On the Distance-Sensitive and Load-Balanced Information Storage and Retrieval for 3D Sensor Networks

Wenping Liu, *Member, IEEE*, Hongbo Jiang, *Senior Member, IEEE*, Jiangchuan Liu, *Senior Member, IEEE*, Xiaofei Liao, *Member, IEEE*, Hongzhi Lin, and Tianping Deng, *Member, IEEE*

***Abstract*—Efficient in-network information storage and retrieval is of paramount importance to sensor networks and has attracted a large number of studies while most of them focus on 2D fields. In this paper, we propose novel Reeb graph based information storage and retrieval schemes for 3D sensor networks. The key is to extract the line-like skeleton from the Reeb graph of a network, based on which two distance-sensitive information storage and retrieval schemes are developed: one devoted to shorter retrieval path and the other devoted to more balanced load. Desirably, the proposed algorithms have no reliance on the geographic location or boundary information, and have no constraint on the network shape or communication graph. The extensive simulations also show their efficiency in terms of sensor storage load and retrieval path length.**

***Index Terms*—3D sensor networks, complex-connected 3D settings, information retrieval, information storage.**

I. Introduction

WIRELESS sensor networks have emerged as an enabling technology for many applications. Among them, we focus on the kind of applications where sensor nodes perform real-time monitoring or tracking, and delivering the generated data to interested users. Despite the large amount of data collected by sensor networks, often the high-level information such as a semantic event report (e.g., the presence of toxic substances, or the current location of the giraffes in a national park, etc.) which is collaboratively aggregated by multiple sensors is more meaningful and of interest. At the same time, the distributed users could pose queries into the network at any time requesting such information, anticipating the results with low access delay and traffic cost. As such, efficient in-network data storage and retrieval is of paramount importance and has attracted a large number of studies [5], [6], [21]–[25], [28], [29]. Most of the studies, however, focus on 2D sensor field, despite the fact that, more recently, there are growing interests for studies on 3D sensor networks [1], [9], [16], [17], [28], [29]. For example, with the advances in vehicle technology, acoustic and optical underwater communication, etc., many underwater/underground monitoring applications, e.g., mine reconnaissance [15], come into reality with cost effectiveness [1], [2].

It it noted that, a desirable information storage and retrieval scheme should be *distance-sensitive* [6], [23] or locality-aware [5]. That is, if the user requesting information (or so-called *consumer*) is of distance d from one source maintaining a copy of information (or so-called *producer*), the consumer is able to retrieve a replication at a cost of $O(d)$ without reliance on the consumer/producer location information. This property is of our great interest since it implies that the consumer travels along the path hitting a replication as soon as possible, thereby introducing low access delay and traffic cost. Please see Fig. 1. Nonetheless, it is not a trivial task and only a few of existing schemes [23], [24] are distance-sensitive. In addition to distance-sensitivity, the *balanced storage load* [5], [23] also makes an information storage and retrieval scheme more practical in that the nodes are supposed to be less powerful with limited storage capacity in sensor networks. Previous studies envision that an uneven storage load distribution will eventually cause high storage cost and traffic cost on a small subset of nodes, thereby consuming their energy quickly.

In literature, however, geographical hash tables (GHT) [21], [31] based schemes are distance-insensitive, possibly resulting in a large traffic cost and retrieval (thus the in-network aggregation for semantic reports) delay, and double-ruling schemes [5], [23], [24] are distance-sensitive and retrieval-guaranteed, yet they are primarily targeted for 2D sensor networks, and the naive extension of double-ruling scheme where the replication plane and retrieval plane are utilized only works in regular-shaped 3D networks. To that end, Yang *et al.* [28],

Manuscript received June 30, 2015; revised November 15, 2015; accepted January 18, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P.-J. Wan. Date of publication February 08, 2016; date of current version December 15, 2016. This work was supported in part by the National Natural Science Foundation of China under Grants 61202460, 61271226, 61272408, 61322210, 61572219, and 61502192; the China Postdoctoral Science Foundation under Grant 2014M552044; the China Scholarship Council (CSC) under Grant 201308420188; the Natural Science Foundation of Hubei Province under Grant 2014CFA040; the Science and Technology Plan Projects of Wuhan City under Grant 2015010101010022; and the Fundamental Research Funds for the Central Universities under Grant 2015QN073. *(Corresponding author: Hongbo Jiang.)*

W. Liu is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Hubei University of Economics, Wuhan 430205, China.

H. Jiang, H. Lin, and T. Deng are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hongbojiang2004@gmail.com).

J. Liu is with the School of Computing Science, Simon Fraser University, BC V5A1S6, Canada.

X. Liao is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China.

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.2016.2523242

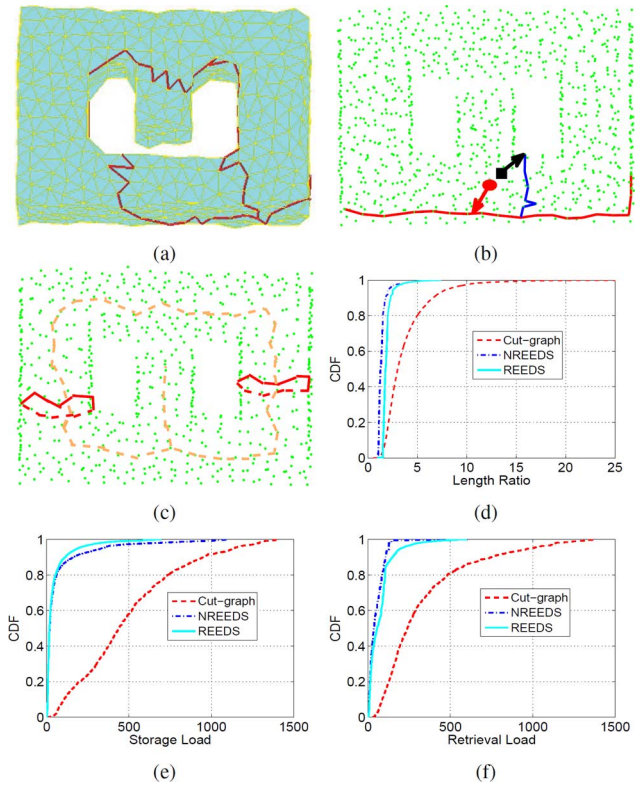


Fig. 1. Illustrative examples on a genus-1 3D sensor network with one concave *tunnel*, e.g., deployed underwater surrounding an island. (a) The cut graph obtained by [28], [29]; (b) The replication curve (in red) of the producer (shown by the solid eclipse) and the retrieval curve (in blue) of the consumer (shown by the solid rectangle) based on the cut graph in (a); (c) Two minimum cuts (indicated by the red polygons) are identified to construct the Reeb graph, which are then used to extract the line-like skeleton (shown by the dashed curve) and decompose the network into a set of level sets where the sizes of level sets are almost balanced. The line-like skeleton intersects with each level set, and the construction of the short retrieval path between two nearby nodes can be guided by the line-like skeleton and the feature function of Reeb graph; (d), (e), and (f) show the CDFs of retrieval path length ratio, storage load and retrieval load, respectively.

[29] proposed to cut open the closed boundary surface of an arbitrary shaped 3D sensor networks to a topological disk along the cut graph (see Fig. 1(a)), followed by mapping it to an aligned planar rectangle such that double-ruling scheme is applicable on boundary surface. Even though the cut-graph based scheme [28], [29] is retrieval-guaranteed, we emphasize that it is not distance-sensitive, as shown in Fig. 1(b). Fig. 1(d) depicts the cumulative distribution function (CDF) of the ratio (so-called *retrieval path length ratio*) of retrieval path length to the shortest path length between each of 10,000 randomly selected pairs of consumers and producers. Besides, due to the frequent usage of boundary nodes for data replication and retrieval, the boundary nodes will be inevitably overloaded, as illustrated in Figs. 1(e) and 1(f).

