BLOW-UP: Toward Distributed and Scalable Space Filling Curve Construction in 3D Volumetric WSNs

CHEN WANG, WEI WEI, HONGZHI LIN, and HONGBO JIANG, Huazhong University of Science and Technology
JOHN C. S. LUI, Chinese University of Hong Kong

In wireless sensor networks (WSNs), a space filling curve (SFC) refers to a path passing through all nodes in the network, with each node visited at least once. By enforcing a linear order of the sensor nodes through an SFC, many applications in WSNs concerning serial operations on both sensor nodes and sensor data can be performed, with examples including serial data fusion and path planning of mobile nodes. Although a few studies have made efforts to find such SFCs in WSNs, they primarily target 2D planar or 3D surface settings and cannot be directly applied to 3D volumetric WSNs due to considerably more complex geometric features and topology shapes that the 3D volumetric settings introduce. This article presents BLOW-UP, a distributed, scalable, and connectivity-based algorithm to construct an SFC for a 3D volumetric WSN (or alternatively to linearize the 3D volumetric network). The main idea of BLOW-UP is to decompose the given 3D volumetric network into a series of connected and closed layers, and the nodes are traversed layer by layer, incrementally from the innermost to the outermost, yielding an SFC covering the entire network, provably at least once and at most a constant number of times. To the best of our knowledge, BLOW-UP is the first algorithm that realizes linearization in 3D volumetric WSNs. It does not require advance knowledge of location or distance information. It is also scalable with a nearly constant per-node storage cost and message cost. Extensive simulations under various networks demonstrate its effectiveness on nodes' covered times, coverage rate, and covering speed.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: 3D volumetric WSNs, space filling curve, network layering, unit tetrahedron cell, spiral-like triangular strip

ACM Reference Format:
Chen Wang, Wei Wei, Hongzhi Lin, Hongbo Jiang, and John C. S. Lui. 2016. BLOW-UP: Toward distributed and scalable space filling curve construction in 3D volumetric WSNs. ACM Trans. Sen. Netw. 12, 4, Article 30 (September 2016), 20 pages.
DOI: http://dx.doi.org/10.1145/2956551

This work was supported in part by the National Natural Science Foundation of China under grants 61572219, 61502192, 61271226, 61272410, 61202460, and 61471408; the National High Technology Research and Development Program ("863" Program) of China under grants 2014AA01A701 and 2015AA011303; the National Natural Science Foundation of Hubei Province under grant 2014CFA040; the China Postdoctoral Science Foundation under grant 2014M560608; the Fundamental Research Funds for the Central Universities under Grants No.2015QN073, No.2016YXMS297 and No.2016JCTD118; and the Science and Technology Plan Projects of Wuhan City under grant 2015010101010022.
Authors' addresses: C. Wang, W. Wei, H. Lin (corresponding author), and H. Jiang, School of Electronic Information and Communications, Huazhong University of Science and Technology, P. R. China; emails: {cwangwhu, hadeswei2012, eihongzhilin2012, hongbojiang2004}@gmail.com; J. C. S. Lui, Computer Science and Engineering Department, Chinese University of Hong Kong (CUHK), Hong Kong; email: cslui@cse.cuhk.edu.hk.

DOI: http://dx.doi.org/10.1145/2956551

1. INTRODUCTION

Recent years have witnessed great success of wireless sensor networks (WSNs) for the capabilities of physical sensing, information gathering, and distributed monitoring [Akyildiz and Vuran 2010; Yang 2014]. Although most earlier studies assume sensor networks on a 2D plane, there has been increasing interest in deploying sensors in a 3D space for applications such as underwater reconnaissance and atmospheric monitoring [Tan et al. 2013; Noh et al. 2013; Li et al. 2014; Xia et al. 2014; Luo et al. 2014; Jiang et al. 2015].

In this article, we focus on distributed algorithms for space filling curve (SFC) construction to linearize a 3D volumetric WSN (i.e., "traversing" the entire 3D volumetric WSN by a single path). In the following, we first introduce the notion of the SFC and summarize its applications in WSNs. Then we review related works on SFC construction and discuss challenges to construct the SFC in 3D volumetric WSNs before presenting our approach.

1.1. The SFC and Its Applications in WSNs

In mathematical analysis, a d-dimensional SFC refers to a surjective, continuous mapping from the unit interval to the unit d-cube [Sagan 1994] (i.e., a curve whose range contains the entire d-dimensional hypercube ($d > 1$)). In 1890, Peano first discovered the existence of the SFC by constructing a mapping from the unit interval onto the unit square, now known as the Peano (space filling) curve [Peano 1890]. In 1891, Hilbert gave the first geometric interpretation of the SFC (i.e., the so-called Hilbert curve [Hilbert 1891]). Henceforth, the curve is deduced to fill the unit cube, or more generally n-dimensional Euclidean space (but only in hypercubes) [Bader 2012]. Figure 1 provides an illustration.

In discrete WSNs, the concept of an SFC is introduced as a path passing through all nodes in the network, with each node visited at least once [Ban et al. 2013; Wang and Jiang 2015]. By enforcing a linear order of the sensor nodes through the SFC, or alternatively linearizing the network, many applications in WSNs concerning serial logical definitions and serial operations on both sensor nodes and sensor data can be performed.

One important such application is *serial data fusion* [Patil et al. 2004; Rajagopalan and Varshney 2006; Mostefaoui et al. 2015], an efficient in-network data aggregation paradigm whose objective is either to obtain a lower/upper value or to derive an estimate of interest [Kar et al. 2012] (e.g., average value [Nedic et al. 2009] or target location [Masazade et al. 2010; Zhu et al. 2010]). In contrast with a parallel fusion mechanism [Viswanathan and Varshney 1997; Blum et al. 1997], where a query starts simultaneously from multiple sensors along multiple paths to the sink, serial fusion combines sensor observation serially from node to node to derive a hypothesis until sufficient evidence has been collected, with the last node making the most correct judgement [Tan et al. 2011]. Often the implementation of serial data fusion in distributed WSNs asks for an SFC to play the role of regulating the query order and assisting in making aggregation decisions [Mostefaoui et al. 2015].

Another major application of SFC in WSNs is *path planning of mobile nodes* such as data mule [Shah et al. 2003; Sugihara and Gupta 2011], mobile sink [Tunca et al. 2014], and mobile charger [Porta et al. 2014; Madhja et al. 2015]. In these applications, the SFC can be utilized to sketch out the trajectories of mobile nodes or tessellate the sensor fields by mapping the multidimensional locations of sensors to a 1D curve, thus offering significant benefits for data dissemination [Ye et al. 2002; Tekdas et al. 2009], battery recharge [Xie et al. 2012], beacon-based localization [Koutsonikolas et al. 2007; Bahi et al. 2008], network coverage [Kamat et al. 2007; Bahi et al. 2008], and sensor node/data indexing [Ahmed and Bokhari 2007; Perera et al. 2014].

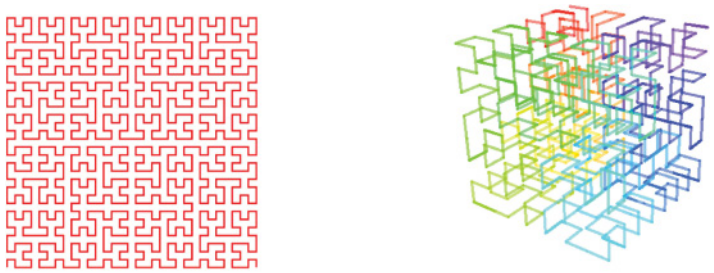


Fig. 1. (a) Hilbert curve in a 2D square. (b) Hilbert curve in a 3D cube.[1]

In light of the preceding applications, it is essential to efficiently construct SFCs in WSNs. It should be noted that the SFC construction we discuss here is much different from traditional multihop routing protocols [Flich et al. 2012; Cadger et al. 2013]: the former aims to find a multihop link between a pair of nodes with a specific destination, whereas the latter considers a global path schedule problem aiming to cover every node in the network. Thus, we pay more attention to a node's coverage completeness with the least reduplicate visits rather than shortening the path stretch or balancing the load.

1.2. Related Work on SFC Construction

In continuous settings, SFCs are often constructed in a recursive way, most if not all for regular topologies, such as a square, circle in the 2D domain, or a cube, sphere in the 3D domain [Bader 2012] (i.e., see Figure 1). To generate an SFC covering the entire hypercube, an infinite number of recursions, mathematically, is needed. For a regular-shaped grid (or cube) WSN with a discrete set of nodes, it is trivial to directly utilize existing mathematical schemes to construct the SFC with a relatively few recursion and short path length. However, the situation becomes intractable when sensor nodes are randomly deployed, which may cause tremendous computation and communication overhead. What makes matters even worse is that when the network shape is irregular, the generated SFC would be cut into many separated segments.

