OnionMap: A Scalable Geometric Addressing and Routing Scheme for 3D Sensor Networks

Kechao Cai, Zhimeng Yin, Hongbo Jiang, *Member, IEEE*, Guang Tan, *Member, IEEE*, Peng Guo, *Member, IEEE*, Chonggang Wang, *Senior Member, IEEE*, and Bo Li, *Fellow, IEEE*

***Abstract*—Geometric routing or geo-routing has been shown as a promising approach to scalable routing in sensor networks. Despite its success in 2-D networks, very few designs are available for 3-D networks that can ensure short routes using only small per-node state, without incurring high load imbalance on the nodes. In this paper, we propose a novel addressing and routing scheme, i.e., *OnionMap*, for 3-D sensor networks that achieve the above goals, using solely connectivity information and at a linear message cost. The key idea is to decompose a 3-D network into a set of connected layers, which are then mapped to a set of concentric sphere structures (similar to an onion). On each sphere, a discrete Ricci flow method is used to assign each node a set of coordinates that permits purely greedy routing within that sphere; across the different spheres, a layer alignment algorithm helps rotate and scale the spheres, to form a coherent global coordinate system that guides global routing. Theoretical analysis and simulation show OnionMap's advantages over state-of-the-art solutions in path stretch, per-node storage, and load balance.**

***Index Terms*—Geometric routing, 3-D sensor networks.**

I. Introduction

FOLLOWING their success in 2-D environments, sensor networks have drawn growing interest among the research community for the 3-D emerging applications in building surveillance, underwater survey, atmospheric monitoring, and so on. As a primitive service for sensor networks, point to point routing remains a critical enabling technology for such applications. Unfortunately, due to the vastly increased complexity of network topology, the shift from 2-D to 3-D brings significant challenges to scalable routing [1], [3] that very few adequate designs if any are available for 3-D cases.

In this paper, we concentrate on *greedy routing* in 3-D sensor networks. Greedy routing is attractive for its simplicity and scalability, as a node only needs to store the coordinates of its neighbors, and makes routing decisions solely based on the coordinates. Although greedy routing has been a well-studied object in 2-D sensor networks, it encounters great difficulties in 3-D network settings. A central problem that any greedy routing algorithm faces is the problem of *local minimum*, where a node cannot find a neighbor that is closer to the destination. In 2-D sensor networks, face routing is a typical way to recover from a local minimum. However, such a strategy requires a planarized network graph, which has no corresponding version in 3-D networks. Therefore, new ideas are required to make 3-D geometric routing possible and efficient.

We propose a new scalable geometric addressing and routing scheme, named *OnionMap*, for 3-D sensor networks. OnionMap offers a number of desirable properties: (1) *Location free*: The algorithm is purely connectivity based and does not need any physical location information; (2) *Boundary detection free*: The algorithm does not need boundary information, thus avoids a large amount of message exchange and performance uncertainty; (3) *Low stretch*: The generated routing paths have close to optimal lengths; (4) *Load balance*: The routing traffic is evenly spread over all the network nodes, which helps prolong the network lifetime; and (5) *High scalability*: The total message cost is linear with network size, and the per-node storage cost is very low and does not change with network size.

The main idea of OnionMap is to organize the network nodes into an onion-like structure (see Fig. 1) consisting of a set of concentric sphere layers. Within each layer, nodes can route greedily with 100% success rate, while the different layers are further organized to allow global routing. At a first glance, this approach is similar to the traditional decomposition based routing schemes [2], [8], [15], [18], in which the network is divided into pieces and routing is performed in an intra- and inter-piece manner. However, the particular problem we are facing contains two new challenges that are not found in previous work: (1) *Layer decomposition*. Layers should be connected and closed surfaces so that their mapping to spheres is possible. The simplest form of such surfaces is concentric spheres, with

Manuscript received January 9, 2014; revised April 16, 2014; accepted June 3, 2014. Date of publication June 9, 2014; date of current version January 7, 2015. The work of G. Tan was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 61103243 and 61379135, and by Shenzhen Overseas High-level Talents Innovation and Entrepreneurship Funds under KQC201109050097A. The work of H. Jiang was supported in part by the NSFC under grants 61073147 and 61173120, and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry). The associate editor coordinating the review of this paper and approving it for publication was L. Lai.

K. Cai and Z. Yin are with the Department of Electronics and Information Engineering and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: caikechao@gmail.com; yinzhimeng@gmail.com).

H. Jiang and P. Guo are with the Department of Electronics and Information Engineering and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hongbojiang2004@gmail.com; guopeng@mail.hust.edu.cn).

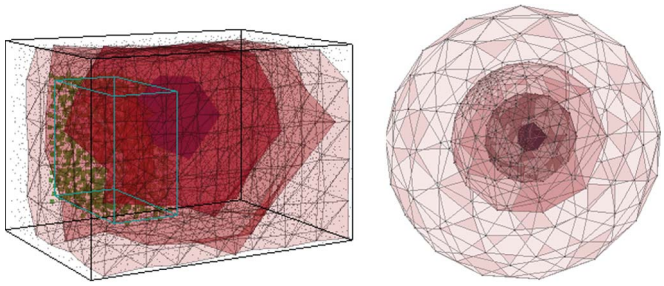
G. Tan is with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: guang.tan@siat.ac.cn).

C. Wang is with InterDigital Communications, Melville, NY 11747 USA (e-mail: cgwang@ieee.org).

B. Li is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Kowloon, Hong Kong, and also with Shanghai Jiao Tong University, Shanghai 200030, China (e-mail: bli@cse.ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2014.2329677



(a) (b)

Fig. 1. Example layers of a 3-D network, and the mapped onion-like structure (i.e., concentric spheres). (a) Example layers. (b) Onion-like structure.

radius defined by hops; however a general network cannot be naturally divided into spheres due to shape irregularity. Therefore, we need a new form of layers with similar properties, and find a way to extract them. (2) *Layer embedding*. Given a likely irregular layer of nodes, how to assign each node a set of coordinates that ensure successful greedy routing is a challenge. Moreover, to achieve truly 3-D geometric routing, instead of an *ad hoc* combination of reduced 2-D versions, the different coordinate systems of the multiple layers should be properly aligned, so that a (nearly) consistent global coordinate system can be provided for the geo-routing algorithm.

To address the first challenge, we propose an *incremental layer construction* method to extract layers that have provable connectedness. The layers also form simple closed surfaces that make geometric processing at later stage possible. For the second problem, we use a discrete Ricci flow method [22], followed by a uniformized stereographic projection, to assign each node within a layer a set of coordinates. After these procedures, each layer is mapped to a unit sphere, on which purely greedy routing between all node pairs is allowed. As an optimization, these unit spheres are scaled and aligned to form a global coordinate system. This final coordinate system makes it possible to route through the whole 3-D volume of network efficiently.

We compare OnionMap against a number of previous solutions, in particular the Bubble Routing scheme (Bubble for short) [20], which represents the state-of-the-art geo-routing solution for general 3-D networks. OnionMap improves over Bubble with significantly lower path stretch (with a reduction up to 41%), without Bubble's reliance on the knowledge of network boundaries, whose detection is highly nontrivial and often expensive [6], [12], [25]. Furthermore, these improvements are achieved at only linear message cost, which is asymptotically optimal.

II. Related Work

It is shown by Durocher *et al.* [1] that there does not exist a deterministic localized routing algorithm in 3-D sensor network. To avoid this obstacle, the Greedy-Random-Greedy (GRG) algorithm [3] uses random walk, which is inefficient in escaping local minima due to its blindness to topology. Liu *et al.* [13] propose a position-based algorithm, GHG, which partitions a network into convex hulls with partial unit Delaunay triangulations. The convex hulls restrict the recovery process in a subspace, thus improve the routing performance. Lam *et al.* [11] present a MDT protocol suite that builds a multidimensional tree structure to guarantee packet delivery in a 3-D space. To support greedy forwarding, MDT constructs virtual links to mask the connectivity irregularity in the physical network. A potential problem of MDT is that the virtual links may be very long, thus imposes high state on the nodes. MDT has been used by an algorithm GDV [16] which shows good performance in certain network scenarios. GDSTR-3-D [27] is another 3-D routing algorithm that constructs convex hull trees to route packets around voids. GDSTR-3-D relies on nodes' physical location information. Zhou *et al.* [26] propose a virtual coordinate assignment algorithm PSVC for 3-D networks. PSVC uses greedy forwarding but does not guarantee delivery.