Pinpointing that the drawbacks of the cut-graph based scheme in [28], [29] stem from the fact that it is merely based on the cut graph of the boundary surface, and the topological information of the interior of the 3D network is totally ignored, in this paper, we strive to design distance-sensitive and load-balanced information storage and retrieve schemes for 3D sensor networks. In contrast to existing schemes, this work takes advantage of a powerful tool—the Reeb graph [20], which is a global abstraction and has the capability of encoding the underlying network topology [8]. Also we are aware that the Reeb graph can be used for line-like skeleton extraction [7], an important infrastructure to reflect the major topological features of the network [16], [17]. It is noted that the line-like skeleton intersects each level set at a skeleton point within that level set, mimicking the retrieval and replication curve in 2D double-ruling base schemes, and two nearby points fall within either the same level set, or two adjacent level sets. *Hence, on top of the Reeb graph (i.e., the level sets) and guided by the skeleton, we can offer retrieval-guaranteed, distance-sensitive and load-balanced data storage/retrieval schemes for 3D sensor networks with an arbitrary shape*.

Note that existing boundary recognition algorithms either require high node density [32], uniformly distributed network [10], [11], or location/distance information [4], [33], which potentially limit their applicability in practice; at the same time, the geographical location information of each node is costly to obtain. As such, in this paper, we first propose to extract the Reeb graph without reliance on boundary and location information. Specifically, we identify the minimum *cut(s)*,[1] as shown in Fig. 1(c), and regard the minimum distance (in hops) of a node to the cut(s) as the feature function such that the nodes having the same distance to the cut(s) within a connected component form a level set, and accordingly the Reeb graph is generated. On top of the Reeb graph, the line-like skeleton can be readily extracted.

With the computed Reeb graph and line-like skeleton, we present two Reeb graph based DCS schemes for 3D sensor networks, entitled *NREEDS* and *REEDS* . In NREEDS, the producer replicates its sensed data in the hashed level set given a hash function, which all nodes are assumed to know, of a certain content attribute, under guide of the Reeb graph and skeleton. In addition, we allow the nodes of the producer's resident level set (referred to as home level set) and the skeleton nodes on the way of replication to store a pointer indicating where the data is originated, such that the retrieval path will intersect the replication path as early as possible, and thus the distance-sensitivity is ensured. The consumer then follows the direction parallel to the line-like skeleton to retrieve data of interest, either from the home or hashed level set, depending on which one is nearer to the consumer. Interestingly, the double-ruling scheme is a special case of NREEDS in 2D settings. Please see Fig. 2.

On the other hand, in REEDS, the producer replicates the data only in a subset of the hashed level set. More specifically, within each level set, a shortest path tree is constructed rooted at the line-skeleton node. Then, the producer replicates its sensed data within the least-loaded sub-tree *rooted* at a child of the line-like skeleton node, of the hashed level set. Similar with NREEDS, the nodes in home level set and the skeleton nodes on the way of replication also store a pointer. To fetch the interested data, the consumer follows the direction parallel to the line-like skeleton until it arrives the home or hashed level set, and moves directly to the producer guided by the pointer of the nodes in the home level set or travels along a *circle* to the targeted sub-tree

[1]If the network is genus-0, only one minimum cut is needed. Each cut corresponds to a level set in the Reeb graph.

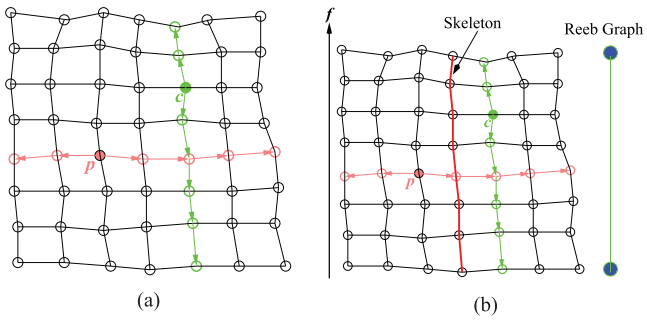


Fig. 2. Double-ruling based storage and retrieval in a 2D sensor network. (a) The producer p leaves the copies of its data along the horizontal replication curve, and the consumer c retrieves the data along the vertical retrieval curve; (b) The double ruling based scheme is the special case of our algorithm in 2D environments where each horizontal curve corresponds to a level set of f, the collection of the centers of horizontal curves serve as the skeleton, and each retrieval path is nearly *parallel* to the skeleton.

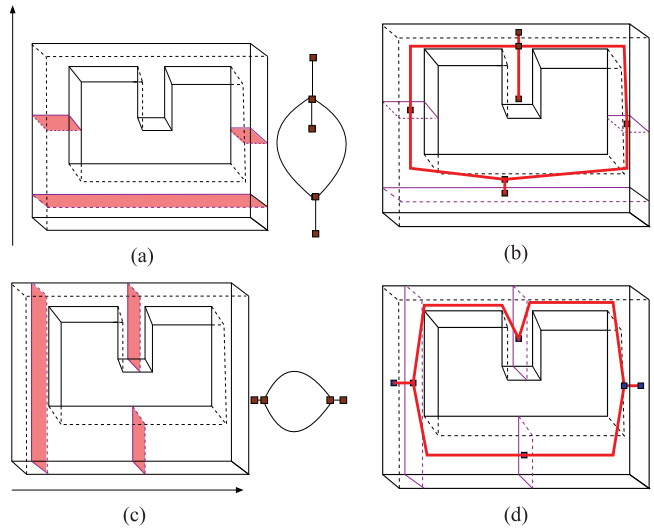


Fig. 3. Illustrative examples on a genus-1 3D object in continuous domain. The *tunnel* is mapped to a loop in the Reeb graph, where critical points are marked as solid rectangles. (a), (c) The Reeb graph w.r.t height function $f = z(x, y)$ and $f = y(x, z)$, respectively, where the shaded areas represent the level sets; (b), (d) The line-like skeleton (indicated by the thick curve) based on the Reeb graph in (a) and (c), respectively.

in the hashed level set. Clearly, the retrieval path by REEDS could be longer than NREEDS, while on the up side the storage overhead by REEDS can be much less. That is, REEDS and NREEDS provide a tradeoff between the retrieval path length ratio (thus the time cost) and storage overhead. Desirably, both NREEDS and REEDS are distance-sensitive and load-balanced as compared with the scheme in [28], [29]. Further, they do not rely on the boundary information as the scheme in [28], [29] does, and thus have wider applicability.

The rest of the paper is organized as follows. In Section II, we introduce the motivations and preliminaries of the paper, and detail the algorithms in Section III. We validate the proposed two algorithms via extensive simulations in Section IV, followed by briefly introducing the related work in Section V. Finally, we conclude the paper in Section VI.

II. Motivations and Preliminaries

The Reeb graph of an object $D \subset R^3$ has been envisioned as a powerful tool for topology-based shape matching [3], and line-like skeleton extraction [7], etc., in computer graphics and computational geometry. In continuous domain, given a smooth function $f : D \to R$ (a.k.a *feature function*) defined on object D, the Reeb graph of D is generated by continuously contracting to a point each connected component of level sets, i.e., the set of points with equal feature function value. In Reeb graph, the vertices are critical points of function f (i.e., minima, saddles and maxima) and the edges are the arcs connecting two adjacent critical points. A loop in the Reeb graph indicates a tunnel, or hole, of the underlying object, and accordingly, the Reeb graph of a genus-n object has n loops. Please see Figs. 3(a)–3(c) for some intuition. As mentioned earlier, the Reeb graph can be used for line-like skeleton extraction, where each line-like skeleton point is approximately computed as the center of a level set, which implies that the derived line-like skeleton intersects with each level set. Thus, in 3D sensor networks, if a sensor node replicates its sensed data within (a part of) the level set, and the consumer retrieves the data along the direction almost parallel to the line-like skeleton as in [16], [17], the retrieve can be guaranteed. Meanwhile, when the producer and the consumer are nearby, the retrieval path will not be very long. Further, if the Reeb graph is properly constructed such that the sizes of the level sets are roughly the same, the storage load can be balanced. Hence, based on the Reeb graph, we can design a load-balanced, retrieve-guaranteed, and distance-sensitive data storage and retrieval protocol, which is the goal of this paper.

