SINUS: A Scalable and Distributed Routing Algorithm with Guaranteed Delivery for WSNs on High Genus 3D Surfaces

Tianlong Yu[1] Hongbo Jiang[1] Guang Tan[2] Chonggang Wang[3] Chen Tian[1] Yu Wu[1]
[1]Department of Electronics and information Engineering, Huazhong University of Science and Technology, China
[2]SIAT, Chinese Academy of Sciences, China, [3]InterDigital Communications, PA, 19406
[1]{yutianlong21,hongbojiang2004,alexandretian,yuwu199111}@gmail.com,[2]guang.tan@siat.ac.cn, [3]cgwang@ieee.org

***Abstract*—In this paper, we put forward a novel scalable and distributed routing algorithm, called *SINUS*, for sensor networks deployed on the surface of complex-connected 3D settings such as tunnels, whose topologies are often theoretically modeled as high genus 3D surfaces. SINUS is carried out by first slicing the genus-n surface along a maximum cut set based on Morse theory and Reeb graph, in order to form a genus-0 surface with $2n$ boundaries. Then, it groups these $2n$ boundaries into two groups each of which is next connected together. By doing so, a genus-0 surface with exactly two boundaries emerges, which can be flattened into a strip, using the Ricci flow algorithm and next mapped to a planar annulus by Möbius Transform. By assigning nodes virtual coordinates on the planar annulus, SINUS finally realizes a variation of greedy routing to enable individual nodes to make local routing decisions. Our simulation results show that SINUS can achieve low-stretch routing with guaranteed delivery, as well as balanced traffic load.**

I. Introduction

Recent years have witnessed a rapid growth of Wireless Sensor Networks (WSNs) with generic tunnel-shape in e-merging applications where nodes are typically deployed on the surface of complex-connected 3D settings. Examples of such applications include monitoring of coal mine tunnels for disaster prevention and rescue,fire detection in the corridors of buildings,as well as monitoring of underground tunnels used in water, sewer or gas systems. These networks are called *3D high genus WSNs* [22] or WSNs on high genus 3D surfaces, as shown in Fig. 1. The sensor network on a high genus 3D surface is often of a complex-connected 3D setting and has non-trivial topology, possibly with high genus (i.e., multiple handles) [22]. While there exist a series of previous studies that achieve scalable routing for 2D networks and simple 3D volume networks (say with only one inner boundary) [20], few of them can work for WSNs on high genus 3D surfaces [22].

This work was supported in part by the National Natural Science Foundation of China under Grant 61073147, Grant 61173120, Grant 61103243, Grant 61202460, Grant 61271226, and Grant 61272410; by the National Natural Science Foundation of China and Microsoft Research Asia under Grant 60933012; by the Fundamental Research Funds for the Central U-niversities under Grant 2012QN078; by the CHUTIAN Scholar Project of Hubei Province; by the Youth Chenguang Project of Wuhan City under Grant 201050231080; by the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry); by the National Natural Science Foundation of Hubei Province under Grant 2011CDB044; by the Fok Ying Tung Education Foundation under Grant 132036; by the Hong Kong Scholars Program; and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry). Guang Tan's work was supported by the Natural Science Foundation of China under Grant 61103243, Youth Innovation Promotion Association, Chinese Academy of Sciences, the Ministry of Science and Technology 863 Key Project No. 2011AA010500, and Shenzhen Overseas High-level Talents Innovation and Entrepreneurship Funds KQC201109050097A. The corresponding author is Hongbo Jiang.

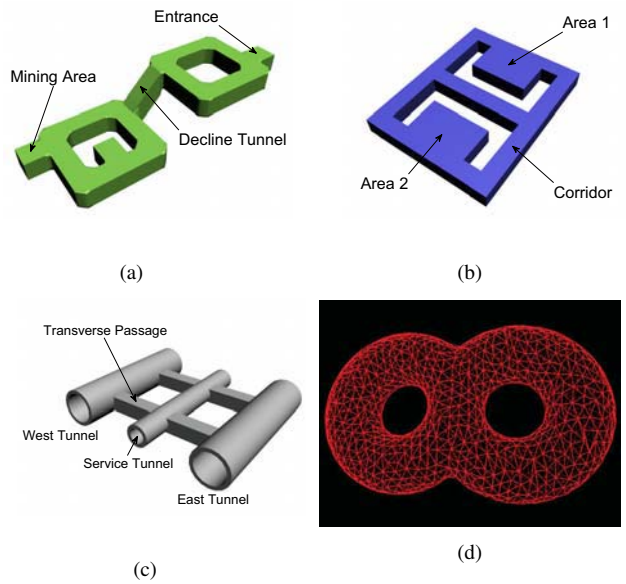


Fig. 1. The networks of (a) coal mine tunnels; (b) corridors of buildings; (c) underground tunnels. These three networks are homotopically equivalent to (d) a 3D genus-2 network.

Existing Work: Greedy routing is appealing for its simplic-ity and scalability to large networks with stringent resource constraints. Unfortunately, its performance suffers from local minima, where greedy forwarding cannot proceed. To deal with this, face routing [14] exploits the fact that a concave void in a 2D planar network is a face with a simple line boundary. When a local minimum is encountered, the packet employs face routing to route along the boundary in either clockwise or counter-clockwise direction, until greedy forwarding is achievable. However, the hole in a 3D topology is not a 2D face and its boundary becomes a surface, yielding an arbitrarily large number of possible paths to be explored [20].

Recent 3D greedy embedding methods [3], [17], [20]map the original sensor network topology to a planar surface or to a virtual sphere to enable greedy routing. However, none of these algorithms can be applied to the sensor networks on high genus 3D surfaces [22]. The reason is that the embedded surface $\mathbf{M}$ should be continuously deformed to the embedding surface $\mathbf{D}$, that is, $\mathbf{M}$ and $\mathbf{D}$ should be in the same homotopy class.

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

However, a high genus 3D surface, as shown in Fig. 1(d), is obviously not homotopically equivalent to a 2D planar surface or a 3D sphere [5], rendering the direct greedy embedding infeasible for WSNs on high genus 3D surfaces.

Yu *et. al.* [22] conducted a pioneer work on scalable routing on high genus 3D surfaces. They proposed to embed the network on a surface with high genus first. Then a canonical hyperbolic metric of the embedded surface is calculated and the network is decomposed into canonical components. The routing follows a two-level paradigm: first to find a sequence of components (corresponding to a shortest path on the adjacency graph of the components), and then to realize the routing with greedy steps within each component. However, a major concern of this approach is its centralized operations during finding the genus and decomposing the network, which make the algorithm impractical for distributed sensor networks. In addition, compact routing among components requires every node to maintain a routing table to all other components [22], possibly resulting in high storage overhead on individual nodes when the network grows large and becomes more complex.

Our Contribution: In this paper, we propose SINUS, a Scalable and dIstributed routing algorithm with guaranteed delivery for WSNs on high geNUs 3D Surfaces. The key contribution of our proposed scheme is to slice the high genus surface to form one single genus-0 surface via a distributed algorithm. Then this genus-0 surface is mapped to a planar annulus for greedy routing.