In an alternative approach, graph embedding techniques are used to assign virtual coordinates to 3-D network nodes for geometric routing. In [17] and [24], a 2-D triangulated domain with holes is mapped to a Euclidean circular disk or a hyperbolic space. The embedding algorithms, however, do not extend to 3-D cases. Yu *et al.* [23] consider a scalable routing algorithm for 3-D high genus networks. This algorithm embeds the network connectivity graphs into a possibly non-zero genus surface. However, the embedding method is iterative and needs expensive message exchanges.

In [21], a deterministic greedy routing algorithm is proposed to embed a 3-D sensor network to a solid ball by using fine tetrahedron mesh structures. This algorithm restricts the shape of the 3-D network to a topological ball, with at most a single hole. The Bubble Routing [20] algorithm (Bubble for short) improves on [21] by allowing the topology to contain multiple holes. The basic idea of Bubble is to decompose the network into a set of Hollow Spherical Cells (HSCs). By mapping the boundaries of HSCs (i.e., the Hollow Spherical Bubbles (HSBs)) to spheres and constructing virtual trees inside each HSC, packets can be greedily routed inside each HSC. Bubble presents an elegant addressing solution for the network. However, its routing algorithm may create suboptimal routes in some cases. If the source and destination are in different HSCs, the packet will first route toward a beacon node on the shared boundary of the adjacent HSBs. When the packet reaches a boundary node, it will be routed along the boundary of the HSC that contains the destination node, potentially creating a detour. Fig. 2 shows an example of Bubble path and OnionMap path. The source node S and destination node T are in different HSCs enclosed by HSBs. The packet will first move toward a beacon node shown by the solid red point on the shared boundary of the adjacent HSBs. Afterwards, the packet will travel along the boundary of the HSC that contains the destination node, before it moves toward T. This indirect movement clearly creates a suboptimal path, as shown by the dotted red line.

Unlike Bubble, OnionMap names all the nodes in a global coordinate system, thus avoids the boundary effect suffered by Bubble. This leads to OnionMap's better performance in path stretch. Another advantage of OnionMap over Bubble is that it does not rely on the knowledge of fine-grained network boundaries; instead it produces network boundaries as a byproduct. Previous work has shown that boundary detection is nontrivial and expensive [6], [12], [25]. For example, the algorithms in

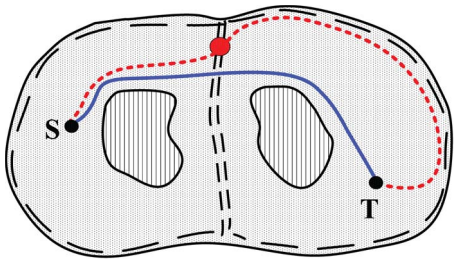


Fig. 2. Comparison between Bubble and OnionMap in a 3-D topology with two holes (cross section view). The solid blue and dotted red lines represent the routing paths between S and T by OnionMap and Bubble, respectively. The long dashed black line is the HSB (Hollow Spherical Bubble) in Bubble Routing. The region enclosed by HSB is HSC (Hollow Spherical Cell). The solid red point is the beacon node in Bubble.

[6], [25] both involve complicated probing procedures to infer the boundary information; the UNFOLD algorithm [12] makes an important advance toward dynamic boundary detection, but requires nodes' location or range information.

III. Layer Decomposition

In this section, we describe how OnionMap decomposes the network into a set of thin layers. The layers are centered at a *center node*, which acts as a coordinator for the layer decomposition process.

The center node can be a randomly selected node from the network. It is preferable to make the center close to the network's physical center, therefore we can select a center from a number of candidates as follows. In the beginning, a randomly chosen starting node floods the network, and selects a small constant number (e.g., 5) of candidate centers on its broadcast tree. Then, each of these nodes floods the network, and obtains its broadcast tree's *inner depth*, defined as the minimum depth of a leaf node on the tree. Finally, the starting node selects the candidate with the maximum inner depth as the final center node. The broadcast tree of the chosen center node serves as a helper structure to be used later.

A. Incremental Layer Construction

The layers are defined and constructed in an inductive way. Let $\mathcal{N}$ be the set of nodes in the network graph G. $\mathcal{N}$ is divided into a set of layers $\mathcal{L}_k$, so we have $\mathcal{N} = \cup_k \mathcal{L}_k$. Ideally, a layer should form a sphere, with radius defined by its hop distance to the center. Unfortunately, in a general network, the set of nodes at a certain hop distance to the center, called a *Hop Set*, are normally neither connected nor closed. Thus we need to find additional nodes, which constitute a *Patching Set*, to help such a hop set to get connected and closed.

Definition 1. (Hop Set and Fragment): The kth *Hop Set*, denoted H_k, is the set of nodes that are k hops away from the center node on the center node's broadcast tree. The ith *Fragment* of H_k, denoted F_k^i, is a connected component of H_k. Thus $H_k = \cup_i F_k^i$.

For example, F_k^i and F_k^j shown in Fig. 3 are the ith fragment and jth fragment in the kth hop set.

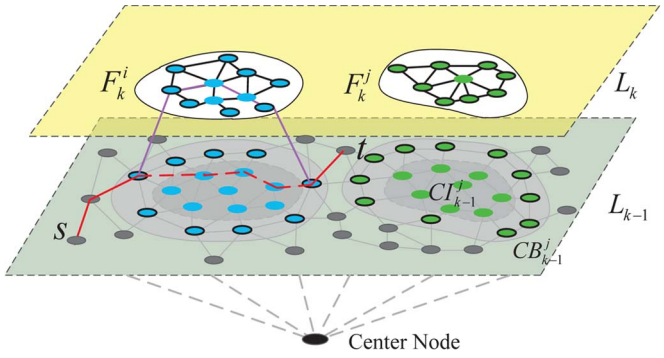


Fig. 3. Two example layers and related node sets.

The basic idea of constructing $\mathcal{L}_k$ on the basis of the established $\mathcal{L}_{k-1}$ is to replace some pieces on $\mathcal{L}_{k-1}$ with the newly found fragments in H_k. Thus we need to define such to-be-replaced pieces. Note that the Hop Sets in different layers are virtually separated even though they could be overlapping physically. Thus such overlap has no impact on the layer construction.

Definition 2. (Covered Set and Uncovered Set): The ith *Covered Set* of F_k^i, denoted C_{k-1}^i, is the set of nodes in $\mathcal{L}_{k-1}$ that have links to the nodes in F_k^i. The ith *Uncovered Set* of $\mathcal{L}_{k-1}$, denoted $\overline{C}_{k-1}$, is the set of all nodes in the $(k-1)$th layer except those in the covered sets, that is, $\overline{C}_{k-1} = \mathcal{L}_{k-1} - \cup_i C_{k-1}^i$.

In Fig. 3, the nodes represented by blue dots form the ith covered set C_{k-1}^i, and the nodes represented by green dots form the jth covered set C_{k-1}^j.

Definition 3. (Covered Boundary Set and Covered Inner Set): The *Covered Boundary Set* of C_{k-1}^i, denoted CB_{k-1}^i, is the set of nodes in C_{k-1}^i that have links to nodes in $\overline{C}_{k-1}$. The *Covered Inner Set* is the remaining nodes in C_{k-1}^i, that is, $CI_{k-1}^i = C_{k-1}^i - CB_{k-1}^i$.

For example, in Fig. 3, the nodes represented by green dots bounded by black circles in the light gray area form the covered boundary set CB_{k-1}^j; the nodes represented by green dots in the dark gray area form the covered inner set CI_{k-1}^j.

Definition 4. (Patching Set): The kth *Patching Set*, denoted P_k, is the set of nodes in $\mathcal{L}_{k-1}$ except those in CI_{k-1}^i, that is, $P_k = \mathcal{L}_{k-1} - \cup_i CI_{k-1}^i$.

