Geographical Quadtree Routing

Chen Avin, Yaniv Dvory, Ran Giladi
Department of Communication Systems Engineering
Ben Gurion University of the Negev
Beer Sheva 84105, Israel
Email: avin@cse.bgu.ac.il, dvory@bgu.ac.il, ran@cse.bgu.ac.il

***Abstract*—In this paper we offer a novel geographical routing algorithm that relies on a well known data structure called Quadtree. Quadtree is an efficient method of mapping a two-dimensional area by recursively partitioning it to disjoint squares. We present a greedy, guaranteed delivery routing algorithm called Greedy-Quadtree-Greedy (GQG). The algorithm is robust to dynamics in the non-Quadtree edges and overcomes local minimums without the use of planarization, face routing, or searching. GQG is a tree-based routing algorithm; it makes greedy forwarding based the location information that is extracted from the Quadtree addresses of the nodes. Bypassing voids is done by a concept of "tree routing with shortcuts", which can significantly improve hop stretch and load balancing. As part of the routing system, we present three algorithms: address distribution, network topology discovery, and geographical routing with guaranteed delivery. We keep all broadcasts bounded to one hop, and the nodes' routing state depends on their degree rather than the overall network size. We prove the correctness of the algorithms and present simulations that show the protocol improvement over simple tree-based routing.**

I. Introduction

As communication networks and data centers continue to grow in size, we are facing networks with a larger number of routing elements. Geographical routing has been proposed as a scalable routing system due to its reduced routing state and forwarding tables. In this type of algorithm nodes have information about the physical location of themselves, their neighbors, and the message's destination. This information is used to route the messages throughout the network. Using this information allows nodes to maintain a small routing state, which greatly reduces memory use and lookup time. Therefore geographical routing algorithms scale much better. The geographical information has recently become more available and accurate, with the development and spread of GPS and other positioning systems.

Initially geographic routing algorithms provided simple non-guaranteed delivery schemes [1]–[3], some packets could not reach their destination and were dropped. Algorithms with different approach relied on wide or bounded message flooding, which required a significant amount of overhead, and might not guarantee delivery. The next generation of geographic routing algorithms [4]–[7] guaranteed delivery using two basic *modes*. First, they route the packet in a *greedy* mode towards the destination, i.e., each node forwards the packet to one of its neighbors that is geographically closer to the destination. Second, when a packet reaches a *local minimum* (a.k.a. void), i.e., a node that is the closest to the destination among its neighbors, it switched to *escape* mode in order to bypass (i.e., escape) the local minimum. Some of the algorithms define switching back to greedy mode after bypassing the void, i.e., when the packet reaches a node closer to the destination than the last local minimum.

The first geographic routing algorithm that achieved guaranteed delivery was initially called *compass routing* [8], and was later called *Face routing*. Face routing is a very popular method to bypass voids and is used in many of the existing geographic routing algorithms [4]–[7], [9] in different variations, which gradually improved the routing hop efficiency. Face routing works only in planar graphs and might be very inefficient in route selections. Distributed planarization works nicely in some graph models such as unit disk graphs, but in general settings, planarization is not always possible, and when it is, it could still be complex and costly [10].

Other existing geographical routing algorithms use preprocessing of the network topology with different methods [11]–[14]. This is done in order to distribute virtual coordinates to the nodes to increase convexity of the voids in the virtual topology, and to reduce the probability of a packet reaching local minimum in first place.

In [15], Leong *et al.* offered GDSTR, a tree-based guaranteed delivery geographical routing algorithm. The tree is build so that each node is the root of an area containing its sub-tree. The reason GDSTR is able to guarantee the delivery of packets in a connected network is that the tree traversal forwarding mode alone is guaranteed to deliver the packet to any node in the network when greedy forwarding fails. One complication of the proposed method is in cases where the destination lies in the intersection of the areas rooted by two or more nodes. In this case, packet delivery is guaranteed by systematic depth first search in all the subtrees with areas containing the destination.

Paper Overview and Main Contributions: We adopt an approach similar to that of [15] and develop a distributed, guaranteed delivery, geographic routing algorithm that uses tree-based routing where greedy routing fails. The basic object we are using to build our tree and to distribute addresses to

The Authors are listed in Alphabetic order, this work is part of the M.Sc Thesis of the second author. This work was partially funded by the European Community's Seventh Framework Programme [FP7/2007-2013], grant agreement no. 215462.

978-1-4577-0681-3/11/$26.00 ©2011 IEEE

nodes is called *Quadtree* [16]. Quadtree is a well known data structure that has applications in data clustering, computational geometry, image processing, and more. The basic idea of Quadtree is to cover a plane region of interest by a square, and then recursively partition it into four smaller squares until each square contains a single point of interest, which, in our case, is simply a vertex.

Based on a Quadtree partitioning, we establish hierarchical partitioning of the network graph into geographical areas with no overlap. For each of the Quadtree squares we choose one or more representative out of the nodes it contains. This procedure well defines a geographical tree hierarchy that allows routing messages with guaranteed delivery and without using tree searches like DFS in order to reach the destination.

