Load Balanced Short Path Routing in Wireless Networks

Jie Gao
Department of Computer Science
Stanford University
Stanford, CA 94305
E-mail: jgao@cs.stanford.edu

Li Zhang
Hewlett-Packard Labs
1501 PageMill Road
Palo Alto, CA 94304
E-mail: l.zhang@hp.com

***Abstract*— In this paper, we study wireless network routing algorithms 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 addition, our algorithms only use local information and can deal with dynamic change and mobility efficiently.**

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

I. Introduction

In a mobile ad-hoc network or a sensor network, devices communicate to their nearby nodes and form an ad-hoc multi-hop communication network. Routing in such a network is challenging due to the lack of central control and the high dynamicity of the network. The previous works have focused on discovering and maintaining routes that enable the connectivity between the nodes and, if possible, that minimize the number of hops on a path.

One important restriction of a wireless network is that the nodes are energy constrained as they are normally powered by batteries. Besides minimizing latency, the shortest path routing is good for overall energy efficiency because energy needed to transmit a packet is correlated to the path length. However, the algorithms that aim to minimize the path length may ignore "fairness" in the 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 die much earlier, thus creating holes in the network, or worse, leaving the network disconnected. This problem is critical in emergency networks, for example, to locate survivors after structure collapse, where depleting the battery of a node may have tragic results. In addition, unbalanced use of the nodes may discourage the nodes to participate in the routing.

Since the biggest energy drain comes from the transmission of packets, we measure the energy consumption of a node by the total size of packets relayed by the node. Then the load-balanced routing can be thought 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 easily construct an example to show that these two goals are indeed conflicting, i.e. the shortest path routing algorithm necessarily creates heavily loaded nodes, and the optimum load-balancing algorithm necessarily uses long paths. In practice, the nodes often distribute 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 with performance 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 [1] 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 routing is done for the nodes on the meta-path. The problem then reduces to the routing for nodes located in a narrow strip.

Compared to the extensive use of the shortest path routing in wireless networks, load-balanced routing has received less attention. This probably should not be too surprising as load-balancing is a much more difficult problem. For example, 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

require global coordination, and the approximation ratio often has only theoretical interests. 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 the shortest path routing, it is appropriate to use the greedy method of sending the packet to the furthest reachable node in the right 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 ratio 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. Then every time the 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;
2) It balances the load: the maximum total size of packets passed on any node is at most three times as much as the optimum.
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 handles dynamic change and mobility efficiently as only a node's neighborhood is affected.
4) It is online: the routing decision of a packet depends only on the previously routed packets, i.e., the current state of the network. It doesn't need to know the packets in the future.

We also consider an important subcase when all the nodes are aligned on a line. In this case, we can achieve an even better approximation factor 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 even when a node has many nodes in its neighborhood.

While we show rigorously that we can achieve constant bounds in terms of both latency and load in the case when the nodes are located in a narrow strip, we should note that this is not generally true when nodes are in the plane by a simple example as in Figure 1.

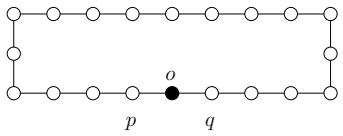

Fig. 1. Each white spot contains m nodes, and the black spot o contains a single node. The packets from spot p to q either go through node o, thus causing o heavily loaded, or route along a long path, thus having large latency.

In addition to providing rigorous analysis, we have also implemented the algorithm and studied the performance by simulation. 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.

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 $O(\log n/\log\log n)$ [4], [5]. 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 [6], [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) [7], Ad-hoc On-demand Distance Vector routing (AODV) [8] and many others. Please refer to the surveys [9] and the references therein.

On the other hand, energy-aware routing algorithms, which try to maximize the network survivability, have attracted a lot of interest [10-24]. The energy aware metrics, such as "maximize time to partition" and "minimize maximum node cost", were first proposed by Singh *et al.* [15]. Chang *et al.* [16], [17] 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], [23]. Li *et al.* [11] 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. Yu *et al.* [19] proposed a method that uses the geographical locations of wireless nodes for energy aware routing. Xu *et al.* [24] proposed an algorithm GAF which is designed to reduce the energy consumption by turning off unnecessary nodes. In [25], [26], the traditional energy-unaware routing protocols such as DSR [7] or AODV [8] were re-visited to take into account the energy-aware metric. All of the energy-aware protocols mentioned above are heuristics and do not provide any guarantee on the performance.