To achieve this goal, the key is to correctly construct the Reeb graph. This is not straightforward since traditionally, the Reeb graph is pose dependent. That is, it will change with the *rotation* of the object, i.e., with different feature functions, and the line-like skeleton extracted based on different feature functions can be very different (see Figs. 3(b)–3(d)). Especially, when applied for data storage, the constructed Reeb graph should better not incur large level sets, otherwise the storage load will be imbalanced. For instance, in Figs. 3(a)–3(c), there are some level sets with relatively large areas, and accordingly, the nodes corresponding to these level sets will be overloaded. As such, the selection of the feature function f is of crucial importance to understand and represent the topology of the object, and to maintain a balanced storage load. This will be much more challenging in sensor networks since the boundary and geographic location information are often unknown.

Our approach is to identify the *minimum cut(s)* of the underlying 3D network. Visually, a minimum cut is a level set with a small volume (in continuous domain, e.g., the level sets in the two *wings* of Fig. 3(a)) or node number (in sensor networks). If we select the distance of a node to the minimum cut(s) as the feature function, the volume (or size) of the level sets will be evenly distributed (shown in Fig. 4(a)) and thus the imbalance of storage load in 3D sensor networks can be alleviated.[2] We first formally introduce the definition of the cut and the weight

[2]One might argue that the feature function can be simply regarded as the distance of each point to an arbitrarily selected point. This method, however, as one can image, will result in an unevenly distributed level set size, and consequently, an imbalance of storage load.

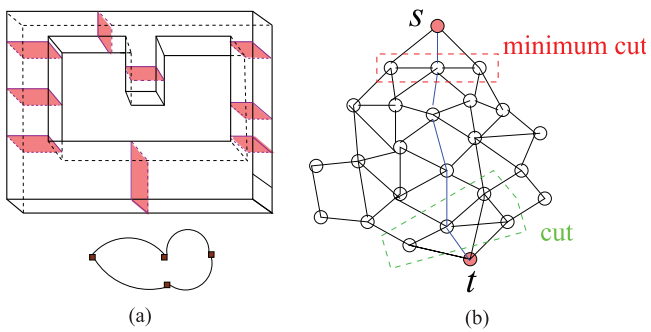


Fig. 4. (a) An illustration in continuous domain. The minimum cut based Reeb graph (lower part) in continuous domain. Two minimum cuts in the two wings of Fig. 3(a) are used to construct the feature function; (b) The cut (bounded by the green dashed polygon) and the minimum cut (bounded by the red dashed rectangle) between source s and destination t.

of a cut, followed by the formulation of *minimum cut problem* (MCP).

Definition 1: Let $G = (V, E)$ be a graph where V denotes the set of vertices and E is the set of edges. Given two *terminal nodes*, i.e., the source s and the destination t, the *cut* (s,t) between s and t is a partition of G into two disjoint parts $\mathcal{S}, \mathcal{T}$ such that $s \in \mathcal{S}, t \in \mathcal{T}$.

Clearly, the cut between s and t is by no means unique: there can be an arbitrarily large number of cuts *separating* two terminals into mutually exclusive parts. In this paper, we propose to identify the so-called minimum cut between pairwise terminals as it is often unique or easy to identify. We denote by $(s,t)_i$ the ith cut and $\mathcal{C}(s,t)$ the set of cuts $(s,t)_i$, i.e., $\mathcal{C}(s,t) = \{(s,t)_i, i = 1, 2, \ldots, K\}$ where K is the number of cuts.

Definition 2: The weight of cut $(s,t)_i$, denoted by $w(s,t)_i$, is the sum of the weights of boundary edges (e.g., the length of boundary edge) in $(s,t)_i$.

Definition 3: Given two terminal nodes s, t of graph G, the minimum cut problem is to identify a cut $(s,t)_k (1 \leq k \leq K)$ such that $w(s,t)_k = \min\{w(s,t)_i | (s,t)_i \in \mathcal{C}(s,t)\}$.

Fig. 4 depicts a cut and the minimum cut between s and t. Obviously, for a genus-0 3D object, i.e., without tunnels or holes, one minimum cut is enough to *block* the terminals; and for an object with n tunnels, there are $n + 1$ minimum cuts. However, the minimum cut identification is rather challenging since MCP cannot be solved in a linear time [26], and in sensor networks, the geographical location information and boundary information are often both unknown. To tackle this challenging problem, in this research, we will present a lightweight and distributed algorithm to approximate the minimum cut(s), which will be detailed in Section III.

With the identified minimum cut(s), each node computes its hop count distance to the nearest minimum cut. The distance function thus serves as a feature function f to establish a Reeb graph. That is, the vertices are the critical nodes, i.e., the minimum, maximum or saddles, of f, and the edges link two adjacent critical nodes; the nodes in a connected component with the same feature function form a *level set*. Note that the nodes in the minimum cut also form a level set since their feature function values are all defined to be zero.

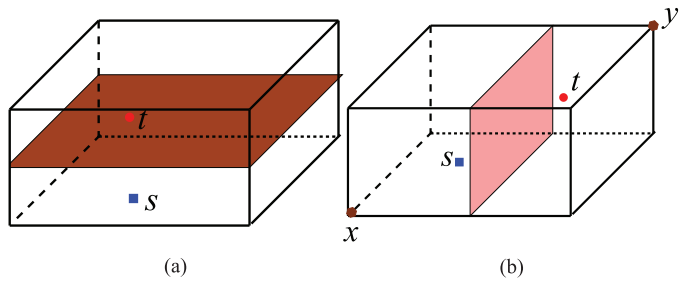


Fig. 5. An illustration of terminals and the minimum cut in a cubic. The shaded areas represent the minimum cut between the terminals. (a) The improperly selected terminals s, t and the minimum cut; (b) The right terminals s, t on the shortest path between two farthest nodes x, y and the minimum cut.

As in a genus-0 network, the geodesic shortest paths between the nearest boundary nodes of a line-like skeleton point decompose the network into more than one connected component, mimic of the boundary of each level set, thus, we can extract the line-like skeleton after the construction of Reeb graph. In Section III, we will detail the implementation of pose-independent Reeb graph construction, line-like skeleton extraction, and on top of them, the design of load-balanced, distance-sensitive storage and retrieval protocols.

III. The Reeb Graph Based Information Storage and Retrieval

In this section, we introduce the details of the pose-independent Reeb graph construction, line-like skeleton extraction, and their applications for information storage and retrieval. We do not require location or boundary information, as they might be difficult to obtain for such scenarios as underground or underwater environments, etc. Generally speaking, our algorithms have three building blocks: Reeb graph construction, line-like skeleton extraction, and information storage and retrieval implementation.

A. Reeb Graph Construction

To construct the pose-independent Reeb graph of a 3D sensor network with an arbitrary shape, the key is to select the right feature function. Our work is based on the minimum cut(s), as mentioned in Section II. Now we introduce the details of minimum cut identification and hereon the Reeb graph construction.

We first consider the 3D sensor network without holes, i.e., genus-0 network. To find the minimum cut, the selection of terminals is important as the improperly selection will result in a Reeb graph not applicable for a protocol with lightweight and balanced storage loads (see Fig. 5(a)). Our approach has the following three phases.

1) Initialization Phase: One randomly selected node initiates a network-wide flooding; the farthest node, say p, marks itself and then broadcasts a message to build a coarse Reeb graph where the distance to p serves as a feature function and the vertices are the critical nodes corresponding to the local/global minimum, local/global maximum or saddle(s) of the distance function. Note that this coarse Reeb graph may cause an unevenly distributed level set size, as mentioned earlier, and thus

cannot be applied to achieve storage load balance of sensor nodes.

