Distributed Information Storage and Retrieval in 3-D Sensor Networks With General Topologies

Yang Yang, Miao Jin, Yao Zhao, and Hongyi Wu

***Abstract*—Distributed in-network data-centric processing aims to reduce energy consumed for communication and establish a self-contained data storage, retrieval, aggregation, and query sensor system that focuses more on the data itself rather than the identities of the individual network nodes. Double-ruling-based schemes support efficient in-network data-centric information storage and retrieval, especially for aggregated data, since all data with different types generated in a network can be conveniently retrieved along any single retrieval curve. Previous double-ruling-based research focuses on two-dimensional (2-D) wireless sensor networks where a 2-D planar setting is assumed. With increasing interests in deploying wireless sensors in three-dimensional (3-D) space for various applications, it is urgent yet fundamentally challenging to design double-ruling-based approach in general 3-D sensor networks because double-ruling-based schemes in general have much harder geometric constraints than other distributed in-network data-centric processing schemes. In this research, we propose a geographic location-free double-ruling-based approach for general 3-D sensor networks with possibly complicated topology and geometric shapes. Without the knowledge of the geographic location and the distance bound, a query simply travels along a simple curve with the guaranteed success to retrieve aggregated data through time and space with one or different types across the network. Extensive simulations and comparisons show the proposed scheme with low cost and a balanced traffic load.**

***Index Terms*—3-D sensor networks, data-centric, in-network, information storage and retrieval.**

I. Introduction

WIRELESS sensor networks have experienced an explosive growth in recent years. In comparison to earlier computer communication systems, the unique and intrinsic challenge in sensor networking is distributed and scalable computation and communication. In particular, an individual sensor is highly resource-constrained, with extremely limited computing, storage, and communication capacities. On the other hand, however, the target applications often require large-scale deployment where the amount of data generated, stored, and transmitted in the network grows proportionally with the network size. This dilemma renders the conventional sensor networking strategy that intends to transmit all sensor data to an external server impractical. To this end, distributed in-network data-centric storage and retrieval have been extensively discussed in the literature [1]–[11]. The new paradigm of distributed in-network data-centric processing focuses more on data themselves rather than the identities of individual sensor nodes. Data are uniquely named, and data processing is achieved using data names instead of network addresses, aiming to establish a self-contained data acquisition, storage, retrieval, and query system.

While a two-dimensional (2-D) planar setting has been assumed in most earlier studies of in-network data storage and retrieval, there has been increasing interest in deploying wireless sensors in three-dimensional (3-D) space for such applications as underwater reconnaissance and atmospheric monitoring. Several explorative 3-D sensor network testbeds have been developed recently (either in space or underwater) [12], [13]. Although they are all in relatively small size, we foresee large-scale deployment will soon be demanded in the near future.

This research focuses on in-network data-centric information storage and retrieval in large-scale 3-D sensor networks. We first summarize existing in-network data-centric storage and retrieval algorithms for 2-D networks, and then discuss the challenges in 3-D networks, followed by an overview of our proposed approach.

A. Overview of Distributed Information Storage and Retrieval Algorithms

Geographical hash table [3], [11], [14]–[16] is a general approach for in-network data-centric storage and retrieval. A basic geographical hash-table-based scheme hashes a datum by its type into geographic coordinates and stores at the sensor node geographically nearest to such coordinates. Queries apply the same hash table with the desired type to retrieve data from the storage node. Delivery of the data is implemented by geographic routing, such as GPSR [17]. To reduce bottleneck at the hash nodes and improve data survivability under node failure, a geographical hash-table-based scheme applies a structured replication with multiple mirrors scattered in the network. Structured replication reduces the cost of storage, but increases the cost of queries.

Manuscript received June 29, 2013; revised January 06, 2014 and April 07, 2014; accepted April 10, 2014; approved by IEEE TRANSACTIONS ON NETWORKING Editor G. Xue. Date of publication April 29, 2014; date of current version August 14, 2015. The work Y. Yang and M. Jin was supported in part by the NSF under Grants CCF-1054996, CNS-1018306, and CNS-1320931. The work of Y. Zhao and H. Wu was supported in part by the NSF under Grants CNS-1018306 and CNS-1320931.

Y. Yang, M. Jin, and H. Wu are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70503 USA (e-mail: yxy6700@cacs.louisiana.edu; mjin@cacs.louisiana.edu; wu@cacs.louisiana.edu).

Y. Zhao was with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70503 USA. He is now with the Research and Development Department, Epic Systems Corporation, Verona, WI 53593 USA (e-mail: kaiser1644@gmail.com)

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/TNET.2014.2317809

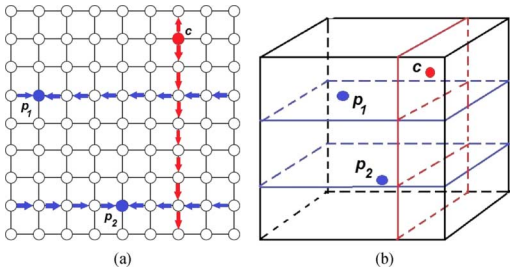

Fig. 1. Simple double-ruling scheme on (a) a 2-D grid sensor network and (b) a 3-D grid sensor network. p_1 and P_2 are two information producers with their replication routes marked in blue. c is an information consumer with its retrieval route marked in red.

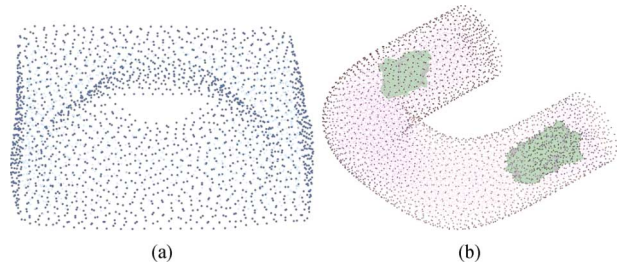

Fig. 2. (a) 3-D sensor network deployed underwater around an island. (b) 3-D sensor network with several coverage holes inside.

Different from geographical hash-table-based schemes, a double-ruling-based scheme works as follows. A datum (or a pointer to the datum) is duplicated along a curve called replication curve, and a query travels along another curve called retrieval curve. Successful retrieval is guaranteed if the retrieval curve intersects the replication curve. A simple double-ruling scheme on a planar grid is illustrated in Fig. 1(a) where nodes are located at lattice points. The replication curves follow the horizontal lines, and the retrieval curves follow the vertical lines. By traveling along a vertical line, a data query, called information consumer, can always find the requested data generated by an information producer.

Double-ruling-based schemes support efficient data retrieval since all data with different types generated in a network can be conveniently retrieved along one simple retrieval curve. This is in a sharp contrast to geographical hash-table-based schemes where an information consumer has to visit multiple nodes scattered in the network to collect data with different types hashed to various locations. Moreover, with modestly increased data replication, a double-ruling-based scheme has well-balanced load across the network, while nodes near the hashed location suffer much higher traffic load than others in a geographical hash-table-based scheme. A double-ruling-based scheme also has better fault tolerance against geographically concentrated node failure by replicating data on nodes that are uncorrelated with node proximity.

Double-ruling-based schemes achieve all the desired properties at the cost of more data duplication and much stronger geometric constraints on the shape of a sensor network than geographical hash-table-based schemes. Previous double-ruling-based schemes either assume networks with 2-D grid shape [1], [4], [18] or with heavy data replication to achieve high probability that the retrieval curve would meet one of the replication curves within the sensor network field [5]. To extend double-ruling scheme to networks with uneven sensor distribution and irregular geometric shapes, landmark-based scheme [9] is proposed to partition the sensor field into tiles. GHT is adopted at the tile level, i.e., a data type is hashed to a tile instead of a single node. Inside each tile, a double-ruling scheme is applied to ensure the intersection of a retrieval path and a replication path. Later, a location-free double-ruling scheme is introduced in [19] based on boundary recognition and the computation of the respective gradient fields. To improve the flexibility of retrieval, a spherical projection-based double-ruling scheme is proposed in [10], where a planar network is mapped to a sphere based on the inverse of stereographic projection. Both the replication and retrieval curves are great circles such that a retrieval curve always intersects all other replication circles.

B. Challenges in 3-D Networks

Although double-ruling has shown highly effective for distributed information storage and retrieval in 2-D sensor networks, it cannot be efficiently applied in 3-D networks. A naive double-ruling-based scheme in 3-D sensor networks is shown in Fig. 1(b). In such a 3-D grid-based cube-shape sensor network, data replication and retrieval are along the horizontal and vertical planes, respectively, such that a retrieval plane intersects all replication planes. Besides an extremely high cost of data replication, such a 3-D grid-based double-ruling scheme requires a network with a regular cube shape and uniform node distribution. Recently, a volumetric parametrization-based double-ruling scheme is introduced in [20] and [21]. They map a 3-D sensor network to a cube and assign each node a virtual coordinates. Their method requires the network shape to be topologically equivalent to a cube.[1] However, many practical 3-D sensor networks are topologically different from a cube. Fig. 2(a) shows a 3-D sensor network with sensors deployed underwater around an island. The topology of the network is equivalent to a donut with a handle. Fig. 2(b) gives another example of a 3-D sensor network with multiple coverage holes inside.

