing of virtual circuits in ATM networks. *IEEE Journal of Selected Areas in Communications*, 13(6):1128–1136, 1995.

[17] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comp. and System Sciences*, pages 130–143, 1988.

[18] 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.

[19] S. Ramanathan and M. Steenstrup. A survey of routing techniques for mobile communications networks. *Mobile Networks and Applications*, 1(2):89–104, 1996.

[20] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proc. Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 166–179, 2001.

[21] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proc. Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 181–190, 1998.

[22] V. Srinivasan, C. F. Chiasserini, P. Nuggehalli, and R. R. Rao. Optimal rate allocation and traffic splits for energy efficient routing in ad hoc networks. In *Proc. of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 950–957, 2002.

[23] I. Stojmenovic and X. Lin. Power aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, 2001.

[24] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personnel Communications*, 4(5):42–47, October 1997.

[25] Y. Xu, J. S. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proc. Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 70–84, 2001.

[26] W. Yu and J. Lee. DSR-based energy-aware routing protocols in ad hoc networks. In *Proc. of the 2002 International Conference on Wireless Networks*, 2002.

[27] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor net-works. Technical report tr-01-0023, University of California, Los Angeles, Department of Computer Science, 2001.

[28] G. Zussman and A. Segall. Energy efficient routing in ad hoc disaster recovery networks. In *IEEE INFOCOM 2003 - The Conference on Computer Communications*, pages 682 –691, 2003.

IX. Appendix

A. Load balancing and short path routing for general node distribution

When the nodes are distributed in the plane, it is not possible to achieve constant bounds in terms of both latency and load. If we insist on using c-short paths, then the load-balancing ratio can be as bad as $\Omega(n/c)$. One such example is shown in Figure 13. There are $4c+1$ spots on a loop. Each spot contains n/c wireless nodes, except one spot has only one node o. The total number of nodes is $4n+1$. Only the nodes in adjacent spots can directly communicate with each other. If we make n/c requests, each from a distinct node on spot p to a distinct node on spot q. Any path that does not pass through o has length at least $4c-1$, i.e., is not c-short path. Therefore, any c-short routing algorithm has to route the requests through o, i.e. o has load $\Theta(n/c)$. On the other hand, the optimal load balancing routing algorithm can route the requests evenly along the path on the longer arc such that each node only passes $O(1)$ packets.

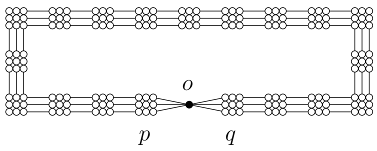


Fig. 13. Each spot contains n/c nodes. The loop has $4c+1$ spots. The packets from spot p to q either go through node o, thus causing the node o to be heavily loaded, or route along a long path with length $\Omega(n/c)$.

B. GREEDY2 gives unbounded maximum load for variable packet size

We show by an example that the algorithm GREEDY2 does not guarantee a constant load balancing ratio if the packets have variable sizes. Assume we have $3n$ nodes distributed on a line. The nodes are organized in three groups, $\{x_1, \cdots, x_n\}$, $\{y_1, \cdots, y_n\}$, $\{z_1, \cdots, z_n\}$. All the x_i's are visible to all the y_i's. All the y_i's are visible to z_i's. But no x_i is visible to z_i, as shown in Figure 14. We have a set of n requests. The first request is $r_n = (x_n, z_n, 1)$, the next $n-1$ requests are $r_i = (x_i, z_i, 2)$, where $i = 1, 2, \cdots, n-1$. The optimum load balanced routing algorithm routes r_i though y_i. So the maximum load is 2. Now let us see how the GREEDY2 works here. The first request $(x_n, z_n, 1)$ is going to be routed on y_n

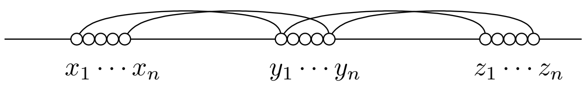
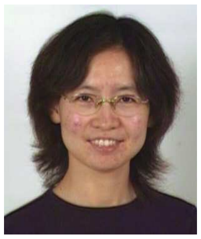
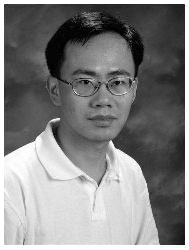


Fig. 14. The optimum load balanced routing algorithm. The maximum load is 2.

since y_n is the furthest node and all the current load of y_i are all 0. The next request $(x_1, z_1, 2)$ will be routed on y_{n-1} since y_{n-1} is the furthest node whose load is strictly smaller than the maximum load among all the y_i's. The third request $(x_2, z_2, 2)$ will be routed on y_n since now the maximum load is 2 and the load of y_n is 1 and is strictly smaller than the maximum load in x_2's right communication range. So the load on y_n is 3. So the next request $(x_3, z_3, 2)$ will be routed on y_{n-1}. After this point, the requests will be routed alternatively on y_n and y_{n-1}. Thus the maximum load produced by GREEDY2 is $\lceil \frac{n-1}{2} \rceil \times 2 + 1$. The load balancing ratio of GREEDY2 in this case is $\Omega(n)$.



Jie Gao received her Ph.D in computer science from Stanford University in 2004, and her BS degree from University of Science and Technology of China in 1999. She is currently a postdoc fellow at the center for the mathematics of information at California Institute of Technology, and will join the State University of New York, Stony Brook as an assistant professor in Fall 2005. Her research interests are algorithms, ad hoc communication and sensor networks, and computational geometry.



Li Zhang received his BE and ME degrees, both in Computer Science, from Tsinghua University, Beijing, China, in 1991 and 1993, respectively, and the PhD degree in Computer Science from Stanford University, California, in 2000. He is currently a researcher at Information Dynamics Laboratory in HP Labs, Palo Alto, California. His research interests include computational geometry, algorithms for ad-hoc and sensor networks, and game theoretical analysis of resource allocation.