To summarize, SINUS presents a neat, distributed, scalable, and general solution with delivery-guaranteed. It is particularly interesting for a 3D network with complex topology. First, SINUS is quite neat, mapping the sliced high genus surface to a *single* connected planar surface — a unit disk with a circular hole. As such, in contrast to [22], no partition is involved and compact routing among components is avoided in SINUS. Second, SINUS is fully distributed. In the embedding stage, approximate geodesic patterns and rotation scheme are utilized to construct Morse function and then slice the high genus surface in a distributed and discrete manner. In contrast to [22], SINUS requires no centralized operations. Third, SINUS is a scalable algorithm, with message complexity during the embedding stage being $O(n \cdot m)$, where n is the genus of $\mathbf{M}$ and m is the number of nodes. So the embedding stage of SINUS is scalable in terms of message cost. Plus, the greedy routing on embedding surface is scalable as well. Besides, the storage complexity on individual nodes is trivial as each node only maintains the virtual coordinate and a 4-tuple to enable local routing decisions. Finally, SINUS is a generic algorithm. It does not rely on the knowledge of any location information, angular information. Nor do we require that the communication graph follows an ideal radio model such as the unit disk graph model or the quasi-unit disk graph model.

The rest of the paper is organized as follows. We describe the theoretical foundation of our proposed algorithm in Section II. In Section III, we introduce our scalable and distributed routing algorithm with guaranteed delivery for WSNs on high genus 3D Surfaces. Simulation results are illustrated in Section IV. Finally, Section V concludes the paper.

II. Theoretical Foundation

A. Genus-n 3D Surfaces

In algebraic topology, a *cut* C is referred to as a non-intersecting closed simple curve on a connected and orientable surface $\mathbf{M}$. A cut *locally* disconnects the topology of $\mathbf{M}$. We call $\mathcal{C}_{\max} = \{C_1, C_2, \ldots, C_n\}$ a *maximum cut set* of $\mathbf{M}$, if and only if: (1) the cut set $\mathcal{C}_{\max}$ do not render $\mathbf{M}$ disconnected; (2) any cut set $\mathcal{C}'$ with a cardinality of $n+1$ or more will render $\mathbf{M}$ disconnected. Accordingly, the *genus* of $\mathbf{M}$ is given:

Definition 1. *The* genus *of* $\mathbf{M}$ *is defined as the cardinality of maximum cut set* $\mathcal{C}_{\max}$*, representing the maximum number of cuts on* $\mathbf{M}$ *without disconnecting* $\mathbf{M}$.

Fig. 2 shows several surfaces with different genus: a sphere is a genus-0 surface and any cut will cause the sphere disconnected; a genus-1 torus can have at most one cut on it without getting disconnected. Generally, one cut C is able to slice a genus-n surface to genus-$(n-1)$.

Fig. 2. From left to right: genus-0, genus-1, genus-2, and genus-3.

Since our goal is to obtain a *single* genus-0 topology, intuitively we can extract the maximum cut set $\mathcal{C}_{max}$ and slice the surface along $\mathcal{C}_{max}$. However, the task of finding a maximum cut set is non-trivial: identifying the genus of an arbitrary surface is NP-hard,and often demands centralized operations.Therefore, this paper only targets at the orientable closed surface (compact and without boundaries) which is the common case of a tunnel-shape scenario [22]. In this case, our proposed method utilizing Morse theory guarantees to find such a maximum cut set, and, more importantly, can be implemented in a distributed manner.

B. Morse Theory

1) Morse Function: A Morse function is defined as a mapping $f : \mathbf{M} \to \mathbb{R}$, from a manifold $\mathbf{M}$ to a real number set $\mathbb{R}$. Fig. 3 illustrates a Morse function on a torus with a mapping from a successive of contour lines to an integer set $\{0, 1, \ldots, I\}$ of the height value.

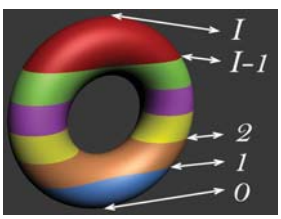


Fig. 3. Morse function.

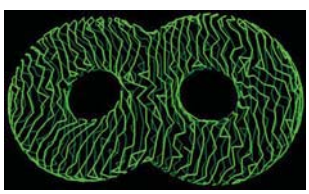


Fig. 4. The successive geodesic patterns of a sensor network.

Unfortunately, it is not straightforward to define the Morse function from mere connectivity information in a discrete network, where no height value can be derived. To address this problem, we propose to take the advantage of successive geodesic patterns [16] in this paper to define the Morse function on high genus 3D surfaces.

2) Successive Geodesic Patterns: Successive geodesic patterns refer to general sequences g_i of successive geodesic lines on a surface and the distances between them [16]. Often the distances are difficult to obtain, so the first order approximation is given as follows on a geodesic curve $g(s)$ where the parameter s is its curve length:

$$g_{i+1}(s) = g_i(s) + \varepsilon \mathbf{v}(s) + \varepsilon^2(\ldots) \quad (1)$$

Here the derivative vector field $\mathbf{v}$ is called a *Jacobi field*, presumably orthogonal to $g(s)$. Accordingly, the Morse function is defined by $f : \mathcal{G} \rightarrow \mathbb{Z}$ from general sequences $\mathcal{G} : g_0, g_1, g_2, \ldots, g_i, \ldots, g_n$ of successive geodesic lines to an integer set $\mathbb{Z}$. $f^{-1}(i)$ is called a i-level set of the Morse function. It is worth noting that calculating a Jacobi field evolves direction calculation, which is infeasible for a discrete sensor network based on connectivity information only. Therefore, an approximation of successive geodesic lines is utilized, which is initiated from an arbitrary node r (detailed in Section III-A). Fig. 4 shows an example of geodesic patterns for a discrete sensor network.

3) Reeb Graph: Based on the Morse function f, the surface $\mathbf{M}$ can be represented by a Reeb graph $\mathbf{R}$, which describes the evolution of the components of the level sets $f^{-1}(\cdot)$ [5]. The number of the connected components of $f^{-1}(\cdot)$ does not change except when a *critical point* emerges, that is, a point where the gradient of f is 0. Theoretically, there are three types of critical points, namely, *minima*, *saddles*, and *maxima*.respectively. The Reeb graph is obtained by contracting the connected components of $f^{-1}(\cdot)$ to critical points. The rest of the Reeb graph consists of arcs connecting these points. Each arc represents a *region* (a connected node set) in $\mathbf{M}$, as illustrated in Fig. 5(a) and Fig. 5(c). The *degree* of a point in Reeb graph is the number of arcs associated with this point. It is noted that, Reeb graph $\mathbf{R}$ in this paper starts from exactly one node r (a *minima* with index 0), since the Morse function is defined by a successive of geodesics initiated from r.

C. Extracting a Maximum Cut Set

Based on the Reeb graph $\mathbf{R}$, we turn to the mechanism of extracting a maximum cut set $\mathcal{C}_{\max}$ from the surface $\mathbf{M}$. First, we have the following result.

Theorem 1. *The Reeb graph $\mathbf{R}$ of a Morse function over a connected orientable 2-manifold of genus-n without boundaries has exactly n loops [2].*

Motivated by Theorem 1, the basic idea behind our algorithms of extracting a maximum cut set is simple: *It suffices to find a cut for each loop in the Reeb graph* of the high genus 3D surface $\mathbf{M}$.