Another challenge to achieve double-ruling in 3-D sensor networks is the delivery of data and query to the mapped geolocalizations for in-network data storage and retrieval. Previous GHT and double-ruling-based schemes on 2-D sensor networks rely on geographical routing schemes, such as GPSR [17]. They require global location information. However, some application scenarios in 3-D prohibit the reception of satellite signals by part or all of the sensors, rendering it impossible to solely rely on global navigation systems. Moreover, the cost to equip a GPS receiver at each node is greatly exacerbated in a 3-D sensor network due to the dramatically increased sensor quantity in order to cover a 3-D space compared to its 2-D counterpart. Even if we assume location information for each sensor node, it is still nontrivial to design a scalable geographic routing scheme that

[1]Two shapes topologically equivalent to each other means they can be continuously transformed one into the other. A continuous mapping exists between them.

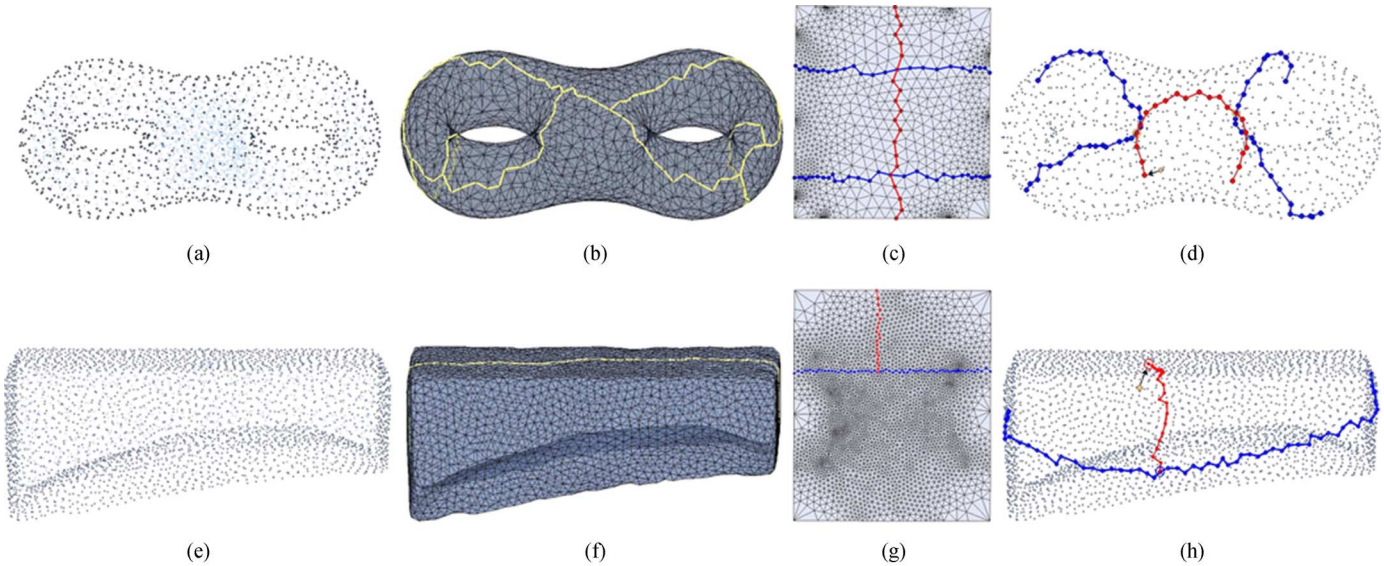

Fig. 3. (a) 3-D sensor network model with two handles. (b) Computed cut graph of the triangulated boundary surface of the network is marked with yellow. (c) Boundary surface of the network is cut open to a topological disk along the cut graph and mapped to an aligned planar rectangle. The horizontal lines marked with blue are two data replication curves with different types. The vertical line marked with red is one data retrieval curve. (d) Query of aggregated data corresponding to (c). (e) 3-D sensor network model. (f) Computed cut graph marked with yellow. (g) Boundary surface of the network is cut open to a topological disk along the cut graph and mapped to an aligned planar rectangle. The horizontal line marked with blue is one data replication curve. The vertical line marked with red is one data retrieval curve. (h) Data query corresponding to (g).

guarantees successful delivery and requires constant storage at each node in general 3-D sensor networks.

C. Our Approach

Considering the boundary surface of a 3-D volume is always a closed surface, the proposed approach is motivated by a topological concept that any closed surface can be cut open to a topological disk along an appropriate set of edges called a cut graph of the surface [22]. Two examples given in Fig. 3 briefly illustrate the basic idea of our approach. The first example is a 3-D volumetric sensor network model with two handles [see Fig. 3(a)]. We first detect the boundary nodes and build a triangular structure of the identified boundary surface of the network [see Fig. 3(a)]. We then compute a cut graph of the boundary surface of the network that is marked with yellow color in Fig. 3(b). We cut the boundary surface of the network open to a topological disk along the cut graph, and then map it to an aligned planar rectangle such that each boundary node of the network is associated with a planar rectangle virtual coordinates [see Fig. 3(c)]. Each nonboundary sensor stores the ID of its neighbor nearest to the boundary of the network. A data generator follows the sequence of IDs to the boundary of the network, and then travels along a horizontal line of the virtual planar rectangle and leaves data copies. The two horizontal lines of the virtual planar rectangle marked with blue color shown in Fig. 3(c) correspond to the two real data replication curves of the network shown in Fig. 3(d) marked with the same color. Similarly, a consumer follows the sequence of IDs to the boundary of the network and collects the aggregated data of different types along a vertical line. The vertical line of the virtual planar rectangle marked with red shown in Fig. 3(c) corresponds to the data aggregation curve of the network shown in Fig. 3(d) marked with the same color. The second example is also a 3-D sensor network deployed underwater [see Fig. 3(e)]. We cut the boundary surface of the network open to a topological disk along a computed cut graph [see Fig. 3(f)], and then assign each boundary node of the network a planar rectangle virtual coordinates [see Fig. 3(g)]. Similarly, Fig. 3(h) gives a data query example.

The proposed cut-graph-based double-ruling approach works for 3-D sensor networks with general topology and geometry shapes. Without the knowledge of the geographic location and the distance bound, the success of data retrieval is always guaranteed because a pair of horizontal and vertical lines surely intersect. Retrieval of aggregated data through time and space with different types is also guaranteed. A consumer travels along a vertical line and then collects all desired information in the network because the vertical line intersects all horizontal lines—replication curves of the network. The proposed algorithm has no constraints on communication models and is fully distributed. Each node only needs to exchange information with its direct neighbors. The amount of extra information stored at individual nodes is constant and small. Simulation results show the proposed approach with low cost and a balanced traffic load.

The rest of this paper is organized as follows. Section II explains in detail the algorithm to compute the cut graph. Section III describes the method to generate the planar rectangle virtual coordinates of boundary nodes of a network. Section IV discusses a number of implementation issues of the proposed cut-graph-based double-ruling approach in 3-D sensor networks. Section V presents the simulation results, and Section VII concludes the paper.

II. Cut Graph of Boundary Surface of the Network

As briefly introduced in Section I, given a sensor network deployed in 3-D volume with distance information within one-hop neighborhood only, we detect its boundary nodes and extract

a triangular boundary surface of the 3-D volume before computing its cut graph. Note that a boundary surface of any 3-D volume is a closed surface (i.e., without any holes).

We apply the algorithm proposed in [23] to identify boundary nodes of a 3-D volume sensor network. The algorithm is fully distributed with no constraints on communication models. Although the algorithm can tolerate a moderate distance measurement error, we do not require accurate detection of all boundary nodes of the network. We only need a connected triangular structure to approximate the boundary surface of the 3-D volume network. We allow some mistakenly detected nonboundary vertices on the triangular structure.

We then apply the algorithm proposed in [24] to extract a triangular structure of the detected boundary nodes of the network. The algorithm takes the connectivity graph of the detected boundary nodes of the network as input. Based on locally estimated distances between nodes within one-hop communication range, a local coordinates system can be constructed. Each edge based on its local coordinates system then computes a weight that measures the number of triangles shared by the edge and the various local neighbor sets for the edge in the initial graph. An iterative algorithm keeps removing edges until each edge in the final graph has weight exactly two—shared by two triangles only. The algorithm is also fully distributed with no constraints on communication models. The constraint of the algorithm is that it assumes that a triangular graph is a subgraph of the initial connectivity graph of boundary nodes of the network.

Before we introduce the algorithm of computing a cut graph of the triangulated boundary surface of a 3-D volume network in Section II-B, we give a brief review of some important topological concepts that are necessary to the algorithm in Section II-A. We refer the interested reader to [22], [25] for formal definitions and further topological background.

A. Topological Background

A surface is orientable if it has two distinct sides. General surfaces in the real world are orientable surfaces. A loop is a continuous function of the circle on a surface. Two loops are homotopic if there is a continuous deformation from one loop onto the other on the surface. Denote M a connected and orientable surface embedded in 3-D, and L a loop on M. L is contractible if it is homotopic to a constant (a loop that can shrink to a single point) as shown in Fig. 4 with L_1; otherwise it is noncontractible. L is surface separating if it can be expressed as the symmetric difference of boundaries of topological disks embedded in surface as shown in Fig. 4 with L_1 and L_2; otherwise it is nonseparating as shown in Fig. 4 with L_3. Any nonseparating loop is a noncontractible loop; similarly, any contractible loop is a separating loop.

The genus of M is the maximum number of disjoint nonseparating loops $L_1, L_2, \ldots, L_g$ in M; that is, any L_i and L_j have no topological intersection if $i \neq j$, and $M \backslash (L_1 \cup \cdots L_g)$ is connected. The genus number is the most basic topology information of a surface and equals the number of handles. For example, a disk and a sphere have a genus 0, and a torus has a genus 1.