The layer construction begins with a CORE which contains the 1, 2, and 3 hop neighbors of the center node. Compared with the network size, the CORE is relatively small. The CORE is designated as layer 0, which can be seen as a virtual closed shell. Then, assuming that the $(k-1)$th layer is already constructed, we start constructing the kth layer, under the instructions from the center node. The process consists of three steps:

1) Identifying Fragments in $\mathcal{L}_k$: The center node floods a message within its k hops, asking the nodes in the k hop set H_k to perform asynchronous floodings within H_k. The flooded message from a node p in H_k contains the ID of p. If a node receives a flooded message before it starts its own flooding, then it cancels its flooding action. A node always forwards a flooded message, unless it has forwarded a previous message containing a larger ID than the current one. In this way, every

connected component, i.e., fragment of H_k, will be able to elect a leader—the flooding node with the largest ID. Every node is then assigned a fragment ID equal to the leader's ID.

2) Identifying Covered Inner Nodes in $\mathcal{L}_{k-1}$: Every fragment leader floods a message within its fragment, asking all its members to determine their covered nodes in $\mathcal{L}_{k-1}$. Each of those covered node then checks if it is a neighbor of an uncovered node in $\mathcal{L}_{k-1}$; if not then it marks itself as a covered inner node.

3) Creating $\mathcal{L}_k$: All the fragment leaders of $\mathcal{L}_k$ asynchronously flood a message containing the value k throughout the kth layer fragments and the $(k-1)$th layer, assigning a layer ID k to all the nodes except the covered inner nodes identified in the last step.

Up to now, all nodes that have been assigned a layer ID k become the desired $\mathcal{L}_k$.

B. Connectedness

The following theorem states that all layers are connected.

Theorem 1: Assume $\mathcal{L}_{k-1}$ is connected, then the union set $\mathcal{L}_k = H_k \cup P_k$ forms a connected graph.

Proof: $\mathcal{L}_k$ is essentially the union of the kth hop set and $\mathcal{L}_{k-1}$, with the covered inner sets on the $(k-1)$th layer removed. Since $\mathcal{L}_{k-1}$ is connected, to see whether $\mathcal{L}_k$ is connected we only need to check whether the removal of the covered inner sets will disconnect a pair of nodes. Consider a path between two nodes s and t that passes through the covered inner set (the dark gray area containing blue dots) in Fig. 3. The removal of this dark gray area will cause the path segment (indicated by the red dashed line) between two nodes in the covered boundary set to be dropped. However, since these two nodes are connected to F_k^i and F_k^i is connected, they can find a path within F_k^i (see the purple line). Thus the connectivity of the original end nodes remains in $\mathcal{L}_k$. Applying the above argument to any pair of nodes in $\mathcal{L}_k$ will show that no two nodes in $\mathcal{L}_k$ are disconnected, thus proving the theorem. □

C. Closedness

Though its connectedness is guaranteed, a layer generated by our construction method does not necessarily form a simple closed surface. For example, in Fig. 4(a), where a normal $\mathcal{L}_{k-1}$ is shown as a red closed surface indicated by a red dotted line in Fig. 4(b). When $\mathcal{L}_k$ is generated, it contains H_k and $\mathcal{P}_k$. While $\mathcal{P}_k$ comes from $\mathcal{L}_{k-1}$, it contains the nodes in the overlapped surface of $\mathcal{L}_k$ and $\mathcal{L}_{k-1}$, i.e., the overlapped area indicated by the overlapped blue dotted line and red dotted line shown in Fig. 4(b). Therefore, such a $\mathcal{L}_k$ is no longer a simple closed surface. In additional to the surface depicted in blue color, it also contains a smaller hole, whose surface consists of two parts: a small piece (in black and enclosed by a yellow circle) which emerges during the layer construction process, and the surface around the egg-shaped inner hole (in red) which is caused by the blindly patching from $\mathcal{L}_{k-1}$ to $\mathcal{L}_k$. Such egg-shaped surface is included because it is not a covered node set by $\mathcal{L}_k$.

Clearly, the surface introduced by the hole should be removed from $\mathcal{L}_k$. Observe that the formation of the hole is triggered by the emergence of the black piece, which we call a *closing set*. Thus, we need to determine the time when this closing set appears, and find a way to eliminate the hole surface. We do this by augmenting the layer construction process with an extra step. First we introduce two additional concepts.

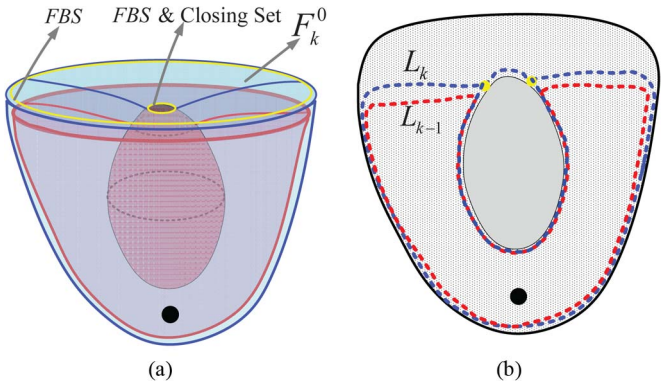


Fig. 4. A pathological layer (blue boundary) that contains an inner hole (in egg shape). The black dot represents the center node. (a) 3-D view; (b) cross sectional view.

Definition 5. (Fragment Boundary Set (FBS)): The *Fragment Boundary Set* (FBS) of a fragment F_k^i, is a set of connected nodes among F_k^i, each of which is a neighbor of some node in CB_{k-1}^i.

In Fig. 4(a), for example, the yellow circles represent the fragment boundary sets of F_k^0.

Definition 6. (Closing Set): A *Closing Set* is a fragment boundary set whose two farthest nodes are no more than l hops away, where l is the landmark spacing parameter.

The closing set is defined with the parameter l because when a component in the network is smaller than l hops in diameter, it is treated as if it were a single node in the landmark-level network. From the viewpoint of the landmark based triangulation routine, this represents a point in time when a new closed surface emerges.

An FBS is easy to detect: every node in a fragment F_k^i checks whether it is linked to a node in CB_{k-1}^i. Those with links to CB_{k-1}^i elect their leaders, called *FBS leaders*, using a distributed leader election algorithm (e.g., the one used by nodes in fragments to elect leaders). Each FBS leader then checks whether the FBS is a closing set by scoped flooding. If an FBS is a closing set, then its leader initiates a flooding within $\mathcal{L}_{k-1}$, with the restriction that the flooded message is only forwarded to uncovered nodes in $\mathcal{L}_{k-1}$. The restriction in effect forces the message to propagate over the egg-shaped surface in Fig. 4(a). We call the set of nodes on such a surface a *hole set*. The nodes in the hole set receiving the flooded message then simply drop themselves from $\mathcal{L}_k$. Thus the egg-shaped surface is removed from $\mathcal{L}_k$.

D. Network Boundaries as a Byproduct

An interesting byproduct of the layer composition process of OnionMap is a set of network boundaries, including the one for the outer network boundary and those for the inner holes. Clearly, the outmost layer constitutes a connected and closed surface for the network's outer boundary. This surface is also tight due to the incremental nature of layer construction:

the outmost layer first includes all the nodes with the largest hop count to the center node; then it finds nodes to ensure connectedness and closedness from the second outmost layer, which in turns finds nodes from more inner layers, in a recursive way. For each inner hole of the network topology, the closing set and its associated hole set constitute a closed and connected surface (Fig. 4(a)).

IV. Layer Embedding

This section describes the main steps in layer embedding, by which each node obtains a set of virtual coordinates.

A. Triangulation

Once the layers are constructed, we extract a triangular mesh structure for each layer. This can be done by means of triangulation. Without physical location information, we have to use hop-count distance metric to get a coarse-grained triangulation. This can be achieved by landmark Voronoi diagram. Each landmark performs a local flooding and measures the hop-distance to the other nodes. Landmark based Voronoi graph is a standard technique that has been used extensively in sensor networks [5], [17]. We first select a maximal l-hop (e.g., $l = 4$) independent set, whose nodes are used as landmarks in our algorithm. Thus any two landmarks in layer 1 are at least l hops apart and any non-landmark node is within l hops of some landmark.

In layer 2, the landmarks are selected in a slightly different manner. If a layer 2 node happens to be a landmark or a neighbor of some landmark in layer 1, then it is given a higher priority of being selected as a landmark in layer 2. From layer 3 onwards, we do landmark selection in the same way as in layer 2. As a result, some landmarks in the kth layer are selected from the neighbors of landmarks in the $(k-1)$th layer. It should be noted that a node may belong to multiple layers on the topological boundaries. This kind of nodes may be selected as landmarks in multiple layers.