In addition, our tree routing enables the use of *shortcuts*. This novel concept provides a sort of greedy routing on the tree while avoiding local minimums, reducing the stretch of the route and increasing load balancing. "Shortcutting" occurs when a node finds one of its neighbors to be closer to the destination on the tree than its trivial tree neighbor (its parent or child). Moreover, the existence of the tree infrastructure is the only condition for the routing algorithm correctness. This means that other edges can be added and removed freely without contradicting the guaranteed delivery of the routing algorithms. The additional edges, which are not part of the infrastructure, are used when routing in greedy mode, and as shortcuts when routing in tree mode. This property gives flexibility of the network topology and allows dynamic addition and removal of edges.

We provide algorithms for network discovery, address distribution, and geographical routing with guaranteed delivery. We prove the correctness of our algorithms and present simulations that show this routing system has very low stretch (i.e., the routes that the algorithm finds are close to the shortest routes in the network) and good load balancing. Due to space constrains, proofs are provided in the full version [17].

II. MultiQuadtree Network and addresses

Quadtree Geographical Address: Consider a squared area in the plane, $\mathcal{S}$, and a set of nodes (communication stations) V distributed in $\mathcal{S}$. As mentioned, the idea of Quadtree is to recursively partition $\mathcal{S}$ into four smaller squares until each square contains only one vertex, i.e., empty sub-squares are not partitioned. Let *Quadtree$_{\mathcal{S}}(V)$* be the unique Quadtree partitioning of V. In the following we assume $\mathcal{S}$ is known and drop it from the notation. See for example Fig. 1 for a Quadtree we generated from a set of $17,168$ weather stations around the world, taken from Mathematica 7.

The $Quadtree(V)$ partitioning can be used to give addresses to squares and nodes. An *empty* Quadtree address, $\emptyset$, represents the whole original square area; then recursively each of the four quarters will get the address of the partitioned square with an added suffix to represent which quarter it is. In the following manner, the bottom left quarter will get the "00" suffix, the top left will get "01", the bottom right will get "10", and the top right will get "11". A node's address is the address

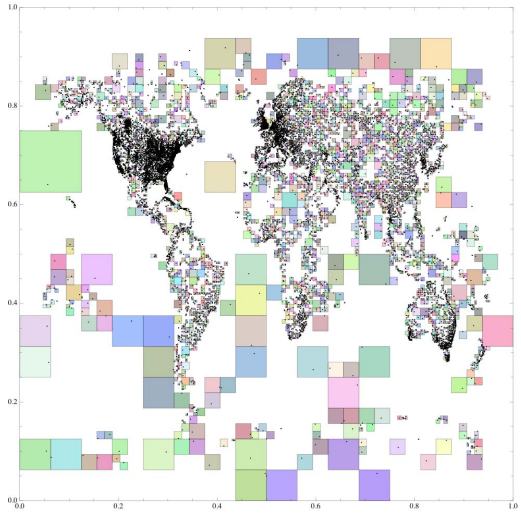


Fig. 1. Quadtree of $17,168$ weather stations around the world.

of the smallest square that it is contained in after the described partitioning is done. For $v \in V$, let $\mathrm{Qadd}(v)$ be the unique Quadtree address of node v resulting from $Quadtree(V)$.

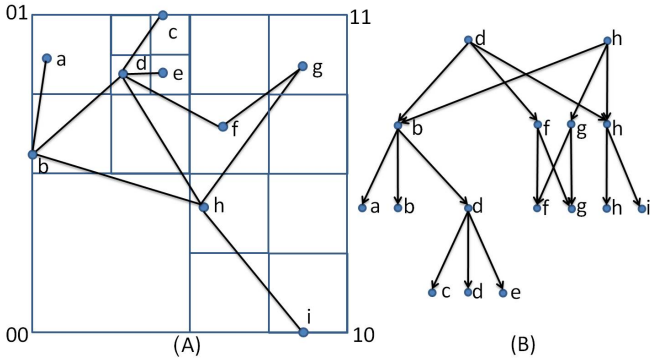
An example, is given in Fig. 2. Node h has the address "10 01" as it is at the top left quarter of the bottom right quarter of the whole square. Node *e* has the address "01 11 10".

This partitioning defines the tree hierarchy of squares: the original square is the Quadtree root and each partitioned square is the parent of its four quarters. A leaf square is a Quadtree square that contains one node only; therefore it is not partitioned.

MultiQuadtree Network: The idea of a *Quadtree network* is to use the square hierarchy of $Quadtree(V)$ in order to create a routing hierarchy. This is done by first assigning *representatives* to each square in the Quadtree hierarchy and connecting representatives according to the Quadtree hierarchy. In a *MultiQuadtree network*, a node can represent one or more squares it is contained in, and a square can have one or more representative nodes. As we show, this increases load balancing, and allows finding short paths while keeping the hierarchy well defined. We now give notation and definitions to be used in this paper.