Any closed surface M (e.g., a surface without a hole) can be opened into a topological disk D (e.g., a surface with one boundary) by cutting along an appropriate set of edges called cut graph. Denote G a cut graph of M. Each edge of G appears twice on the boundary of D, and we can obtain M by gluing together these corresponding boundary edges of D. Fig. 5 shows cut graphs of genus-0, genus-1, and genus-2 surfaces, respectively. The three closed surfaces are cut open to topological disks along the given cut graphs. Denote g the genus number of M. The number of base loops of a cut graph is $2g$.

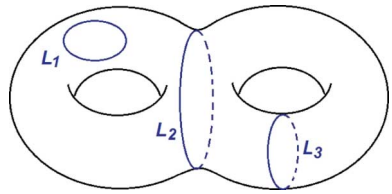

Fig. 4. Loops on a genus-2 surface: L_1 is contractible and surface separating; L_2 is noncontractible and surface separating; L_3 is noncontractible and nonseparating.

B. Cut Graph of the Boundary Surface of a Network

The triangulated boundary surface of a 3-D sensor network is connected, orientable, and closed. We abuse the symbol M to denote a connected and orientable triangulated surface embedded in 3-D. Specifically, we denote $M = (V, E, F)$ a triangulated surface embedded in 3-D, consisting of vertices V, edges E, and triangle faces F. Denote $v_i \in V$ a vertex with ID i; $e_{ij} \in E$ an edge with two ending vertices v_i and v_j; $f_{ijk} \in F$ a triangle face with vertices v_i, v_j, and v_k.

The problem of computing the shortest cut graph of a general topology surface is studied in [26], which proves the problem NP-hard and provides a polynomial-time approximation algorithm. Canonical systems of loops, a system of $2g$ (g is the genus number of the surface) loops with a single base vertex to cut a surface into a disk, is discussed in [27]. Erickson and Whittlesey [28] provide a greedy algorithm to compute a shortest system of such canonical loops. Shortest cut graph with prescribed vertex set is discussed later in [29].

Instead of adopting the above optimization algorithms, we propose a fully distributed two-step algorithm motivated by that of [30], aiming to balance the size of the cut graph measured by the number of edges and the computation cost and the communication cost measured by the number of exchanged messages. The basic idea of the first step introduced in Section II-B.1 is to grow triangles with the width first way. At each step of the growing, all the marked triangles always form a topological disk, and the marked edges form the boundary of the disk. After all the triangles have been marked, we can cut the closed surface into a topological disk along the marked edges. The size of the cut graph can be largely reduced by trimming away those unnecessarily marked edges at the second step introduced in Section II-B.2. For genus-0 surfaces, there are no marked edges left after trimming, which provides a convenient way to automatically identify the topology of a 3-D sensor network. We discuss the computational cost of the two-step algorithm in Section II-B.3.

1) Computing the Cut Graph: The algorithm starts from one randomly chosen triangle f_{ijk} of M, which can be the one with

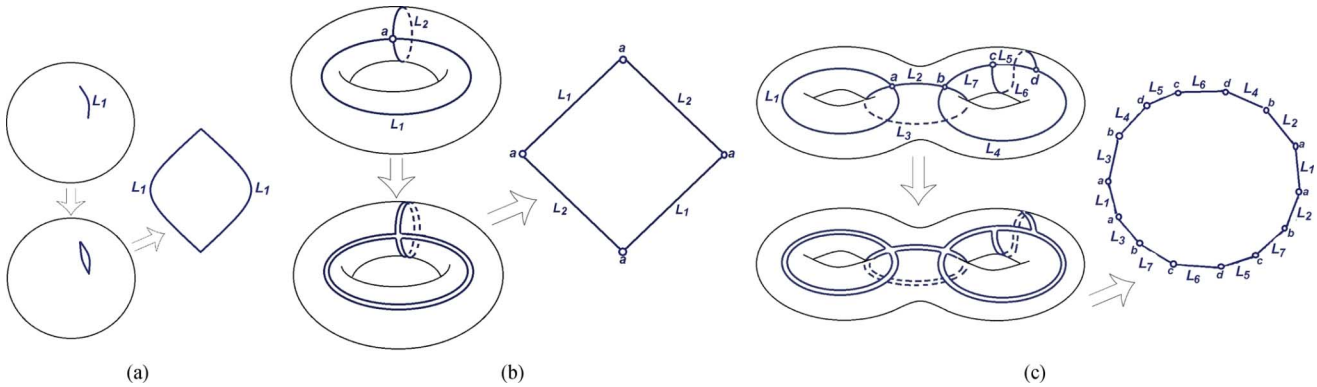

Fig. 5. Any closed surface M can be cut open to a topological disk D along one cut graph of the surface G: (a) Network I—a genus-0 surface; (b) Network II—a genus-1 surface; (c) Network III—a genus-2 surface. Notice that the edges on G appear twice on the boundary of D. D has one-to-one mapping with M except the boundary of D.

the smallest node ID. f_{ijk} marks itself and its three edges e_{ij}, e_{jk}, and e_{ki}. Each of the marked edges checks whether it is shared by two marked triangles. For example, edge e_{ij} finds its neighboring triangle f_{jil} unmarked. e_{ij} then removes mark from itself but adds mark on triangle f_{jil} and edges e_{il} and e_{lj}. Note that it is possible that e_{il} or e_{lj} may have been marked already. The propagation algorithm stops when all the triangles of M have been marked. Let all the marked edges be G, which form a cut graph of M. We prove the correctness of the algorithm as follows.

Theorem 1: A closed surface M can be cut into a topological disk D along the cut graph G computed by the above algorithm.

Proof: We can prove it by way of induction. The algorithm starts from one triangle of M with its edges marked. This single triangle is topologically equivalent to a disk with boundary edges marked. We denote it as D_1. After the $i-1$ steps, $i-1$ triangles have been marked and added. We have D_{i-1}. Suppose D_{i-1} is a topological disk with boundary edges marked. At the step i, an unmarked triangle is identified, which means this triangle does not belong to D_{i-1} but must share at least one of the boundary edges of D_{i-1}. By adding this triangle into D_{i-1} and updating the marked edges as the way described by the algorithm, the newly formed D_i is still a topological disk. Marked edges form the boundary of D_i. When all the triangles of M have been marked, D_m has included all the triangles and is still a topological disk with its boundary edges marked. The surface M can then be cut open to a topological disk $D = D_m$ along the marked edges by splitting each marked edge and its two ending vertices to two. ■

Theorem 2: The cut graph computed by the above algorithm is connected.

Proof: This can be easily proved by way of contradiction. If the computed cut graph is disconnected, then the boundary of D_m in the proof of Theorem 1 is disconnected. It is impossible that the boundary of a topological disk is disconnected. ■

2) Trimming: If we cut a closed surface M open along the cut graph computed by the above propagation algorithm, the boundary of the topological disk surface D would be extremely zigzagged. The size of the boundary edges of D in the worst case can equal the size of the triangles of M. The reason is that, in the worst case, the size of the marked edges is increased by one each time when one triangle of M is marked.

To control the size of the cut graph, we need to trim away those unnecessarily marked edges. Marked edges forming nonsegmenting loops of M are necessary because the loops correspond to the cut open of each handle. Meanwhile for those marked and dangling tree edges that do not belong to or connect any loops, M can still be cut open to a topological disk after removing them from the cut graph.

The algorithm of trimming is straightforward. Each marked edge checks whether its two ending vertices connect to other marked edges. If one of its two ending vertices does not connect to any other marked edges, the edge is identified as a dangling tree edge and can be unmarked—removed from the cut graph. The unmarked edge will then send messages to its neighboring marked edges through the other ending vertex. Its neighbors then conduct the same checking when receiving the message. The trimming process stops when there is no marked, dangling tree edges. Note that it is impossible that the two ending vertices of a marked edge do not connect to any other marked edges because we proved that the marked edges are connected in Theorem 2.

Theorem 3: The removal of marked, dangling tree edges does not disconnect the cut graph.

Proof: Each dangling tree edge connects to G with only one ending vertex when it is removed. Hence, the removal of dangling tree edges does not disconnect G. ■

We have the following theorem, which said that all marked edges will be unmarked by the end of the trimming process for a genus-0 surface M.

Theorem 4: After the trimming process, there is no marked edge left for a closed genus-0 surface M (i.e., a topological sphere surface).

Proof: We first use way of contradiction to prove the cut graph of a genus-0 surface M computed by the above propagation algorithm is a tree. Considering the fact that M is topologically equivalent to a sphere, every loop on M is contractible, therefore surface-separating, If there exists a loop in the computed cut graph of M, the loop will separate M to two disconnected parts. This contradicts what we have proved in Theorem 1: At each step i of the propagation, D_i is always a topological disk. Thus, the computed cut graph of a genus-0 surface M should be a tree without loops. Each marked edge will be identified as dangling tree edge and removed eventually

by the proposed trimming algorithm. There will be no marked edges left at the end of the trimming process. ■

For a boundary surface of a network-detected genus 0, the boundary node of the network with the smallest ID conducts a simple flooding on the boundary surface of the network to find one boundary node with the longest shortest path measured by hops on the boundary surface. We then cut the surface open to a topological disk surface (i.e., a simply connected domain) along the shortest path between the pair of nodes.

3) Computational Cost: The algorithm to compute a cut graph of a boundary surface of a network involves many topological concepts, but the computational complexity of the algorithm is actually low.