Next, for each layer, we obtain a closed triangular mesh with the chosen landmarks using a method similar to [6]. More specifically, in a given layer, a set of disjoint virtual edges that connect the landmarks in the same layer are extracted first. Then the triangles are generated from a distributed triangulation process, which guarantees that every virtual edge is associated with only two triangles. For example, Fig. 5(a) shows a closed triangular mesh for the outmost surface of a cube.

B. Virtual Coordinates Generation

We then map the closed triangular mesh surfaces to unit spheres, following a three-step implementation similar to the procedure in [20].

The first step of the mapping is to map the triangular surface into a plane. By removing a node and its neighboring triangular faces, a topological sphere is converted into a topological disk. Then the discrete Ricci flow method [22] is applied to get a planar disk. In this step, all landmarks in a layer except the removed node are assigned a set of planar coordinates. The details of the coordinate assignment by Ricci flow can be found in [7], [17]. For example, in Fig. 5(a), the triangles in magenta

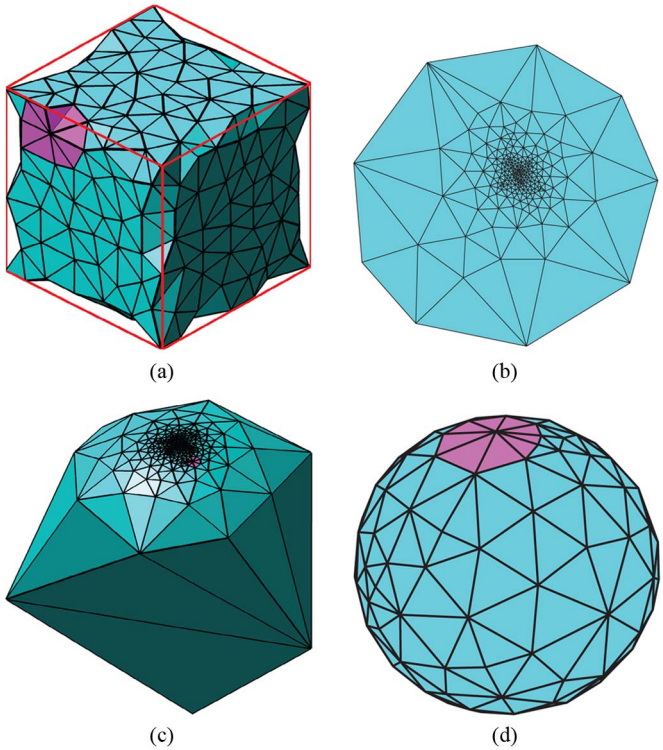


Fig. 5. Uniformized spherical mapping. (a) A triangulated layer; (b) mapped unit disk; (c) mapped non-uniform sphere; (d) mapped uniform sphere.

are removed first, then the triangles in cyan are mapped to a unit disk shown in Fig. 5(b).

The second step is mapping the plane obtained in the first step to a unit sphere. This can be easily done by a standard stereographic projection. In this step, all the landmarks including the removed one are projected on to a unit sphere. In Fig. 5(d), for example, triangles in the unit disk shown in Fig. 5(b) are mapped to a sphere shown in Fig. 5(c), and the removed node is relocated to the center of its neighboring landmarks. Then, the removed triangles in magenta are attached to the triangles in cyan, forming a closed spherical surface.

The third step is a Möbius transformation on a unit sphere. It can be seen that landmark nodes on the spherical surface shown in Fig. 5(c) are not uniformly distributed. This step is to make sure that the landmarks are uniformly distributed on the spherical surface. Otherwise, the stretch factor and load balancing performance would be poor.

After the three steps, each triangular mesh surface is transformed into a unit sphere where landmarks are uniformly distributed (shown in Fig. 5(d)). Each layer has a unique spherical coordinate system, and every landmark belongs to at least one coordinate system. A landmark that belongs to multiple layers would store the IDs of all the layers it resides in, as well as the corresponding coordinates.

C. Layer Alignment

The spherical mapping of the layer results in a set of independent, concentric unit spheres. So far these spheres are misaligned, which may result in inefficient cross-layer routing. To address this problem, we need to (roughly) align them into one coordinate system.

Quaternion Rotation Based Layer Alignment: The idea of layer alignment is based on the fact that the adjacent layers have either neighboring or shared landmarks. The connections among these landmarks can serve as a skeleton of the whole topology. We do alignment as follows. First, we fix $\mathcal{L}_1$ as a reference layer, and then we align the other layers one by one. Specifically, we randomly select a landmark on $\mathcal{L}_{i+1}$. The landmark calculates the centroid C_i of its neighboring landmarks in l hops on $\mathcal{L}_i$, and the centroid C_{i+1} of its neighboring landmarks in l hops on $\mathcal{L}_{i+1}$. Then it computes an offset vector $\mathbf{v_i}$ from the origin to C_i, and the other offset vector $\mathbf{v_{i+1}}$ from the origin to C_{i+1}. The two vectors allow us to compute the rotation $\mathbf{R}$ from $\mathbf{v_{i+1}}$ to $\mathbf{v_i}$ using the *quaternion rotations* [10]. The landmark then floods a message containing the rotation $\mathbf{R}$ to all the other landmarks on $\mathcal{L}_{i+1}$. Each of these landmarks adjusts its position vector $\mathbf{p}$ (points from the origin to its virtual coordinate) to a new position vector $\mathbf{p'}$ by performing the *quaternion product* $\mathbf{RpR^{-1}}$. Finally, each landmark node on $\mathcal{L}_{i+1}$ scales its radius by multiplying its layer number, for example, $i+1$ in this case. Till now, the landmarks on $\mathcal{L}_{i+1}$ are aligned with the landmarks on $\mathcal{L}_i$.

Such alignment operations are performed from $\mathcal{L}_2$ to the maximum layer one by one for several times. Each time, a leader is selected on each layer to sum up the errors of the virtual Euclidean distances among the landmark neighbors in different layers, where the error equals to the absolute value of difference between the virtual Euclidean distance and the hop count (i.e., 1) of each pair of neighboring landmarks in different layers. Such errors are used as the metric to evaluate the extent of alignment. In our experiments, three rounds of layer alignment are sufficient for the errors to be minimized and stable. Thus, we can align all the layers and obtain a global coordinate system for the whole network.

State Compression: A landmark lying on the network boundaries tends to belong to multiple layers. Since it has to maintain coordinates for each layer, the storage cost may be high. We found that most of these landmarks share almost the same radial direction in the aligned Onion-like structure. Such property can be utilized to compress the state information on this kind of landmarks. Specifically, if a landmark finds that it belongs to multiple layers, and that its radial directions in those layers have difference within a small threshold (e.g., 5° in both polar angle and azimuth angle), it compresses its states by only maintaining one copy of the virtual coordinates and a range of layers it belongs to. In this way, a landmark can save a significant amount of storage.

V. Routing in OnionMap

In this section, we describe the routing method of OnionMap.

A. State Information Maintained by Nodes

During the layer decomposition step described in Section III, the center node has built a broadcast tree spanning the whole network like [19]. Every node can then learn its depth d, and report its depth back up the tree. (A delay proportional to depth may be introduced in this process so that the reports from most nodes are suppressed by those from deeper levels.) The center determines the maximum depth d_M, and makes it known to all other nodes by a second flooding. This flooding also commands all the nodes of depth d_M, which we call the *edge nodes*, to perform an *edge flooding* in the network. Specifically, all the edge nodes broadcast a packet containing a unique ID at approximately the same time. Every non-edge node, upon receiving the message for the first time, will broadcast the message. A non-edge node then records its *downward parent*, the node from which it receives the flooded message for the first time. The purpose of the edge flooding is to establish links for nodes to route toward any level below itself on the broadcast tree (hence any layer). We call the downward parent links and the (upward) parent links on the original broadcast tree *inter-layer links*, which guarantee reachability between any two layers.

Recall that in OnionMap there are two types of nodes: landmark nodes and non-landmark nodes. A non-landmark node p stores several pieces of state information: 1) depth d with respect to the center node. 2) the ID of p's parent on the center's broadcast tree; 3) the ID of its downward parent; 4) the IDs of its neighbors; 5) the ID and coordinates of its home-landmark on p's innermost layer, and the path to that home-landmark. The home-landmark of a node is the landmark in the Voronoi cell that the node belongs to. Within every Voronoi cell, the home-landmark builds a local shortest path tree rooted at itself. Each non-landmark node can learn the parent to its home landmark. Note that a non-landmark node only keeps track of its home-landmark on its innermost layer, though it may belong to multiple layers.