To that end, we first identify all the loops in $\mathbf{R}$. In a Reeb graph $\mathbf{R}$, a loop is associated with two degree-3 nodes - a *loop-start node* n_{st} and a *loop-end* node n_{ed}. n_{st} separates one arc (region) into two, and thus the number of components of the corresponding level set is increased by 1. Accordingly, n_{ed} merges two arcs (regions) into one, thus the number of components of the corresponding level set is decreased by 1. Therefore, we present

Definition 2. *A region (arc in Reeb graph) is defined as a **loop-end region** I_m, if two regions I_α and I_β are merged into I_m by a loop-end node n_{ed}.*

For example, in Fig. 5(a), *loop* 1 is associated with loop-start node B and loop-end node C, while *loop* 2 is associated with loop-start node D and loop-end node E. Loop-start node B separates arc AB into arc a and arc b, while loop-end node C merges arc a and arc b into arc CD. Correspondingly, in Fig. 5(c), the blue region and the yellow region are merged to a loop-end region region in cyan. Consequently, we identify all the loops in a Reeb graph $\mathbf{R}$ by the following observation:

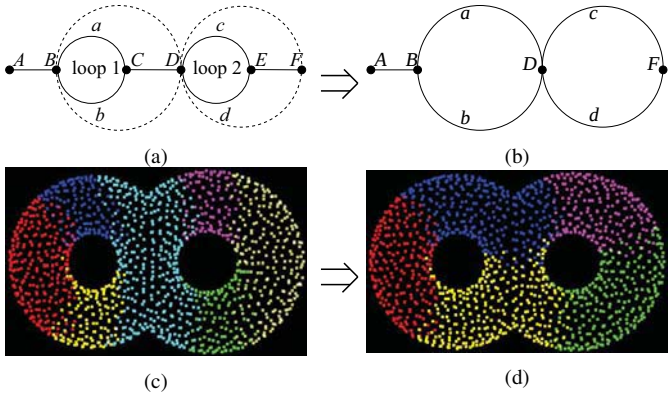


Fig. 5. (a) The Reeb graph of the network in Fig. 4; (b) The Reeb graph after the bisection operation; (c) The regions of the Reeb graph in (a), which are differentiated by colors; (d) The regions of the Reeb graph in (b), which are differentiated by colors.

Theorem 2. *Each loop L_i corresponds to one loop-end region (arc) I_m.*

Proof: According to Definition 2, equivalently we need to prove that each loop L_i corresponds to one loop-end node n_{ed}. First we prove that, there exists a loop-end node n_{ed} in each loop L_i. In Morse theory, a loop corresponds to at least two saddle points; each of them is a degree-3 node that merges two arcs into one, that is, a loop-end node n_{ed}.

We next prove that there exists a loop for every loop-end node n_{ed}. In $\mathbf{R}$, arcs I_α and I_β are connected with loop-end node n_{ed}. Since the Reeb graph $\mathbf{R}$ is a connected graph, and starts from exactly one node r, there exists a path connecting I_α and r without passing through node n_{ed}. Accordingly, there exists a path in $\mathbf{R}$, which connects I_β and r without passing through node n_{ed}. So there are two different paths that connect n_{ed} and r (distinguished by I_α and I_β). Therefore, there exists a loop associated with the loop-end node n_{ed}. ■

This observation motivates us, in order to identify a cut C_i for each loop L_i, we can find a bisection in the corresponding *loop-end region* I_m, that disconnects the loop L_i. It is noted that I_α and I_β are connected by I_m, which constitutes part of loop L_i. Therefore, cut C_i can be obtained by bisecting I_m at n_{ed}. Consequently, we define a bisection operation at n_{ed} as follows: for a point p in *loop-end region* I_m, if the minimum distance from p to points in I_α is less than the minimum distance from p to points in I_β, it is assigned to arc I_α, otherwise it is assigned to I_β. This bisection operation separates I_m into two parts I_{m1} and I_{m2} and "glues" them to I_α and I_β respectively. After the bisection operation, I_α and I_β become newly *merged regions* I'_α and I'_β, then we have:

Theorem 3. *Suppose loop L_i is identified by loop-end region I_m. If a cut C_i in the loop-end region I_m separates the merged region I'_α from merged region I'_β, then C_i should be a cut for loop L_i.*

Proof: Apparently, cut C_i disconnects one of the two distinguished paths between I_α and I_β, which is part of the loop L_i. That is, cut C_i is a cut for loop L_i. Based on Theorem 2, this theorem holds. ∎

Fig. 5 demonstrates the bisection operation. In Fig. 5(a) and Fig. 5(c), loop-end node C merges arc a (region in blue) and arc b (region in yellow) into *loop-end region* CD (cyan). So a and b are connected by CD, which is part of the *loop* 1. The result of the bisection operation is presented in Fig. 5(b) and Fig. 5(d). After the bisection, the two cuts are obtained by separating region a (blue) from region b (yellow), as well as region c (pink) from region d (green). The two cuts are presented in Fig. 6(a). Finally, all the cuts $C_i(i = 1, 2, ..., n)$ form a maximum cut set $\mathcal{C}_{max}$ for $\mathbf{M}$.

D. Surface Ricci Flow and Möbius Transform

Given a genus-0 surface $\mathbf{S}$ with exactly two boundaries, surface Ricci flow and Möbius transform can be applied, aiming at mapping $\mathbf{S}$ to a planar annulus $\mathbf{D}$ where the greedy routing is allowed.

1) Riemannian Metric and Curvature: Let $\mathbf{S}$ be a surface in $\mathbb{R}^3$, and it has a Riemannian Metric $g = (g_{ij})$, induced from the Euclidean metric of $\mathbb{R}^3$. Gaussian curvature K and geodesic curvature k_g are also determined by Riemannian Metric g, while the total curvature is topological invariant:

Theorem 4. *(Gauss-Bonnet) Suppose $\mathbf{S}$ is a compact 2-manifold with boundary $\partial\mathbf{S}$. Then the total curvature is given by $\int_{\mathbf{S}} KdA + \int_{\partial\mathbf{S}} k_g ds = 2\pi\chi(\mathbf{S})$, where K is the Gaussian curvature on interior points, k_g is the geodesic curvature on boundary points $\partial\mathbf{S}$, $\chi(\mathbf{S})$ is the Euler characteristic number of $\mathbf{S}$.*

Suppose $u : \mathbf{S} \to \mathbb{R}^3$ is a scalar function defined on $\mathbf{S}$. Then $\bar{g} = e^{2u}g$ is another Riemannian metric on $\mathbf{S}$, and $\bar{g}$ is conformal to g.

2) Surface Ricci Flow: As an intrinsic geometric flow, the Ricci flow [6] deforms the metric of a Riemannian manifold in the manner analogous to heat diffusion, smoothing out irregularities of the metric. It can be represented by curvature evolution: $\frac{K(t)}{dt} = -\Delta_{g(t)}K(t)$, where $K(t)$ is the Gaussian curvature of metric $g(t)$, and $-\Delta_{g(t)}$ is the Laplace-Beltrami operator induced by $g(t)$. Let $g(t) = e^{2u}g(0)$, we have $\frac{du(t)}{dt} = -2K(t)$.