Let $\overline{x}$ be a Quadtree address and $|\overline{x}|$ be the length of the Quadtree address, defined as the number of *bit pairs* in $\overline{x}$. A *k-Qsquare* is a squared area in $Quadtree(V)$, with an address $\overline{\alpha}$ such that $|\overline{\alpha}| = k$. Let α be a k-Qsquare, the *parent* of α is the unique $(k-1)$-*Qsquare* that contains α. α's parent address has α's address removing two-bit suffix. The *sub-Qsquares* of α are the four $(k+1)$-*Qsquares* contained in it. Their addresses are α's address with added two-bit suffixes. Let the *Ancestors* of α be all the *i-Qsquares* containing α such that $0 \leq i < k$. α is a *Populated Qsquare* iff it contains at least one node. α is a *Leaf Qsquare* iff it contains exactly one node.

For a Qsquare α, let *RepSet(α)* denote the set of the nodes that are the representatives of α. Recall that a representative of a Qsquare must belong to the Qsquare. We now define when



	a	b	c	d	e	f	g	h	i
Qadd(x)	01 01	01 00	01 11 11	01 11 00	01 11 10	11 00	11 11	10 01	10 10
TSet(x)	01 01	01 00 01	01 11 11	01 11 00 01 11 empty	01 11 10	11 00 11	11 11 11	10 01 10 empty	10 10

(C)

Fig. 2. MultiQuadtree network example: (A) $G(V,E)$ and $Quadtree(V)$. (B) The MultiQuadtree hierarchy. (C) Qadd and Tset table

a graph G is considered a *MultiQuadtree*:

Definition 1. *A graph G(V,E) is a MultiQuadtree network iff: There exists an assignment of representatives to all populated Qsquares in $Quadtree(V)$ s.t.*

1) *Any representative of populated k-Qsquare in $Quadtree(V)$, where $k > 0$, has a link (i.e., edge $e \in E$) to at least one representative of its parent Qsquare.*
2) *Any representative of populated k-Qsquare in $Quadtree(V)$ that is not a leaf Qsquare, has links (i.e., edges $e \in E$) to at least one representative of each of its populated sub-Qsquares.*

In the above definition we consider nodes as connected to themselves. A *root* of a MultiQuadtree network is a representative of the 0-Qsquare. Note that a graph that is a MultiQuadtree must have certain edges according to definition 1, but have no limitation on additional edges. The following lemma is a consequence of the definition:

Lemma 1. *A MultiQuadtree network is connected.*

The definition of a MultiQuadtree network allows nodes to represent several Qsquares in the hierarchy and therefore they may have few addresses. For $v \in V$, let $\mathrm{Tset}(v)$ be the set of addresses of all the Qsquares v represents, including its leaf Qsquare with the address $\mathrm{Qadd}(v)$. Formally, for any Qsquare α such that $v \in RepSet(\alpha)$, v will keep the Quadtree address $\overline{\alpha}$ in its $\mathrm{Tset}(v)$.

Fig. 2 illustrates the above definitions: (A) present graph $G(V,E)$, which is a MultiQuadtree network according to Definition 1 and the $Quadtree(V)$ partitioning. (B) shows the MultiQuadtree network hierarchy; arrows are stretched from Qsquare representatives to their populated sub-Qsquares representatives. (C) The Qadd and Tset of all the nodes, for example: nodes d and h represent the 0-Qsquare (they are both roots for this MultiQuadtree network), as they are both connected to representatives from all populated 1-Qsquare, which are b, f, h for node d and b, g, h for node h. Node h, as a representative for Qsquare "10", is also connected to all of its populated sub-Qsquares representatives i and h itself. Node f represents 11 as it is connected to the representatives of its populated sub-Qsquares g and itself, in the same way node g also represents 11. As one can see, a node can represent few different Qsquares it is contained in. For example node d represent three different Qsquares, their addresses are in TSet(d). Moreover, a Qsquare can be represented by few nodes. For example the Qsquare with the address 11 is represented by nodes f and g, and the whole square is represented by the two roots d and h.

In the full version of the paper we offer a distributed algorithm for allocating the Quadtree addresses to nodes in a MultiQuadtree Network. Nodes, in this algorithm, locally negotiate with neighbors to determine their exact Quadtree addresses until each node v in the network is familiar with its exact $\mathrm{Qadd}(v)$, and the Quadtree addresses of its neighbors. Assigning representatives is described later in section IV.

One concern of a Quadtree Network is the existence of "long" edges. We claim that the average length of edges in a Quadtree network is of the same order as the minimum spanning tree (MST). For example, the average edge length of the MST of n points located at $\sqrt{n}\times\sqrt{n}$ grid on the unit square is $\frac{1}{\sqrt{n}}$, while the average edge length of a Quadtree Network of these points is $\frac{2}{\sqrt{n}}$. In addition, we ran both MST and random Quadtree network construction on the $17,168$ points of Fig. 1; the average edge length in the MST came to 0.0018, while in the Quadtree Network it was 0.00503, which is only 2.78 times bigger. We now turn to present our routing algorithm.