In view of the preceding challenges, a few studies have made efforts to find such SFCs in sensor networks [Ban et al. 2013; Mostefaoui et al. 2015; Wang and Jiang 2015]. Ban et al. [2013] conducted a pioneer work on constructing an SFC that densely covers any 2D sensor network with possibly holes. One particular feature of this approach is that the generated SFC is able to visit the nodes with progressive density and thus can accommodate different budgets of the path length. However, the proposed method is developed only for 2D WSNs and cannot be applied to 3D volumetric networks due to the considerably more complex geometric features that a higher-dimensional space introduces. Mostefaoui et al. [2015] proposed a novel localized serial algorithm theoretically ensuring that the generated curve covers all nodes in the network. The basic idea behind their algorithm is the so-called boundary traversal process. To be more concrete, the network is regarded as a composition of a series of boundary faces, and the serial process is conducted by traversal from the outmost boundary in sequence to inner ones. Although 2D boundary faces can be traversed by a local deterministic algorithm in either the clockwise or counterclockwise direction, the boundary of a 3D volumetric network is not a face but a surface, where the 2D boundary traversal process is no longer valid.

[1]Source: https://en.wikipedia.org/wiki/Hilbert_curve.

The only attempt to construct SFCs in a 3D sensor space is discussed in Wang and Jiang [2015], where a novel serial traversal algorithm was put forward based on established isocontours and network decomposition. However, it is designed for 3D surface WSNs with a high genus, where sensor nodes are deployed on the surface of a high-genus WSN. The proposed surface traversal approach cannot even be employed to cover the boundary surface of a 3D WSN with general topology (which is a genus-0 surface) in that a high-genus 3D surface is obviously not homotopically equivalent to a genus-0 3D surface, let alone adaptable to the entire 3D volumetric network space.

There are three potential solutions for SFC construction in 3D volumetric WSNs: Hamiltonian path, random walk, and tree-based traversal. In graph theory, a Hamiltonian path asks for a path that visits all vertexes in the network and visits each vertex once and only once. However, not all graphs have a Hamiltonian path, and it is NP-hard to determine whether a Hamiltonian path exists in a specified graph [Garey et al. 1976]. Random walk [Spitzer 2001], as its name suggests, traverses the network by randomly choosing one of its neighbors as the next hop. Although the scheme is simple, its drawbacks also stem from its randomness: it cannot ensure deterministic node coverage due to its essentially blind and luck-dependent traversal. Tree-based traversal (e.g., preorder traversal [Liang et al. 2013]) can guarantee node coverage by a depth-first-like traversal along a tree built from a root node, but one major concern is that an ordinary depth-first traversal takes the worst case of $2(\mathcal{N}-1)$ hops to cover a network with $\mathcal{N}$ nodes, indicating a large number of reduplicate visits. What is more, tree-based traversal is more likely to overload those nodes near the father nodes (as well as the root node), resulting in the so-called energy hole problem [Kuhn et al. 2003; Gao and Zhang 2006; Zhang and Shen 2009], as they are shared by different branches and thus subject to excessive visits during the traversal.

1.3. Overview of Our Approach

In this article, we present BLOW-UP, a distributed and scalable SFC construction algorithm for 3D volumetric WSNs, yielding a path provably ensuring that any node is covered at least once and at most a constant number of times. The main idea behind BLOW-UP is to decompose the 3D volumetric sensor network into a series of connected and closed layers, based on a unit tetrahedron cell (UTC) structure [Xia et al. 2011] of the given 3D volumetric network. The nodes are traversed/covered layer by layer, incrementally from the innermost to the outermost, finally yielding an SFC of the 3D volumetric network. Note that the whole process is something like putting an inflating balloon into a closed and arbitrary shaped box. Assume that the surface of the balloon is resilient enough that it will never be blown out. Then as the balloon blows up, the surface of the balloon and the shell of the box begin to touch gradually. In the end, the surface of the balloon and the shell of the box would stick together without any gap. Thus, we name our algorithm *BLOW-UP* to actually imply the basic idea of how we construct the SFC of a volumetric WSN.

Given the UTC structure, BLOW-UP involves a two-step process, as shown in Figure 2. The first step is to decompose the network into a set of connected and closed layers radiated from a so-called center node. This is done by a distributed procedure that starts from the center node with iterations, based on the UTC structure. The center node first grows to be a polyhedron composed of UTCs directly surrounding the center node. The exterior surface of the polyhedron is regarded as the first layer. Then the polyhedron continually assimilates the neighboring UTCs, forming expanding polyhedrons whose exterior surfaces are the incrementally expansive layers. This process stops when each sensor node is finally located on at least one layer.

The second step is to traverse the nodes' intra- and interlayers. For each layer we obtain from the first step, we construct a spiral-like triangular strip on it, along which

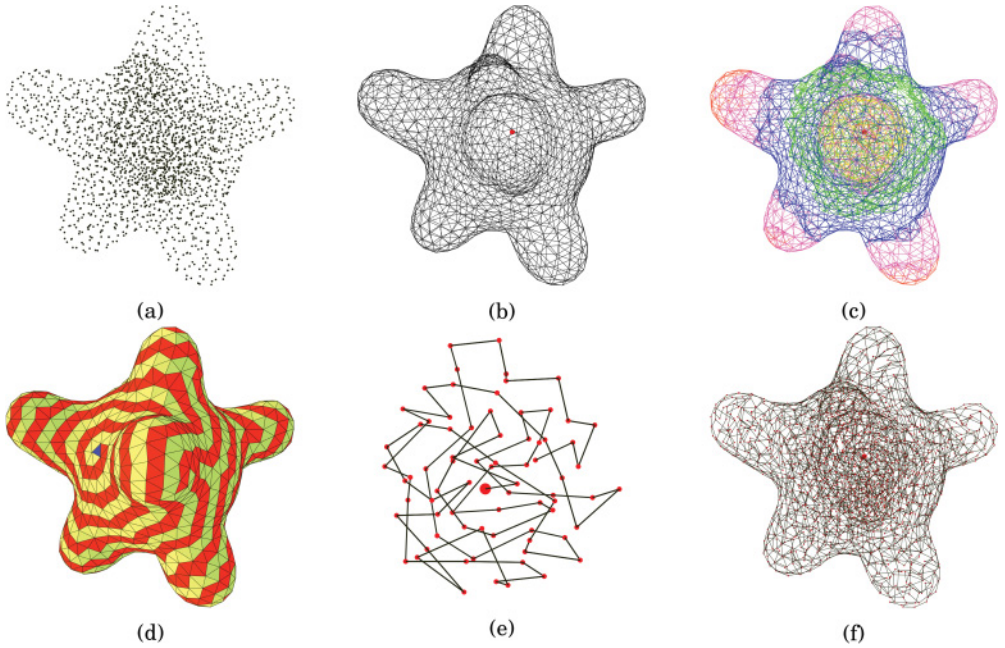


Fig. 2. Pipeline of the BLOW-UP algorithm. (a) A 3D volumetric network model with 2,290 nodes. (b) Boundary mesh; the center node is marked in red. (c) A series of connected and closed layers. (d) The spiral-like triangular strip (in red) on the outermost layer. (e) The segment of an SFC near the center node. (f) The constructed SFC of the network.

nodes in the layer can be traversed in a cyclic order with predefined orientation on the layer. In other words, if we move along the edge of the strip in a fixed direction, we will go back to the start node after covering all nodes on the layer. With the generated strips, lightweight intra- and interlayer traversal schemes are conducted to construct the final SFC for the network, provably covering each node at least once but no more than a bounded number of times.

To the best of our knowledge, BLOW-UP is the first algorithm that realizes SFC construction in 3D volumetric WSNs. It has several salient features:

—*Coverage guaranteed*: The generated SFC by BLOW-UP ensures that every node in the network is covered at least once and at most a constant number of times.
—*Distributed*: Each step of BLOW-UP is conducted in a distributed manner without reliance on a central controller.
—*Scalable*: In BLOW-UP, every sensor requires only local available information with a nearly constant per-node storage cost and message cost.
—*Location free*: BLOW-UP is based on connectivity information only with no reliance on a node's location information or any particular communication radio models.

The remainder of the article is organized as follows. In Section 2, we describe how to decompose the network to a series of connected and closed layers. In Section 3, we provide details on SFC construction in the 3D volumetric network on the basis of the established layers. Further discussions are conducted in Section 4, and performance evaluation of the proposed algorithm is presented in Section 5. Finally, Section 6 concludes the article.