3) Möbius Transformation: To achieve a conformal mapping and embed the surface from the strip to a planar annulus, we make use of the Möbius Transformation with a rational function of the form $f(z) = \frac{az+b}{cz+d}$. Specifically, for a point on the plane $p = (x, y)$: $p \leftarrow e^{\frac{2\pi}{h}(x+iy)}$ where h is the width of the strip.

III. SINUS Algorithm

This section presents SINUS algorithm which deals with a fundamental problem: embedding a network on a high genus surface into a planar surface as a whole. The basic idea behind SINUS is simple: slicing the genus-n surface $\mathbf{M}$ to a simpler surface for embedding. The preprocess of SINUS is to compute a triangulation from the original network via a simple distributed algorithm in [17], [23]. The triangulated structure, or *mesh* for short, forms a shape representation of the high genus surface, as shown in Fig. 1(d). Without leading to confusion, we still call the triangulated mesh as the high genus surface, denoted by $\mathbf{M}$ henceforth. Overall SINUS mainly consists of four steps:

1) We identify a maximum cut set $\mathcal{C}_{max}$ from $\mathbf{M}$, which is used to slice $\mathbf{M}$ to form a genus-0 surface $\mathbf{G}$ with $2n$ boundaries. To this end, a flooding across $\mathbf{M}$ is initiated by an arbitrary node r. This implicitly approximates the geodesic patterns on $\mathbf{M}$ and constructs a Morse function by assigning each node a level index. Based on the Morse function, a Reeb graph $\mathbf{R}$ starts at node r is also constructed in the process, by assigning each node a region ID. Thereafter, all the *loop-end* regions $I^i_m(i = 1, 2, ..., n)$ are identified, and next a bisection operation is performed to extract a cut C_i within each I^i_m. All the cuts $C_i(i = 1, 2, ..., n)$ form a maximum cut set $\mathcal{C}_{max}$, which slices the genus-n surface $\mathbf{M}$ to a genus-0 surface $\mathbf{G}$ with $2n$ boundaries, as illustrated in Fig. 6(a).
2) We further slice $\mathbf{G}$ to a 'pipe' $\mathbf{S}$ — a genus-0 surface with exactly two boundaries. To do so we propose to cluster the aforementioned $2n$ boundaries of $\mathbf{G}$ into two groups, each with n boundaries. After a Depth-First Search (DFS) procedure is performed among the regions of $\mathbf{G}$, n boundaries in each group are connected and sliced open with $(n-1)$ shortest paths. As such, two boundaries $\partial\mathbf{S}_1$ and $\partial\mathbf{S}_2$ are extracted, as illustrated in Fig. 6(b).
3) Using the Ricci flow algorithm, $\mathbf{S}$ is flattened into a strip $\mathbf{P}$, with the edge of the strip being a path connecting the two boundaries $\partial\mathbf{S}_1$ and $\partial\mathbf{S}_2$. Thereafter, by Möbius Transform, strip $\mathbf{P}$ is mapped to a planar annulus $\mathbf{D}$, with $\partial\mathbf{S}_1$ and $\partial\mathbf{S}_2$ mapped to the inner boundary and outer boundary of $\mathbf{D}$ accordingly, denoted by $\partial\mathbf{D}_{inner}$

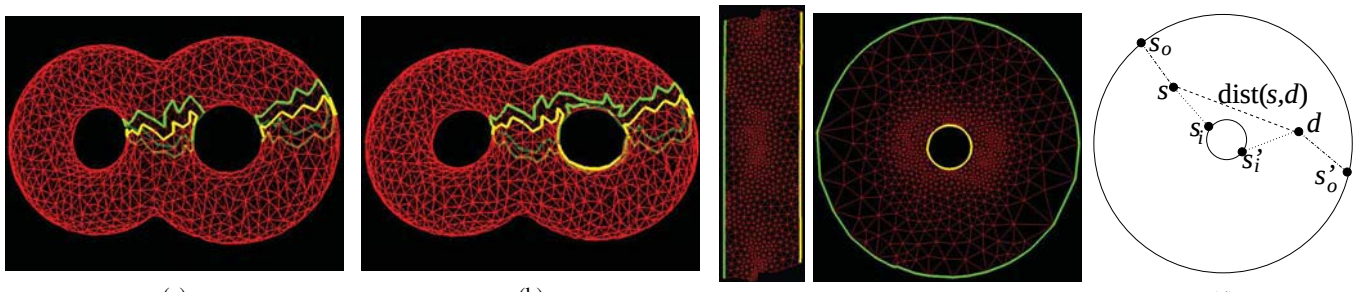


Fig. 6. (a) The genus-0 surface $\mathbf{G}$ with $2n$ boundaries; (b) Connecting two boundary groups — A 'Pipe' $\mathbf{S}$ with exactly two boundaries; (c) The planar annulus $\mathbf{D}$; (d) SINUS realizes a variation of greedy routing.

and $\partial\mathbf{D}_{outer}$. Every node in the network is then given a virtual coordinate in $\mathbf{D}$, as illustrated in Fig. 6(c).

4) In the virtual coordinate system, it is observed that, a source node s can route to a destination node d via three kinds of paths: (1) across only the interior nodes; (2) across the inner boundary $\partial\mathbf{D}_{inner}$; (3) across the outer boundary $\partial\mathbf{D}_{outer}$. Therefore, having obtained the virtual coordinates, the three paths are estimated and a shortest path is chosen for routing by the source node s, as illustrated in Fig. 6(d).

Fig. 6 shows an example pipeline of SINUS.

A. Extracting a Maximum Cut Set in Discrete Settings

For a genus-n surface $\mathbf{M}$, the first step is to find a maximum cut set $\mathcal{C}_{max}$, aiming at slicing $\mathbf{M}$ to a genus-0 surface $\mathbf{G}$ based on Morse theory and Reeb graph. To this end, we exploit the discrete geometry characteristics of $\mathbf{M}$.

In order to set up the Morse function in $\mathbf{M}$, an arbitrary root node r initiates a flooding across the whole network. By doing so, every node learns its hop distance i from r, and records its level index as i. All the nodes with a level index of i belong level-i of $\mathbf{M}$. Also, after a flooded message from r reaches a node p, p records the parent from which it receives the message.

A Morse function is then defined by approximating the geodesic patterns on $\mathbf{M}$, as mentioned in Section II-B. Recall that the final mapped planar annulus $\mathbf{D}$ is a *developable surface* [16], on which the Gaussian curvature K of $\mathbf{D}$ is zero everywhere. As such, the Jacobi field $\varepsilon\mathbf{v}(s)$ is a constant, indicating that the distance between two successive geodesics $g_i(s)$ and $g_{i+1}(s)$ is a constant. In this case, the hop-count distance between nodes of two adjacent levels (level-i and level-$(i+1)$) is exactly one hop. Based on this observation, we propose to use nodes in level-i in the discrete settings to approximate $g_i(s)$ in the continuous settings. Consequently, the Morse function in discrete settings is defined as a mapping $f(p) \to i$, where p is in level-i. Therefore, all the nodes in level-i are denoted by $f^{-1}(i)$. We denote the max hop count of nodes from r as I, that is, $i \leq I$.