III. Quadtree-Based Routing Algorithms

The main routing algorithm we offer is a combination of two routing modes, the first is *greedy mode*, in which a node forwards a message to the neighbor that is physically closest to the destination (and closer than the routing node itself). The greedy forwarding based the location information extracted from the Quadtree addresses of the nodes. This mode use local information, uses any available edge, but fails when reaching a local minimum. This requires a way to calculate the physical distance based on the Quadtree addresses, which are the only information a node has on the location of its neighbors and of the destination. We call this function G_{dist}.

The second routing mode is called *Quadtree mode*. It is used for overcoming local minimums in order to provide the guaranteed delivery quality, and is done by a new concept of "tree routing with shortcuts". In this mode a node routes a message to the neighbor that is closest in the MultiQuadtree hierarchy to the destination. We call the function of calculating this type of distance MQ_{dist}.

Let w, d be two nodes in V. Let $G_{\mathrm{dist}}(w,d)$ be the Greedy distance between nodes w and d.

Algorithm 1 Q: Quadtree Routing

A source node s is sending a message m to a target node t with $\overline{t} = Qadd(t)$. Each node $v \neq t$ that receives m does the following:

1) find neighbor $u^* = \underset{u \in N(v)}{argmin}(MQ_{\mathrm{dist}}(u, \overline{t}))$
2) If $MQ_{\mathrm{dist}}(u^*, \overline{t}) < MQ_{\mathrm{dist}}(v, \overline{t})$
   a) route the message to u^*.
3) else:
   a) destination unreachable - terminate the message.

$G_{\mathrm{dist}}(w, d)$:= The Euclidean distance between the bottom left corners of the squared area represented by $\mathrm{Qadd}(w)$ and $\mathrm{Qadd}(d)$, i.e., the Quadtree addresses of the nodes.

Now, since leaf Qsquares are disjoint, clearly:

Lemma 2. *$G_{\mathrm{dist}}(w, d) \in \mathbb{R}^+$ and $G_{\mathrm{dist}}(w, d) = 0$ iff $w = d$.*

In order to define MQ_{dist}, let $\mathrm{lmp}(\overline{x}, \overline{y})$ to be the longest, even lengthed, matching prefix of $\overline{x}$ and $\overline{y}$, where $\overline{x}, \overline{y}$ are two Quadtree addresses. A closer look at $\mathrm{lmp}(\overline{x}, \overline{y})$ shows us that by definition, it is the Quadtree address of the smallest common ancestor Qsquare of the squares represented by addresses $\overline{x}$ and $\overline{y}$. Let $MQ_{\mathrm{dist}}(w, \overline{\alpha})$ be the MultiQuadtree network distance between node w and Qsquare α with address $\overline{\alpha}$. $MQ_{\mathrm{dist}}(w, \overline{\alpha})$ is calculated in the following way:

$$MQ_{\mathrm{dist}}(w, \overline{\alpha}) := \underset{\overline{w} \in \mathrm{Tset}(w)}{min}(|\overline{w}| + |\overline{\alpha}| - 2 \times |\mathrm{lmp}(\overline{w}, \overline{\alpha})|)$$

For example, the MQ_{dist} between b and $\overline{e}$ in Fig. 2, where $\overline{e} = \mathrm{Qadd}(e)$ is:

$$\begin{aligned} MQ_{\mathrm{dist}}(b, \overline{e}) = \\ & min(|0100| + |011110| - 2 \times \mathrm{lmp}(0100, 011110) \\ & , |01| + |011110| - 2 \times \mathrm{lmp}(01, 011110)) \\ & = min(2 + 3 - 2 \times 1, 1 + 3 - 2 \times 1) = 2. \end{aligned}$$

Again the distance is zero only when the two are the same:

Lemma 3. *Let $\overline{d} = \mathrm{Qadd}(d)$, then $MQ_{\mathrm{dist}}(w, \overline{d}) = 0$ iff $w = d$.*

Lemma 2 and 3 present two properties that will allow us to route messages in the network with guaranteed delivery.

A. Quadtree Routing with Shortcuts

Quadtree mode routing works using edges that are not necessarily the tree edges, as long as the node we route the message to is closer to the destination on the tree. This is a novel concept of greedy-like routing that is based on the tree distance we defined earlier - MQ_{dist}. We calculate the tree distance from a node to the Quadtree address of the target based on the Tset of the node, which defines the locations of the node in the MultiQuadtree network hierarchy. In this way a node looks at all its neighbors, and may use any link it has when routing in Quadtree mode rather than just looking at its parent and sub-Qsquares in the hierarchy. This property leads to a better hop efficiency and better load balancing as simulations show. This algorithm can be implemented independently, and provides guaranteed delivery.

Let $N(v)$ denote the neighbors of v, i.e., all nodes that v is connected to. The Quadtree mode routing is defined in Algorithm 1. The following theorem states Algorithm 1's correctness and bounds its performance.