The first step of computing the cut graph starts from marking one randomly chosen triangle of the boundary surface of the network and then continues to mark others with the width first way, so each triangle will be visited only once before this step ends. The time complexity of this step is linear to the size of the triangles of the boundary surface. During the growth of the marked triangles, only nodes at the two ends of a marked edge exchange messages. They exchange messages at most twice: the first time when the marked edge is shared by only one marked triangle, and the second time when the marked edge is shared by two marked triangles. Each edge will be marked only once, so the communication cost is twice of the size of the edges of the boundary surface of the network.

The second step of trimming is to remove those unnecessary marked edges. Both the time complexity and communication cost are determined by the number of marked edges at the end of the first step. The size of the marked edges in the worst case can equal the size of the triangles of the boundary surface of the network. For genus-0 surfaces, denote m the size of the boundary nodes; an extra $O(m \log m)$ cost is spent to find one longest shortest path on the boundary surface of the network after all marked edges are trimmed out.

Given an arbitrary triangular surface, the sizes of its triangles and edges are both linear to the size of the nodes. In summary, for non-genus-0 surfaces, both the time complexity and the communication cost of the whole algorithm are linear to the size of the boundary nodes of a network. For genus-0 surfaces, we need an extra $O(m \log m)$ cost. Note that the size of the boundary nodes of the network is far less than the size of the network in general.

III. Generating Planar Rectangle Virtual Coordinates

After we virtually cut the outside boundary surface of a network M to a topological disk D along the computed cut graph. We then apply discrete surface Ricci flow to compute the planar rectangle virtual coordinates of boundary nodes. We introduce briefly discrete surface Ricci flow in Section III-A, and then the algorithm in Section III-B.

A. Discrete Surface Ricci Flow

To briefly introduce the concept of discrete surface Ricci flow, we start from the definitions of circle packing metric and discrete Gaussian curvature. Given a topological disk triangulated surface $D = (V, E, F)$ as defined in a similar way in Section II-B, we assign each v_i a circle with radius γ_i and denote the radius function $\Gamma : V \to \mathbb{R}^+$. For each edge e_{ij}, the two circles at v_i and v_j intersect with an acute angle ϕ_{ij}. We call ϕ_{ij} the *weight* of e_{ij}, and denote the weight function $\Phi : E \to [0, \frac{\pi}{2}]$.

Definition 5 (Circle Packing Metric): [31] A circle packing metric of D includes Γ and Φ.

Denote l_{ij} the length of e_{ij}. l_{ij} can be computed from γ_i, γ_j, and ϕ_{ij} from the following cosine law:

$$l_{ij}{}^2 = \gamma_i{}^2 + \gamma_j{}^2 + 2\gamma_i\gamma_j \cos\phi_{ij}. \tag{1}$$

Discrete Gaussian curvature measures how curved a discrete surface is embedded in $\mathbb{R}^3$.

Definition 6 (Discrete Gaussian Curvature): Denote θ_i^{jk} the corner angle attached to v_i belonging to f_{ijk}, ∂D the boundary of D, and K_i the discrete Gaussian curvature at vertex v_i. K_i can be computed as the angle deficit at v_i

$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \notin \partial D \\ \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \in \partial D. \end{cases} \tag{2}$$

Definition 7 (Discrete Surface Ricci Flow [32]): Denote (Γ_0, Φ) an initial circle packing metric of M, $\overline{K}_i$ and K_i the target and current Gaussian curvatures at v_i, respectively, and t the time. Let u_i be the logarithm of γ_i for each v_i. The discrete surface Ricci flow is defined as

$$\frac{du_i(t)}{dt} = (\overline{K}_i - K_i). \tag{3}$$

Discrete surface Ricci flow deforms the initial circle packing metric such that the final circle packing metric induces edge lengths satisfying the target Gaussian curvatures. Chow and Luo prove the convergence of discrete surface Ricci flow in [32]. We refer readers to our previous work [33] for details of discrete surface Ricci flow.

B. Computing Planar Rectangle Virtual Coordinates

A planar rectangle has zero Gaussian curvature everywhere except its four corner points with Gaussian curvature $\frac{1}{2}\pi$. Hence, we uniformly pick four vertices along the boundary of the topological disk triangulated surface D and assign their target Gaussian curvatures: $\overline{k_i} = \frac{1}{2}\pi$. For all other vertices of D, we assign: $\overline{k_i} = 0$.

We initialize a circle packing metric on D such that each circle associated with a vertex has a unit radius, i.e., $\gamma_i = 1, u_i = \log\gamma_i = 0$ for each v_i, and $\phi_{ij} = \frac{\pi}{2}$ for each e_{ij}. Discrete surface Ricci flow deforms the circle packing metric on D such that the induced edge lengths from final circle packing metric satisfy our predefined Gaussian curvatures. Isometric embedding of D on a plane based on the computed edge lengths generates a planar rectangle mapping of D. The mapping is diffeomorphism that provides planar rectangle virtual coordinates for vertices of D. The detail of the algorithm is as follows.

1) Initialization of circle packing metric: For each v_i, $u_i = 0$. For each e_{ij}, $\phi_{ij} = \frac{\pi}{2}$.
2) Compute edge length for each e_{ij}: $l_{ij} = e^{u_i} + e^{u_j}$.

3) Compute each θ_i^{jk} according to the law of cosines:
$$\theta_i^{jk} = \cos^{-1} \frac{l_{ij}^2 + l_{ki}^2 - l_{jk}^2}{2 l_{ij} l_{ki}}.$$
4) Compute current Gaussian curvature k_i for each v_i as (2).
5) Denote ϵ a threshold and set to $1e-4$. If all $|\overline{k_i} - k_i| < \epsilon$, the algorithm goes to the next step; otherwise, $u_i = u_i + \delta(\overline{k_i} - k_i)$, where δ is a small constant and set to 0.1, and the algorithm goes back to step 2.
6) Isometric embedding: Denote p_i the planar coordinates of each v_i. Start from a boundary edge e_{ij} with v_i one of the four chosen boundary vertices: We assign $p_i = (0, 0)$, $p_j = (l_{ij}, 0)$. In a breadth-first-search way, if f_{ijk} has exactly two vertices (e.g., v_i and v_j) with planar coordinates (e.g., p_i and p_j), compute p_k as one intersection point of two circles centered at p_i and p_j with radii l_{ik} and l_{jk}, respectively, and satisfying $(p_k - p_i) \times (p_j - p_k) > 0$.[2] Repeat the above process until every vertex has its planar coordinates. The planar rectangle is automatically aligned with x-axis.

Note that each boundary node of the network only needs to exchange information with its direct neighbors when implementing the above algorithm.

IV. Implementation

A. Data Replication

Since a network is location-free, we let each nonboundary node store the ID of its neighbor nearest to the boundary of the network. A producer follows a sequence of nodes to the nearest boundary node of the network denoted as p. The boundary surface of the network has been mapped to a virtual planar rectangle, so each boundary node has planar rectangle virtual coordinates. We assume a data replication curve is along a horizontal line of the virtual planar rectangle. The horizontal line through p is unique, solely determined by the y-coordinate of the planar rectangle virtual coordinates of p. The producer travels and leaves pointers or copies of the data at nodes along the line with two directions—one with the increased and the other with the decreased x values. At each step, the producer simply checks the planar rectangle virtual coordinates of its one range neighbors and chooses the one with the closest distance to the line and along the current direction. Once finishing data replication, the producer turns back and follows the reversed path back.

B. Data Retrieval

Without awareness of the knowledge of the producer's location and the distance, a consumer follows a sequence of nodes to the nearest boundary node denoted as p. We assume a data retrieval curve is along a vertical line of the virtual planar rectangle. A vertical line passing through p is determined solely by the x-coordinate of the planar rectangle virtual coordinates of p. The consumer simply travels along the line with two directions—one with the increased and the other with the decreased y values. At each step, similarly, the producer simply checks the planar rectangle virtual coordinates of its one range neighbors and chooses the one with the closest distance to the line and along the current direction. Note that the boundary surface of the network is only virtually cut open and mapped to a planar rectangle. Once a consumer hits the boundary side of the rectangle, the consumer can cross the boundary side and keep traveling along the line with the same direction. The consumer stops as soon as it hits the replication curve of its desired data. If there are multiple producers and different types of data, the consumer travels along a full vertical line to collect all the aggregated data in the network. Once data has been collected, the consumer turns back and follows the reversed path back.

[2]The direction of the cross product of the two planar vectors points outside instead of inside.

C. Delivery of Data and Query

As a preprocessing, each of the boundary nodes of the network sends messages recording its minimum hop count to boundary (initialized to zero) to its neighbors. A nonboundary node receives a message and compares it to its current record (initialized to infinity). If the received count has more than one hop count less, the node updates its current one and records the ID of its neighbor sending this message. The node also updates the count of the message and then sends to its neighbors. Otherwise, the node simply discards the message. When there is no message in the network, each of the nonboundary nodes of the network has recorded the ID of its neighbor nearest to boundary. It is then straightforward for a producer or a consumer to travel along the shortest path to the boundary according to the sequences of IDs.

D. Storage

We have very limited information stored at the nodes of the network. For each of the nonboundary nodes, it only stores the ID of its neighbor nearest to boundary; for each of the boundary nodes, it stores the computed planar rectangle virtual coordinates. For the data replication, we can leave copies of data on either all the nodes along the replication curve; or just a small portion of nodes sampled along the replication curve, which is a tradeoff between the storage cost and the retrieval cost as discussed in Section V-D.

E. Time Complexity and Communication Cost

We measure the communication cost by the number of messages. Denote the size of all the nodes of a network as n and the size of its boundary nodes as m. We summarize the time complexity and the communication cost of the major steps of the cut-graph-based double-ruling scheme.