To construct a Reeb graph from the defined Morse function, a distributed algorithm similar to that in [13] is carried out, which involves two major steps:

First, the algorithm identifies the nodes in each connected component in $f^{-1}(i)$ (level-i) with a component ID. To this end, a randomly selected node q on $f^{-1}(i)$ claims itself as a landmark and floods a message within $f^{-1}(i)$, which contains the node ID of q and its level index i. The flooding is constrained within $f^{-1}(i)$ as follows: when a node p receives the flooded message, it compares the level index i with its Morse function value $f(p)$, and if the value is not equal, the message is discarded. Then by a landmark selection process similar to [15], each component selects a dominating landmark, denoted by $DM(\cdot)$, with the smallest ID. Therefore, the nodes in each component in $f^{-1}(i)$ are notified of a component ID.

Second, all components should be transformed to *regions* (arcs) of the Reeb graph, as shown in Fig. 5. This process starts from $f^{-1}(1)$ to $f^{-1}(I)$. We call a component P_i in $f^{-1}(i)$ is connected with a component P_{i+1} in $f^{-1}(i+1)$, if there exists a node p in the component P_i that has a neighbor p' in the component P_{i+1}. Then the nodes p and p' will notify the dominating landmarks $DM(P_i)$ and $DM(P_{i+1})$ in the components P_i and P_{i+1} respectively of this connectivity. According to Morse theory, there only exist three cases: (1) P_i is connected with P_{i+1} and another component Q_{i+1} in level-$(i+1)$; (2) P_i corresponds to P_{i+1} only; (3) P_{i+1} is connected with P_i and another component Q_i in level-i, which is notified to $DM(P_i)$ and $DM(P_{i+1})$ respectively. The three cases are illustrated in Fig. 7. The dominating landmarks will notify the nodes in the component P_{i+1} if P_{i+1} only corresponds to the component P_i. If so, the nodes in P_{i+1} are assigned the same *region* ID as the nodes in component P_i, otherwise, the nodes in component P_{i+1} are assigned a new region ID. After this process, every node is notified with its region ID. The result of constructing the Reeb graph is given in Fig. 5(c), the Reeb graph regions are distinguished by colors.

With the Morse function and Reeb graph constructed, all the *loop-end* regions are notified in a straight forward manner: if the dominating landmark $DM(P_{i+1})$ in P_{i+1} is notified that P_{i+1} is connected with P_i and Q_i in level-i (as shown in Fig. 7(c)), then component P_{i+1} and all the components in the same *region* are notified to be in a loop-end region.

Then to extract the maximum cut set $\mathcal{C}_{\max}$, each *loop-end* region I_m^i performs a bisection operation to extract a cut C_i

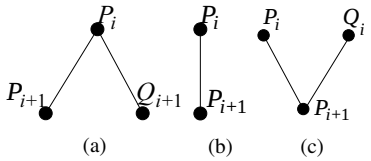


Fig. 7. P_i, Q_i are in level-i; P_{i+1}, Q_{i+1} are in level-$(i+1)$. (a) The component P_i is connected with P_{i+1} and Q_{i+1}; (b) The component P_i corresponds to component P_{i+1} only; (c) The component P_{i+1} is connected with P_i and Q_i.

as follows. Suppose a *loop-end* region I_m^i consists of a set of components $\{P_h\}$ where $H_1 \leq h \leq H_2$, then the bisection operation in I_m^i is initiated from P_{H_1} to P_{H_2}. The nodes in P_{H_1} reset its region ID to its parent's region ID. Since P_{H_1} is the first component of I_m^i, it is the neighbor of two components P_i and Q_i (Fig. 7(c)), and the component P_i and Q_i belong to *region* I_α and I_β respectively (as described in Section II-C). The nodes in P_{H_1} change their *region* ID to the region ID of I_α or I_β respectively. Therefore, P_{H_1} is bisected and assigned to *region* I_α or I_β. This process is carried out from P_{H_1} to P_{H_2}. Consequently, *loop-end* region I_m^i is bisected and two *newly merged region* I'_α and I'_β (as described in Section II-C) are generated, as shown in Fig. 5(d).

Next, the cut C_i is obtained by disconnecting I'_α and I'_β. In a continuous domain, a cut is a curve that disconnects I'_α and I'_β. In contrast, in discrete settings, disconnecting I'_α and I'_β will generate a pair of curves, as illustrated in Fig. 6(a).

Definition 3. *Suppose (p_A, p_B) is a pair of neighboring nodes, such that $p_A \in I'_\alpha$ and $p_B \in I'_\beta$, then (p_A, p_B) is called a cut pair.*

Definition 4. *In discrete settings, a cut C_i in a loop-end region I_m is a pair of simple closed curves C_i^A and C_i^B, which are generated by disconnecting all the cut pairs in I_m.*

Accordingly, the process of identifying a cut C_i is as follows: (1) each node p in *loop-end* region I_m will send a message to its neighbor p' in I_m, if p' has a different region ID with p, (p, p') is notified to be a cut pair; (2) by disconnecting all the cut pairs in *loop-end* region I_m, a cut C_i is identified. Since each cut generates two simple closed curves in $\mathbf{M}$, the generated genus-0 surface $\mathbf{G}$ has exactly $2n$ boundaries, as shown in Fig. 6(a).

B. A 'Pipe' $\mathbf{S}$ with Exactly Two Boundaries

Unfortunately, the genus-0 surface $\mathbf{G}$ with $2n$ boundaries is still complex: it is not homotopically equivalent to any simple planar topology, such as a planar annulus $\mathbf{D}$.Therefore, we propose to further slice $\mathbf{G}$, to a 'pipe' $\mathbf{S}$ — a genus-0 surface with exactly two boundaries $\partial\mathbf{S}_1$ and $\partial\mathbf{S}_2$, which is homotopically equivalent to a planar annulus $\mathbf{D}$.

To this end, we propose to cluster the $2n$ boundaries of $\mathbf{G}$ into two groups, each group with n boundaries. Then, by connecting the n boundaries with $(n-1)$ shortest paths within each group, two boundaries $\partial\mathbf{S}_1$ and $\partial\mathbf{S}_2$ are generated. Since all the loops are disconnected in $\mathbf{G}$, the Reeb graph is transformed to a tree structure $\mathbf{T}$, as illustrated in Fig. 8(b), in which a node represents an aforementioned *region*. In each region, one dominating landmark is selected (the process is quite similar to that of selecting a dominating landmark in a component). It is well-known that, given such a tree structure $\mathbf{T}$ that contains many nodes (regions), it can be traversed in a manner analogous to a Depth-First Search (DFS) as shown in Fig. 8(b). The arrow in Fig. 8(b) illustrates the search order: $c \to a \to d \to b \to AB$.

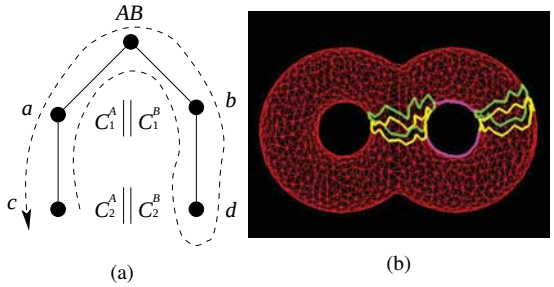


Fig. 8. (a) The tree structure $\mathbf{T}$ transformed from the Reeb graph in Fig. 5(a); (b) The shortest paths (in pink) connecting the boundaries.