Theorem 1. *Let u and d be nodes in a MultiQuadtree network where $\overline{d} = \mathrm{Qadd}(d)$. Routing a message m from u to d with Algorithm 1 has guaranteed delivery. Moreover, $MQ_{\mathrm{dist}}(u, \overline{d})$ is an upper bound of the number of hops in the route.*

Theorem 1 shows that when a node routes a message in tree mode, it will always find a parent or a child that is closer than itself to the destination (in terms of MQ_{dist}). Note however, that additional edges, or as we call them *shortcuts*, to nodes on the tree can only help by connecting the current node to a neighbor with even a lower MQ_{dist} (than the parent or child it must have). Recall that in step 1 of the algorithm the current node chooses the neighbor with the minimal MQ_{dist} to the destination so using these shortcuts lowers the upper bound on the number of hops to the destination. The usage of these additional edges takes place only when they are helpful, but there is no necessity in having them to guarantee delivery. Therefore there is no need to recalculate the network hierarchy and addresses when adding and removing shortcuts; this makes the protocol robust to dynamics on the topology of the non-tree edges. This is a major advantage over planarization-based routing that demands recalculating the planar graph when the topology changes.

B. GQG: Greedy-Quadtree-Greedy Routing Algorithm

We now define the **G**reedy-**Q**uadtree-**G**reedy (GQG) routing algorithm for routing a message in a graph $G(V, E)$ that is a MultiQuadtree network.

A source node *s* is sending a message m to a target node *t* with $\overline{t} = \mathrm{Qadd}(t)$. The message m holds the following fields: (1) $m.source$, (2) $m.target$, (3) $m.mode$, and (4). $m.localmin$. (1) is used to identify the message source. (2) is used to identify the target Quadtree address. (3) is used to identify the routing mode, which can be $\{$*tree/greedy*$\}$. (4) stores the G_{dist} between the target and the last local minimum node, and is used to determine whether it is possible to change the routing mode from *tree* back to *greedy*. The message is initialized with the following parameters: $m.source = \mathrm{Qadd}(s)$, $m.target = \mathrm{Qadd}(t)$, $m.mode = greedy$, and $m.localmin = G_{\mathrm{dist}}(s, t)$. *GQG* is presented in Algorithm 2 and Theorem 2 below states the correctness of the GQG algorithm.

Theorem 2. *GQG has guaranteed delivery in finite Multi-Quadtree networks.*

IV. MultiQuadtree Network Discovery

Our routing algorithms are based on the concept that each node v knows its $\mathrm{Qadd}(v)$ and its $\mathrm{Tset}(v)$, namely what

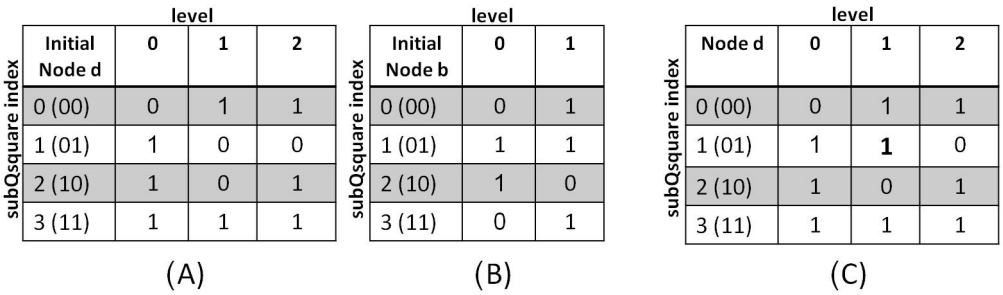


Fig. 3. Example of population maps of nodes from the network shown in Fig. 2): (A) The initial population map of node d, (B) The initial population map of node b, and (C) The population map of node d after receiving and learning the population map of node b (after executing $\mathrm{LearnMap}_d(b)$. Consequently node d now knows it can not represent the qsquare 01, because qsquare 0101 is populated but d has no neighbor in that qsquare, and in particular no neighbor that is a representative candidate for the sub-Qsquare 0101.

Algorithm 2 GQG: Greedy-Quadtree-Greedy Routing

The algorithm starts at *s* and defines the behavior of any node *v* that receives *m* destined to node $t \neq v$: (otherwise *v* is the target and the algorithm stops).