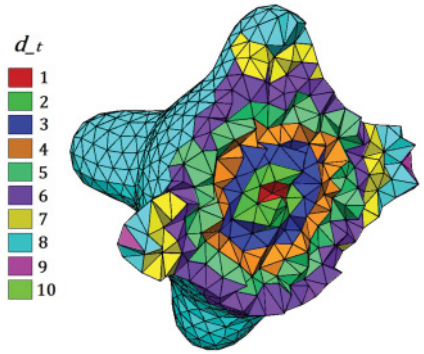


Fig. 3. Cross section of the tetrahedral flooding result. UTCs are marked by different colors according to their distance counters d_t; those UTCs with the same value of d_t form the crust.

2. NETWORK LAYERING

Given a WSN deployed in 3D volumetric environments, BLOW-UP first extracts a UTC structure of the network (detailed in Section 2.1), by which it is enabled to conduct tetrahedral flooding, so as to decompose the given WSN into connected and closed layers (detailed in Section 2.3). Note that in our algorithm, we assume a connected 3D volumetric network without (exterior or interior) holes.

2.1. UTC Structure Construction

A 3D volumetric WSN can be represented by a connectivity graph $G(V; E)$, where V denotes the nodes and E indicates the communication links in the network. It is well known that a tetrahedron consists of four nodes, six edges, and four triangular faces, where each face consists of three angles, and each angle is made up of one node and two edges. Formally speaking, we have the following definition.

Definition 2.1. A *unit tetrahedron cell* is a tetrahedron formed by four nodes and does not intersect with any other tetrahedrons [Xia et al. 2011].

The UTC structure, as a representative of the whole network, is constructed by integrating all of the UTCs, using a simple algorithm similar to that in Xia et al. [2011]. Specifically, we obtain the first UTC by choosing an arbitrary tetrahedron containing only its own vertex nodes and removing edges intersecting with this tetrahedron. Then the three nodes on one of the faces of the first UTC search for a common neighbor and form another UTC that neither overlaps any existing UTCs nor contains any other nodes. These steps run in a recursive way until all nodes are involved in the constructed UTCs. Finally, we can obtain a UTC structure of the network. It is noted here that during the UTC structure construction process, the connectivity between two nodes may be changed, as some intersecting edges may be removed. However, having this change of nodes' connectivity, the network is still connected; even the nodes are nonuniformly distributed in the space.

From the UTC structure construction process we have the following lemma.

LEMMA 2.2. *The UTC structure of a 3D volumetric WSN is solid.*

Figure 3 provides an illustration of the UTC structure. We can see that the UTC structure is fulfilled with UTCs with no interior hollow. This greatly facilitates the construction of connected and closed layers for layer traversal, as will be discussed in Section 2.3.

2.2. Preliminary

For the sake of the statement convenience in the rest of the article, we first present several definitions from Xia et al. [2011].

Definition 2.3. A face is a *boundary face* if it is contained in one UTC only.

Definition 2.4. A UTC is a *boundary UTC* if it contains at least one boundary face.

Definition 2.5. A node is a *boundary node* if it is one of the three nodes that form a boundary face.

Definition 2.6. If a node (or face or UTC) is not a boundary node (or face or UTC), it is defined as an *interior node (or face or UTC)*.

With these definitions, we propose to redefine the neighbor relationship in the following definition.

Definition 2.7. Two nodes are neighbors if and only if they are two endpoints of an edge; two faces are neighbors if and only if they share an edge; two UTCs are neighbors if they share either a face, an edge, or a node.

Then we have the following lemma.

LEMMA 2.8. *Given a node a, the set of UTCs containing a, denoted as ngb*(a)*, forms a solid polyhedron, and its neighboring nodes are on the surface of the polyhedron.*

PROOF. We prove the correctness by contradiction. Recall from Lemma 2.2 that the UTC structure of a 3D volumetric network is solid with no interior hollow, and thus a is wrapped by UTCs that surround it. Assume that b is one of a's neighbors and does not lie on the triangular face of the polyhedron formed by $ngb(a)$. In this case, the edge e_{ab} either passes through the faces of the polyhedron or lies in the interior of the polyhedron. In both cases, there must exist at least one UTC containing the edge e_{ab}, intersecting with the UTC that belongs to $ngb(a)$. This contradicts Definition 2.1 and in turn validates Lemma 2.8. □

Lemma 2.8 can be regarded as a basic principle to reveal the geometric relations between a node and its neighbors. It also provides theoretical foundations for layer construction and layer traversal in the SFC construction process.

2.3. Layer Construction

Given the UTC structure of the network, we now show how the network can be decomposed into connected and closed layers by grouping together the UTCs that share the same geometric characteristic.

First, we seek for a so-called center node, the node farthest from the boundary, as a start point of the layer construction. The center node is identified by boundary flooding, where boundary nodes can be identified according to Definition 2.5 (Figure 2(b) shows an example). Specifically, each node on the boundary floods a message containing its own ID and a hop counter (h_t) indicating in how many hops the message has been delivered. When an interior node receives such a message, it updates its h_t value if the message contains a smaller h_t than its maintained one; otherwise, it simply discards the message. After this process, each node knows its distance to the nearest boundary. Then the node with the maximum h_t value is elected as the center node. Note that there might be more than one node declaring the role of the center node. In this case, the node with the largest ID stands out, ensuring that there is only one node taking the role of the center node.

Second, we turn to tetrahedral flooding, an upper-class flooding method that is similar to traditional flooding. The main difference between tetrahedral flooding and traditional flooding is that the information exchange is carried out between a pair of UTCs rather than a pair of nodes. The UTCs directly surrounding the center node first record the distance to the center node as one, and then each of them forwards to its UTC neighbors a message containing its own ID and a distance counter, denoted by d_t. The newly flooded UTCs update their distance counter values by adding one. This process is repeated until all UTCs are flooded.

After tetrahedral flooding, we can obtain the so-called crust [Gold 1999; Gold and Snoeyink 2001], which is composed of the UTCs sharing the same value of d_t, as shown in Figure 3. We observe that the crust with a small d_t is near the center node and looks like a hollow sphere that is attached to the crust with $d_t - 1$. This motivates us to extract an exterior surface of the crust when d_t is small, on which we can propose a scheme to generate a traversal path on the exterior surface. However, the farther from the center, the more disconnected the crust (as well as its exterior surface) would be. Note that in Figure 3, the crust is connected when $d_t \leq 5$, whereas it is disconnected when $d_t \geq 6$. This is caused by the concavity variation of the topology that could slice the crust into separate patches as d_t increases. In this case, it is required to traverse the nodes, patch by patch passing through UTCs with different d_t, which is more challenging than traversing on the exterior surface of the connected crust.

To deal with this technical challenge, our approach is based on the following observation: the exterior surface of the polyhedron formed by the UTCs with $d_t \leq n$ is connected and closed (to be proved in Section 2.4). Denote P_n as the polyhedron formed by the UTCs with $d_t \leq n$ and S_n as the exterior surface of P_n. Exactly, S_n is composed of a set of triangular faces. Among the set of UTCs that form P_n, each triangular face in S_n is only contained in one of the UTCs. By using this property, these faces can be identified in a similar way that we obtain the boundary surface of the 3D volumetric network. Therefore, a series of S_n can be identified as P_n evolves. Then the network can be disintegrated into a series of layers (i.e., S_1, S_2...) forming a layering structure of the network, as shown in Figure 2(c).

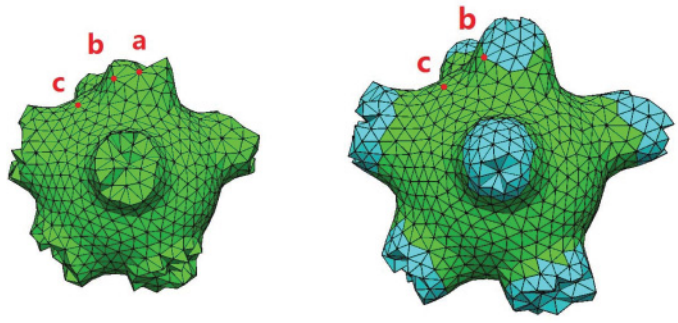
2.4. Theoretical Insights

Earlier, we described the process of layer construction. In this section, we reveal several properties of the constructed layer S_n, which is requisite for traversing the nodes within each layer, as will be discussed in the next section.

One essential property of the constructed layer S_n is its closeness and connectivity as mentioned previously. Here we validate this property by the following theorem.