A landmark node maintains similar state to that of a non-landmark node except the home-landmark related part (i.e., the 5th part). In addition, a landmark node maintains some other information: 1) a range of layer IDs it is associated with; 2) its virtual coordinates corresponding to the layers. Due to state compression, a landmark often maintains only a single copy of the polar angle for its associated layers. 3) neighboring landmarks and their virtual coordinates on each layer; 4) the shortest paths to the neighboring landmarks within l ($l = 4$ in our implementation) hops of the network.

B. Routing

OnionMap routing works at two levels: the landmark level and local level (within a landmark's Voronoi cell). Each landmark establishes a shortest path tree within its Voronoi cell on a layer, and routing within the cell is easily guided by the tree. For brevity of exposition, in the following we consider only the general case where the source and destination nodes are in different cells. We will also omit the trivial part of routing between the end nodes and their home-landmark nodes. This allows us to focus on the routing on the high-level landmark network, whose nodes we call *L-nodes*.

Routing on the L-nodes works in two modes: greedy mode and recovery mode. By default, the packet is forwarded greedily toward the destination L-node under the guidance of the global coordinate system. Note that there always exists a greedy path between any two L-nodes on the same layer. Since the

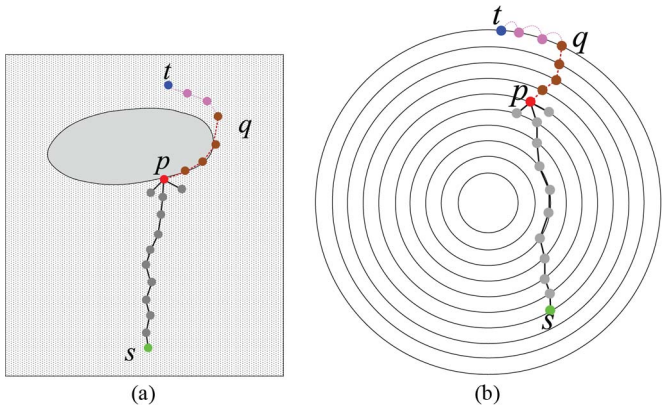


Fig. 6. A typical routing path in an actual network and its OnionMap structure. (a) Actual routing path; (b) routing path in OnionMap.

global coordinate system is not necessarily uniform due to the distortion of layer embedding (described in Section IV), greedy routing may fail. In this case, the routing enter would enter the recovery mode. The packet follows the inter-layer links (described in Section V-A) to reach the layer that contains the destination L-node, and then switches back to greedy mode.

Fig. 6 gives an example of routing path between the source node s (green node) and destination node t (blue node) in an actual physical network and its OnionMap structure. The path marked with grey dots represent the greedy path in the guidance of OnionMap. The greedy forwarding, however, gets stuck at the red node p as routing comes across a barrier (the grey area) as shown in Fig. 6(a), where it cannot find any neighboring node nearer to t and thereafter switches to the recovery mode. The packet then moves along the inter-layer links to reach the node q (path marked with brown dots), which is on the same layer as t. Afterwards the packet routes on the sphere greedily until it reaches t (path marked with pink dots). Theorem 2 guarantees the reachability of the packet under OnionMap.

Theorem 2: OnionMap guarantees packet delivery for any pair of nodes in a stable network.

Proof: Again we consider only two L-nodes. The reachability is ensured by two facts: the inter-layer links that guarantee reachability between any two layers, and the coordinates within a layer that guarantee the success purely greedy routing. Thus we have the theorem. □

VI. Message Cost Analysis

We briefly analyze the complexity of message cost of OnionMap, in comparison with Bubble Routing [20]. Let n be the network size, l_k the nodes of the kth layer. The message cost is measured by the total number of messages exchanged during the addressing process.

Theorem 3: OnionMap costs $O(n)$ message transmissions to complete the addressing process.

Proof: In its first step, OnionMap chooses a center node among a small constant number of nodes, which perform flooding in the network. Thus, the relevant cost is $O(n)$.

In the layer construction step, the nodes in $\mathcal{L}_k$ check their connectedness and closeness by asynchronous flooding within $\mathcal{L}_k$, thus the message cost is $O(l_{k-1} + l_k)$. Summing up the costs of all the layers gives a total cost of $O(n)$.

For each layer k, a discrete Ricci flow is used to compute virtual coordinates which are then mapped to a unit sphere. According to [20], this step needs $O(l_k)$ message cost. Thus the network's total cost of the coordinate generation is $O(n)$. Finally, during the coordinate alignment process, in order to adjust $\mathcal{L}_k$ with reference to $\mathcal{L}_{k-1}$, the message cost is $O(l_k)$. Therefore, the total cost is $O(n)$.

Put things together, OnionMap uses $O(n)$ message exchanges to establish a global coordinate system for routing. □

In comparison, Bubble Routing needs $O(n_b)$ messages for addressing, where n_b denotes the number of boundary nodes. Notice that Bubble Routing relies on a boundary detection process which takes at least $O(n)$ message cost [25]. So the total message cost of Bubble Routing is $O(n)$. In summary, both schemes are asymptotically optimal in message cost.

VII. Simulation

We have implemented OnionMap using C++ for simulation study. Fig. 7 shows the five 3-D topologies used in our simulation. The nodes are randomly distributed in a 3-D space with holes. The number of nodes of the former four topologies ranges from to 12 000 to 17 000. The last topology m-Hole is an example of a medium sized network which has 2735 nodes. Table I shows the size of the out boundary of each topology and the corresponding size of each hole. The unit is the communication range R. The average node degree is from 15 to 20. The degree setting appears to be higher than typical 2-D settings, but it is reasonable as a third dimension is introduced to the network. Further reduction of the average node degree tends to disconnect the network. We conduct experiments under two different radio models: Unit Ball Graph (UBG) and Quasi-UBG [9]. In the UBG model, a link exits between every two nodes if the Euclidean distance between them is less than R. In the Quasi-UBG radio model, two parameters, α $(0 \leqslant \alpha < 1)$ and probability p $(0 < p < 1)$, are used to control connectivity patterns. Specifically, a link exists between two nodes if the distance between two nodes is less than $(1-\alpha)R$; a link exists with probability p if the distance is between $(1-\alpha)R$ and $(1+\alpha)R$; no link exists when the distance is greater than $(1+\alpha)R$.

We compare OnionMap with alternative solutions from different angles. For this purpose we have implemented four other schemes:

1) OnionMap-, a variant of OnionMap, in which layer alignment is disabled. Through comparison with OnionMap-, we want to study the benefit of layer alignment;
2) Bubble [20], representing the state of the art of 3-D geo-routing for general network topology (with holes);
3) BVR [4], representing the *topology-ignorant approach* to *ad hoc* routing for general network topology. It uses distances to a number of landmark nodes as virtual coordinates for the nodes, based on which routing is performed;
4) Greedy Random Greedy (GRG) [3], representing the *randomized approach* for 3-D geo-routing algorithm (in