1) Check *m.mode*:
   a) greedy: perform step 2.
   b) tree: perform step 3.
2) Greedy mode:
   a) find neighbor $u^* = \underset{u \in N(v)}{argmin}(G_{dist}(u,t))$
   b) if $G_{dist}(u^*,t) < G_{dist}(v,t)$
      i) route the message to u^*.
   c) else:
      i) *m.localmin*$\longleftarrow G_{dist}(v,t)$.
      ii) change *m.mode* to *tree.*
      iii) perform step 4.
3) Mode examine:
   a) If $G_{dist}(v,t) < m.localmin.$
      i) change *m.mode* to *greedy*
      ii) perform step 2.
   b) else :
      i) perform step 4.
4) Quadtree mode:
   a) Follow Algorithm 1

Qsquares it represents. As mentioned earlier, algorithm QAD (not presented here) goal is to provide to each node in V its Qadd, based on each node's location information. The task of selecting representatives is more complex, we offer an additional, local, distributed, algorithms for MultiQuadtree Network discovery and representatives selection called *MQD* (MultiQuadtree Network Discovery Algorithm). MQD is initiated after the QAD algorithm is finished and all the nodes in the network have their own and their neighbors' Quadtree addresses. When MQD is finished, all nodes in the network are familiar with their *TSets*, and can start routing messages with Algorithms 1 or 2. Due to space constrains the MQD algorithm is given in the full version only [17]. In this version we give a general overview, discuss the complications and properties and proof the correctness.

The MQD algorithm is built from two main phases:

Phase I: Works bottom-up from the MultiQuadtree network leaves towards its root(s). This phase insures that in order for v to become a *representative candidate* of Qsquare α, the following two conditions must be satisfied: 1. v is contained in α; and 2. v is connected to a representative candidate of each of the populated sub-Qsquare of α (i.e., condition 2 in definition 1). Messages in this phase are sent only by candidates; initially all nodes are candidates for their leaf Qsquares, but their quantity decreases as the algorithm climbs the Quadtree hierarchy.

Phase II: Works its way from the MultiQuadtree network root(s) towards the leaves. The purpose of this phase is that a *representative candidate* for a Qsquare α will become a *representative* for α if and only if it is connected to a representative of α's parent Qsquare (i.e., condition 1 in definition 1), unless α is the root Qsquare and becomes a representative without having a parent.

The selection of representative is complex since we want a distributed and local algorithm, i.e., nodes only know about their direct neighbors and can only send messages to them. The complexity arises since in order for a node to become a *representative candidate* for a Qsquare α it has to know which of α's sub-Qsquares are populated. The problem is that a node which populate a certain sub-Qsquare can still be several hops away from a potential representative. We overcome this challenge by defining a data structure called *population map* (Pmap). An example for this problem is given in Fig. 3.

Population Maps: A population map of a node v is a matrix of bits denoted by $\mathrm{Pmap}(v)$, which keeps the knowledge v has about population in the square. The size of the map is determine by the address length of v. Node v uses its population map in the first phase of the MQD algorithm to inform its neighbors of the existing populated Qsquares it knows of. Next, Pmaps are aggregated and combined in such a way that when a node decides whether or not it can become a representative candidate for Qsquare α, it already knows which of α's sub-Qsquares are populated. Therefore it only needs to

check if, for each of α's populated sub-Qsquares, it received a Phase I message from a neighbor that is a representative candidate of that sub-Qsquare.

To define the Pmap data structure we need to define the *Sibling* relation: two Qsquares, α and β, are **k-Sibling Qsquares** if $\mathrm{lmp}(\overline{\alpha}, \overline{\beta}) = k$. Similarly, two nodes, u and w are **k-Sibling nodes** if $\mathrm{lmp}(\mathrm{Qadd}(u), \mathrm{Qadd}(w)) = k$. A node u is a **k-Sibling** of a Qsquare α if $\mathrm{lmp}(\overline{\alpha}, \mathrm{Qadd}(u)) = k$

The population map for node v, $\mathrm{Pmap}(v)$, is a matrix of bits of size $4 \times |\mathrm{Qadd}(v)|$. Rows present the 4 sub-Qsquares and are numbered 0 to 3 and columns present Qsquares level and numbered from 0 to $|\mathrm{Qadd}(v)| - 1$. Node v will have '1' in its population map at cell $[i, j]$ iff v knows that its j-Sibling Qsquare with index i is populated and '0' otherwise.

For example in Fig. 3 (A) we see the initial population map of node d in the network shown in Fig. 2 (A). Node d has $\mathrm{Qadd}(d)$= 01 11 00, it knows that the j-Qsquares ($j = 0, 1, 2$) it is contained in are populated, therefore cells $[1, 0], [3, 1], [0, 2]$ in $\mathrm{Pmap}(d)$ are set to '1'. Its neighbor e has Qadd 01 11 10, and is a 2-sibling of d, therefore d know that the 2-Qsquare 01 11 with index 2 (i.e., 10) is populated and cell $[2, 2]$ is set to '1'. In the same way cell $[3, 2]$ set to '1' because of its neighbor c. Neighbor b with address 01 00, is 1-sibling therefore cell is $[0, 1]$ set to '1'. And the 0-sibling neighbors h, f with addresses 10 01, 11 00 sets cells $[1, 0], [0, 0]$. In Fig. 3 (B) we can see the initial population map of node b in the network of Fig. 2 (A).