Both the time complexity and the communication cost to compute the cut graph are linear to the size of the boundary nodes of the network, $O(m)$, including the trimming step.

We apply discrete Ricci flow to compute the planar rectangle virtual coordinates. The number of iterations, as shown by Fig. 6 for the model in Fig. 3(a), determines the time complexity of computing the edge lengths, given by $-C\frac{\log \epsilon}{\lambda}$ where C is a constant, ϵ is the threshold of curvature error (set to $1\mathrm{e}-4$ in our implementation), and λ is the step length of each iteration (set to 0.1 in our implementation) [32]. Since each vertex only needs to exchange messages with its one range neighbors at one iteration, the corresponding communication

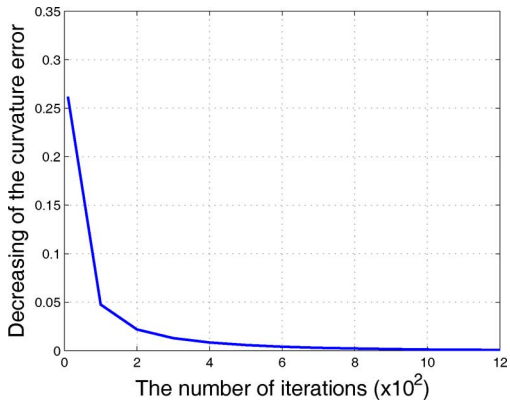

Fig. 6. Convergence rate of discrete Ricci flow.

cost is $-C\frac{\log\epsilon}{\lambda}d'm$ where d' is the average vertex degree of the triangulated boundary surface. With the computed edge lengths, both the time complexity and the communication cost of computing the planar rectangle embedding are linear to the size of the boundary nodes, i.e., $O(m)$.

Let the boundary nodes of the network synchronize among themselves and start to send messages recording the minimum hop count to boundary to its neighbors at roughly the same time [34], [35]. The total communication cost for each nonboundary node to record the ID of its neighbor nearest to boundary is $O(dn)$, where d is the average number of neighbors of each node. The time complexity is $O(n)$.

F. Bound on Data Replication and Retrieval Costs

Data replication and retrieval costs, without considering the cost to travel to the nearest boundary node for inner nodes, are bounded by the number of nodes along the horizontal and vertical lines of the virtual planar rectangle, respectively.

The algorithm to compute the cut graph in Section II starts from one randomly chosen triangle of the boundary surface of a network and propagates to grow the marked triangles in a width-first way until all triangles of the boundary surface have been marked. When cutting the boundary surface open to a topological disk along the computed cut graph, the longest radius of the topological disk is bounded by the longest shortest path of the nodes of the starting triangle on the boundary surface. We then uniformly pick four nodes along the boundary of the topological disk and map the topological disk to a well-aligned planar rectangle with corners of the four chosen nodes in Section III using Ricci flow. The mapping is a conformal map (i.e., a diffeomorphism) that preserves angles and local shape. Specifically, the mapping preserves both the neighborship of a boundary node and the relative positions of its neighbors when virtually mapped to a planar rectangle. Therefore, the horizontal and vertical lines of the rectangle should be bounded by the diameter of the topological disk. It is twice of the longest shortest path of the nodes of the starting triangle.

Since we randomly choose the starting triangle to compute the cut graph, the horizontal and vertical lines of the vertical rectangle should be bounded by twice of the longest shortest path of a pair of nodes on the boundary surface of a network.

V. Simulations

A set of 3-D sensor networks with representative topologies and various sizes is simulated in this work as shown in Figs. 2 and 3. The genus numbers of these network models range from genus 0 to genus 2. One network model is particularly designed to have several inner holes (uncovered regions). We evaluate the performance of the proposed location-free cut-graph-based double-ruling scheme on these network models. Specifically, we denote the one given in Fig. 3(d) as Network I model. It is genus 0 with $4k$ number of nodes, and the average number of neighbors of each node 13.79. We denote the one given in Fig. 2(a) as Network II model. It is genus 1 with $4k$ number of nodes, and the average number of neighbors of each node 13.43. We denote the one given in Fig. 3(a) as Network III model. It is genus 2 with $6k$ number of nodes, and the average number of neighbors of each node 13.67. We denote the one given in Fig. 2(b) as Network IV model. It is genus 0 with two inner holes. The size of Network IV model is close to $8k$, and the average number of neighbors of each node is 13.7181.

Several parameters are particularly important for the performance of a distributed in-network data storage and retrieval scheme. One parameter is the cost to store data, and another parameter is the cost to retrieve data. We call them the producer and consumer costs measured by the number of hops traveled to store or retrieve data. In general, there is a balance between the two costs. If one distributed in-network data processing scheme favors data storage, a producer on average will spend less cost than a consumer, and the vice versa. For non-double-ruling-based distributed in-network data processing schemes, the number of data types generated inside a network and the frequency of the requests for aggregated data retrieval will greatly increase the consumer cost. Section V-A is designed to evaluate the costs of the producer and the consumer with different scenarios. Some scenarios that require frequent aggregated data retrieval are designed to favor double-ruling-based schemes, while some scenarios do not.

Traffic load is also a very important parameter to evaluate the performance of a distributed in-network data storage and retrieval scheme. We measure the traffic load on each node by the number of messages passing through it. A balanced traffic load is preferred; otherwise nodes with extremely high traffic load will run out of battery very soon. Section V-B is designed to evaluate traffic load with various scenarios. Similarly, some scenarios favor double-ruling-based schemes, while some scenarios do not.

It is desirable that a consumer spends a lower cost to retrieve data from a nearby producer than from a faraway one. Assume that the distance between a producer and a consumer measured by hops is d. A distance-sensitive retrieval scheme would require that the retrieval cost of the consumer is $O(d)$. Such distance-sensitive retrieval is discussed in Section V-C.

The motivation of double-ruling-based schemes is to trade storage cost for efficient and successful data retrieval, especially the aggregated data retrieval. To find the balance between storage cost and data retrieval, another parameter—the data storage cost—is evaluated in Section V-D.

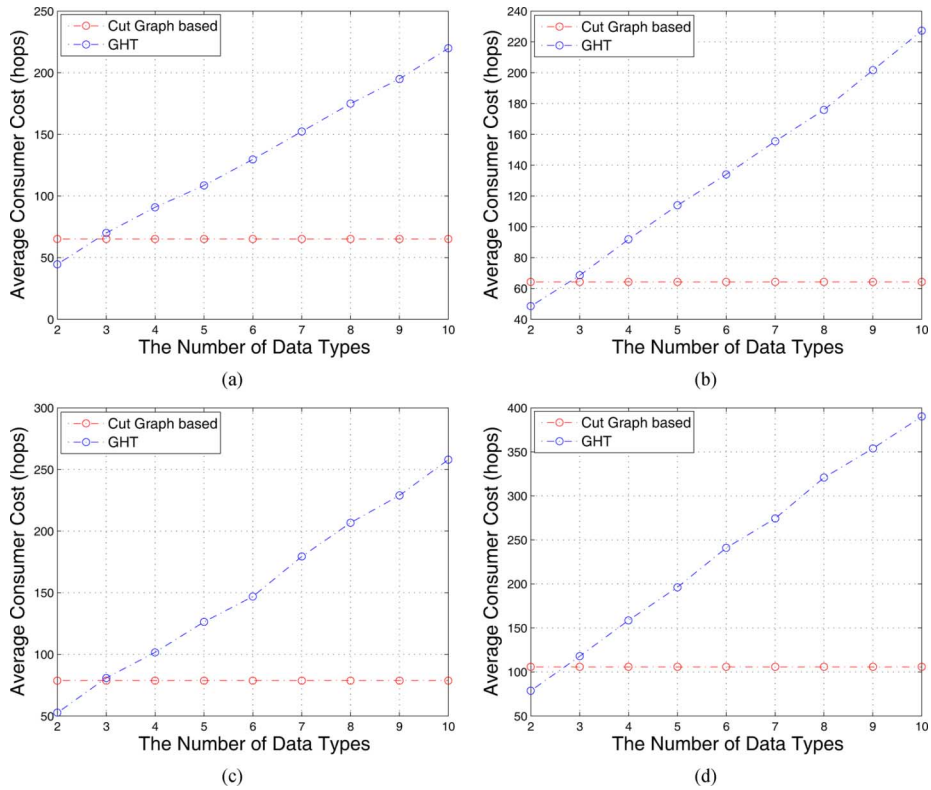

Fig. 7. Comparison of average consumer costs with the increase of data types in the network. (a) Network I. (b) Network II. (c) Network III. (d) Network IV.

TABLE I
COMPARISON OF AVERAGE PRODUCER AND CONSUMER COSTS OF SINGLE TYPE OF DATA

	Network I			Network II			Network III			Network IV		
	Cut Graph	GHT	SR-GHT	Cut Graph	GHT	SR-GHT	Cut Graph	GHT	SR-GHT	Cut Graph	GHT	SR-GHT
Producer cost	67. 6919	22.5738	19.4178	63.7369	23.6468	16.0400	78.6357	24.6926	13.7566	105.4353	41.0066	24.23068
Consumer cost	20.3451	22.8416	94.4715	19.3566	23.8186	93.0521	25.9980	24.7264	101.8150	28.2378	41.2531	157.83

Different choices of parameters when computing the cut graph in Section II and mapping the cut open boundary surface to a planar rectangle in Section III may lead to different producer and consumer costs for the same given network. We discuss and evaluate such possibility in Section V-E.