In practice, the clustering process is initiated by a node with the smallest level index on one curve of a cut. For example, in Fig. 8(b), a node p on C_2^A initiates the clustering procedure. Then p notifies the dominating landmark in the region c to start to traverse $\mathbf{T}$. In the process $c \to a$, the curve C_2^A is clustered with C_1^A, therefore 2 boundaries are clustered to a group for a genus-2 torus. So the dominating landmark in a continues to traverse $\mathbf{T}$ to cluster another group. Then from $d \to b$, C_2^B is clustered with C_1^B. C_2^A and C_2^B find a shortest path to C_1^A and C_1^B respectively. The two shortest paths of the 'pipe' $\mathbf{S}$ are depicted in pink in Fig. 8(b). The local topology is then disconnected according to the shortest paths, and a genus-0 surface $\mathbf{S}$ with exactly two boundaries $\partial\mathbf{S}_1$ and $\partial\mathbf{S}_2$ is extracted. The result of the 'pipe' $\mathbf{S}$ is given in Fig. 6(b).

C. Discrete Surface Ricci Flow

Given a genus-0 surface $\mathbf{S} = (V, E, F)$ (where V, E, F represent the vertices, edges and face) with exactly two boundaries, Ricci flow algorithm is then applied to embed $\mathbf{S}$ into a planar annulus $\mathbf{D}$.

1) Circle packing metric: For a vertex $v_i \in V$, a circle with radius r_i is assigned. A function that assigns a radius r_i to each vertex v_i is denoted as: $\Gamma : V \to \mathbb{R}^+$. We also define a weight function: $\Phi : E \to [0, \frac{\pi}{2}]$ by assigning a positive weight w_{ij} to each edge e_{ij}. The pair of vertex radius and edge weight function on the mesh $\mathbf{S}$, (Γ, Φ), is called a *circle packing metric* of $\mathbf{S}$.

2) Discrete Ricci flow: Discrete Gaussian curvature is an intrinsic measure of curvature, and its value depends only on the distance metric of the surface.

Definition 5. *The discrete Gaussian curvature is defined as the angle deficit on a mesh. The discrete Gaussian curvature K_i on a vertex v_i is defined as:*

$$K_i = \begin{cases} 2\pi - \Sigma_{f_{ijk} \in F} \theta_i^{jk}, v_i \notin \partial \mathbf{S} \\ \pi - \Sigma_{f_{ijk} \in F} \theta_i^{jk}, v_i \in \partial \mathbf{S} \end{cases} \quad (2)$$

where θ_i^{jk} represents the corner angle attached to v_i in face $f_{ijk} \in F$, and $\partial \mathbf{S}$ is the boundary of $\mathbf{S}$.

Definition 6. *Suppose $u_i = logr_i$ for vertex v_i, then the discrete Ricci flow can be defined as follows:*

$$\frac{du_i(t)}{dt} = (\bar{K}_i - K_i) \quad (3)$$

where K and $\bar{K}_i$ are the current and target Gaussian curvatures at v_i, and t is the evolving time of the Ricci flow. To deform the initial circle packing metric $\mathbf{S}(\Gamma, \Phi)$ to flatten surface $\mathbf{S}$, the target curvature is set as follows:

$$\bar{K}_i = \begin{cases} 0, v_i \notin \partial \mathbf{S}, \\ \frac{2\pi}{L}, v_i \in \partial \mathbf{S}, \end{cases} \quad (4)$$

where L is the length of the inner boundary.

Ricci flow algorithm is performed to flatten the surface $\mathbf{S}$ to a strip $\mathbf{P}$. Then the Möbius transformation is applied to embed it into a planar annulus $\mathbf{D}$ as we mentioned in Section II-D3. Finally, every node p is assigned a virtual coordinate $Coord(x_p, y_p, 0)$ in the annulus, as shown in Fig. 6(c).

D. Routing Scheme

With the virtual coordinates, intuitively greedy routing can be applied directly on the annulus $\mathbf{D}$ with guaranteed delivery. However, there are still two problems. First, the cut pairs are disconnected in the embedding process, possibly creating long paths between some of nodes that are close to each other, as shown in Fig. 9(a); second, many packets tend to travel along the inner boundary by greedy routing, as shown in Fig. 9(b), potentially overloading the boundary nodes, which is also a vital problem in 2D greedy embedding [17], [18].

To solve the problems, we propose a variation of greedy routing based on the following observation. Suppose (s_i, s_i') and (s_o, s_o') are two cut pairs associated with the source node s, there are only three possible shortest routing paths between the source node s and the destination node d, as shown in Fig. 6(d): (1) s routes the packet to d all by interior nodes; (2) s first routes the packet to the node s_i on the inner boundary, and then to d from s_i'; (3) s first routes the packet to a node s_o on the outer boundary, and then to d from s_o'. Based on this observation, our routing scheme includes the following steps:

First, the inner boundary nodes of $\mathbf{D}$, denoted by $\partial \mathbf{D}_{inner}$, initiate flooding. In the process, every node s in $\mathbf{D}$ records a root node s_i, which is the closest to s among all nodes on $\partial \mathbf{D}_{inner}$. Also, the node s records an arbitrary s_i' such that (s_i, s_i') is a cut pair. It is noted that here s_i' could be in $\partial \mathbf{D}_{inner}$ or $\partial \mathbf{D}_{outer}$. Similarly, the outer boundary nodes $\partial \mathbf{D}_{outer}$ perform flooding, and every node s in $\mathbf{D}$ records a root node s_o and also records a node s_o'. It is worth noting that, every node requires only maintaining a 4-tuple $((s_i, s_i'), (s_o, s_o'))$, where

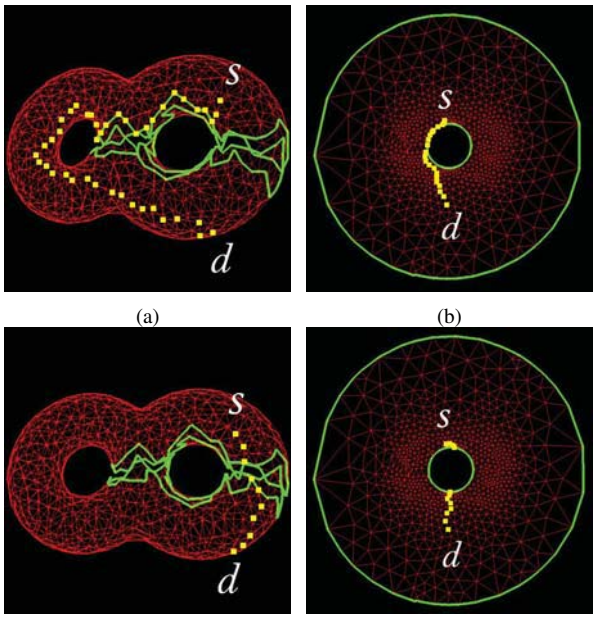


Fig. 9. (a) Greedy routing path in original topology; (b) Greedy routing path in the virtual coordinate system; (c) SINUS routing path in original topology; (d) SINUS routing path in the virtual coordinate system.