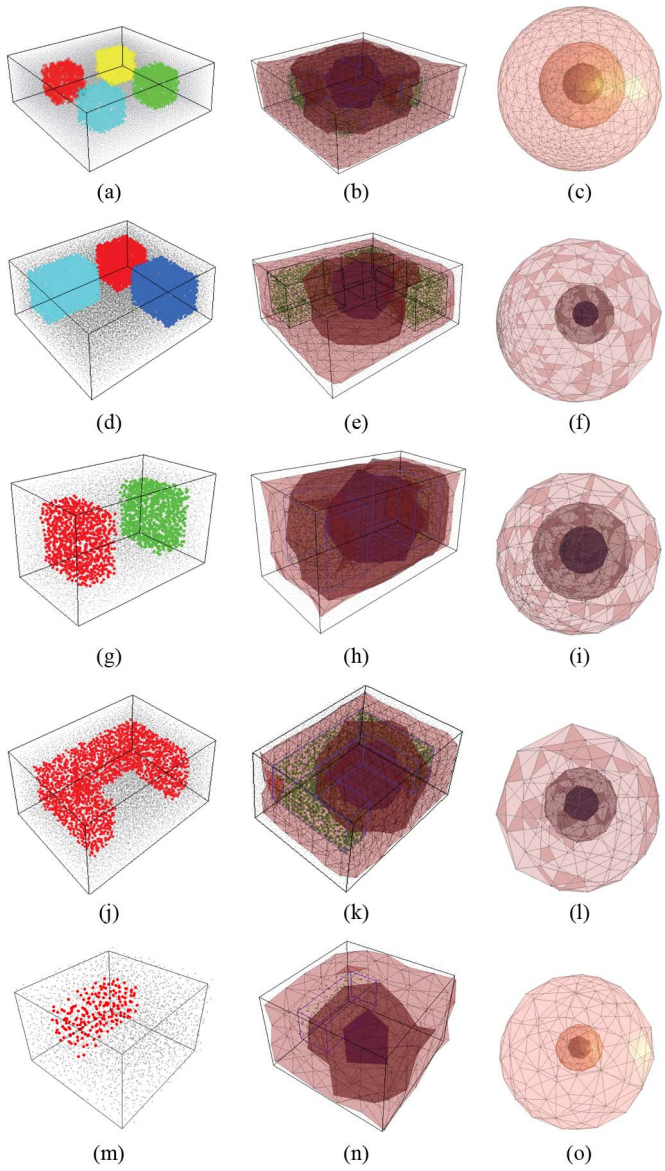


Fig. 7. 3-D network topologies (left column), example layers (second column), and concentric spherical mappings (third column). The outmost layers shown in the second row represent the outer boundaries for the networks. For clarity, the boundaries shown include only the landmarks on their layers, so are only an abstract version of the real boundaries. (a) Four-hole; (b) example layers; (c) example spheres; (d) three-hole; (e) example layers; (f) example spheres; (g) two-hole; (h) example layers; (i) example spheres; (j) U-hole; (k) example layers; (l) example spheres; (m) m-hole; (n) example layers; (o) Example spheres.

TABLE I
DIMENSION SIZE OF DIFFERENT TOPOLOGIES

Hole No.	Four-Holes (38×38×20)	Three-Holes (32×32×20)	Two-Holes (35×20×20)	U-Hole[1] (28×28×20)	m-Hole (15×15×15)
1	14×15×10	20×9×11	5×11×10	5×14×10	5×7×10
2	15×15×9	9×11×10	5×10×11	5×5×10	*
3	14×15×11	11×20×9	*	5×16×10	*
4	15×15×10	*	*	*	*

contrast to our deterministic approach). GRG uses random walk to recover from local minima.

5) S4 [14], representing the hierarchical routing scheme for general network topologies. In S4, several landmarks are selected to flood the network and every node records the hop count distances to these landmarks and then a worst-case stretch 3 routing scheme is established.

TABLE II
PATH STRETCH OF DIFFERENT ALGORITHMS

		Four-Holes	Three-Holes	Two-Holes	U-Hole	m-Hole
OnionMap	Avg	**1.26**	**1.25**	**1.23**	**1.24**	**1.25**
	95%	2.04	1.82	1.71	1.83	**1.76**
OnionMap–	Avg	2.42	2.1	1.67	1.74	1.68
	95%	3.58	3.13	2.74	3.0	2.71
Bubble	Avg	**2.14**	**1.94**	**1.6**	**1.37**	**1.48**
	95%	3.27	2.98	2.7	2.33	2.26
BVR	Avg	8.2	6.64	2.51	3.07	2.93
	95%	27.1	15.36	9.05	11.04	10.1
GRG	Avg	5.82	2.83	1.88	1.85	1.9
	95%	16.05	11.14	2.56	3.36	2.95
S4	Avg	1.15	1.12	1.1	1.08	1.06
	95%	1.38	1.32	1.3	1.3	1.31

For BVR, we randomly select 1% of the nodes as landmark nodes, following the default setting in its original paper. For GRG, we feed the physical coordinates of the nodes to GRG since it relies on location information. For S4, we randomly select $\sqrt{n}$ nodes as landmark nodes, where n is the size of the network.

A. Path Stretch

Table II shows the path stretch results of the six schemes. Path stretch is defined as the ratio of the routing path hop count to the shortest path hop count. Each time we randomly select 50 000 pairs of source and destination nodes, and report the aggregate statistics over 10 runs with different random seeds. As can be seen from Table II, OnionMap achieves mildly worse stretch compared with S4, with a significant improvement over the other four schemes. For example, for the Four-Holes case, OnionMap produces an average stretch of 1.26, achieving a reduction of 41% compared to the second best scheme Bubble; for the Three-Holes case, the reduction is 36%. The reason for Bubble's disadvantage has been explained in Section II. Its problem lies in its particular way of dealing with multiple holes, where the packet may make a detour on its way to the destination. When the network contains a single hole, e.g., in the U-Hole topology, the performance difference becomes smaller, and even smaller for a medium scale network, m-Hole.

OnionMap- performs worse than OnionMap because it does not align the layers, thus there is no guidance from a global coordinate system. In OnionMap-, when the source and destination nodes are in different layers, the packet will be routed along the inter-layer links until it reaches the destination's layer. This "vertical" routing, however, lacks global direction and may well land on a point very far away from the destination, thus generating a big detour.

Table II also shows that OnionMap dramatically outperforms BVR and GRG. BVR generates a large stretch because it requires scoped flooding to guarantee packet delivery in the final step when greedy routing fails. The flooding operations

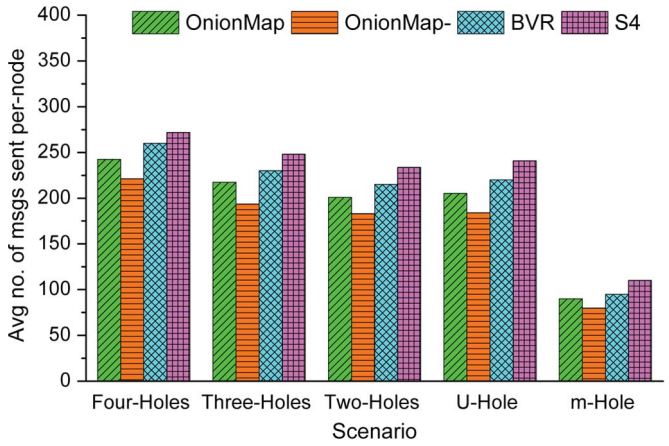


Fig. 8. Control message cost in routing setup.

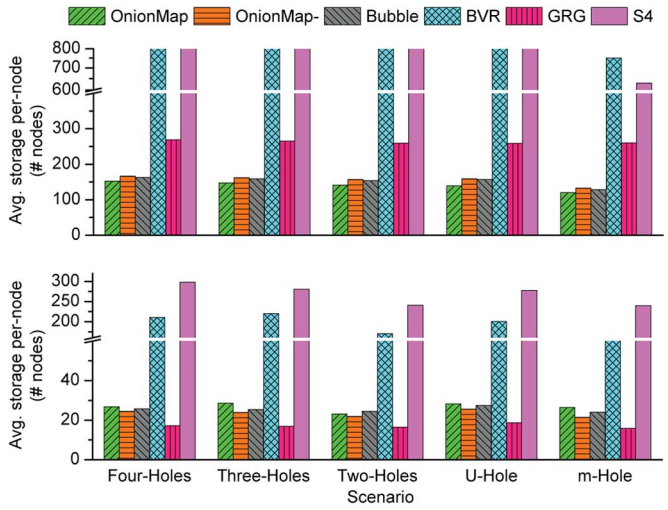


Fig. 9. Comparison of per-node storage cost.

can be very expensive when the failing point is far away from the destination. The stretch of GRG varies greatly for different topologies, due to its using blind random walk to escape local minima. When the topology become complex (with many holes), the stretch performance degrades very quickly.

B. Communication and Storage Cost

Routing Setup Cost: Except for GRG which does not need setup and simply uses random walk in routing, the other schemes all require a setup procedure to enable routing. Since the communication cost of detecting boundaries in Bubble is unclear, we only compare the control message cost of the four schemes in routing setup as shown in Fig. 8. As has been explained in Section VI, OnionMap limits the flooding of control messages mainly on layers while S4 and BVR both require all the landmarks flood the whole network. Thus OnionMap needs less control message cost for routing setup. Also, without the additional cost to align the layers, OnionMap- requires lower routing setup cost than OnionMap.