There are very limited algorithms with which to compare because all previous double-ruling-based schemes cannot work on 3-D sensor networks with general topology and geometry shapes including the naive 3-D grid-based double-ruling scheme and the volumetric parametrization-based double-ruling scheme [20], [21]. Hashing-based schemes can tolerate different topology but require geographic information. Our implementation of the GHT scheme for comparison has actually considered geographic information to design the hash function and stored heavy routing information on each node (shortest path tree rooted at each node) to guarantee the routing path a shortest one from the producer and the consumer to the hashed location, and hence all "improved GHT" approaches will not help to achieve better performance in our comparison.

A. Producer and Consumer Costs

1) Single Type of Data: We compare the proposed scheme to the GHT one with and without structured replication. For GHT with structured replication (SR-GHT), we apply 1 level hierarchy with extra three mirror points scattered in network to store the nearby data. Table I lists the average producer and consumer costs with one type of data generated in network. For cut-graph-based scheme, the producer cost is the highest, and the consumer cost is the lowest; a producer needs to travel and leave copies of data along the whole replication curve, while a consumer can stop immediately when its retrieval curve intersects the replication curve. For SR-GHT scheme, on the contrary, the producer cost is the lowest, and the consumer cost is the highest; a producer can store data at the closest location, but a consumer has to travel to both the hashed location and its three mirror points to collect data.

2) Aggregated Data: If there are more than one data type in network, as shown by Fig. 7, the consumer cost of cut-graph-based scheme is fixed; a consumer collects all different types of data by simply moving along a retrieval line. While the consumer cost of GHT scheme increases proportionally to the number of data types, a consumer has to travel to different hashed locations for different types of data. Note that the cost of GHT scheme may decrease because we simply take a round trip to each hashed location in our implementation. However, to find a minimum tour to visit all of the locations is the traveling salesman problem, which is NP-hard. The

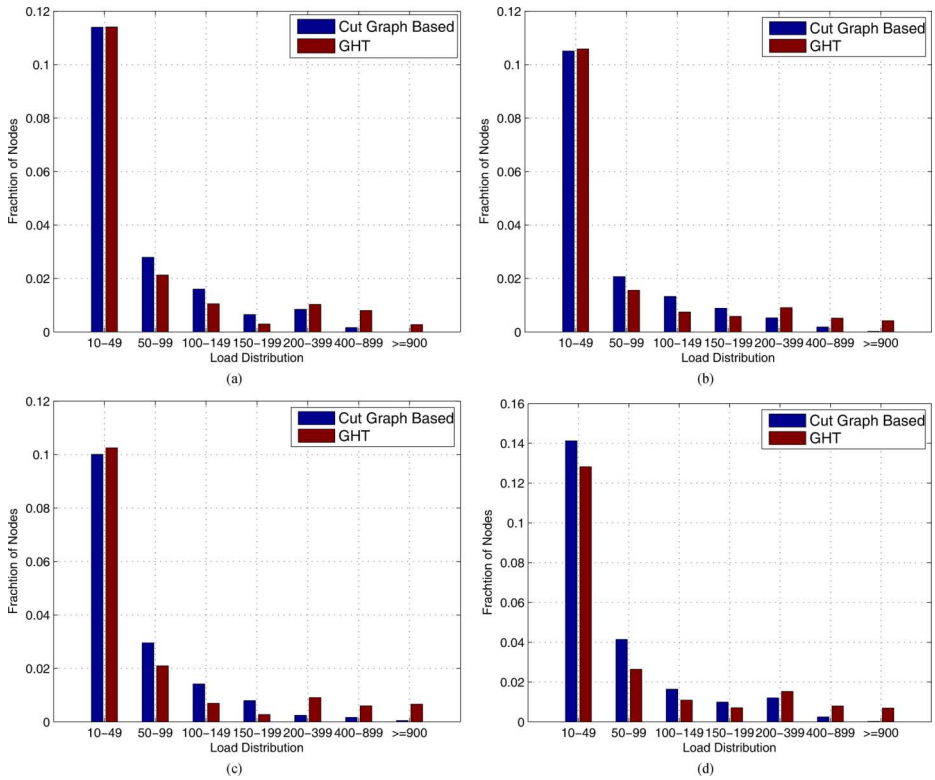

Fig. 8. Comparison of load distribution with one information producer and one data type in the network. (a) Network I. (b) Network II. (c) Network III. (d) Network IV.

producer cost does not change for either cut-graph-based and GHT schemes with the increase of data types.

Fig. 7 clearly shows that the cut-graph-based scheme performs the best for retrieval of multiple types of data generated in the network. When there is only one type of data in the network, cut-graph-based scheme and GHT schemes have a tradeoff between the producer and consumer costs. Meanwhile with the increases of data replication, cut-graph-based scheme has a better fault tolerance and a more balanced load distribution across the network as discussed in Section V-B. The length of the data replication curve of cut-graph-based scheme is bounded (Section VI).

B. Load Distribution

We simulate different scenarios to evaluate the load distribution of a cut-graph-based scheme and compare to GHT scheme. The first scenario is one data producer with one data type in a network. Each node in the network has equal probability to request for data. We randomly choose the data producer from the network and repeat the tests for 10 times and get the average. For both GHT and cut-graph-based schemes, the load on the majority of the nodes is within a small number. Specifically, the loads on roughly 83% of nodes in Network I, roughly 84% of nodes in Networks II and III, and roughly 79% of nodes in Network IV are below 10. Fig. 8 shows the distribution of high traffic load on the remaining nodes. For GHT scheme, nodes near the hashed location suffer much higher traffic load, while for cut-graph-based scheme, boundary nodes take a little bit more traffic load since the load has been evenly distributed among the boundary. The node suffering the highest traffic has a load of 4368 with GHT scheme and 813 with cut graph scheme for Network I; a load of 4297 with GHT scheme and 140 with cut graph scheme for Network II; a load of 6193 with GHT scheme and 981 with cut graph scheme for Network III: and a load of 14 077 with GHT scheme and 996 with cut graph scheme for Network IV.

The second scenario is 100 data producers with 10 data types in a network. We randomly choose the data producers from the network. Each node in the network has equal probability to request for aggregated data. Fig. 9 shows the distribution of the total traffic load of data storage and retrieval. For GHT scheme, a data consumer has to travel a long path to collect different types of data scattered in a network, which generates high traffic load in the network; while for cut-graph-based scheme, a data consumer has fixed cost for aggregated data retrieval so that the majority of the traffic load of the network is still low. The node suffering the highest traffic has a load of 10 211 with GHT scheme and 1542 with cut graph scheme for Network I; a load of 10 540 with GHT scheme and 1404 with cut graph scheme for Network II; a load of 22 935 with GHT scheme and 1614 with cut graph scheme for Network III: and a load of 48 074 with GHT scheme and 3778 with cut graph scheme for Network III.

C. Distance-Sensitive Retrieval

It is desirable that a consumer only needs to spend a low travel cost to retrieve data generated from a nearby producer. Such property is called distance sensitivity of retrieval. GHT-based schemes in general do not have such property because a consumer travels to a hashed position by the desired data type and the data stored there may be produced by a nearby or a faraway producer.

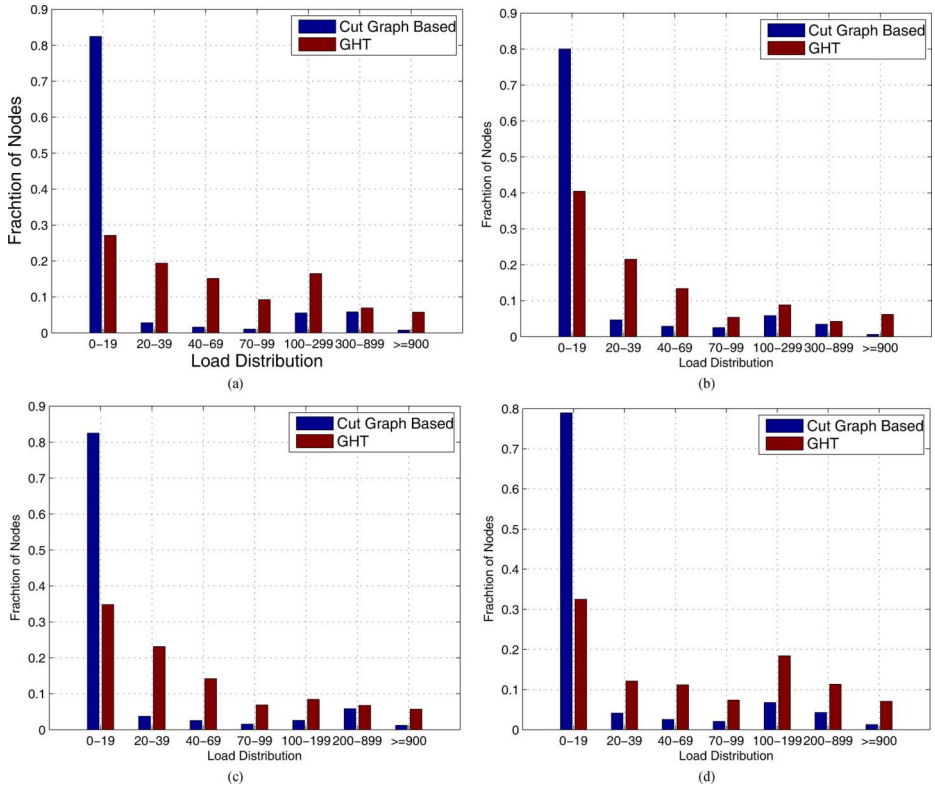

Fig. 9. Comparison of load distribution with 100 information producers and 10 data types in the network. (a) Network I. (b) Network II. (c) Network III. (d) Network IV.