When a node v receives form a neighbor u its $\mathrm{Pmap}(u)$ and its quadtree address $\mathrm{Qadd}(u)$ it update its own $\mathrm{Pmap}(v)$ with what u knows according to the function $\mathrm{LearnMap}_v(u)$:

$$\mathrm{LearnMap}_v(u) :=$$

$$\begin{cases} \mathrm{Pmap}(v)[i,j] & j > \mathrm{lmp}(\bar{v}, \bar{u}), \\ & i \in \{0,1,2,3\} \\ \mathrm{Pmap}(v)[i,j] \vee \mathrm{Pmap}(u)[i,j] & j \leq \mathrm{lmp}(\bar{v}, \bar{u}), \\ & i \in \{0,1,2,3\} \end{cases}$$

where $\bar{v} = \mathrm{Qadd}(v)$, $\bar{u} = \mathrm{Qadd}(u)$.

Fig. 3 (C) presents the population map of node d after receiving and learning the population map of node b. Bit $[2, 1]$ is now set to one since node d received the information about the existence of node a from the population map of node b. Consequently node d now knows it can not represent the Qsquare 01, because it has no neighbor its sub-Qsquare 01 10 and in particular no neighbor that is a representative candidate for the sub-Qsquare 0101.

A Maximal MultiQuadtree(V, E) is the unique MultiQuadtree(V, E) where each node that can be a representative of Qsquare α according to Definition 1, will be a Qsquare α representative. Namely, Maximal MultiQuadtree(V, E) is the maximal assignment of representatives such that conditions 1 and 2 in Definition 1 are met.

We show that our algorithm finds the *Maximal MultiQuadtree(V,E)* network.

Formally, we prove the following about the MQD algorithm:

Theorem 3. *Node $v \in V$ becomes representative of a Qsquare α in MQD algorithm iff v is a representative of α in the Maximal MultiQuadtree(V,E).*

Using the Maximal MultiQuadtree(V, E) found by MQD algorithm is very important for load balancing since it is better to use more edges and different nodes when routing in Quadtree mode. Note that after the completion of the algorithm, each node is familiar with its (maximal) Tset and ready to perform routing.

V. Simulation

We implemented Algorithms MQD, QAD, Q, and GQG in the C++ based simulation platform OMNET++ 4. We studied these algorithms using different topologies and network models, but here we report only a subset of the results. For each simulation we first distributed N nodes uniformly at random in the unit square. Then we ensure the MultiQuadtree network condition by creating random representative trees. Tree construction is made by randomly choosing *T* roots to be level 0 representatives, followed by recursively connecting each representative to a randomly chosen representative for each of its populated sub-squares. In this paper we consider two topologies for additional edges. i) *Pure tree*: we do not add any edges; this model is used for comparison and allows us to eliminate use of the non-tree edges (i.e., *shortcuts*). ii) *Distance related* model: edges are added randomly with probability proportional to the reciprocal of the length of the edge to the power of two (i.e., shorter edges have greater probability). The probability is normalized to achieve a desired average degree, D, in the network.

For each model the final MultiQuadtree network to be used is discovered by the MQD algorithm described in section IV. To test the algorithms, we choose uniformly at random 5% of the $\binom{N}{2}$ pairs to communicate, and compare the performance of both the Q algorithm (Quadtree routing, Algorithm 1), and the GQG algorithm (Greedy-Quadtree-Greedy, Algorithm 2). The results reported are an average of the simulations.

Main Results: Fig. 4 (A) shows the average hop stretch of the different algorithms as a function of the network size N. Each point on this graph is the average hop stretch of 5% random pairs of source and destination (out of all possible pairs), and for 10 different random networks (node set and topology). Note that the same random networks and traffic patterns were used for testing the different algorithms. We can see the best stretch was achieved by the Q algorithm with five roots, $T = 5$. The worst was found when routing on the tree edges only (without shortcuts) with one root (i.e., $T = 1$). The number of roots reduces the stretch achieved by Q routing, but does not influence on the GQG algorithm since most of the routing is done in greedy mode. In general the stretch does not increase significantly when increasing the network size.

One concern of tree-based routing algorithms is an unbalanced load distribution. This is caused by high traffic at the