THEOREM 2.9. *S_n is a connected and closed triangular mesh structure.*

PROOF. Theorem 2.9 can be proved by mathematical induction. Lemma 2.8 interprets the basic case when $n = 1$. Now assuming that Theorem 2.9 is true when $n = k$ ($k > 1$), we next prove its correctness when $n = k + 1$. The situation can be distinguished according to whether a node of S_k is an interior node or not, as shown in Figure 4. Suppose that an interior node a lies on the exterior surface of P_k, and there must be at least one UTC belonging to both $ngb(a)$ and P_k. Then we have $ngb(a) \bigcap P_k \neq \varnothing$, and hence $ngb(a) \bigcup P_k$ forms a new solid polyhedron. Thus, $P_{k+1} = (\bigcup_{v \in S_k} ngb(v)) \bigcup P_k$, is solid, and naturally has a connected and closed triangular mesh structure since P_{k+1} is composed by a number of UTCs side by side. On the other hand, for a boundary node, there are two cases: (a) it is surrounded by UTCs contained in P_k (see node c in Figure 4), or (b) some UTCs are contained in P_k whereas the others are not (see node b in Figure 4). In the former case, there will be no newly flooded UTC and the portion



(a) S_k (b) S_{k+1}

Fig. 4. Evolvement of S_k to S_{k+1}. The UTC colored in green has a $d_t \leq k$, whereas the UTC with a $d_t = k+1$ is colored in cyan.

of the UTC structure around b would not change. For the latter case, $ngb(b) \bigcap P_k \neq \varnothing$ still remains valid. All in all, Theorem 2.9 stands. □

Another indispensable property of the layer S_n is that any node in the network must lie on at least one layer. With this property, if we traverse all layers in the network and ensure that nodes in each layer are all covered, there would be no uncovered node remaining. This property can be deduced by the following theorem.

THEOREM 2.10. *Let us classify the set of nodes that are k-hop away from the center node into two kinds: interior nodes I_k and boundary nodes B_k. Denote the set of nodes in the network as N and the nodes on S_k as N_{S_k}, then we have (a) $N_{S_m} = (\bigcup_{k=1}^{m} B_k) \bigcup I_m$; (b) $N = \bigcup_k N_{S_k}$.*

PROOF. We would demonstrate our proof in sequence. (a) Based on the definition of the neighborship of UTCs, no matter whether a node is an interior node or not, it must be included in N_{S_k} if its distance to the center node is equal to k. Meanwhile, if such a node is an interior node, it is not included in $N_{S_{k+1}}$ because it is wrapped by some UTCs with $d_t = k+1$ and thus cannot lie on S_{k+1} (i.e., see node a in Figure 4). On the contrary, if such a node is a boundary node, the node itself is the outmost one of the network and would locate on S_k, S_{k+1}, S_{k+2} ... continuously. Above all, the first formula holds. (b) It is obvious that $N = \bigcup_k (I_k \bigcup B_k)$. Considering what we have proved in (a), we can derive that $(I_k \bigcup B_k) \subseteq N_{S_k}$ and $\bigcup_k (I_k \bigcup B_k) \subseteq \bigcup_k N_{S_k}$ (i.e., $N \subseteq \bigcup_k N_{S_k}$); on the other hand, $N \supseteq \bigcup_k N_{S_k}$. Therefore, we have $N = \bigcup_k N_{S_k}$. □

Theorem 2.10 implies that the network can be decomposed into a series of connected and closed layers, each of which consists of the interior nodes that are exactly n-hop away from the center node and the boundary nodes with the hop distance to the center node no more than n. Initially when n is small, S_n would look like an arbitrary-shaped polyhedron, as the nodes around the center node are interior nodes and randomly distributed in the space. But as n increases, some concave parts of the topology stay stationary as P_n evolves. That is because for a general 3D volumetric network with an irregular topology, we cannot decompose it into a set of connected and closed layers that look like a set of concentric spheres. Thus, in our design, some boundary nodes have to reappear in different layers (e.g., see nodes b and c in Figure 4; each of them are simultaneously on both S_k and S_{k+1}), serving to render each layer connected and closed, as well as providing great benefits, as we will illustrate in the next section, for extracting a traversal path on these layers.

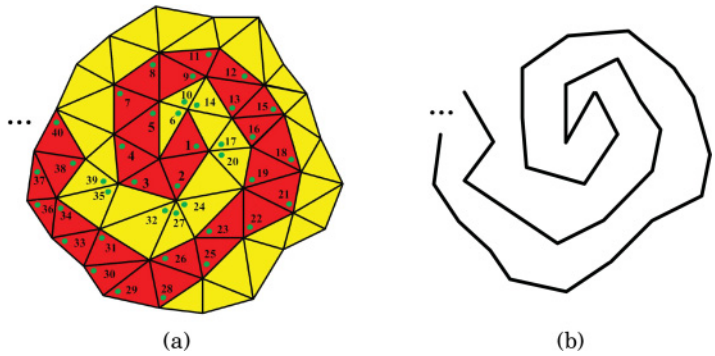


Fig. 5. (a) Strip construction: the angles are numbered in the order as they are labeled, starting from the start angle; red triangular faces are labeled. (b) The traversal loop based on the strip.

3. SFC CONSTRUCTION

Given the layers obtained in the previous section, here we put forward a novel distributed scheme for SFC construction. We first strive to construct a spiral-like triangular strip within each layer, where a traversal path (or a local SFC) can be achieved by moving along the edge of the strip. Next, we show with the generated strips how to construct the final SFC by conducting lightweight intra- and interlayer traversal schemes.

The main idea of the traversal within one layer is to select a set of edges, yielding a continuous path that realizes full-layer coverage. To achieve this goal, we propose to classify the triangular faces on the layer into two kinds: labeled and unlabeled. The labeled region makes up a spiral-like triangular strip, which is bounded by a number of edges, as shown in Figure 5(a). This spiral-like strip actually specifies a loop (see Figure 5(b)) such that if we move along the loop in a fixed direction, we will go back to the start point, after covering most, and ideally all, nodes on the layer. It is noted that our traversal scheme, as a distributed algorithm, may not generate a complete curve with a full coverage on the layer, possibly leaving a few nodes isolated (the so-called dead ends). We will discuss how to deal with the dead ends at the end of this section.

The spiral-like strip is constructed in a greedy manner—that is, a labeled face selects one unlabeled neighbor and sets it labeled, and this is repeated by the newly labeled face until there is no alternative to continue. The whole process can be divided mainly into three steps: (a) set up a direction on the layer, (b) define the neighborship of angles, and (c) sketch out a strip. We note that the former two steps can be regarded as preparations for the final strip construction, and the details of each step are presented as follows.

The first step is to inform all faces on the layer of a universal direction to ensure that the strip winds in a fixed orientation. To accomplish the uniformity of direction, we are motivated by the observation that in a pair of neighboring faces, the direction that the common edge stands for is opposite. For example, in Figure 6(a), the edge e_{bc} (from b to c) represents clockwise in face f_{abc} but counter-clockwise in face f_{bcd}. Denote the layer to be traversed as S_k and the first node on S_k to be covered as the start node s_k. We implement the uniformity of direction as follows. Initially, all faces and edges on S_k are marked as undirected, and one of the triangular faces associated with s_k is randomly selected and labeled as the start face, denoted by f_{s_k}, to be the embryonic strip. Then the direction is initialized in f_{s_k} by storing an edge direction chart in each node of f_{s_k}, and f_{s_k} forwards the information to its adjacent faces. When an undirected face receives such a message, it extracts the original direction that the common edge

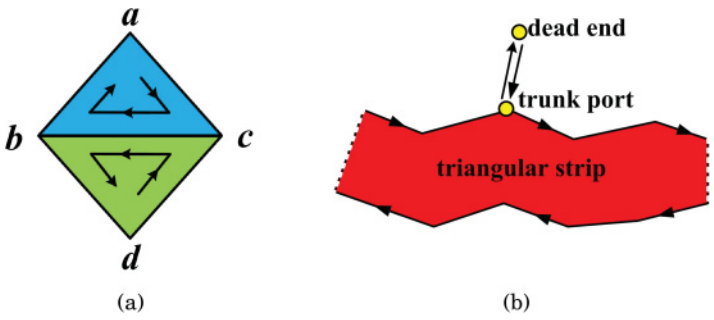


Fig. 6. (a) Establishing the universal direction. (b) Dealing with the dead end.

represents and sets its own with an opposite value. After that, it sets itself as directed and continues the forwarding as mentioned earlier until all faces are marked directed. Note that the direction we define here does not exist in reality but can guarantee that the strip extends without self-intersection. Either clockwise or counterclockwise would be feasible for our scheme.

Based on the universal direction set earlier, we next define the neighborship of angles, which helps an angle in a face differentiate the other two by their relative positions. These settings are capitalized to determine which face is further labeled and indicate how the strip extends. To provide an illustration of the scheme, we assume a clockwise direction. An angle identifies the other two as its n-neighbor/p-neighbor to be the next/previous clockwise angle around it, and a pair of angles are mutually o-neighbors if they share the same opposite edge. For instance, θ_{bac} in Figure 6(a) identifies θ_{acb}, θ_{abc}, θ_{bdc} as its n-neighbor, p-neighbor, and o-neighbor, respectively. Note that since each layer is connected and closed, as stated in Theorem 2.9, all angles on one layer have these three kinds of neighbors without exception.