Recall that our approach is to locate the minimum cut between two terminal nodes which can be identified as follows. To that end, an arbitrary node farthest from p (e.g., q) is selected to build the shortest path between p and q, and the terminals s, t can be any two nodes on the shortest path,[3] as illustrated in Fig. 5(b). We denote by $sp(s,t)$ the shortest path between s and t. Then, the nodes on $sp(s,t)$, except s and t, mark as *visited*, followed by building a new shortest path between s and t without using the visited nodes. The process is repeated until there is no path connecting s and t among the unvisited nodes. Denote by $sp_i(s,t)$ the ith shortest path between s and t by using unvisited nodes, where $sp_1(s,t)$ represents the (global) shortest path $sp(s,t)$. Note that there might be multiple shortest paths with an equal length. With a little abuse of notation, we still denote them as $sp_i(s,t)$.

2) Minimum cut Identification: According to the Definition 3, a minimum cut is the cut with the smallest weight, i.e., the length of boundary edges. To that end, we locate the *boundary shortest paths* defined as follows.

Definition 4: A shortest path is a boundary shortest path if it intersects with the network boundary.

Clearly, the identification of boundary shortest path is simple when boundary information is given, while in our boundary-free case, it is very challenging. Note that a boundary shortest path often has a large *deviation* from the shortest path. Thus, we propose a metric named *deviation degree* of a shortest path as follows.

Definition 5: The deviation degree of a shortest path $sp_i(s,t)$, denoted by $DD_i(s,t)$, is the maximum of hop count distance between nodes on $sp_i(s,t)$ and those on $sp_1(s,t)$. Namely,

$$DD_i(s,t) = \max_{p \in sp_i(s,t), q \in sp_1(s,t)} \{d_h(p,q)\} \tag{1}$$

where $d_h(p,q)$ represents the distance between p and q.

Lemma 1: For two shortest paths $sp_i(s,t), sp_j(s,t)$, if $i > j$, then the length of $sp_i(s,t)$ is larger than that of $sp_j(s,t)$, and vice versa.

Lemma 2: For two different shortest paths $sp_i(s,t)$ and $sp_j(s,t)$, if the length of $sp_i(s,t)$ is larger than that of $sp_j(s,t)$, then $DD_i(s,t) > DD_j(s,t)$.

Proof: As stated in Lemma 1, if the length of $sp_i(s,t)$ is larger than that of $sp_j(s,t)$, there is a larger void when building $sp_i(s,t)$, which implies the *diameter* of the void of $sp_i(s,t)$ is larger than $sp_j(s,t)$. Hence, $DD_i(s,t) > DD_j(s,t)$. □

Theorem 3: A shortest path $sp_i(s,t)$ with a locally maximal deviation degree is a boundary shortest path.

Proof: We prove this by contradiction. If $sp_i(s,t)$ is not a boundary shortest path, then there is no boundary nodes, which implies that there is at least one shortest path $sp_j(s,t)(j > i)$. According to Lemma 1, the length of $sp_j(s,t)$ is larger than that of $sp_i(s,t)$, and according to Lemma 2, the deviation degree $DD_j(s,t) > DD_i(s,t)$ which contradicts with the assumption that $DD_i(s,t)$ is locally maximal. □

With Theorem 3, the boundary shortest paths can be easily identified in a distributed fashion. However, the minimum cut identification is not straightforward, as there are many nodes on the boundary shortest paths, and not all of them consists of the minimum cut; our strategy is to identify some key nodes (referred to as *key minimum cut nodes*) on the boundary shortest paths and also on the minimum cut.

Definition 6: A node is said to be a key minimum cut node if 1) it locates in a boundary shortest path, and 2) the ratio of the number of its unvisited neighbors to that of visited neighbors is locally minimal.

After identifying some key minimum cut nodes, we conduct a greedy operation to connecting these nodes trying to move along the boundary paths, and the minimum cut between terminals can be achieved.

3) Reeb Graph Establishment: The minimum cut is used to define a feature function, hereby construct the Reeb graph. More specifically, the nodes in the minimum cut issue simultaneously an in-network flooding, and each node computes its distance to the minimum cut. The distance of a node then serves as a feature function f to construct a Reeb graph where, a vertex is a critical points of f, an edge connects two adjacent critical points, and a level set is a connected component of nodes with the same distance function.

For the genus-n network, the Reeb graph construction can be done in a similar way. First, in the initialization phase, the coarse Reeb graph with $n+2$ vertices is constructed. Between each edge of the Coarse Reeb graph, we conduct the minimum cut identification process in the above-mentioned way, and accordingly, $n+1$ minimum cuts can be obtained. Then, each node computes its distance to the nearest minimum cut; the nodes within a connected component and having an equal distance function value form a level set.

B. Line-Like Skeleton Extraction

The Reeb graph can be used for approximating the line-like skeleton point, as mentioned earlier in Section II. Recall that the feature loop(s), formed by the geodesic shortest path(s) between the feature nodes, of a line-like skeleton point can decompose the boundary into more than one connected components [16], [17] in a genus-0 network. The feature loop, together with the interior nodes nearest to the same feature loop, can be naturally treated as a level set here. In this subsection we will present the details of line-like skeleton extraction based on the constructed Reeb graph.

As the line-like skeleton is locally symmetric to the underlying network, the line-like skeleton node should locate medially in the level set. Thus, in each level set, we simply find its *center* in the following way. Initially, one node, say q, is randomly chosen to initialize a flood within the level set, and the farthest node q_1 then floods in the level set to locate its farthest node, and so on. The process continues for a few rounds, and the node, with the largest number of shortest paths passing through it, identifies itself as the center of the level set, which is approximately regarded as the line-like skeleton node of this

[3]We do not necessarily select p, q as terminals because, as can be seen in the following minimum cut identification process, the farther the terminals separate, the more cost for the shortest path construction. Thus, we can select two nodes having a separation of, e.g., 4 or 5 hops, to reduce the time and message complexity.

level set. Eventually, according to their feature function values, these skeleton nodes are orderly connected to derive the line-like skeleton.

C. Information Storage and Retrieval Implementation

Now we propose to apply the constructed Reeb graph and extracted line-like skeleton for the design of load-balanced and distance-sensitive information storage and retrieval protocol for an arbitrarily shaped 3D network. Note that each level set is associated with a unique edge of Reeb graph, i.e., locates between two adjacent critical nodes. We thus assign a unique name for each level set based on the feature function value and the associated edge, and let each node store the Reeb graph such that they know where to replicate or retrieve data. We first present a naive Reeb graph based algorithm, named *NREEDS*, followed by an alternative approach *REEDS*.

1) Naive Algorithm: NREEDS: A simple and intuitive strategy based on the Reeb graph works as follows:

Step 1: Information Storage: Given a hash function $H : \mathcal{C} \rightarrow \mathcal{LS}$ where $\mathcal{C}$ denotes the set of possible content attributes and $\mathcal{LS}$ the set of named level sets, the data with the same content attribute is hashed to the same level set (referred to as *hashed level set*), disregarding where the source is. Specifically, each producer replicate its data within the hashed level set in the following way.