For our scheme, we first consider the case of a pair of a consumer and a producer close on the boundary surface. They are not necessarily boundary nodes. If they are inner nodes, we measure the hops between the two boundary nodes nearest to the producer node and the consumer node, respectively. The following lemma says that if a pair of a consumer and a producer is close on the boundary surface, the consumer travels a distance less than the one to the producer on the boundary surface to access the data.

Lemma 1: The distance a consumer travels on the boundary surface of a network to hit the data replication curve of a producer is no more than the distance between the consumer and the producer on the boundary surface.

Proof: The mapping we conducted in Section III is conformal mapping. It is diffeomorphism preserving the neighborship of a boundary node and the relative positions of its neighbors when virtually mapped to a planar rectangle. The shortest path between a pair of a producer and a consumer on the boundary surfaces can be approximated by a straight line between their nearest boundary nodes on the planar rectangle. A data replication curve is a straight line passing through the boundary node nearest to the producer on plane. The distance between the boundary node nearest to the consumer and the data replication curve is always less than the distance between the pair of the boundary nodes. The two distances equal each other only when the line between the pair of the boundary nodes is perpendicular to the data replication curve.

Two boundary nodes, one nearest to a producer and the other nearest to a consumer, may be close on the boundary surface but far away on the planar rectangle if they are located exactly on the two sides of the cut graph. Such a case will not affect the distance the consumer travels to retrieve the data because the boundary surface is only virtually cut open and mapped to a planar rectangle. We design the data retrieval scheme as once a consumer hits the boundary side of the rectangle; the consumer can cross the boundary side and keep traveling along the same line with the same direction. ■

We randomly select 1000 pairs of producers and consumers on each network model. Fig. 10(a) shows the simulation results based on the distances of the pairs of producers and consumers on the boundary surface of a network. It is obvious that the average consumer cost increases with the distance between the producer and the consumer on the boundary surface of a network. Such a trend does not always hold when the distance of a pair of a consumer and a producer is large. They may be far away on the boundary surface, but the consumer is still close to the data replication curve.

However, if we consider the distance between a pair of a producer and a consumer in the volume of a network, such distance-sensitive retrieval is not necessarily applied. It is possible that a consumer is geographically close to a producer in the 3-D volume of a network, but the consumer has to go a long distance to hit the data replication curve by the producer. Such an extreme case happens when the consumer and the producer are located very close to each other but at the two sides of the medial axis of the 3-D volume network,[3] so their nearest boundary nodes are far away from each other on the

[3]The medial axis of a shape is the set of all points that have more than one closest point to the boundary of the shape.

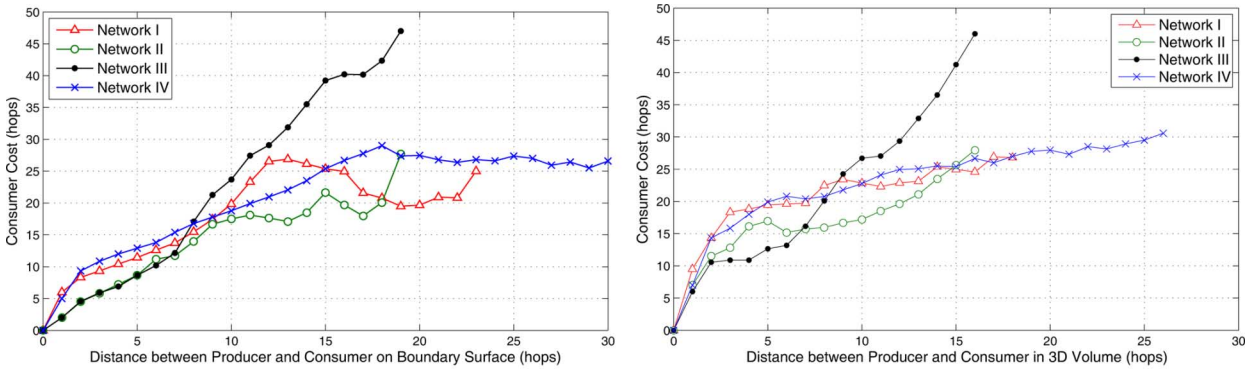

Fig. 10. (a) Consumer costs versus distance between producer and consumer on boundary surface of a network. (b) Consumer costs versus distance between producer and consumer in 3-D volume of a network. (a) Distance on boundary surface of a network. (b) Distance in 3-D volume of a network.

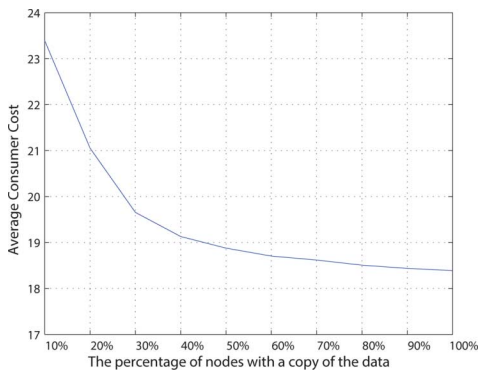

Fig. 11. Tradeoff between storage cost and consumer cost.

boundary surface of the network. We still randomly choose 1000 pairs of producers and consumers on each network model. Fig. 10(b) shows the simulation results based on the distances of the pairs of producers and consumers in the 3-D volume of a network. The average consumer cost does not show a strong linear correlation with the distance.

D. Tradeoff Between Storage Cost and Consumer Cost

As discussed in Section IV, it is a tradeoff between the storage cost and the consumer cost for the proposed cut-graph-based double-ruling approach for information storage and retrieval in general 3-D sensor networks. Fig. 11 shows clearly that the average consumer cost drops dramatically when the percentage of nodes stored with a copy of the data along the data replication curve increases from 10% to 40%, and then decreases slowly when the percentage of nodes stored with a copy of the data is over 50% for Network II model. A balance between the storage cost and the consumer cost would be to store copies of data at half of the nodes along the data replication curve.

E. Different Choices of Parameters

As we introduced in Section II, the algorithm to compute the cut graph starts from one randomly chosen triangle of the triangular boundary surface of a network. The algorithm marks the triangle and keeps a width-first propagation with the center of the first marked triangle until all triangles of the network have been marked. We assume the algorithm always picks the triangle

TABLE II
EVALUATION OF THE PERFORMANCE UNDER DIFFERENT CHOICES OF PARAMETERS

	Starting Triangle		Corner Nodes	
	Producer cost	Consumer cost	Producer cost	Consumer cost
μ	58.2514	20.7714	61.7551	21.3348
σ	10.9340	4.0029	14.691962	4.7569
$\tilde{x}$	57.2104	20.2214	57.7746	20.9780

with the smallest boundary node ID as the starting triangle in our previous simulations. To evaluate the effect of the choice of the starting triangle, we first randomly select 1000 pairs of producers and consumers from Network II model. We randomly pick a triangle of the boundary surface as the starting triangle to compute the cut graph, and then compute the average producer and consumer costs of the chosen 1000 pairs. The testing is repeated 10 times with the same set of pairs of producer and consumer but 10 different triangles as the starting triangle. Denote μ the average producer and consumer costs of the 10 tests, σ the standard deviation, and $\tilde{x}$ the median. Table II shows the mean, the standard deviation, and the median of the producer and consumer costs of the 10 tests under different starting triangles.

After we cut the closed boundary surface open to a topological disk along the computed cut graph, we uniformly pick four nodes along the boundary of the topological disk and map the topological disk to a planar rectangle with four corner points exactly of the four chosen nodes in Section III. We assume the algorithm always chooses the node with the smallest ID along the boundary of the topological disk as the first corner node and then uniformly chooses the other three in our previous simulations. To evaluate the effect of the choice of the four corner nodes, similarly, we first randomly select 1000 pairs of producers and consumers from Network II model. We randomly pick a node along the boundary of the topological disk as the first corner node and then uniformly select the other three to do the mapping. We compute the average producer and consumer costs of the chosen 1000 pairs. The testing is repeated 10 times with the same set of pairs of producers and consumers, but 10 different sets of corner nodes when mapping the cut open boundary surface to a planar rectangle. Table II shows the mean, the standard

deviation, and the median of the producer and consumer costs of the 10 tests under different sets of corner nodes.

In summary, the different choices of the parameters slightly affect the performance of the proposed cut-graph-based double-ruling scheme.

VI. Discussions

A. Network Model

The proposed solution does not require any global position information of a network. It has no constraints on communication models of the network either. The network only requires one-hop neighborhood distance information to detect boundary nodes and construct a triangular structure at the preprocessing step.

The proposed cut-graph computation algorithm is independent of the complexity and irregularity of a 3-D volume where a set of sensors is deployed because the boundary surface of a 3-D volume is always a closed surface. The algorithm cuts a closed surface with any geometric shape or topology to a topological disk and then virtually maps it to a planar rectangle. It is possible that a 3-D network degenerates. One example is that a 3-D volume sensor network degenerates to a 3-D surface network. The algorithm can simply be applied to the surface network directly, which is not necessarily closed. Another example is that part of a 3-D network degenerates to a single line, neither a volume nor a surface. We can apply 3-D network segmentation algorithm [36] to identify the bottleneck and then segment the network to parts. Double-ruling approach can be applied at individual parts.