With the preceding two steps, we now explain how the strip is constructed. Denote the start face as f_{abc} and the start angle (a randomly selected angle in f_{abc}) as θ_{abc}. We use the angle θ and the face f_θ (θ belongs to f_θ) to guide the labeling. To start with, all faces and vertices are set unlabeled. θ is updated according to whether the node of θ is labeled or not, as shown in Figure 5(a). If the node of θ has not been labeled (see angle 1 in Figure 5(a)), the three nodes of f_θ together with f_θ are set as labeled and θ is updated by the o-neighbor of its p-neighbor, expanding the strip clockwise. Otherwise, if the node of θ has been labeled (see angle 5 in Figure 5(a)), we do not label f_θ but update θ with the o-neighbor of its o-neighbor's p-neighbor. These steps run in a recursive way until θ returns to the start angle. The whole process of strip construction is sketched in Algorithm 1, and the results are illustrated in Figures 2(d) and 7.

After the strip construction, we can now focus on SFC construction, which includes the traversals' intra- and interlayers. Recall that the strip construction divides each layer into two parts: labeled faces and unlabeled faces (see Figure 7). The labeled faces make up a spiral-like triangular strip, along which the traversal intralayer can be conducted in a cyclic order with predefined direction on the layer: any node on the edge of the strip can initiate the traversal on the layer, and the traversal proceeds by the current node selecting the uncovered neighbor on the strip as its next hop. This kind of cyclic traversal ensures that every node on the edge of the strip is covered only once. However, as we mentioned earlier, the spiral-like strip of each layer cannot guarantee that all nodes are on the edge of the strip, possibly leaving a few isolated nodes uncovered. We denote these isolated nodes as dead ends. Once this situation occurs, a trivial distributed scheme is conducted: the dead end simply chooses one of its neighbors on the strip as a trunk port to construct a pair of back-and-forth loops

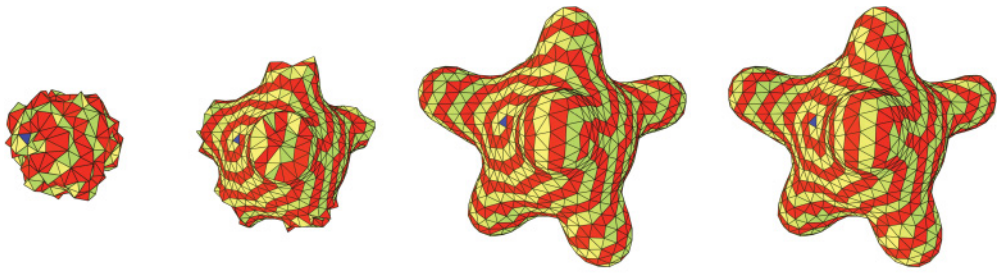


Fig. 7. (a)–(c) Spiral-like triangular strips for different layers; red faces are labeled, and the start face is marked in blue. (d) Counterclockwise strips compared to the clockwise one in (c); both provide a path for the complete coverage of the layer. It is noted that (c) is the same result as Figure 2(d).

ALGORITHM 1: Strip Construction

Input: Triangular mesh M, start face f_{abc}, start angle θ_{abc}.
Output: Triangular strip for intralayer traversal.
for *each triangular face* f_{ijk} *and each edge* e_{ij} **do**
    $f_{ijk} \leftarrow$ undirected, $e_{ij} \leftarrow 0$;
end
set the direction for f_{abc} such that $e_{ab} \leftarrow 1, e_{bc} \leftarrow 1, e_{ca} \leftarrow 1, f_{abc} \leftarrow$ directed;
for *each undirected face* f_{ijk} **do**
    if *one of its neighboring faces, say* f_{jkl}, *has been directed* **then**
        $e_{jk} \leftarrow -e_{jk}, e_{ki} \leftarrow e_{jk}, e_{ij} \leftarrow e_{jk}, f_{ijk} \leftarrow$ directed;
    end
end
for *an angle* θ_{ijk} *in a directed face* **do**
    if $e_{ij} = 1$ **then**
        θ_{ijk}'s n-neighbor $\leftarrow \theta_{ikj}$, θ_{ijk}'s p-neighbor $\leftarrow \theta_{jik}$;
    else
        θ_{ijk}'s n-neighbor $\leftarrow \theta_{ikj}$, θ_{jik}'s p-neighbor $\leftarrow \theta_{ijk}$;
    end
end
for *a pair of angles* θ_{jik}, θ_{jlk} *that share the same opposite edge* e_{jk} **do**
    θ_{jik}'s o-neighbor $\leftarrow \theta_{jlk}$, θ_{jlk}'s o-neighbor $\leftarrow \theta_{jik}$
end
$\theta = \theta_{abc}, f_\theta = f_{abc}$;
repeat
    if *the node of* θ *is unlabeled* **then**
        $\theta \leftarrow \theta$'s p-neighbor, $f_\theta \leftarrow$ labeled, all vertices of $f_\theta \leftarrow$ labeled;
    else
        $\theta \leftarrow$ the p-neighbor of θ's o-neighbor;
    end
    $\theta \leftarrow \theta$'s o-neighbor;
until $\theta = \theta_{abc}$;

between them (see Figure 6(b)). If one dead end finds that the neighbor it selects has already been selected by another dead end, it simply changes the neighbor for a new one. In this case, the nodes taking the role of connecting the dead end to the strip will be covered twice.

With the method for traversing the intralayer, we then clarify how to traverse interlayers, by which the traversal paths are tied together, taking shape in a complete SFC

for the network. Similar to s_k, we denote the last node on S_k to be covered as the end node e_k. To begin with, the center node randomly chooses a neighbor as its next hop (i.e., s_1), which obviously lies on S_1. After all nodes on S_1 are traversed, e_1 randomly chooses a neighbor on S_2 as s_2 and then it can proceed in a recursive way. Note that during this process, when s_k finds itself a dead end, it simply finds a non–dead-end node to proceed by using a way similar to the expanding ring algorithm [Johnson and Maltz 1996]. It is also noted that if e_k is a boundary node, it simply makes itself as s_{k+1}, as it must locate on S_{k+1}. By doing so, this iterative process can end up associating the traversal path on the outmost layer and then the SFC of the whole network is finally constructed, as shown in Figure 2(f).

4. DISCUSSION

4.1. Node's Covered Times

For SFC construction in WSNs, one major concern is how many times a node has been covered by the constructed SFC. In this section, we expound upon this issue of our proposed algorithm.

As mentioned in the previous section, BLOW-UP guarantees that each node in the network is covered at least once. Meanwhile, as some boundary nodes could be recovered on different layers, they are inevitably covered more than once. Therefore, we intend to figure out the upper bound of the number of a node's covered times by the generated SFC. In the following, we prove that the number of covered times of each node is relevant to two parameters concerning the network topology and is bounded within a constant number for a given network. For the sake of convenience, hereinafter we use d_{max} (d_{min}) to denote the hop count distance between the center node and the farthest (nearest) boundary node from it.

Recall that Theorem 2.10 indicates that a boundary node may stay on different layers while each interior node locates on a unique one. On the other hand, the nodes associated with the polyhedron are interior nodes until the polyhedron "grows" to $P_{d_{min}}$. Thereafter, boundary nodes emerge asynchronously, according to their distances to the center node. Thus, the maximum number of times that a node occurs on different layers is bounded by $d_{max} - d_{min}$. In addition, each node exactly only on one layer is covered at most twice. Therefore, the number of covered times of an arbitrary node can be bounded by $2(d_{max} - d_{min})$. In fact, in most applications for large-scale 3D volumetric WSNs, such as the underwater detection or atmospheric monitoring, the gap between d_{max} and d_{min} is considered negligible compared to d_{max} or d_{min} itself. However, when it comes to those networks with narrow line-like bottlenecks, the difference between d_{max} and d_{min} would significantly affect the performance of the proposed algorithm. For those cases, a more advisable method is to segment the network into a set of subnetworks [Zhou et al. 2011; Tan et al. 2014; Jiang et al. 2015] followed by applying our algorithm in each subnetwork.

4.2. Storage Cost, Message Cost, and Time Complexity

In this section, we study the storage cost and message cost of BLOW-UP, which are important factors for the scalability of a distributed algorithm. First, we introduce the following lemma.