(s_i, s_i') and (s_o, s_o') are two cut pairs. That is, the storage overhead for local routing decisions is trivial.

Second, if we denote the Euclidean distance between two nodes by the virtual coordinate as $\texttt{dist}(\cdot, \cdot)$, the routing path is determined as follows: in the virtual coordinate system, the length of the routing path $s \to d$ is estimated by $L_A = \texttt{dist}(s, d)$; the length of the routing path $s \to s_i \to s_i' \to d$ is estimated by $L_B = \texttt{dist}(s, s_i) + \texttt{dist}(s_i', d)$; and the length of the routing path $s \to s_o \to s_o' \to d$ is estimated by $L_C = \texttt{dist}(s, s_o) + \texttt{dist}(s_o', d)$. Therefore, by finding the minimum value in the set $\{L_A, L_B, L_C\}$, the source node s chooses the corresponding route to deliver the packet.

Fig. 9(a) and Fig. 9(b) illustrate the routing path by directly applying greedy routing in the virtual coordinate. Fig. 9(c) and Fig. 9(d) illustrate the routing path via SINUS.

IV. Simulations and Evaluations

We have implemented a simulator and conducted a series of simulations on various 3D topologies. Simulation results presented in Fig. 10 depict four 3D topologies — a genus-1 corridor with 710 nodes, a genus-2 bowknot with 837 nodes, a genus-3 smile with 1,102 nodes, and genus-4 window with 5,429 nodes. It is observed that despite the variation in scale and complexity, SINUS extracts appropriate maximum cut sets for the networks by which it embeds them to the planar annuluses, which guarantees packet delivery between any pair of nodes. It is noted that, SINUS generates more perceptible "cuts" for large-scale sensor networks such as Fig. 10(h) and Fig. 10(i), as they are closer to smooth surfaces. The delivery rate of Greedy routing is 65% for Corridor, 76% for Bowknot, 70% for Smile and 81% for Window, while SINUS constantly

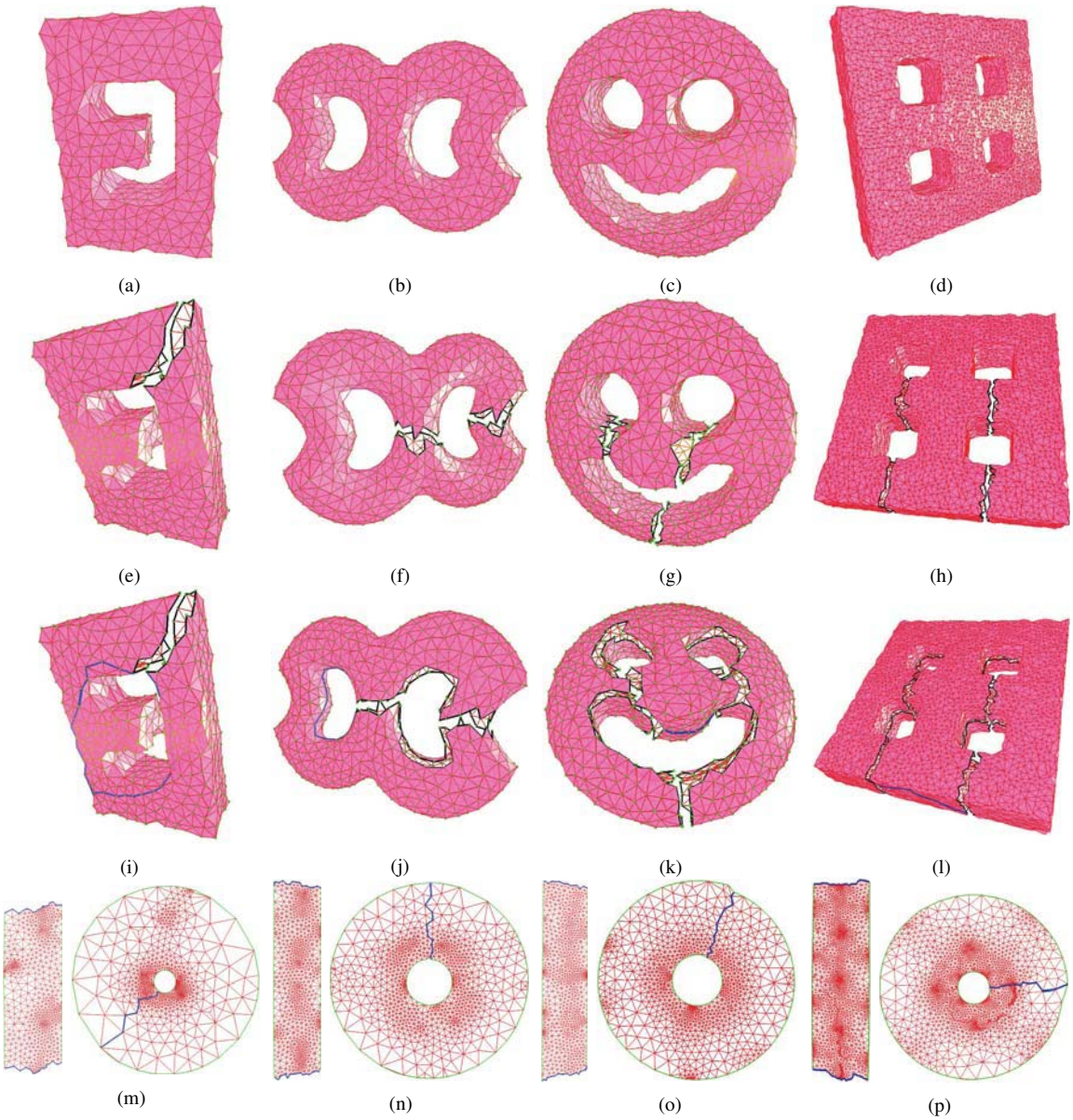


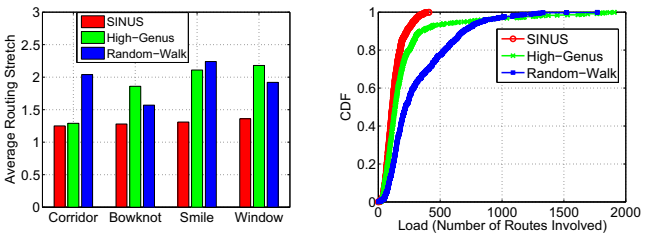
Fig. 10. Columns from left to right: (a) A genus-1 corridor network with 710 nodes; Avg deg is 8.92; (b) A genus-2 bowknot network with 837 nodes; Avg deg is 9.35; (c) A genus-3 smile network with 1,102 nodes; Avg deg is 10.01; (d) A genus-4 window network with 5,429 nodes; Avg deg is 9.74. Rows: (1) the original network; (2) the genus-0 surface $\mathbf{G}$ with 2n boundaries; (3) the 'pipe' $\mathbf{S}$ with exactly two boundaries; (4) the planar annulus $\mathbf{D}$.

produces a delivery rate of 100%. Therefore we focus on two factors for routing performance evaluation: routing stretch and load distribution. For comparison, we also implemented two other routing algorithms for 3D high genus surfaces with guaranteed delivery — the High-Genus algorithm in [22] and the Random-Walk algorithm in [4].