The paper is organized as follows. In Section II, we introduce some definitions and notations. In Section III and IV, 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 V. In Section VI, we show the simulation results of our algorithm.

II. Notations and Definitions

Wireless nodes can be modeled as a set of points S in the plane. Let n denote the size of 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 nodes 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 $s(P) \leq \alpha$, P is called *α-short*.

A routing request has the form $r = (s, t, \ell)$ where s, t, ℓ represent the source, destination, and packet size, respectively. To satisfy 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 $\mathcal{P} = \{P_r \,|\, r \in R\}$ where P_r satisfies r. Similarly, 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 v, i.e.

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

The maximum load $\ell(\mathcal{P})$ of $\mathcal{P}$ is then defined to be $\max_{v \in S} \ell(v)$. 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 present an efficient distributed implementation of the algorithm in the next section.

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 the optimum load balancing is difficult even for a simple network, as shown in Figure 2(i). Suppose that each node x_i wishes to send a packet with size ℓ_i to the 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 exactly the knapsack problem, a well-known NP-hard problem.

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 [27].

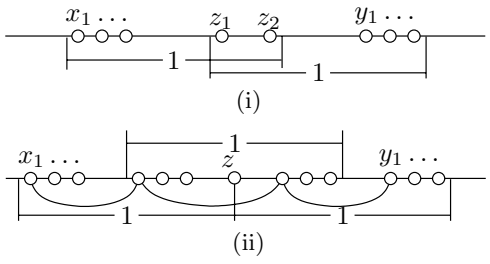

Fig. 2. Load-balanced routing is hard. The problem in (i) is equivalent to the knapsack problem. In (ii), the shortest path from x_i to y_i all pass through z, but one can evenly distribute the load by using the path as shown in the figure.

Next, we show that it is impossible to optimize both the stretch factor and the load-balancing ratio. In Figure 2(ii), when x_i sends a packet to y_i, if we insist to use 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.

B. Requests with unit packet size

Suppose that all the nodes lie on the real 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 load 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. Whenever 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, it then sends the request to the furthest node among all the nodes in its right communication range whose load is strictly smaller than $\ell_r^*(p_i)$. If all the nodes in $I_r(p_i)$ have the same load, then p_i simply sends the request to the furthest node in $I_r(p_i)$. In this case, $\ell_r^*(p_i)$ is increased by 1.

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

Theorem 3.1. *GREEDY2 is* 2*-short and* 2*-balanced.*

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 a, b, c, d are all mutually visible to each other. Since c, d are both in $I_r(b)$ and b chooses c instead of d to be the next node along the path, we must have that $\ell(c) < \ell(d)$ by the greedy forwarding strategy. Therefore $\ell(c) < \ell(d) \leq \ell_r^*(a)$. That is, a should have chosen c or a node to the right of c to be the next node, contradicting with that b is the next neighbor of a in the path.

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 in GREEDY2, one shortcut is sufficient. Therefore, the stretch factor of GREEDY2 is 2 because any unit interval can only cover at most two nodes in a path produced by GREEDY2 while for the shortest path, there must be at least one node in each unit interval.

As for the load-balancing ratio, we consider the first time when the maximum load is created. Suppose that it is on the node i and caused by the request r. That is, right before i relays the request, no node had more than $\ell(\mathcal{P})-1$ load on it, and $\ell(i) = \ell(\mathcal{P}) - 1$. Assume the forwarding direction of r is from left to right.

If there is no node to the right of i, then i must be the destination of all the requests and therefore the load-balancing ratio is 1. Otherwise, suppose that j is the node to the immediate right of i in S, and k the node to the left of i on the path P_r(Figure 3 (i)). Denote by $I' = I_l(i) \setminus I_l(j)$. Then k must be inside I' because k and j cannot be visible to each other — otherwise k would have chosen j, instead of i to relay the packet.

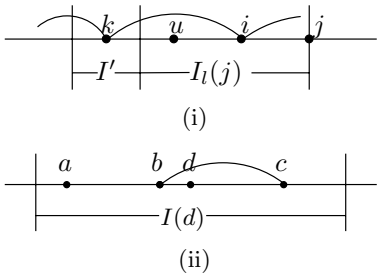

Fig. 3. (i) load-balancing competitive ratio of GREEDY2; (ii) The bridge bc of d.