LEMMA 4.1. *For a polyhedron with triangular faces, the number of faces is approximately twice that of nodes if the number of nodes is sufficiently large.*

PROOF. We begin the proof with the Euler equation in graph theory that yields for any polyhedron $V - E + F = 2$, where V, E, and F are respectively the numbers of nodes, edges, and triangular faces in the polyhedron. Since every edge is shared by

two different faces, we thus have $E = 3F/2$. By substitution, we have $F = 2(V - 2)$. Therefore, if the number of nodes is sufficiently large, we can see that there are approximately twice as many faces as nodes. In other words, Lemma 4.1 holds. □

THEOREM 4.2. *The per-node storage cost and message cost of BLOW-UP is at most* $O(1)$ *and* $O(d_{max})$*, respectively.*

PROOF. Let us see the storage cost at first. For a node a, it stores local information of its neighbor nodes, the UTCs that surround a, and the set of angles associated with node a. According to Lemma 2.8, the nodes on the polyhedron formed by $ngb(a)$ stand for the neighboring nodes of a, and the number of the polyhedron's exterior faces represents the size of UTCs—that is, two times as many as a's neighbors, as we proved in Lemma 4.1. In addition, the number of angles associated with a is equal to that of the faces inside the polyhedron and is three times that of a's neighbors. Overall, each node maintains the local information with $O(1)$ storage cost.

Next, we turn to the message cost. First, to identify a center node, every node updates its h_t value and discards the message containing a h_t larger than its maintained one. Thus, the message cost for each node at this step is related to its distance to the boundary, which is at most $O(d_{max})$. Second, the center node initiates a tetrahedral flooding, and every node communicates to its neighbors with a message cost $O(1)$. Third, in the process of constructing a triangular strip for a layer, a flooding is performed to set a universal direction, which generates $O(1)$ message cost. Additionally, according to Lemma 4.1, there are approximately twice as many faces as nodes, and thus the number of angles is six times that of nodes. Since each angle is visited at most once, this introduces $O(1)$ message cost. Similar steps are executed on each layer, so the overall message complexity depends on how many times a node occurs on different layers (i.e., at most $O(d_{max} - d_{min})$). Last, the traversal paths are tied together by the start/end nodes with a message cost $O(1)$. Overall, the per-node message cost of BLOW-UP is at most $O(d_{max})$. □

As the storage cost of a node depends on the number of its neighbors, and the message cost of a node is dominated by its hop count distance to the boundary, according to Theorem 4.3, the storage cost and the message cost of each node are both close to a constant and therefore scale well with the network size.

We further explore the time complexity of BLOW-UP as follows.

THEOREM 4.3. *The time complexity of BLOW-UP is* $O(nd_{max})$*, where n is network size.*

PROOF. First, the time complexity to identify the center node depends on the communication of the two farthest nodes in the network, which is $O(d_{max})$. Second, the time complexity of the tetrahedral flooding is related to the distance from the center node to the boundary, which is $O(d_{max})$. Third, the time complexity for constructing a triangular strip for each layer is $O(nd_{max})$, as the strip covers all nodes in the network, with duplicated coverage of part boundary nodes. Finally, the traversal process has a time complexity of $O(n)$. To sum up, the time complexity of BLOW-UP is thus $O(nd_{max})$. □

4.3. BLOW-UP for Adaptive Coverage

In reality, there are some cases in which the in-network data aggregation is implemented in a parallel manner—for instance, several mobile sinks are adopted to simultaneously tour around the network collecting data stored by each sensor, as in Wang et al. [2012] and Ban et al. [2013]. In such collaborative cases, mobile sinks may be far away from each other, and it will be hard to conduct mutual communications. Thus, there might be a large amount of duplicate visits by different sinks. Thanks to the layering strategy of our scheme, we could assign each mobile sink to cover the nodes

Table I. Comparisons of the Three 3D SFC Construction Schemes

Scheme	Feature	Traversal Method
Preorder traversal	Tree based	By traversing along one of its subtrees before visiting another.
Random walk	Randomized	By randomly choosing one of its neighbor as its next traversal node.
BLOW-UP	Well guided	By constructing a spiral-like triangular mesh on each layer, along which to traverse the node's intralayer.

on several layers (or only one layer) respectively, according to the number of layers and sinks. As proved in Theorem 2.10 that only a few boundary nodes reappear in different layers, this notion would greatly reduce the interference among different sinks.

Similar to breaking down the traversal task by layers and allocating each portion to different sinks, we can also derive a coarse-grained traversal by visiting nodes every other layer (or every several layers). This kind of adaptive coverage is useful when we only have one sink at hand and are faced with restricted budgets of travel length or fusion delay [Ban et al. 2013]. In such cases, some nodes are abnegated to be uncovered to meet the restrictive requirements. We argue that the to-be-covered nodes are representative enough to depict the whole network as they are relatively distributed uniformly among all corners of the deployment area. In this point, it is consistent with our initial intention to quickly "browse" the network.

5. PERFORMANCE EVALUATION

We have applied the proposed BLOW-UP algorithm to various 3D volumetric topologies in different sizes and shapes. In addition to the network model shown in Figure 2, Figure 8 illustrates several other examples, where the first row shows original networks; the second row illustrates the set of layers acquired by tetrahedral flooding; the third row exhibits the strip on the outmost layer, providing a traversal path of the layer; and the fourth row depicts the final SFC. Sensor nodes are randomly distributed and have an average degree around 12. It can be seen that despite the variation in scale and complexity, BLOW-UP, after decomposing the original network into layers, is always able to construct an SFC via traversing the network layer by layer.

In the following, to further evaluate the performance of BLOW-UP, we mainly focus on three factors: a node's covered times, coverage rate (the ratio of the number of covered nodes to the number of all nodes in the network), and covering speed (the number of newly covered nodes during a fixed interval of the path length). Although several linearization algorithms [Ban et al. 2013; Mostefaoui et al. 2015; Wang and Jiang 2015] have been proposed for WSNs, none of them can be directly applied in a 3D volumetric space. Therefore, we compare our algorithm only with two other comparable schemes: tree-based preorder traversal [Liang et al. 2013] and random walk [Spitzer 2001] (with their the differences noted in Table I). We first construct a spanning tree rooted at the center node and then adopt preorder traversal. In particular, preorder traversal is a kind of depth-first search and achieves a shorter path length than the breadth-first one. When a node and its children are all covered, this node returns to its father node and continues to find other children of its father, doubling all of the edges to generate a continuous path, as shown in the fifth row of Figure 8. For random walk scheme, the current node simply chooses one of its neighbors to be the next hop. Because of the stochastic characteristics, we conduct the experiments 100 times to obtain average performance.

The histogram in Figure 9 illustrates the distribution of a node's covered times based on our proposed algorithm, preorder traversal, and random walk under different network models. From simulations, we find out that in our scheme, most nodes are covered only once or twice and the maximum number of node's covered times over