Assume the network has been decomposed into $M \in (0, n)$ level sets. Each skeleton node first broadcasts a message within its resident level set $ls_i (i \in [1, M])$, whereafter every node $l \in ls_i$ keeps the record of its distance to the skeleton node, denoted by $d_i(l)$. Let $d^i_{\max} = \max\{d_i(l), l \in ls_i\}$. Then the distance $d_i(l)$ is normalized to be $\hat{d}_i(l) = \frac{d_i(l)}{d^i_{\max}}$. The producer p, say in ls_i, first broadcasts a message within its resident level set (referred to as *home level set*) about its content attribute and other nodes in ls_i stores a pointer about the content attribute and $p's$ resident level set name. 1) If the home and hashed level set fall within the same edge of Reeb graph, p first travels to an intermediate node $p_1 \in ls_j (j \neq i)$ satisfying that $|f(p_1) - f(h)| < |f(p) - f(h)|$ where $f(h)$ denotes the feature function value of any node h in the hashed level set, and $|\hat{d}_i(p) - \hat{d}_j(p_1)| = \min_{p' \in N_1(p)}\{|\hat{d}_i(p) - \hat{d}_j(p')|\}$ where $N_1(p)$ denotes the neighbors of p. Afterwards, p_1 sends to the skeleton node of its level set a pointer indicating the content attribute and the source, while p keeps traveling to $p'_1 s$ intermediate node p_2 until it hits the hashed level set, say at p'. Finally, p' replicates the received content within the hashed level set. 2) If the home and hashed level set share different edges, then p first finds the other end (i.e., critical node) of the edge closer to the hashed level set based on the locally stored Reeb graph, and then travels to the level set corresponding to this end in above-mentioned way, until it reaches a level set locating in the same edge of the hashed level set. Then, p travels to the hashed level set since they are now associated with the same edge. Note that the hashed level set and the producer might separate faraway. In this case, if the consumer is close to the producer but fetches the data from the hashed level set, the retrieval path can be very long and thus the protocol is not distance-sensitive. As such, to ensure the distance-sensitivity, we let the nodes residing in the home level set and the skeleton nodes of the level sets on the replication path store a pointer indicating the content attribute and source.

Step 2: Information Retrieval: Similar with [5], we assume that each node is aware of the hash function of a certain data type such that they know where to retrieve the data of its interest, and also we assume that the consumer does not reside in the home or hashed level set, as in this case, the retrieval is rather simple. Further, here we only consider the case that the consumer locates in the same edge of the hashed level set for ease of statement, as the retrieval process can be similarly done in the other case. Then there are three possible cases of a consumer's relative position: between home and hashed level set, not between them but nearer to the hashed level set (i.e., hashed level set sided), or home level set sided. When the consumer retrieves the interested data, it first checks whether the skeleton node of its resident level set has a pointer or not. If yes, it must locate between the home and hashed level set, and thus retrieves the data from the closer level set according to the pointer information[4] under the guide of the skeleton and feature function value as described in step 1; otherwise, the consumer travels toward the hashed level set. For each intermediate node on the way of the travel, if there is a pointer, showing that the consumer is actually closer to home level set where the current intermediate node resides, the consumer can easily find the producer and fetch the data by using the pointer information. Otherwise, the consumer will keep moving until it reaches the hashed level set and return when the data is fetched.

Clearly, $NREEDS$ protocol has the following properties:

Proposition 4: NREEDS is retrieval guaranteed, distance-sensitive(i.e., if the consumer locates near the producer, the retrieval path will be short as well), and load-balanced.

Proof: As the line-like skeleton surely intersects with each level set, the retrieval is always guaranteed. next we prove its distance-sensitivity. If the consumer locates between the home and hashed level set, then there must be a pointer stored in the skeleton node of its resident level set. The consumer can fetch the data either from the producer, or the nodes in hashed level set, based on the difference of feature function values of the consumer and the producer or the hashed level set. As such, the retrieve path will not be long if the consumer and producer are nearby. If the consumer does not locate between the home and hashed level set, it will fetch the data from the nearer one, which again shows that the retrieve path will be not long, and thus the protocol is distance-sensitive.

As the distance of a point to the skeleton (or, the distance to the skeleton point in its level set) is normalized, the protocol can avoid boundary nodes' overloading and thus balance the retrieval load. Besides, the storage load is bounded by $\max(|ls_i|, i = 1, 2, \ldots, M)$, and each level set has an almost equal size by construction in most cases. Therefore, the storage and retrieval load can be balanced. □

[4]The consumer knows the feature function value of the hashed level set, the producer and itself, thus it is easy to know who is closer.

2) An Alternative Approach: REEDS: Despite of these properties of NREEDS, the undesirability is that each node replicates the data within the hashed evel set which might result in some nodes heavily loaded if there are large level sets. Next we introduce an alternative called *REEDS* which evenly distributes the storage overhead. The key is to let a producer replicate the data in a small part of the level set, i.e., a sub-tree, instead of the whole level set.

Step 1: Information Storage: First, the line-like skeleton node of each level set broadcasts a message within the level set to build a shortest path tree, hereby a number of sub-trees rooted at each child of the skeleton node. At the same time, each child node of a skeleton node stores the storage load of its sub-tree. Similar with NREEDS, each data is first hashed to the hashed level set based on a hash function of the content attribute. But here the producer only replicates its information within the sub-tree with the smallest load. In addition, as described in REEDS, each node stores its normalized distance to the root skeleton node, and the nodes residing in the home level set and the skeleton nodes on the way of replication stores a pointer to show where the data is replicated.

Step 2: Information Retrieval: The consumer p first travels to the hashed level set of producer q in the same way as in NREEDS,[5] and hits the hashed level set, say at l. Then, p conducts a *circle searching operation* within the level set to reach the sub-tree of a child of the skeleton node in the hashed level set. That is, l travels to the best intermediate node l' such that 1) l and l' locate in different sub-trees, 2) $|\hat{d}_i(l) - \hat{d}_i(l')| = min_{l' \in N_1(l)}\{|\hat{d}_i(l) - \hat{d}_i(l')|\}$, 3) $f(l) = f(l')$, and 4) l has not visited the resident level set of l'; the process is repeated until p travels to the targeted sub-tree.

As the sub-tree size is much smaller than the level set size, no nodes will be heavily loaded. If the broadcasted message travels roughly at the same speed, each sub-tree will be as *thin* as a line. In the end, the balance of storage load can be achieved. Besides, the retrieval can also be guaranteed since p can successfully reach the hashed or home level set and each level set is regularly shaped such that circle searching operation will not fail. Hence, we have

Proposition 5: REEDS is retrieval-guaranteed, storage and retrieval load balanced, and distance sensitive.

To highlight the difference between REEDS and NREEDS, we first define the metric named *Load Ratio* as follows.

Definition 7: Let the network of size n be decomposed into $M \in (1, n)$ level sets, and $|ls_i|$ be the size of the ith level set ls_i. Then the load ratio of nodes in ls_i to that in $ls_j (i \neq j)$, denoted by $LR(i, j)$, is defined as

$$LR(i, j) = \frac{|ls_i|}{|ls_j|}. \tag{2}$$

Theorem 6: The upper bound of the storage load ratio by NREEDS is not smaller than that by REEDS.

Proof: We prove this in continuous domain. Let $ls_k, ls_m (k, m \in M)$ be the level set with the largest and smallest area (since each level set is a 2-manifold sheet), and $R_{\max}, R_{\min}$ denote the maximal distance of the line-like skeleton point of ls_k and ls_m to other nodes in ls_k and ls_m, respectively. Clearly, ls_k is bounded by a disk centered at the line-like skeleton point of ls_k with radius $R_{\max}$, and the disk centered at the line-like skeleton point in ls_m with radius $R_{\min}$ is totally included in ls_m. Hence, the storage load ratio by NREEDS has an upper bound of $\frac{\pi R_{\max}^2}{\pi R_{\min}^2} = \frac{R_{\max}^2}{R_{\min}^2}$, and the upper bound by REEDS is $\frac{R_{\max}}{R_{\min}}$. Since $R_{\max} \geq R_{\min}$,[6] we have $\frac{R_{\max}^2}{R_{\min}^2} \geq \frac{R_{\max}}{R_{\min}}$, which proves the claim. □

Corollary 7: Let $ls_m (m \in M)$ be the level set with the smallest area. For each level set $ls_i (i \neq m)$, the load ratio by NREEDS is no smaller than that by $REEDS$.

Proposition 8: REEDS is more storage-balanced and less distance-sensitive than REEDS in irregular-shaped networks.

Proof: In most cases except the cylinder-shaped network, the network is often irregular-shaped, and the sizes of the level sets are not always the same. Hence, generally speaking, the upper bound of REEDS is much smaller than NREEDS, and the difference of the storage load between two nodes in different level set is larger for NREEDS than that for REEDS. In addition, the retrieval path by REEDS includes the path by NREEDS and the greedy shortest path built in the last mile. That is, the retrieval path length is larger than NREEDS, implying that it is less distance-sensitive. □

Thus, by proposing NREEDS and REEDS, we offer a tradeoff between storage load balance and retrieval path length, i.e., distance-sensitivity.