Routing Stretch: The routing stretch for a pair of nodes (s, d) is the ratio of the actual path length to the shortest path length between s and d. The average routing stretches of Random-Walk [4], High-Genus [22] and SINUS, are summarized in Fig. 11(a). In our simulation, 10,000 pairs of nodes are randomly selected to calculate the average routing stretch for each 3D network. It is observed that, SINUS yields a much smaller average routing stretch (1.30, the average of the 4 topologies in Fig. 11(a)) than Random-Walk [4] (1.94) and High-Genus [22] (1.86).

We also notice that, for the topology of the genus-1 corridor, the routing stretch of High-Genus is close to SINUS. This is because for genus-1 surfaces, High-Genus adopts the algorithm in [21] to embed the topology to a standard genus-1 torus without introducing decomposition. When decomposition is introduced for High-Genus, that is, when the topology is genus-2 or higher, the stretch of High-Genus increases significantly. The reason is that the adjacency graph does not consider the information of component size. So for routing between different components, a packet may travel through a series of large components even if there exists a much shorter path through small components. Consequently, the stretch of

High-Genus increases as the genus of the topology increases. It is also observed that, Random-Walk has larger stretches in Corridor and Smile than in Bowknot and Window. This is due to the fact that Random-Walk has to traverse the nodes in the concave region. When the concave region is large (say Corridor and Smile), the stretch increases significantly. In contrast to High-Genus and Random-Walk, SINUS maintains a relatively stable low routing stretch despite the diversity in the genus or topology concavity, since it embeds the topology as a whole with the concave region flattened.



(a) (b)

Fig. 11. (a) Average routing stretch; (b) Load distribution.

Load Balance: For algorithms that enable greedy routing by embedding, such as [17], [20], there exists a critical issue – the nodes on the embedded hole boundary tend to experience a higher traffic load. In SINUS, the case that the packet is routed between non-neighbor boundary nodes has been considered. Therefore, SINUS achieves good load balance and its boundary nodes are not overloaded. We simulated the traffic load of Random-Walk [4], High-Genus [22] and SINUS, with randomly selected 100,000 routes from the four tested topologies. Fig. 11(b) shows the Cumulative Distribution Function (CDF) of the loads of the nodes, which is measured by the number of routes involved.

It is observed that, in SINUS, all the nodes evolve a number of routes less than 450, and the CDF rapidly increases to 1, which means SINUS generates no overloaded nodes and the nodes involve relatively fewer routes in SINUS than Random-Walk and High-Genus. For Random-Walk, it has to traverse the concave region within a certain radius. Therefore, the nodes of the concave region attract a lot of traffic and may be overloaded. As for High-Genus, the boundary nodes in each region may be overloaded, since the six boundaries of a component [22] are concave, so routing within each region tends to route along these concave boundaries. In contrast, SINUS achieves a better load balance with the concave region flattened and a fairly traffic load on boundaries.

V. Conclusion

We have presented SINUS, a novel scalable and distributed routing algorithm with guaranteed delivery for sensor networks on high genus 3D surfaces. By slicing high genus surfaces to simpler ones for embedding, SINUS finally realizes a variation of greedy routing. The proposed algorithm is appealing as it requires no centralized operation, and offers a trivial storage overhead for routing on high genus surfaces. In the future we will study its uses in applications such as data processing [7]–[10], skeleton extraction [1], [11], [12], and localization [19], especially for 3D sensor networks, which have attracted the attention of many researchers.

References

[1] J. Bruck, J. Gao, and A. A. Jiang. Map: Medial axis based geometric routing in sensor networks. *Wireless Networks*, 13(6), 2007.

[2] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. In *Proceedings of ACM Symposium on Computational Geometry*, 2003.

[3] R. Flury, S. V. Pemmaraju, and R. Wattenhofer. Greedy routing with bounded stretch. In *Proceedings of INFOCOM*, 2009.

[4] R. Flury and R. Wattenhofer. Randomized 3d geographic routing. In *Proceedings of IEEE INFOCOM*, 2008.

[5] A. Gray, E. Abbena, and S. Salamon. *Modern differential geometry of curves and surfaces with Mathematica*. Chapman & Hall CRC, 2006.

[6] R. S. Hamilton. The ricci flow on surfaces. *Contemp. Math.*, 71(1), 1988.

[7] H. Jiang, J. Cheng, D. Wang, C. Wang, and G. Tan. Continuous multi-dimensional top-k query processing in sensor networks. In *Proceedings of IEEE INFOCOM*, 2011.

[8] H. Jiang, J. Cheng, D. Wang, C. Wang, and G. Tan. A general framework for efficient continuous multi-dimensional top-k query processing in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(9), 2012.

[9] H. Jiang, S. Jin, and C. Wang. Parameter-based data aggregation for statistical information extraction in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 59(8), 2010.

[10] H. Jiang, S. Jin, and C. Wang. Prediction or not? an energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(6), 2011.

[11] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, Y. Wu, and W. Liu. CASE: Connectivity-based skeleton extraction in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, 2009.

[12] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, Y. Wu, and W. Liu. Connectivity-based skeleton extraction in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(5), 2010.

[13] H. Jiang, T. Yu, C. Tian, G. Tan, and C. Wang. Consel: Connectivity-based segmentation in large-scale 2d/3d sensor networks. In *Proceedings of IEEE INFOCOM*, 2012.

[14] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM MOBICOM*, 2000.

[15] A. Nguyen, N. Milosavljevic, Q. Fang, J. Gao, and L. J. Guibas. Landmark selection and greedy landmark-descent routing for sensor networks. In *Proceedings of IEEE INFOCOM*, 2007.

[16] H. Pottmann, Q. Huang, B. Deng, A. Schiftner, M. Kilian, L. Guibas, and J. Wallner. Geodesic patterns. In *Proceedings of ACM SIGGRAPH*, 2010.

[17] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu. Greedy routing with guaranteed delivery using ricci flows. In *Proceedings of IEEE IPSN*, 2009.

[18] R. Sarkar, W. Zeng, J. Gao, and X. D. Gu. Covering space for in-network sensor data storage. In *Proceedings of IPSN*, 2010.

[19] G. Tan, H. Jiang, S. Zhang, and A.-M. Kermarrec. Connectivity-based and anchor-free localization in large-scale 2d/3d sensor networks. In *Proceedings of ACM MOBIHOC*, 2010.

[20] S. Xia, X. Yin, H. Wu, M. Jin, and X. D. Gu. Deterministic greedy routing with guaranteed delivery in 3d wireless sensor networks. In *Proceedings of ACM MOBIHOC*, 2011.

[21] X. Yu, X. Ban, W. Zeng, R. Sarkar, X. Gu, and J. Gao. Spherical representation and polyhedron routing for load balancing in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, 2011.

[22] X. Yu, X. Yin, W. Han, J. Gao, and X. Gu. Scalable routing in 3d high genus sensor networks using graph embedding. In *Proceedings of INFOCOM*, 2012.

[23] H. Zhou, H. Wu, S. Xia, M. Jin, and N. Ding. A distributed triangulation algorithm for wireless sensor networks on 2d and 3d surface. In *Proceedings of IEEE INFOCOM*, 2011.