Assume there are $m \geq 1$ 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 $\ell(i) - 1$, because otherwise k would have chosen u instead of i as the next neighbor on P_r (there is no node between i, j as j is to the right of i in S.). The total load summed over all the nodes in $I_l(j)$ is therefore $(m-1)(\ell(i)-1) + \ell(i)$. Since a path generated by GREEDY2 has at most two nodes inside any unit interval, the number of requests that pass the interval $I_l(j)$ is at least $(m\ell(i) - m + 1)/2$. Therefore the optimum value, $\ell^*(R)$, is at least $(m\ell(i) - m + 1)/(2m)$. This proves that $\ell(\mathcal{P}) \leq 2\ell^*(R) + 1$. □

The above analysis on the load-balancing ratio is tight as we can construct examples to achieve the bound.

C. Requests with variable packet sizes

The above greedy algorithm fails for requests with variable size. For example, with variable sized packets, we can force GREEDY2 to alternate between two nodes while it is possible to evenly distribute the loads among nearby nodes. Here, we show a different greedy strategy with stretch factor of 2 and the load-balancing ratio of 3. The idea is to define bridges which can be used to guarantee progress and then choose from the "lightest" bridge. This idea will also be used for dealing with strips.

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(Figure 3 (ii)). The load of a bridge $\ell(bc)$ is defined as $\max(\ell(b), \ell(c))$. Then lightest bridge $B(d)$ is defined to be the lightest bridge among all the bridges over d. Now, the algorithm GREEDY3 works as follows. Whenever a node a receives a request from left, we again 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 $B(d) = bc$ to route the request. The same process is repeated at the node c.

Similarly, we can show that GREEDY3 has the property that for any four adjacent nodes a, b, c, d on the path produced by GREEDY3, a, d are not visible to each other

since they are separated by a bridge. Using the same technique as in the previous section, 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, the stretch factor of GREEDY3 is 2. We now argue that

Theorem 3.2. *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. The claim is clearly true when $t = 1$. Suppose that after the t-th request is processed, we have that $\ell(R_t) \leq 3\ell^*(R_t)$. We now argue that it is still true for $t+1$. We prove this by contradiction. Suppose that it were not true. Consider the first time when the condition is violated when routing the $t+1$-th request r_{t+1}. Suppose that it is when a receives r_{t+1} and routes it through bc, the lightest bridge over d.

Let ℓ_{t+1} denote the size of r_{t+1}. Then, $\ell_t(bc) + \ell_{t+1} > 3\ell^*(R_{t+1})$. For every bridge B over d, $\ell_t(B) \geq \ell_t(bc)$ since bc is the lightest bridge over d. A node u is *heavy* if there exists a bridge $B = uv$ or $B = vu$ over d such that $\ell_t(B) = \ell_t(u)$. Denote by D the set of heavy nodes in $I(d)$. Since each bridge has to pass at least one heavy node and can pass at most two heavy nodes, the average load on all the nodes in D is at least $\ell_t(bc)/2$. Any algorithm has to use one node in D to route those requests. Therefore $\ell^*(R_t) \geq \ell_t(bc)/2 > (3\ell^*(R_{t+1}) - \ell_{t+1})/2$. Since $\ell_{t+1} \leq \ell^*(R_{t+1})$, the above formula implies that $\ell^*(R_t) > \ell^*(R_{t+1})$, a contradiction. Thus the result holds for $t+1$. □

IV. 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 localization methods [28], [29], [30], [31]. 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 distance two from p. Our implementation has the following properties:

- A wireless node makes the routing decision by using only local information.
- Each node only stores $O(\log h_1(p))$ bytes.
- A node p makes the routing decision in $O(\log h_1(p))$ ($O(\log^2 h_2(p))$) time, for the case of unit (variable) packet size.
- Any dynamic update, including changing load on p, adding or deleting a node p, takes $O(\log h_1(p))$ time.

A. 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 simple 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 instead of using a central 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 imagine 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 4). At the end of the process, each node has a rank which is the highest level it is on.

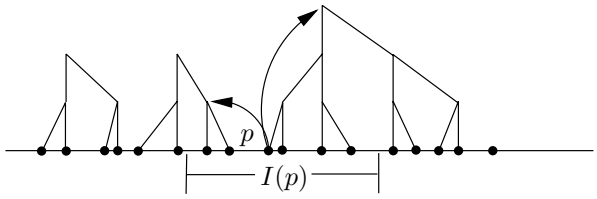

Fig. 4. Algorithm for requests with unit packet size.