Storage Cost of Average Nodes: In Fig. 9, we compare the per-node storage cost of the six schemes in both bytes and number of nodes. The storage cost of OnionMap consists of the data structures that we have described in Section V. We assume that every ID and double data type takes 4 bytes in the six schemes. (The specific number of bytes may vary on a different hardware architecture, but does not affect the following analysis in general.) From Fig. 9 we can see that OnionMap takes about 150 bytes, which is the least among all schemes. The storage cost of OnionMap- is slightly larger than OnionMap mainly because the absence of coordinate alignment (thus state compression). Bubble requires more storage than OnionMap because each node in Bubble has to store the coordinates of neighbors of the same depth in a local shortest path tree, while OnionMap which only requires landmarks to maintain coordinates.

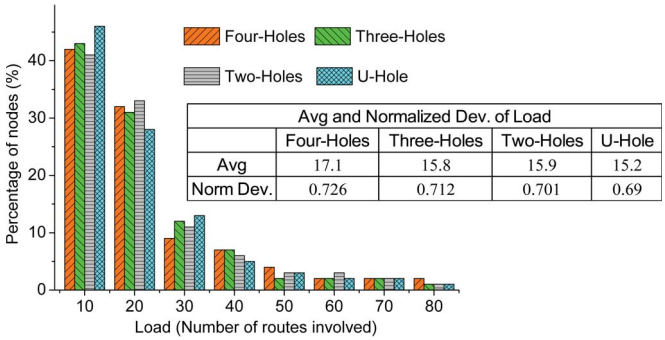
To put things in perspective, recall that the nodes' average degree is 15 to 20, amounting to a 60–80 bytes' storage cost for merely the neighbors table. OnionMap's cost of about 150 bytes suggests that it takes only two times as much as a neighbor table's size to support efficient routing in a large scale network. Moreover, the cost does not depend on network size, thus providing high scalability for routing.

For BVR, it incurs a large amount of storage because each node in BVR has to maintain a distance vector for each of its neighbors, the size of which is proportional to the number of the landmarks. S4 also requires high per-node state as each node has to maintain a large routing table for intra-cluster and inter-cluster routing. As shown in Fig. 9, the per-node state of OnionMap can be an order of magnitude lower than that of S4, while the stretch of OnionMap is less than 18% larger than that of S4. GRG is a position-based routing algorithm, in which each node has to maintain the IDs and geographical coordinates of all of its neighbors. In comparison using bytes count, the three connectivity-based routing algorithms, namely OnionMap, OnionMap- and Bubble, do not need the geographical information, thus saving storage substantially.

Storage Cost of Landmarks: In OnionMap, landmarks are distinguished nodes that maintain more state than other nodes. For a landmark that belongs to only one layer, it needs to maintain only the IDs of its neighbors, the coordinates of itself and its landmark neighbors on its home layer. These add up to about 140 bytes. For a multi-layer landmark belonging to multiple layers, thanks to the storage compression technique (Section IV-C), it does not have to store information for each associated layer. Since it only maintains one copy of information for multiple layers, the storage it requires would not increase very much. In our simulation, the storage for such landmarks is about 150 bytes. For a multi-layer landmark that lies near to the inner boundaries, its layers may not be perfectly aligned, thus its state cannot be compressed. In our simulations, such a node takes about 220 bytes. Fortunately, the percentage of such kind of landmarks is very small (about 0.5% of the total number of nodes). Therefore, the average per-node state of OnionMap can stay very low.

C. Load Balance

To obtain the load distribution, we count the number of routes a node participates in routing by randomly selecting 200 000 routes from each topology shown in Fig. 7. The results are given in Fig. 10. We omit the result of m-Hole as its scale is much different from other topologies. We can see that load in



Avg and Normalized Dev. of Load				
	Four-Holes	Three-Holes	Two-Holes	U-Hole
Avg	17.1	15.8	15.9	15.2
Norm Dev.	0.726	0.712	0.701	0.69

Fig. 10. Load distribution of OnionMap.

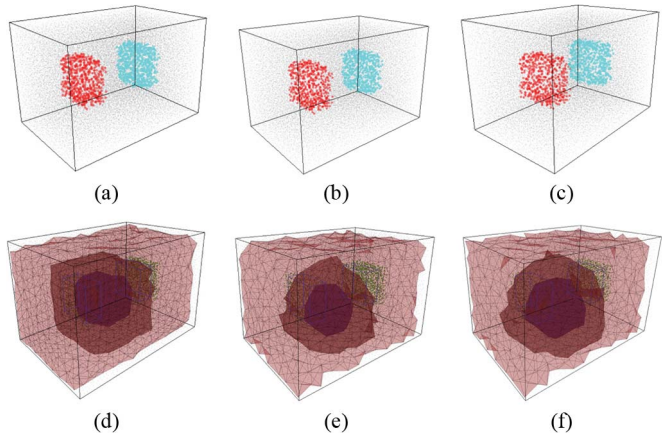


Fig. 11. Network with varying degrees of radio irregularity. (a) $\alpha = 0.0$; (b) $\alpha = 0.1$; (c) $\alpha = 0.3$; (d) example layers for $\alpha = 0.0$; (e) example layers for $\alpha = 0.1$; (f) example layers for $\alpha = 0.3$.

OnionMap is well balanced. Specifically, a node is involved $200\,000/(12\,000 \sim 17\,000) < 20$ routes on the average. As shown in Fig. 10, we can see that more than 40% of nodes in different topologies are involved less than 10 routes and around 70% less than 20 routes, a load quite close to the average one. Actually, as the table in Fig. 10 shows, there are only minor differences in the normalized standard deviation (all around 0.7) of the load on sensors on the four different topologies. It also indicates that the routing paths of OnionMap are distributed almost uniformly. Moreover, the load balancing property is not affected by the topologies as the trends of load under different topologies are nearly the same.

D. Robustness to Connectivity Irregularity

Irregular Radio Model: We use the Quasi-UBG model with different parameters to verify that OnionMap adapts to irregular radio models. Specifically, we compare the routing performance of OnionMap with α set to 0.0, 0.1, and 0.3. We have found that OnionMap attains stable performance across these variations. Fig. 11 shows that the decomposed layers of OnionMap become more irregular for a larger α, but OnionMap can still generate virtual coordinates that support efficient routing. Table III shows that the performance in stretch, storage, and load balance varies little when α becomes larger. These results indicate OnionMap's stability against radio irregularity.

Non-Uniform Node Distribution: OnionMap is also insensitive to node distribution, thanks to its nature of being location free. This can be seen from Fig. 12, where the network in the Two-Hole scenario is adjusted so that the left half and right half have average node densities of 15.6 and 23.4, respectively. Fig. 12(b) shows a cross section of the original network, in which the shade of the color of a node is proportional to its degree (number of neighbors). Notice both local and global non-uniformity of node density. The simulation result (Fig. 12(c)) shows that OnionMap can still obtain well constructed layers. Moreover, the average stretch generated is 1.28, with an average per-node state of 146.8 bytes. These results are comparable with those of the uniform case, suggesting that OnionMap is robust to non-uniformity node distribution.

TABLE III
PERFORMANCE OF ONIONMAP FOR DIFFERENT DEGREES OF RADIO IRREGULARITY

	Avg stretch	Per-node state(bytes)	Max node state(bytes)	Norm dev. of load
$\alpha = 0.0$	1.23	141.2	220	0.711
$\alpha = 0.1$	1.29	142.1	236	0.704
$\alpha = 0.3$	1.33	143.4	244	0.702

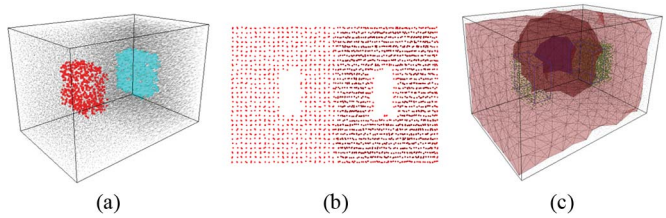


Fig. 12. A network topology with non-uniform node density. The left and right halves have average node densities 15.6 and 23.4, respectively. (a) 3-D topology; (b) a cross section; (c) example layers.