D. Complexity Analysis

Theorem 9: The algorithms have an $O(N)$ time and message complexity where N denotes the network size.

Proof: The algorithms consist of three steps.

1) During the first step, the coarse Reeb graph construction and the selection of terminals both incur an $O(N)$ complexity. If the terminals are very close, e.g., four hops away, the minimum cut identification is only conducted locally, incurring a constant complexity for each minimum cut, and at most $O(N)$ complexity in total. To construct the feature function, each node computes its distance to the minimum cut(s), which can be done via a simultaneous flooding from the minimum cut(s) such that the complexity is linear to the network size.
2) For the second step, to extract the line-like skeleton, each level set computes its center, i.e., the line-skeleton node. As the size of each level set is constant compared with the network size, this step incurs a constant complexity within a level set and at most $O(N)$ in total. Since the skeleton node number is far less than N, the connection of line-like skeleton nodes has a complexity of at most $O(N)$.
3) Finally, for storage and retrieval, the construction of tree is conducted within each level set. Each node is affiliated with one skeleton tree, and therefore this step only incurs an $O(N)$ complexity in total.

In summary, both NREEDS and REEDS have a linear time and message complexity. □

[5]If p hits the home level set, it can fetch the data from q by NREEDS.

[6]The equality holds only when the network is cylinder-shaped everywhere.

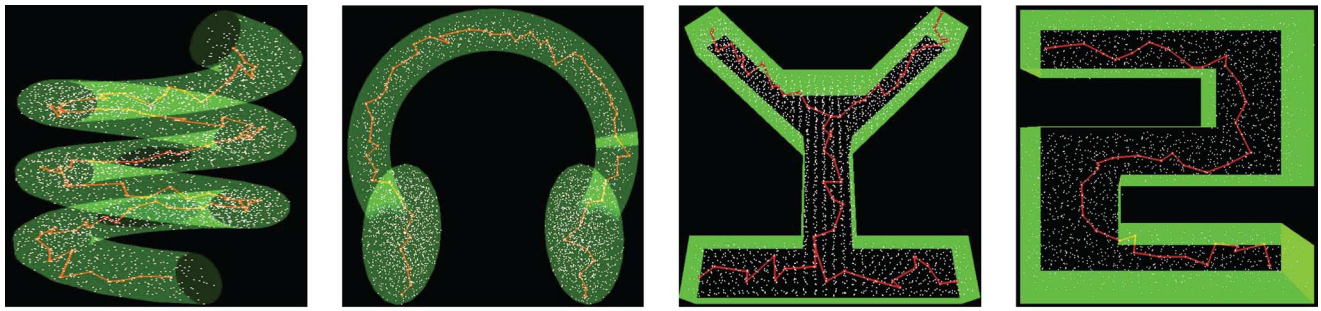


Fig. 6. The line-like skeleton extraction under 3D sensor networks with different shapes, where boundary information of each scenario is unknown. The curves represent the line-like skeleton. (a) Spiral, 3,464 nodes, avg. deg 15.42; (b) Headset, 3,260 nodes, avg.deg 16.94; (c) Y, 2,475 nodes, avg. deg. 18.22; (d) S, 2,244 nodes, avg. deg 17.25.

IV. Performance Evaluation

Previously we have presented Reeb-graph based data-centric storage and retrieval protocols, named NREEDS and REEDS. In this section, we will examine their performance via extensive simulations. We first show the line-like skeleton computed based on the construction of the boundary-free and pose-independent Reeb graph. Then, to quantitatively measure the performance of NREEDS and REEDS, we conduct a large number of data storage and retrieval simulations on different scenarios, and compare with the cut-graph based scheme (referred to as *Cut Graph)* in [28], [29].

A. Simulation Setup

In our simulations, sensors are randomly deployed in the sensing space; each sensor has the same communication radio range, and the communication radio model follows by default the unit disk graph (UDG). We do not assume the boundary information or geographic location information.

In the data storage and retrieval simulations, we randomly select 10,000 pairs of producers and consumers; for each pairwise producer and consumer, the producer first replicates its data within the replication plane, i.e., the hashed level set (in NREEDS) or replication sub-tree (in REEDS).

For fair comparison, to implement the Cut-graph algorithm [28], [29], we manually detect the boundary of the 3D networks; after a triangulation process, the boundary surface is mapped to a rectangle virtual coordinate system, whereby a simple 2D double rulings applies; an interior node then follows along the greedy path to the boundary and conducts the 2D double-ruling based scheme to store and retrieve data.

For quantitative comparison, we use two metrics: length ratio and storage load. The length ratio of a retrieval path is the ratio of the retrieval path length to the length of the shortest path between a pair of producer and consumer, and the storage load of a node is the number of packets the node received and stored locally; for comparison, we compute the distribution of storage load and retrieval length ratio by NREEDS, REEDS and Cut Graph, respectively.

B. Simulation Results

Line-Like Skeleton Extraction: We first evaluate the performance of the Reeb-graph based line-like skeleton extraction, where the geographic location and boundary information are all assumed to be unknown. Fig. 6 presents the results of the algorithm under four different 3D scenarios, i.e., Spiral (see Fig. 6(a)), HeadSet (see Fig. 6(b)),Y (see Fig. 6(c)) and S (see Fig. 6(d)). From Fig. 6(a), we see that there are five *zig zags*, and the extracted line-like skeleton by our algorithm correctly reflects these four *sudden turns* and locate medially inside the network, accurately capturing the main topological features of the underlying network. From Figs. 6(b)–6(d), we observe the similar results. That is, the extracted line-like skeletons are all locally symmetric to the underlying 3D networks, showing that our algorithm can correctly extract the line-like skeleton, without reliance on the boundary information.

Performance Comparison of Information Storage and Retrieval: Fig. 7 depicts the comparison studies on data storage and retrieval path length by conducting three algorithms, namely Cut Graph, NREEDS and NREEDS, on the four networks in Fig. 6. For the Spiral network (see Fig. 7(a)), we can see that most of the retrieval paths by REEDS and NREEDS have a small length ratio to the shortest path (see row 1, Fig. 7(a)). More specifically, among the 10,000 pairwise producers and consumers, there are more than 9,500 paths having a length ratio less than 2, where NREEDS produces a slightly smaller length ratio than REEDS. This is because after *hitting* the home or hashed level set, REEDS conducts a greedy operation to locate the least-loaded sub-tree. But we also can see from Fig. 7(a) (row 2), that the storage load of REEDS is much smaller than NREEDS. In the network with 3,464 nodes, about 3,000 nodes have a storage load smaller than 50, while about 2,000 nodes by NREEDS store less than 50 packets. Clearly, compared with REEDS, NREEDS produces a small length ratio but the storage load is relatively larger since it allow the producers to replicate their data within the hashed level sets, instead of the sub-trees as in REEDS. In other words, NREEDS and REEDS provide for us a tradeoff between retrieval length ratio and storage load. On the other hand, the result by Cut Graph is the worst. Only a few retrieval path have a length ratio smaller than 2, and about 5,500 retrieval paths have a length ratio smaller than 4. This is because each interior node has to firstly follow the greedy path to the nearest boundary node, and then applies double ruling on the boundary surface; even though the producer might be very close to the consumer, the