The construction of $\mathcal{F}$ can be done by using leader election algorithms such as the randomized algorithm used in [32]. The virtual forest $\mathcal{F}$ is stored distributedly 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 F is stored implicitly such that each node stores the ID's of its parent and children. Now, we consider the communication range $I(p)$ of a node p. By the process $\mathcal{F}$ is created. 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 at each node is $O(\log h_1)$ in total.

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 $O(\log h_1)$ as well.

B. 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 a function $f_p(x) = \min_{x_p \le x_u \le x_p + x} \ell(u)$ and $g_p(x) = \min_{x_p + x - 1 \le x_u \le x_p} \ell(u)$. 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 5. Then we claim that,

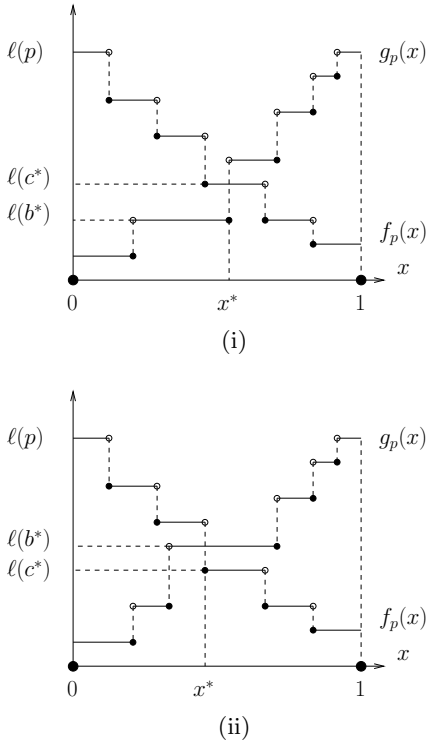

Fig. 5. (i) $g_p(x^*) = \ell(b^*) \le f_p(x^*) = \ell(c^*)$; (ii) $g_p(x^*) = \ell(b^*) > f_p(x^*) = \ell(c^*)$.

Lemma 4.1. *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 5 (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 contradiction. The case when $\ell(b^*) > \ell(c^*)$ can be proved in a similar way. □

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.

C. Handling dynamic changes

There are two types of events that may cause the dynamic change. One is when two nodes start or stop to become visible to each other due to either insertion/deletion of nodes or mobility. The other is when the load on a node changes. In both cases, such an event only affects a constant number of trees in the forest. If we use a dynamic balanced binary tree, then each dynamic change can be done in $O(\log h_1(p))$ time.

In the previous section, we describe our algorithms in a per-packet basis. Its main purpose is to provide a rigorous analysis. In practice, the control overhead can be high if we try balance load on a per-packet basis. Instead, we maintain for each node the level of its energy use and ran our algorithm on the levels. This way, a route can be cached, and a load update is only needed when the energy level of a node changes.

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

The load-balanced routing algorithm can be extended to a strip with width $w \le \sqrt{3}/2 \approx 0.86$(recall that 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 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 6(i)). The load of a bridge is then defined as $\max(\ell(b), \ell(c))$, and 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 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 5.1. *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 6(ii)). □

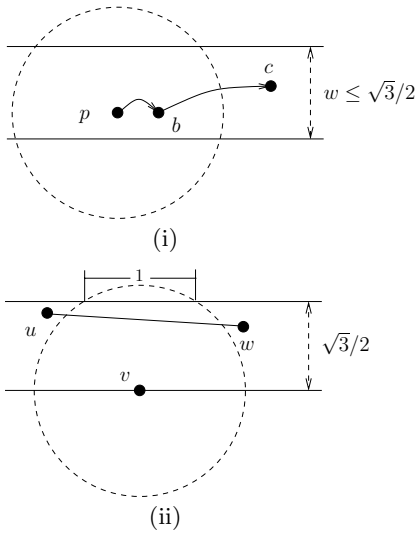

Fig. 6. (i) bc is a right bridge of p. (ii) u, w cannot see each other if they are on different sides of v and outside of v's communication range.

We now claim that

Theorem 5.2. *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 cannot stop at a node that is to the left of t. Otherwise, assume that the algorithm is stuck at a node p, i.e. p does not have a right bridge and does not see t. 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 the assumption(Figure 7). 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. By Lemma 5.1, 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 is a path.