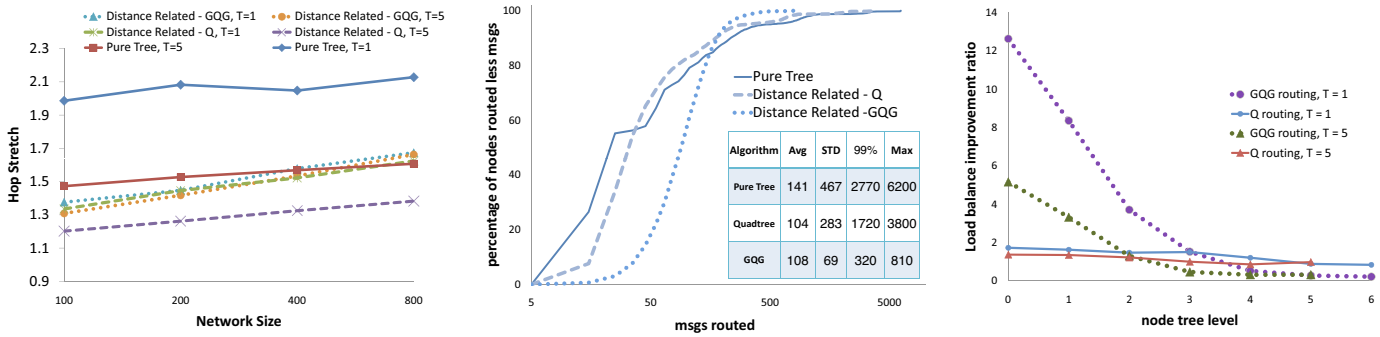


Fig. 4. (A) D=10, Average found path length for different algorithms as a function of the network size. (B) 400 nodes , D=10, T=1, Accumulating Load balance. (C) 400 nodes, D=10, Distance Related graph, Pure Tree/GQG routing load balance ratio.

root and the nodes close to it. We show that this problem is significantly reduced when we use shortcuts and greedy routing instead of pure tree routing. Fig. 4 (B) presents the cumulative load distribution using different routing algorithms over the distance related graph. The X-axis is the number of messages forwarded by a node presented in logarithmic scale. The Y-axis shows the cumulative percentage of nodes that forward equal or smaller number of messages. The maximal X value is the maximum number of messages passing through any node, i.e., maximum load. The standard deviation value is an important value as well; the smaller it is the better the load balancing. An optimal load balancing graph would be a straight vertical line on the average. The pure tree routing is worst with the largest maximum value of 6200 and with very large standard deviation; this is since most of the about 8,000 messages ($5\% * 400^2$) traverse through the tree root. Routing with the Quadtree algorithm greatly improves the load balancing, with a maximum value of less than 3900 messages, and a standard deviation value that improve by a 1.65 ratio. Using GQG reduces the maximum value to 810 messages, and the standard deviation reduces by a 4.1 ratio from the Quadtree routing on the same graph. This indicate a significant improvement in the load balancing. Fig. 4 (C) shows the ratio of load balancing when using the GQG or Q algorithms vs. using routing based on the tree edges only, as a function of the nodes position in the MultiQuadtree network hierarchy. The X-axis is the node level in the MultiQuadtree network, and the Y-axis is the ratio between the average quantity of messages a node routes when using the tree edges only, to the average amount of messages a node routes when using GQG or Q algorithms. Here we see clearly that the GQG distributes the load better by reducing the load of nodes closer to the roots and adding load to nodes at a higher level in the hierarchy.

VI. Conclusions

In this paper we offer a Quadtree-based network hierarchy and a geographical address system that are then used in two routing algorithms with guaranteed delivery. In the results here (and in [17]) our local, distributed algorithm presents a low routing stretch and good load balancing. Future research is in place to determine the use of Quadtree-based networks in the context of data centers, wireless networks, peer-to-peer networks, and cases where the Quadtree-based network infrastructure is not feasible and only connectivity is guaranteed.

References

[1] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *Communications, IEEE Transactions on*, vol. 32, no. 3, pp. 246 – 257, 1984.

[2] T.-C. Hou and V. Li, "Transmission range control in multihop packet radio networks," *Communications, IEEE Transactions on*, vol. 34, no. 1, pp. 38 – 44, 1986.

[3] G. Finn, "Routing and addressing problem in large metropolitan-scale internetworks," ISI Research Report, University of Southern California, Tech. Rep. ISI/EE-87-180, March 1987.

[4] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *DIALM '99*, 1999, pp. 48–55.

[5] B. Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *MOBICOM-00*, 2000, pp. 243–254.

[6] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: of theory and practice," in *PODC '03*, 2003, pp. 63–72.

[7] B. Leong, S. Mitra, and B. Liskov, "Path vector face routing: Geographic routing with local face information," in *Network Protocols, IEEE International Conference on*, 2005, pp. 147–158.

[8] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *CCCG*, 1999.

[9] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing," in *MobiHoc '03*, 2003, pp. 267–278.

[10] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic routing made practical," in *NSDI'05*, 2005, pp. 217–230.

[11] B. Leong, B. Liskov, and R. Morris, "Greedy virtual coordinates for geographic routing," in *ICNP 2007*, 2007, pp. 71 –80.

[12] J. Newsome and D. Song, "Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information," in *SenSys '03*, 2003, pp. 76–88.

[13] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *MobiCom '03*, 2003, pp. 96–108.

[14] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. Gu, "Greedy routing with guaranteed delivery using ricci flows," in *Information Processing in Sensor Networks, 2009. IPSN 2009. International Conference on*, 2009, pp. 121–132.

[15] B. Leong, B. Liskov, and R. Morris, "Geographic routing without planarization," in *NSDI'06*, 2006, pp. 25–25.

[16] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 03 1974.

[17] Y. Dvory, "Arithmetic geographical coding and routing," Master's thesis, Ben Gurion University of the Negev, 2010. [Online]. Available: http://www.bgu.ac.il/~avin/papers/gqr_thesis.pdf