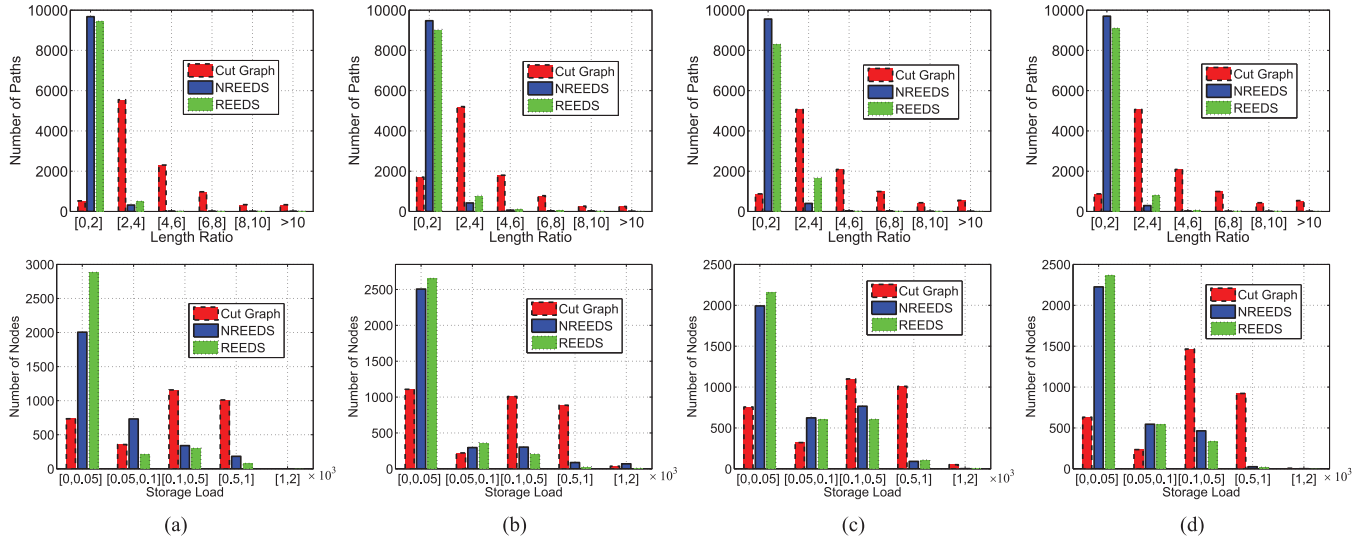


Fig. 7. The performance comparison under 3D sensor networks in Fig. 6. Row 1: length ratio distribution; Row 2: storage load distribution. (a) Spiral; (b) Headset; (c) Y; (d) S.

TABLE I
THE COMPARISON STUDY OF THE AVERAGE LOAD PER NODE

Scenarios	NREEDS	REEDS	Cut Graph
Spiral	95.37	21.49	88.51
S	103.47	22.73	99.83
Y	98.27	22.64	95.44
$Headset$	95.95	21.87	94.67

retrieval path could be very long, as mentioned in Section I. Accordingly, the storage load of boundary nodes, together with the interior nodes nearest to the boundary nodes, will be very large, which is confirmed by our experiment (see row 2, Fig. 7(a)). For the other three investigated scenarios, we see the similar trend: NREEDS provides a smallest length ratio while the storage load by REEDS is the smallest; Cut Graph delivers the worst results in terms of both length ratio and storage load.

Table I presents the comparison results of the three algorithms on the average load per node. As expected, we observe that the average load per node by REEDS is smaller than REEDS and Cut Graph. Counter-intuitively, the average load per node by NREEDS is larger than but still comparable with Cut Graph. This is because in the four investigated scenarios, the networks all exhibit a long corridor shape, resulting in a long replication curve in the virtual planar rectangle by Cut Graph, and the number of nodes on the replication curve is comparable with the level set size.

Overall, both NREEDS and REEDS can guarantee retrieval delivery, even without reliance on boundary information or mesh structure like Cut Graph. NREEDS is more distance-sensitive while REEDS yields more balanced and smaller storage load. They provide a tradeoff between length ratio (or distance sensitivity) and storage overhead, both of which outperform Cut Graph.

V. RELATED WORK

In this section, we briefly introduce related work of distributed data-centric storage/retrieval (DCS) protocols with good data persistence in 2D/3D sensor networks, and classify them into two categories: GHT-based [18], [21], [31] and double ruling-based schemes [5], [19], [23], [24], [28]–[30].

A. GHT-Based DCS Protocols

Ratnasamy *et al.* [21] first proposed to exploit the geographical hash tables (GHT) for data storage in 2D sensor networks. In GHT, the data item generated by a node (producer) is mapped to a position, and the node (named *home node*) closest to the position stores the data. For data persistence, the data is also replicated locally on rendezvous nodes such that no sensed data will be lost and the hotspot of communication can be avoided. Then, the nodes (consumers) interested in the data of a home node retrieve the data according to GPSR [12] policy. The performance of GHT in [21], primarily designed for 2D sensor networks, heavily relies on the computed bounding box, and it suffers from the irregularity of a complex network if the bounding box is not accurately computed based on the convex hull, resulting in the boundary nodes being overloaded. To this end, Zhou *et al.* [31] proposed to identify the bottleneck(s) of 3D sensor networks by computing the injectivity radius of the boundary nodes which implies the narrowness of an interested region of the boundary surface, and then compute the bounding box to improve the performance of GHT. The undesirability is that when a complex network has no bottleneck, it is incapable of computing an accurate bounding box. Noticing that the irregularity of the boundary surface at the concave nodes will bend the surface skeleton, Liu *et al.* [18] proposed to locate the concave nodes by extracting the surface skeleton, on top of which the computed bounding box can be very tight. Note that GHT is distance-insensitive, i.e., the retrieval path between consumer and producer with a short distance may be long, possibly resulting in a large retrieval delay and traffic cost. As a result, the in-network aggregation for semantic reports suffers from a significant access delay.

B. Double Ruling-Based DCS Protocols

Sarkar *et al.* [23], [24] developed a double-ruling scheme where a producer leaves copies of the sensed data along a

replication curve, and the consumer retrieves the data following a retrieval curve until it hits the replication curve; theoretically, the double-ruling scheme is distance-sensitive and retrieval-guaranteed. However, such a double-ruling scheme is prone to fail for a network with multiple holes. Fang *et al.* [5] proposed a divide-and-conquer policy to solve this problem. After partitioning the network into tiles, a two-level routing scheme is designed by combining the GHT (when across tiles) and the double ruling scheme (if within the same tile). Despite their success in 2D environments, however, the double ruling based schemes in [5], [23], [24] are difficult to utilize in 3D spaces, and the naive extension of them often does not work well in irregular-shaped 3D sensor networks. Recently, Luo *et al.* [19] proposed to parameterize a 3D network volume to a solid cube, and design a quorum system accordingly. Clearly, such a volumetric parametrization-based double-ruling scheme is only applicable for the network topologically equivalent to a solid cube, and cannot apply for the practical scenarios where the volume usually cannot be mapped to a solid cube. Zhang *et al.* [30] proposed to marry the quorum system with harmonic fields such that it can work for complex 3D networks with holes. Yang *et al.* [28], [29] conducted a 3D-capable work to cut open the closed boundary surface of the 3D sensor networks to a topological disk along the cut graph of the surface, and then map the boundary surface to an aligned planar rectangle where each boundary node has a virtual coordinate such that double-ruling scheme is applicable on boundary surface. Afterwards, each producer first sends the data to its nearest boundary node and then replicates the data along the horizontal line of the virtual planar rectangle; the consumer follows the sequence of ID to the boundary surface and retrieves the data along the vertical line. Such a cut-graph based scheme in [28], [29] is retrieval-guaranteed yet load-imbalanced and distance-insensitive: as the double ruling mainly applies on the boundary surface, the boundary nodes are supposed to be overloaded, and when the producer and consumer are nearby (e.g., on opposite sides of the line-like skeleton), the retrieval path can be arbitrarily long.

VI. Conclusions

In this paper, we study the problem of information storage and retrieval for 3D sensor networks, and proposed two novel Reeb graph based schemes, named NREEDS and REEDS, without reliance on the geographic location or boundary information. These two schemes are both distance-sensitive and load-balanced while they provide a tradeoff between the length ratio (thus the time cost) and the storage overhead. NREEDS produces a shorter retrieval path yet heavier (and slightly less balanced than REEDS) storage load, and REEDS yields a slightly longer retrieval path but the storage load will be much smaller and more balanced than NREEDS.