Now, we bound the stretch factor of the algorithm. For a right bridge bc of p, since c is outside the communication range of p, we have that $x_c \ge x_p + 1/2$. Clearly, for a packet from s to t, it has to take at least $x_t - x_s$ hops, while

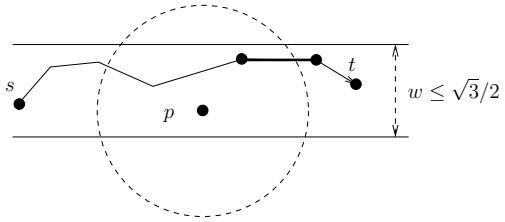

Fig. 7. The path from s to t has to cross the right boundary of p's communication range. The thickened edge is a right bridge of p.

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 3.2. 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 or destination, see Figure 8. 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.

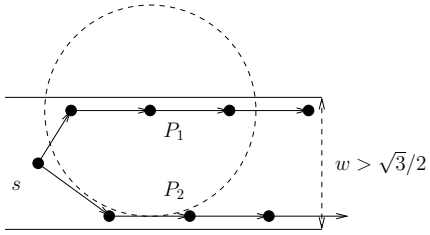

Fig. 8. A strip of width $w > \sqrt{3}/2$, P_1, P_2 are two paths from s.

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 like 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 the number of nodes in a neighborhood is typically small, it is unnecessary to use those heavy machinery.

VI. Simulations

We evaluate the load-balanced routing algorithm under various wireless nodes distributions and traffic patterns. The shortest path routing algorithm minimizes the total energy the network consumed but does not balance the loads on different nodes. So a node could easily be

overloaded under the shortest path routing. We compare GREEDY3 with the shortest path routing algorithm.

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.

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.

a) Unlimited energy and random traffic: We generate 1000 random packets, each with size randomly chosen between 1 and 10. In Figure 9, 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 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 is that we achieve substantially in terms of the maximum load ratio by paying a little higher price on the maximum and average delay.

b) 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,

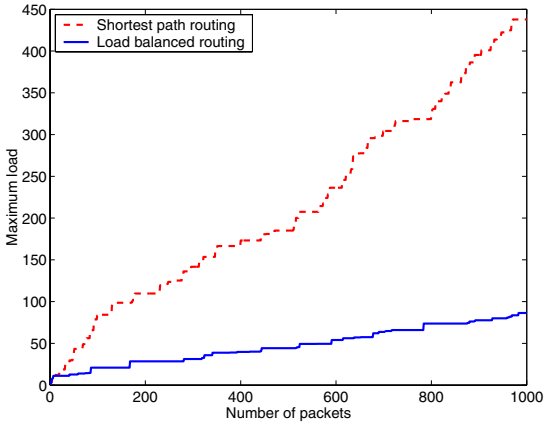

Fig. 9. The maximum node load in terms of the number of packets delivered under the random traffic pattern. The dashed line curve is for the shortest path routing, and the solid line curve for the load-balanced routing. The communication range has radius 5.

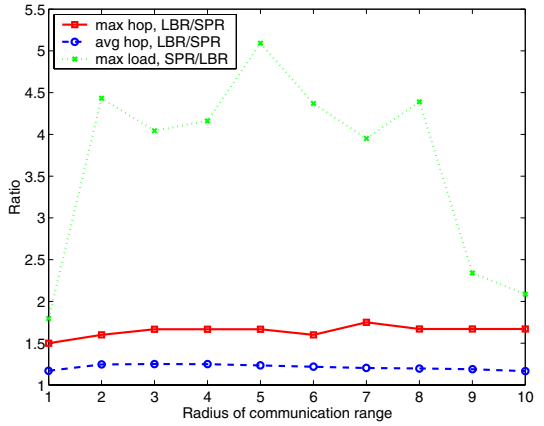

Fig. 10. The worst case and average ratio of the length of the paths produced by our algorithm to the shortest path length under different communication ranges, under the random traffic pattern. We also show the ratio of the maximum load under the shortest path routing to that under the load-balanced routing for different communication ranges.

if the communication range has radius 5.

c) 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 13. 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.

d) 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 14, the result for load-balanced routing is better than the case for random traffic.