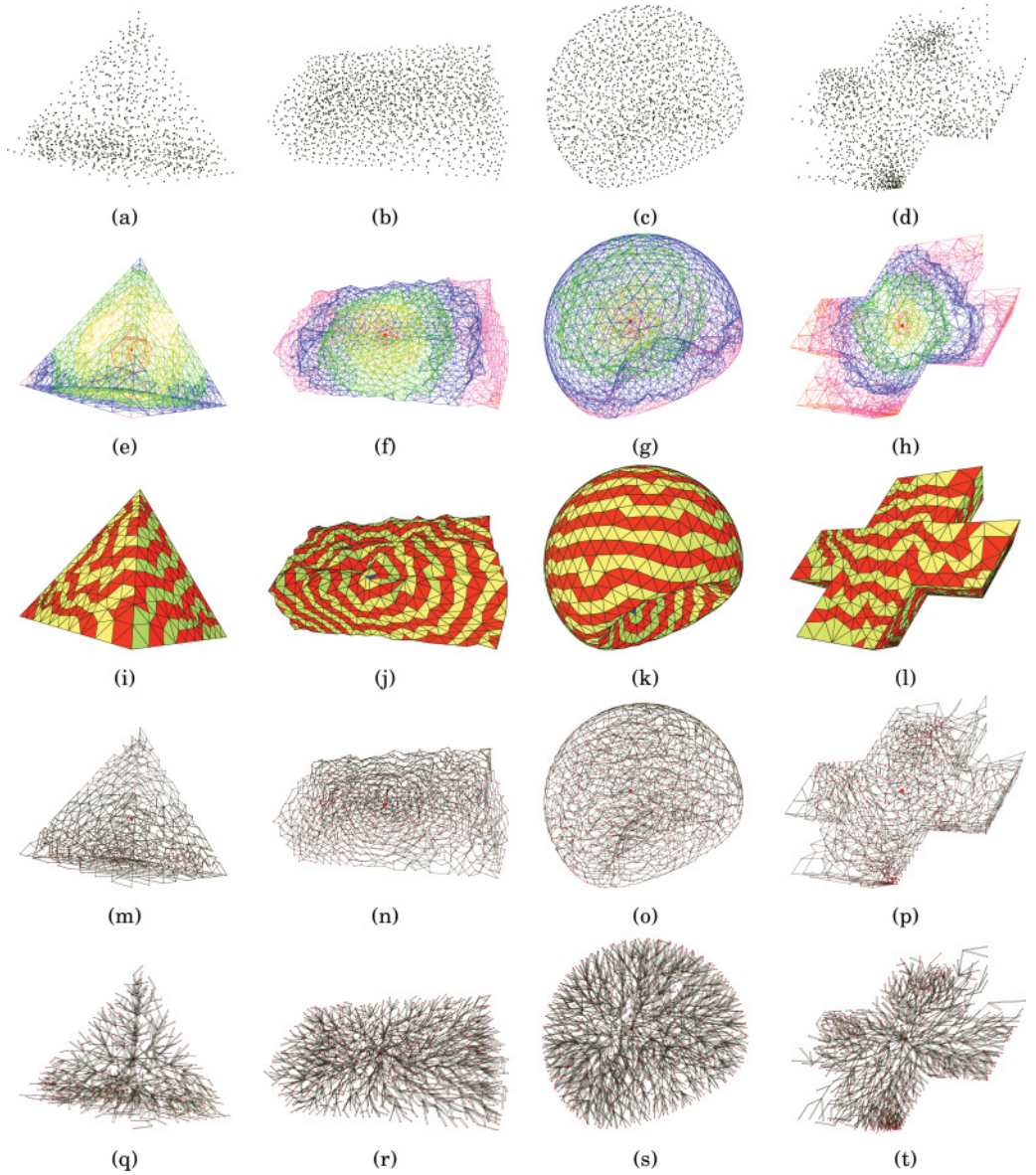


Fig. 8. Columns from left to right: (a) A pyramid network with 993 nodes; average degree is 11.63; $d_{min} = 3$, $d_{max} = 9$. (b) A pool network with 1,721 nodes; average degree is 11.72; $d_{min} = 4$, $d_{max} = 11$. (c) A stadium network with 1,922 nodes; average degree is 12.04; $d_{min} = 5$, $d_{max} = 8$. (d) A crossroad network with 1,608 nodes; average degree is 12.29; $d_{min} = 4$, $d_{max} = 11$. Rows: (1) The original network. (2) The connected and closed layers. (3) The spiral-like triangular strip on the outmost layer. (4) The constructed SFC by BLOW-UP. (5) The traversing path based on preorder traversal.

all scenarios is around 5 to 7, indicating that the number of dead ends is not very significant. We emphasize that the upper bound of the number of a node's covered times is $2(d_{max} - d_{min})$, which only stands for the worst case, such as networks with narrow line-like bottlenecks as discussed in Section 4.1. On the other hand, a node's

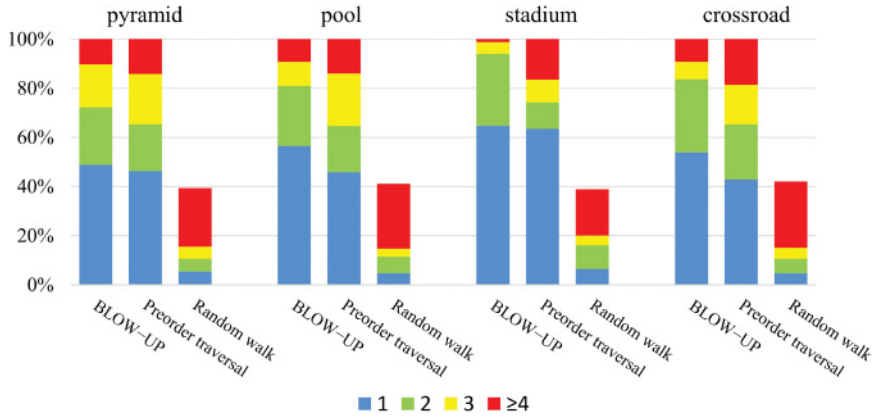
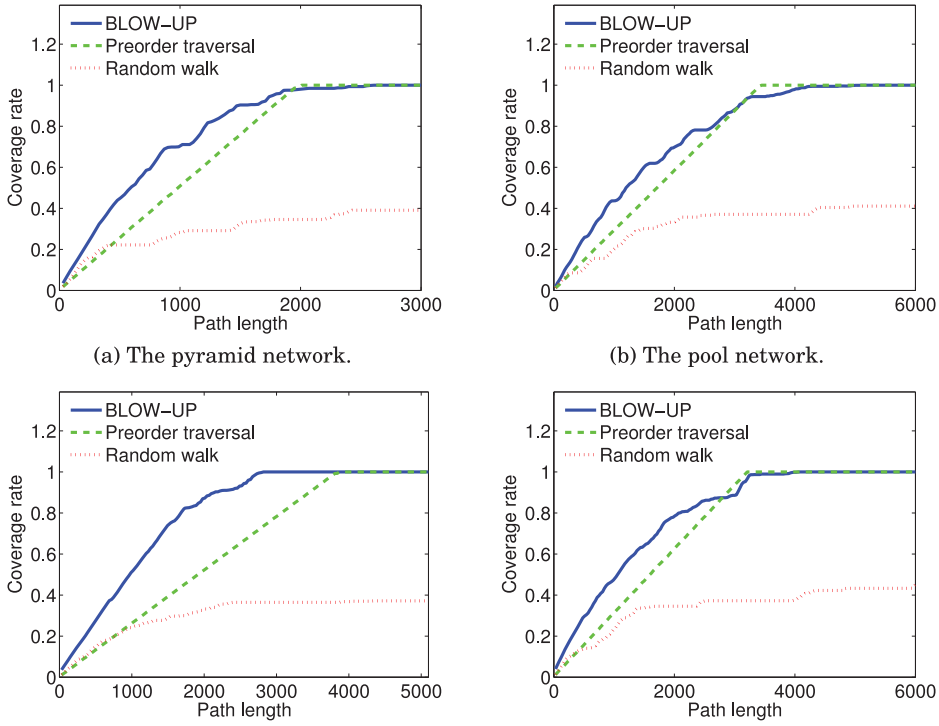


Fig. 9. Distribution of a node's covered times. For the four networks (pyramid, pool, stadium, and crossroad), (i) the average numbers of a node's covered times using BLOW-UP are 1.93, 1.83, 1.44, and 1.82; (ii) the average numbers of a node's covered times using preorder traversal are 2.17, 2.21, 2.03, and 2.33; and (iii) random walk cannot achieve a full coverage, and thus the average numbers of a node's covered times are neglected.

covered times in preorder traversal is equal to the number of its children plus one. That is because of the backtracking on itself when a subtree is traversed, which undermines a traversal's efficiency. As for random walk, it can be easily observed that it heavily suffers from reduplicate visits and incomplete coverage, as more than half of the nodes are left uncovered and most of the already-covered nodes are visited more than four times.

The network coverage rates of our proposed algorithm, preorder traversal, and random walk, as the paths move forward in different networks, are shown in Figure 10. For preorder traversal, every node nearly has the similar amount of children and executes the single stereotype of addressing. Thus, its covering speed approximately stays consistent, as shown in Figure 10 where the first half of the curve is nearly a straight line. Meanwhile, as the tree structure preserves the relationship between neighboring nodes, it is natural that the preorder traversal finally achieves 100% coverage. For random walk, as can been seen, it is far less effective both in coverage rate and covering speed. Because of its memorylessness and aimlessness, random walk is more likely to take a lot of time carrying out repeated visits in the already covered area, thereby missing information of the uncovered regions. That is why its covering speed becomes much slower, almost to zero in the final stage. Thus, the coverage rate of random walk reaches nearly 40%, whereas that of the other two schemes achieves full coverage.

Compared to the other two approaches, BLOW-UP introduces a faster and more efficient traversal. It is well guided to visit as few additional hops as possible. The superiority is even more prominent in a network where d_{max} is close to d_{min} (e.g., the stadium network). The reason is that a small value of $(d_{max} - d_{min})$ leads to a small number of covered times on boundary nodes, as we discussed in Section 4.1. Sometimes the coverage percentage of BLOW-UP at a certain path length may lag behind that of preorder traversal. Nevertheless, this has little impact on the effectiveness of BLOW-UP, as more than 90% of the nodes have already been covered before that time. We note that in sensor network applications such as serial data fusion, the ultimate goal is to have a quick tour of the network coarsely and provide appropriate responses. From this perspective, BLOW-UP outperforms the other two methods in that it traverses the network efficiently and provides a good representation of the network quickly at the early stage.



(a) The pyramid network.

(b) The pool network.

(c) The stadium network.

(d) The crossroad network.

Fig. 10. Comparisons on coverage rate and covering speed for the four networks.

6. CONCLUSION

In this article, we have presented BLOW-UP, a novel distributed and scalable algorithm for SFC construction in 3D volumetric WSNs. BLOW-UP is based on the idea of divide and conquer: a given 3D volumetric network is first decomposed into a series of connected and closed layers, then the nodes are traversed layer by layer, incrementally from the innermost to the outermost, yielding an SFC covering the entire network, provably at least once and at most a constant number of times. BLOW-UP is based on connectivity information only, with no reliance on a node's location information or any particular communication radio models. It is also scalable with a nearly constant per-node storage cost and message cost. Extensive simulations under various networks demonstrate its effectiveness on nodes' covered times, coverage rate, and covering speed. In the future, we plan to design linearization schemes applicable to 3D volumetric networks with more general topologies, say with holes.