In future, we would like to explore more applications based on the Reeb graph of 3D sensor networks such as network segmentation [8], [9], k-area coverage problem [13], [14], and path planing for mobile sinks [27], etc.

References

[1] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: Research challenges," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 257–279, 2005.

[2] S. Basagni *et al.*, "Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle," in *Proc. IEEE INFOCOM*, 2014, pp. 988–996.

[3] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno, "Reeb graphs for shape analysis and applications," *Theor. Comput. Sci.*, vol. 392, no. 1–3, pp. 5–22, 2008.

[4] D. Dong, Y. Liu, and X. Liao, "Fine-grained boundary recognition in wireless ad hoc and sensor networks by topological methods," in *Proc. ACM MobiHoc*, 2009, pp. 135–144.

[5] 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.

[6] S. Funke, L. J. Guibas, A. Nguyen, and Y. Wang, "Distance-sensitive information brokerage in sensor networks," in *Distributed Computing in Sensor Systems*. Berlin, Germany: Springer, 2006, vol. 4026, Lecture Notes in Computer Science, pp. 234–251.

[7] M. Hassouna and A. Farag, "Robust centerline extraction framework using level sets," in *Proc. IEEE CVPR*, 2005, pp. 458–465.

[8] H. Jiang, T. Yu, C. Tian, G. Tan, and C. Wang, "CONSEL: Connectivity-based segmentation in large-scale 2D/3D sensor networks," in *Proc. IEEE INFOCOM*, 2012, pp. 2086–2094.

[9] H. Jiang, T. Yu, C. Tian, G. Tan, and C. Wang, "Connectivity-based segmentation in large-scale 2D/3D sensor networks: Algorithm and applications," *IEEE/ACM Trans. Netw.*, vol. 23, no. 1, pp. 15–27, Feb. 2015.

[10] H. Jiang, S. Zhang, G. Tan, and C. Wang, "CABET: Connectivity-based boundary extraction of large-scale 3D sensor networks," in *Proc. IEEE INFOCOM*, 2011, pp. 784–792.

[11] H. Jiang, S. Zhang, G. Tan, and C. Wang, "Connectivity-based boundary extraction of large-scale 3D sensor networks: Algorithm and applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 4, pp. 908–918, Apr. 2014.

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

[13] F. Li, J. Luo, W. Wang, and Y. He, "Autonomous deployment for load balancing k-surface coverage in sensor networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 279–293, Jan. 2015.

[14] F. Li, J. Luo, S. Xin, W. Wang, and Y. He, "LAACAD: Load balancing k-area coverage through autonomous deployment in wireless sensor networks," in *Proc. IEEE ICDCS*, 2012, pp. 566–575.

[15] M. Li and Y. Liu, "Underground coal mine monitoring with wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 5, no. 2, pp. 528–539, 2009.

[16] W. Liu, H. Jiang, Y. Yang, and Z. Jin, "A unified framework for line-like skeleton extraction in 2D/3D sensor networks," in *Proc. IEEE ICNP*, 2013, pp. 1–10.

[17] W. Liu *et al.*, "A unified framework for line-like skeleton extraction in 2D/3D sensor networks," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1323–1335, May 2015.

[18] W. Liu *et al.*, "Surface skeleton extraction and its application for data storage in 3D sensor networks," in *Proc. ACM MobiHoc*, 2014, pp. 337–346.

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

[20] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas, "Robust on-line computation of Reeb graphs: Simplicity and speed," *ACM Trans. Graphics*, vol. 26, no. 3, pp. 58:1–58:9, 2007.

[21] S. Ratnasamy *et al.*, "Data-centric storage in sensornets with ght, a geographic hash table," *Mobile Netw. Appl.*, vol. 8, no. 4, pp. 427–442, 2003.

[22] S. Ratnasamy *et al.*, "GHT: A geographic hash table for data-centric storage in sensornets," in *Proc. Int. Workshop Wireless Sensor Netw. Appl.*, 2002, pp. 78–87.

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

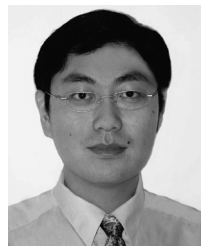
[24] R. Sarkar, X. Zhu, and J. Gao, "Double rulings for information brokerage in sensor networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1902–1915, Dec. 2009.

[25] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," *Comput. Commun. Rev.*, vol. 33, no. 1, pp. 137–142, 2003.

[26] M. Stoer and F. Wagner, "A simple min-cut algorithm," *J. ACM*, vol. 44, no. 4, pp. 585–591, 1997.
[27] R. Sugihara and R. K. Gupta, "Path planning of data mules in sensor networks," *ACM Trans. Sensor Netw.*, vol. 8, no. 1, pp. 1:1–1:27, 2011.
[28] Y. Yang, M. Jin, Y. Zhao, and H. Wu, "Cut graph based information storage and retrieval in 3D sensor networks with general topology," in *Proc. IEEE INFOCOM*, 2013, pp. 465–469.
[29] Y. Yang, M. Jin, Y. Zhao, and H. Wu, "Distributed information storage and retrieval in 3D sensor networks with general topologies," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1149–1162, Aug. 2015.
[30] C. Zhang *et al.*, "Harmonic quorum systems: Data management in 2D/3D wireless sensor networks with holes," in *Proc IEEE SECON*, 2012, pp. 1–9.
[31] 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. IEEE SECON*, 2011, pp. 494–502.
[32] H. Zhou, H. Wu, and M. Jin, "A robust boundary detection algorithm based on connectivity only for 3D wireless sensor networks," in *Proc. IEEE INFOCOM*, 2012, pp. 1602–1610.
[33] H. Zhou, S. Xia, M. Jin, and H. Wu, "Localized and precise boundary detection in 3-D wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1742–1754, Dec. 2015.

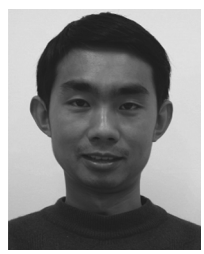
Wenping Liu is currently an Associate Professor with Hubei University of Economics, Wuhan, China. He received the Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 2012. His research interests include topological recognition and its applications in sensor networks. He is a member of the IEEE.

Hongbo Jiang received the B.S. and M.S. degrees from Huazhong University of Science and Technology, Wuhan, China. He received the Ph.D. degree from Case Western Reserve University, Cleveland, OH, USA, in 2008. After that, he joined the faculty of Huazhong University of Science and Technology, where he is now a Full Professor. His research concerns computer networking, especially algorithms and protocols for wireless sensor networks and mobile computing. He is a Senior Member of the IEEE.



Jiangchuan Liu received the B.Eng. degree from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from The Hong Kong University of Science and Technology, Hong Kong, in 2003. He is a co-recipient of the Best Student Paper Award of IWQoS2008 and the Multimedia Communications Best Paper Award from the IEEE Communications Society. He is currently an Associate Professor with Simon Fraser University, Vancouver, BC, Canada, and was an Assistant Professor with The Chinese University of Hong Kong, Hong Kong, from 2003 to 2004. His research interests include cloud computing, peer-to-peer systems, multimedia communications, and wireless networking. He is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA and an editor of *IEEE Communications Surveys and Tutorials*. He is a Senior Member of the IEEE.

Xiaofei Liao received the Ph.D. degree in computer science and engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2005. He is now a Professor with the School of Computer Science and Engineering, HUST. He has served as a reviewer for many conferences and journal papers. His research interests are in the areas of system software, P2P system, cluster computing, and streaming services. He is a member of the IEEE and the IEEE Computer Society.



Hongzhi Lin received the B.S., M.S., and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2000, 2003, and 2008, respectively. He is now an Assistant Professor with Huazhong University of Science and Technology. His current research interests are in the areas of wireless networking and digital signal processing.

Tianping Deng received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China, in 1998, and the M.S. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2003 and 2007, respectively. He then joined the faculty of Huazhong University of Science and Technology as an Assistant Professor. His research interests include computer networking and wireless communication. He is a member of the IEEE.