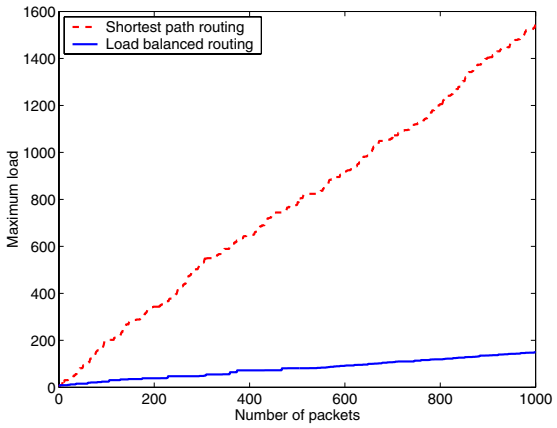

Fig. 11. The maximum node load in terms of the number of packets delivered under the aligned traffic pattern. The communication range has radius 5.

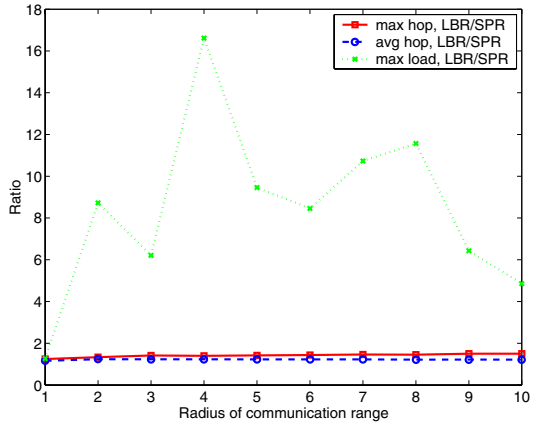

Fig. 12. The worst case and average ratio of the length of the paths produced by our algorithm to the shortest path length under different communication ranges, under the aligned traffic pattern. We also show the ratio of the maximum load under the shortest path routing to that under the load-balanced routing for different communication ranges.

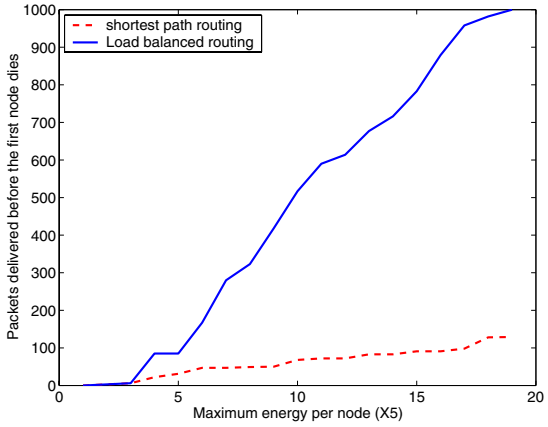

Fig. 13. The number of packets delivered when the first node dies in terms of the maximum energy of each node under the random traffic pattern. Again, dashed line curve is the shortest path routing, and sold line curve the load-balanced routing.

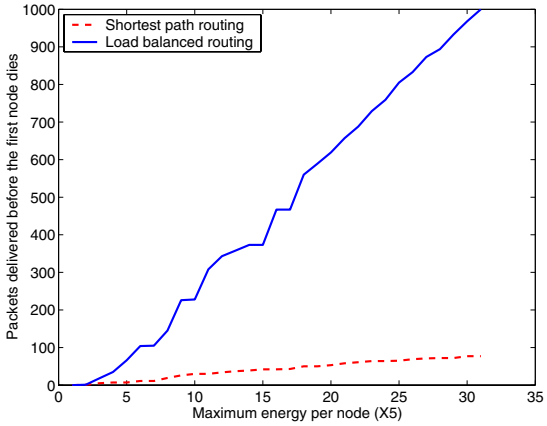

Fig. 14. The number of packets delivered when the first node dies in terms of the maximum energy of each node under the aligned traffic pattern.

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.

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. One interesting problem is to extend the study to the other node distribution. For such an extension, we may have to make assumption on the traffic patterns too as the load-balanced routing is difficult even for nodes that form a regular grid.

References

[1] Hannes Hartenstein, Bernd Bochow, André Ebner, Matthias Lott, Markus Radimirsch, and Dieter 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')*, 2001, pp. 259–262.

[2] Yossi Azar, "On-line load balancing," in *On-line Algorithms: The State of the Art*, A. Fiat and G. Woeginger, Eds., pp. 178–195. LNCS 1442, Springer, 1998.

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

[4] 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*, 1985, pp. 79–87.