B. Network Dynamics

As discussed in Section IV-E, both the time complexity and communication cost of the proposed cut-graph-based double-ruling approach are dominated by computing the planar rectangle virtual coordinates of the boundary surface using discrete surface Ricci flow. For a network with possible nodes' failures, we do not need to restart the computation of Ricci flow each time a sensor node runs out of its battery. We only need to replace a dead boundary node with its nearest active sensor node. The process can be triggered by nodes with dead communication to one common node. They conduct a local flooding to find one node nearest to the dead one. Note that this new one is not necessarily a boundary node. Denote this new node v_i. v_i initializes its $\gamma_i = 1, u_i = \log \gamma_i = 0$. v_i and its direct neighbors recompute the weights of edges neighboring to v_i. Discrete surface Ricci flow continues until the convergence.

C. Network Density

As we mentioned in Section II, the algorithm provided in [24] to construct a triangular boundary surface of a network assumes that a triangular graph is a subgraph of the initial connectivity graph of detected boundary nodes. Such assumption is true only when the node density of the network is not too low. The node densities of the network models we choose in simulations are all around 13. We have no problem to construct such triangular surfaces. In our previous simulations, however, such assumption may not be true when the node density of a network is around or lower than 8.

VII. Conclusion

We present a location-free cut-graph-based double-ruling approach for 3-D sensor networks with general topology and geometry shapes. An information consumer simply travels along a simple curve with the guaranteed success to retrieve aggregated data through time and space with different types across the network. We conduct extensive simulations and comparisons that further show the proposed approach with low cost and a balanced traffic load.

References

[1] I. Stojmenovic and B. Vukojevic, "A routing strategy and quorum based location update scheme for ad hoc wireless networks," University of Ottawa, Ottawa, ON, Canada, Tech. Rep., 1999.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. MobiCom*, 2000, pp. 56–67.

[3] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A geographic hash table for data-centric storage in sensornets," in *Proc. 1st ACM Workshop Wireless Sensor Netw. Appl.*, 2002, pp. 78–87.

[4] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *Proc. ACM MobiCom*, 2002, pp. 148–159.

[5] D. Braginsky and D. Estrin, "Rumor routing algorithm for sensor networks," in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw. Appl.*, 2002, pp. 22–31.

[6] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker, "DIFS: A distributed index for features in sensor networks," in *Proc. 1st IEEE Int. Workshop Sensor Netw. Protocols Appl.*, 2003, pp. 163–173.

[7] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proc. 1st SenSys*, 2003, pp. 63–75.

[8] J. Gao, L. J. Guibas, J. Hershberger, and L. Zhang, "Fractionally cascaded information in a sensor network," in *Proc. 3rd Int. Symp. Inf. Process. Sensor Netw.*, 2004, pp. 311–319.

[9] Q. Fang, J. Gao, and L. J. Guibas, "Landmark-based information storage and retrieval in sensor networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.

[10] R. Sarkar, X. Zhu, and J. Gao, "Double rulings for information brokerage in sensor networks," in *Proc. ACM MobiCom*, 2006, pp. 286–297.

[11] R. Sarkar, W. Zeng, J. Gao, and X. D. Gu, "Covering space for in-network sensor data storage," in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2010, pp. 232–243.

[12] J. Allred *et al.*, "Sensorflock: An airborne wireless sensor network of micro-air vehicles," in *Proc. 5th SenSys*, 2007, pp. 117–129.

[13] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "The challenges of uilding scalable mobile underwater wireless sensor networks for aquatic applications," *IEEE Netw.*, vol. 20, no. 3, pp. 12–18, May–Jun. 2006.

[14] F. Bian, R. Govindan, S. Shenker, and X. Li, "Using hierarchical location names for scalable routing and rendezvous in wireless sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 305–306.

[15] F. Araújo, J. Kaiser, C. Mitidieri, L. Rodrigues, and C. Liu, "Chr: A distributed hash table for wireless ad hoc networks," in *Proc. Int. Conf. Distrib. Comput. Syst. Workshops*, 2005, vol. 4, pp. 407–413.

[16] M. Albano, S. Chessa, F. Nidito, and S. Pelagatti, "Data centric storage in non-uniform sensor networks," in *Proc. 2nd INGRID*, 2007.

[17] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. ACM MobiCom*, 2000, pp. 243–254.

[18] X. Liu, Q. Huang, and Y. Zhang, "Balancing push and pull for efficient information discovery in large-scale sensor networks," *IEEE Trans. Mobile Comput.*, vol. 6, pp. 241–251, Mar. 2007.

[19] S. Funke and I. Rauf, "Information brokerage via location-free double rulings," in *Proc. 6th Int. Conf. Ad-Hoc, Mobile Wireless Netw.*, 2007, pp. 87–100.

[20] J. Luo, F. Li, and Y. He, "3DQS: Distributed data access in 3D wireless sensor networks," in *Proc. IEEE ICC*, 2011, pp. 1–5.

[21] C. Zhang, J. Luo, L. Xiang, F. Li, J. Lin, and Y. He, "Harmonic quorum systems: Data management in 2D/3D wireless sensor networks with holes," in *Proc. IEEE SECON*, 2012, pp. 1–9.
[22] J. Munkres, *Topology*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.
[23] H. Zhou, S. Xia, M. Jin, and H. Wu, "Localized algorithm for precise boundary detection in 3D wireless networks," in *Proc. IEEE ICDCS*, 2010, pp. 744–753.
[24] 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 *Proc. IEEE INFOCOM*, 2011, pp. 1053–1061.
[25] A. Hatcher, *Algebraic Topology*. Cambridge, U.K.: Cambridge Univ. Press, 2001.
[26] J. Erickson and S. Har-Peled, "Optimally cutting a surface into a disk," *Discrete Comput. Geom.*, vol. 31, no. 1, pp. 37–59, 2004.
[27] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust, "Computing a canonical polygonal schema of an orientable triangulated surface," in *Proc. 17th Annu. Symp. Comput. Geom.*, 2001, pp. 80–89.
[28] J. Erickson and K. Whittlesey, "Greedy optimal homotopy and homology generators," in *Proc. 16th Annu. ACM-SIAM Symp. Discrete Algor.*, 2005, pp. 1038–1046.
[29] ?. C. De Verdière, "Shortest cut graph of a surface with prescribed vertex set," in *Proc. 18th Annu. Eur. Conf. Algor., Part II*, 2010, pp. 100–111.
[30] T. K. Dey, "A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection," in *Proc. 10th Annu. Symp. Comput. Geom.*, 1994, pp. 277–284.
[31] W. P. Thurston, *Geometry and Topology of Three-Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 1976, Lecture notes.
[32] B. Chow and F. Luo, "Combinatorial Ricci flows on surfaces," *J. Differential Geom.*, vol. 63, no. 1, pp. 97–129, 2003.
[33] M. Jin, J. Kim, F. Luo, and X. Gu, "Discrete surface Ricci flow," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 5, pp. 1030–1043, Sep.–Oct. 2008.
[34] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in *Proc. IPDPS, Workshop Parallel Distrib. Comput. Issues Wireless Netw. Mobile Comput.*, 2001, pp. 1–6.
[35] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 138–149.
[36] H. Zhou, N. Ding, M. Jin, S. Xia, and H. Wu, "Distributed algorithms for bottleneck identification and segmentation in 3D wireless sensor networks," in *Proc. 8th Annu. IEEE SECON*, 2011, pp. 494–502.

Yang Yang received the B.S. degree in computer science from Northwestern Polytechnical University, Xi'an, China in 2009, and the M.S. and Ph.D. degrees in computer sciences from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 2011 and 2014, respectively.

His research focuses on designing geometric algorithms for wireless sensor networks in both areas of in-network information processing and localization.

Miao Jin received the B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2000, and the M.S. and Ph.D. degrees from the State University of New York at Stony Brook, Stony Brook, NY, USA, in 2006 and 2008, respectively, all in computer science.

She has been an Assistant Professor with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA, USA, since Fall 2008. Her research results have been used as cover images of mathematics books and licensed by Siemens Healthcare Sector of Germany for virtual colonoscopy. Her research interests are computational geometric and topological algorithms with applications in wireless sensor networks, computer graphics, computer vision, geometric modeling, and medical imaging.

Dr. Jin received the NSF CAREER Award in 2011.

Yao Zhao received the bachelor's degree in automatic control from Zhejiang University, Hangzhou, China, in 2008, and the master's degree in computer science from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 2011, and is currently pursuing the Ph.D. degree in computer science, which is to be awarded in May 2014.

He worked as a full-time Researcher with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, from 2008 to 2012. He now works as a member of the Research and Development Department, Epic Systems Corporation, Verona, WI, USA. His research interest focused on automatic localization in sensor networks, especially on 3-D surfaces.

Hongyi Wu received the B.S. degree in scientific instruments from Zhejiang University, Hangzhou, China, in 1996, and the M.S. degree in electrical engineering and Ph.D. degree in computer science from the State University of New York (SUNY) at Buffalo, Buffalo, NY, USA, in 2000 and 2002, respectively.

Since then, he has been with the Center for Advanced Computer Studies (CACS), University of Louisiana at Lafayette (UL Lafayette), Lafayette, LA, USA, where he is now a Professor and holds the Alfred and Helen Lamson Endowed Professorship in Computer Science. His research spans delay-tolerant networks, radio frequency identification (RFID) systems, wireless sensor networks, and integrated heterogeneous wireless systems.

Prof. Wu received the NSF CAREER Award in 2004 and the UL Lafayette Distinguished Professor Award in 2011.