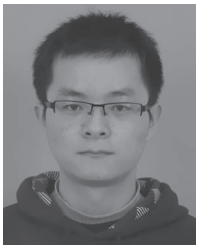
VIII. CONCLUSION

In this paper, we have presented an addressing and routing scheme, OnionMap, for 3-D sensor networks. The key idea is to decompose a given network into a set of connected and closed layers, which are then mapped to a set of concentric spheres. On each sphere, nodes are assigned coordinates that allow purely greedy routing. The spheres are further aligned to form a global coordinate system, on which geometric routing is realized. We have demonstrated OnionMap's effectiveness through extensive simulations.

REFERENCES

[1] S. Durocher, D. Kirkpatrick, and L. Narayanan, "On routing with guaranteed delivery in three-dimensional ad hoc wireless networks," in *Distributed Computing and Networking*. Berlin, Germany: Springer-Verlag, 2008, pp. 546–557.
[2] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang, "Glider: Gradient landmark-based distributed routing for sensor networks," in *Proc. IEEE INFOCOM*, 2005, pp. 339–350.
[3] R. Flury and R. Wattenhofer, "Randomized 3d geographic routing," in *Proc. IEEE INFOCOM*, 2008, pp. 1508–1516.
[4] R. Fonseca *et al.*, "Beacon vector routing: Scalable point-to-point routing in wireless sensornets," in *Proc. USENIX NSDI*, 2005, pp. 329–342.
[5] S. Funke and N. Milosavljevic, "Network sketching or: 'How much geometry hides in connectivity?—Part II,'" in *Proc. SODA*, 2007, pp. 958–967.
[6] H. Jiang, S. Zhang, G. Tan, and C. Wang, "Cabet: Connectivity-based boundary extraction of large-scale 3d sensor networks," in *Proc. IEEE INFOCOM*, 2011, pp. 784–792.

[7] M. Jin, G. Rong, H. Wu, L. Shuai, and X. Guo, "Optimal surface deployment problem in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2012, pp. 2345–2353.
[8] A. M. Kermarrec and G. Tan, "Greedy geographic routing in large-scale sensor networks: A minimum network decomposition approach," in *Proc. ACM MobiHoc*, 2010, pp. 161–170.
[9] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Ad-hoc networks beyond unit disk graphs," in *Proc. 2003 Joint Workshop Found. Mobile Comput.*, 2003, pp. 69–78.
[10] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer With Applications to Orbits, Aerospace, and Virtual Reality*. Princeton, NJ, USA: Princeton Univ. Press, 1999.
[11] S. Lam and C. Qian, "Geographic routing in d-dimensional spaces with guaranteed delivery and low stretch," in *Proc. ACM SIGMETRICS*, 2011, pp. 257–268.
[12] F. Li, J. Luo, C. Zhang, S. Xin, and Y. He, "Unfold: Uniform fast on-line boundary detection for dynamic 3d wireless sensor networks," in *Proc. ACM MobiHoc*, 2011, pp. 1–14.
[13] C. Liu and J. Wu, "Efficient geometric routing in three dimensional ad hoc networks," in *Proc. IEEE INFOCOM*, 2009, pp. 2751–2755.
[14] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith, "S4: Small state and small stretch routing protocol for large wireless sensor networks," in *Proc. NSDI*, 2007, p. 8.
[15] A. Nguyen, N. Milosavljevic, Q. Fang, J. Gao, and L. J. Guibas, "Landmark selection and greedy landmark-descent routing for sensor networks," in *Proc. IEEE INFOCOM*, 2007, pp. 661–669.
[16] C. Qian and S. S. Lam, "Greedy distance vector routing," in *Proc. IEEE ICDCS*, 2011, pp. 857–868.
[17] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. Gu, "Greedy routing with guaranteed delivery using ricci flows," in *Proc. ACM/IEEE IPSN*, 2009, pp. 121–132.
[18] G. Tan, M. Bertier, and A. Kermarrec, "Convex partition of sensor networks and its use in virtual coordinate geographic routing," in *Proc. IEEE INFOCOM*, 2009, pp. 1746–1754.
[19] Y. Wang, J. Gao, and J. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proc. ACM MobiCom*, 2006, pp. 122–133.
[20] S. Xia, M. Jin, H. Wu, and H. Zhou, "Bubble routing: A scalable algorithm with guaranteed delivery in 3d sensor networks," in *Proc. IEEE SECON*, 2012, pp. 245–253.
[21] S. Xia, X. Yin, H. Wu, M. Jin, and X. Gu, "Deterministic greedy routing with guaranteed delivery in 3d wireless sensor networks," in *Proc. ACM MobiHoc*, 2011, pp. 1–10.
[22] X. Yin, M. Jin, F. Luo, and X. D. Gu, "Discrete curvature flows for surfaces and 3-manifolds," in *Emerging Trends in Visual Computing*. Berlin, Germany: Springer-Verlag, 2009, pp. 38–74.
[23] X. Yu, X. Yin, W. Han, J. Gao, and X. Gu, "Scalable routing in 3d high genus sensor networks using graph embedding," in *Proc. IEEE INFOCOM*, 2012, pp. 2681–2685.
[24] W. Zeng, R. Sarkar, F. Luo, X. Gu, and J. Gao, "Resilient routing for sensor networks using hyperbolic embedding of universal covering space," in *Proc. IEEE INFOCOM*, 2010, pp. 1694–1702.
[25] H. Zhou, H. Wu, and M. Jin, "A robust boundary detection algorithm based on connectivity only for 3d wireless sensor networks," in *Proc. IEEE INFOCOM*, 2012, pp. 1602–1610.
[26] J. Zhou, Y. Chen, B. Leong, and B. Feng, "Practical virtual coordinates for large wireless sensor networks," in *Proc. IEEE ICNP*, 2010, pp. 41–51.
[27] J. Zhou, Y. Chen, B. Leong, and P. Sundaramoorthy, "Practical 3d geographic routing for wireless sensor networks," in *Proc. ACM SenSys*, 2010, pp. 337–350.



Kechao Cai received the B.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 2010. He is currently working toward the M.S. degree in the Department of Electronics and Information Engineering and at Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology. He is also with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His current research interests include wireless sensor networks.

Zhimeng Yin received the B.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 2011. He is currently working toward the M.S. degree with the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology. He is also with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His current research interests include wireless sensor networks.

Hongbo Jiang received the B.S. and M.S. degrees from Huazhong University of Science and Technology, Wuhan, China, and the Ph.D. degree from Case Western Reserve University, Cleveland, OH, USA. He is currenty a Professor with the Department of Electronics and Information Engineering and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology. His research interests include computer networking, particularly algorithms and architectures for high-performance networks and wireless networks.

Guang Tan received the B.S. degree from Chongqing University of Posts and Telecommunications, Nan'an, China, in 1999; the M.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 2002; and the Ph.D. degree in computer science from the University of Warwick, Coventry, U.K., in 2007. From 2007 to 2010, he was a Postdoctoral Researcher at INRIA-Rennes, France. He is currently an Associate Researcher with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, where he works in the area of distributed systems and networks.

Peng Guo received the B.S., M.S., and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2000, 2003, and 2008, respectively. He is currently an Associate Professor with the Department of Electronics and Information Engineering and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology. His research interests include wireless sensor networks and research and development of embeded systems.

Chonggang Wang received the Ph.D. degree in computer science from Beijing University of Posts and Telecommunications, Beijing, China. He has conducted research with NEC Laboratories America, AT&T Labs Research, University of Arkansas, and Hong Kong University of Science and Technology. He is currently with InterDigital Communications, Melville, NY, USA. His research interests include future Internet, machine-to-machine communications, and cognitive and wireless networks.

Bo Li received the B.Eng. degree in computer science from Tsinghua University, Beijing, China, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst, MA, USA. From 1993 to 1996, he was with IBM Networking System Division, Research Triangle Park, NC, USA. From 1999 to 2005, he was an Adjunct Researcher at Microsoft Research Asia-MSRA, where he spent his sabbatical leave (2003–2004). He is currently a Professor with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Kowloon, Hong Kong. He is also currently a Cheung Kong Chair Professor with Shanghai Jiao Tong University, Shanghai, China. His recent research interests include large-scale content distribution in the Internet, peer-to-peer media streaming, the Internet topology, cloud computing, and green computing and communications.