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

[6] Serge A. Plotkin, "Competitive routing of virtual circuits in ATM networks," *IEEE Journal of Selected Areas in Communications*, vol. 13, no. 6, pp. 1128–1136, 1995.

[7] David B Johnson and David A Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds., vol. 353, pp. 153–181. Kluwer Academic Publishers, 1996.

[8] Charles E. Perkins and Elizabeth M. Royer, "Ad hoc on-demand distance vector routing," in *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 90–100.

[9] S. Ramanathan and Martha Steenstrup, "A survey of routing techniques for mobile communications networks," *Mobile Networks and Applications*, vol. 1, no. 2, pp. 89–104, 1996.

[10] N. Bambos, "Toward power–sensitive network architectures in wireless communications: Concepts, issues, and design aspects," *IEEE Personal Communications*, vol. 5, pp. 50–59, 1998.

[11] Qun Li, Javed Aslam, and Daniela Rus, "Online power-aware routing in wireless ad-hoc networks," in *Proc. ACM MobiCom*, 2001, pp. 97–107.

[12] A. Goldsmith and S.B. Wicker, Eds., *Special Issue: Energy-aware ad hoc wireless networks*, vol. 9, IEEE Wireless Comm., Auguest 2002.

[13] Christine E. Jones, Krishna M. Sivalingam, Prathima Agrawal, and Jyh-Cheng Chen, "A survey of energy efficient network protocols for wireless networks," *Wireless Networks*, vol. 7, no. 4, pp. 343–358, 2001.

[14] C. Petrioli, R.R. Rao, and J. Redi, Eds., *Special Issue: Energy Conserving Protocols*, vol. 6, Mobile Networks and Applications, June 2001.

[15] Suresh Singh, Mike Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *Proc. ACM/IEEE MobiCom*, 1998, pp. 181–190.

[16] Jae-Hwan Chang and Leandros Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proc. INFOCOM*, 2000, pp. 22–31.

[17] Jae-Hwan Chang and Leandros Tassiulas, "Fast approximation algorithms for maximum lifetime routing in wireless ad-hoc networks," in *Networking, LNCS 1815*, 2000, pp. 702–713.

[18] Javier Gomez, Andrew T. Campbell, Mahmoud Naghshineh, and Chatschik Bisdikian, "PARO: Power-aware routing in wireless packet networks," in *Proc. 6th IEEE International Workshop on Mobile Multimedia Communications*, November 1999.

[19] Yan Yu, Ramesh Govindan, and Deborah 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.

[20] C.-K. Toh, "Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks," *IEEE Communications Magazine*, pp. 138–147, June 2001.

[21] R. C. Shah and J. M. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Proc. IEEE WCNC'02*, March 2002.

[22] Vikram Srinivasan, Carla F. Chiasserini, Pavan Nuggehalli, and Ramesh R. Rao, "Optimal rate allocation and traffic splits for energy efficient routing in ad hoc networks," in *IEEE INFOCOM*, 2002.

[23] Gil Zussman and Adrian Segall, "Energy efficient routing in ad hoc disaster recovery networks," in *IEEE INFOCOM*, 2003.

[24] Ya Xu, John S. Heidemann, and Deborah Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proc. ACM MOBICOM*, 2001, pp. 70–84.

[25] Wei Yu and Jangwon Lee, "DSR-based energy-aware routing protocols in ad hoc networks," in *Proc. of the 2002 International Conference on Wireless Networks*, 2002.

[26] Nishant Gupta and Samir Ranjan Das, "Energy-aware on-demand routing for mobile ad hoc networks," in *Proc. IWDC*, 2002, pp. 164–173.

[27] Susanne Albers, "Better bounds for online scheduling," in *Proceedings of the ACM Symposium on Theory of Computing*, 1997, pp. 130–139.

[28] Jeffrey Hightower and Gaetano Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–667, August 2001.

[29] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proc. MobiCom*, 2001, pp. 166–179.

[30] Andreas Savvides and Mani B. Strivastava, "Distributed fine-grained localization in ad-hoc networks," *IEEE Transactions of Mobile Computing*, 2003.

[31] A. Ward, A. Jones, and A. Hopper, "A new location technique for the active office," *IEEE Personnel Communications*, vol. 4, no. 5, pp. 42–47, October 1997.

[32] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu, "Discrete mobile centers," *Discrete and Computational Geometry*, vol. 30, no. 1, pp. 45–65, 2003.