REFERENCES

M. Ahmed and S. Bokhari. 2007. Mapping with space filling surfaces. *IEEE Transactions on Parallel and Distributed Systems* 18, 9, 1258–1269.

I. F. Akyildiz and M. C. Vuran. 2010. *Wireless Sensor Networks*. John Wiley & Sons.

M. Bader. 2012. *Space-Filling Curves: An Introduction with Applications in Scientific Computing*. Texts in Computational Science and Engineering, Vol. 9. Springer.

J. M. Bahi, A. Makhoul, and A. Mostefaoui. 2008. Hilbert mobile beacon for localisation and coverage in sensor networks. *International Journal of Systems Science* 39, 11, 1081–1094.

X. Ban, M. Goswami, W. Zeng, X. Gu, and J. Gao. 2013. Topology dependent space filling curves for sensor networks and applications. In *Proceedings of the IEEE INFOCOM Conference (INFOCOM'13)*. 2166–2174.

R. S. Blum, S. A. Kassam, and H. V. Poor. 1997. Distributed detection with multiple sensors II. Advanced topics. *Proceedings of the IEEE* 85, 1, 64–79.

F. Cadger, K. Curran, J. Santos, and S. Moffett. 2013. A survey of geographical routing in wireless ad-hoc networks. *IEEE Communications Surveys and Tutorials* 15, 2, 621–653.

J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho. 2012. A survey and evaluation of topology-agnostic deterministic routing algorithms. *IEEE Transactions on Parallel and Distributed Systems* 23, 3, 405–425.

J. Gao and L. Zhang. 2006. Load-balanced short-path routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems* 17, 4, 377–388.

M. R. Garey, D. S. Johnson, and R. E. Tarjan. 1976. The planar Hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing* 5, 4, 704–714.

C. Gold. 1999. Crust and anti-crust: A one-step boundary and skeleton extraction algorithm. In *Proceedings of ACM Symposium on Computational Geometry*. 189–196.

C. Gold and J. Snoeyink. 2001. A one-step crust and skeleton extraction algorithm. *Algorithmica* 30, 2, 144–163.

D. Hilbert. 1891. Ueber die stetige abbildung einer line auf ein fachenstuck. *Mathematische Annalen* 38, 3, 459–460.

H. Jiang, T. Yu, C. Tian, G. Tan, and C. Wang. 2015. Connectivity-based segmentation in large-scale 2-D/3-D sensor networks: Algorithm and applications. *IEEE/ACM Transactions on Networking* 23, 1, 15–27.

D. B. Johnson and D. A. Maltz. 1996. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*. Kluwer International Series in Engineering and Computer Science, Vol. 353. Kluwer, 153–181.

M. Kamat, A. S. Ismail, and S. Olariu. 2007. Modified Hilbert space-filling curve for ellipsoidal coverage in wireless ad hoc sensor networks. In *Proceedings of the IEEE ICSPC Conference (ICSPC'07)*. 1407–1410.

S. Kar, J. M. F. Moura, and K. Ramanan. 2012. Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication. *IEEE Transactions on Information Theory* 58, 6, 3575–3605.

D. Koutsonikolas, S. M Das, and Y. C. Hu. 2007. Path planning of mobile landmarks for localization in wireless sensor networks. *Computer Communications* 30, 13, 2577–2592.

F. Kuhn, R. Wattenhofer, and A. Zollinger. 2003. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the ACM MobiHoc Conference (MobiHoc'03)*. 267–278.

F. Li, C. Zhang, J. Luo, S. Xin, and Y. He. 2014. LBDP: Localized boundary detection and parametrization for 3-D sensor networks. *IEEE/ACM Transactions on Networking* 22, 2, 567–579.

W. Liang, P. Schweitzer, and Z. Xu. 2013. Approximation algorithms for capacitated minimum forest problems in wireless sensor networks with a mobile sink. *IEEE Transactions on Computers* 62, 10, 1932–1944.

Y. Luo, L. Pu, M. Zuba, Z. Peng, and J.-H. Cui. 2014. Challenges and opportunities of underwater cognitive acoustic networks. *IEEE Transactions on Emerging Topics in Computing* 2, 2, 109–211.

A. Madhja, S. Nikoletseas, and T. P. Raptis. 2015. Distributed wireless power transfer in sensor networks with multiple mobile chargers. *Computer Networks* 80, 7, 89–108.

E. Masazade, R. Niu, P. K. Varshney, and M. Keskinoz. 2010. Energy aware iterative source localization for wireless sensor networks. *IEEE Transactions on Signal Processing* 58, 9, 4824–4835.

A. Mostefaoui, A. Boukerche, M. A. Merzoug, and M. Melkemi. 2015. A scalable approach for serial data fusion in wireless sensor networks. *Computer Networks* 79, 14, 103–119.

A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis. 2009. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control* 54, 11, 2506–2517.

Y. Noh, U. Lee, P. Wang, B. S. C. Choi, and M. Gerla. 2013. VAPR: Void-aware pressure routing for underwater sensor networks. *IEEE Transactions on Mobile Computing* 12, 5, 895–908.

S. Patil, S. R. Das, and A. Nasipuri. 2004. Serial data fusion using space-filling curves in wireless sensor networks. In *Proceedings of the IEEE SECON Conference (SECON'04)*. 182–190.

G. Peano. 1890. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen* 36, 1, 157–160.

C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos. 2014. Sensor search techniques for sensing as a service architecture for the Internet of Things. *IEEE Sensors Journal* 14, 2, 406–420.

T. L. Porta, C. Petrioli, C. Phillips, and D. Spenza. 2014. Sensor mission assignment in rechargeable wireless sensor networks. *ACM Transactions on Sensor Networks* 10, 4, 60:1–60:39.

R. Rajagopalan and P. K. Varshney. 2006. Data aggregation techniques in sensor networks: A survey. *IEEE Communications Surveys and Tutorials* 8, 4, 48–63.

H. Sagan. 1994. *Space-Filling Curves*. Springer-Verlag, New York, NY.

R. C. Shah, S. Roy, S. Jain, and W. Brunette. 2003. Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks* 1, 2, 215–233.

F. Spitzer. 2001. *Principles of Random Walk*. Vol. 34. Springer.

R. Sugihara and R. K. Gupta. 2011. Path planning of data mules in sensor networks. *ACM Transactions on Sensor Networks* 8, 1, 1:1–1:27.

G. Tan, H. Jiang, J. Liu, and A.-M. Kermarrec. 2014. Convex partitioning of large-scale sensor networks in complex fields: Algorithms and applications. *ACM Transactions on Sensor Networks* 10, 3, 41:1–41:23.

G. Tan, H. Jiang, S. Zhang, Z. Yin, and A.-M. Kermarrec. 2013. Connectivity-based and anchor-free localization in large-scale 2D/3D sensor networks. *ACM Transactions on Sensor Networks* 10, 1, 6:1–6:21.

R. Tan, G. Xing, J. Wang, and B. Liu. 2011. Performance analysis of real-time detection in fusion-based sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 22, 9, 1564–1577.

O. Tekdas, V. Isler, L. J. Hyun, and A. Terzis. 2009. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications* 16, 1, 22–28.

C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy. 2014. Distributed mobile sink routing for wireless sensor networks: A survey. *IEEE Communications Surveys and Tutorials* 16, 2, 877–897.

R. Viswanathan and P. K. Varshney. 1997. Distributed detection with multiple sensors I. Fundamentals. *Proceedings of the IEEE* 85, 1, 54–63.

C. Wang and H. Jiang. 2015. SURF: A connectivity-based space filling curve construction algorithm in high genus 3D surface WSNs. In *Proceedings of the IEEE INFOCOM Conference (INFOCOM'15)*. 981–989.

C. Wang, H. Ma, Y. He, and S. Xiong. 2012. Adaptive approximate data collection for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 23, 6, 1004–1016.

S. Xia, H. Wu, and M. Jin. 2014. Trace-routing in 3D wireless sensor networks: A deterministic approach with constant overhead. In *Proceedings of the ACM MobiHoc Conference (MobiHoc'14)*. 357–366.

S. Xia, X. Yin, H. Wu, M. Jin, and X. D. Gu. 2011. Deterministic greedy routing with guaranteed delivery in 3D wireless sensor networks. In *Proceedings of the ACM MobiHoc Conference (MobiHoc'11)*. 1–10.

L. Xie, Y. Shi, Y. T. Hou, and H. D. Sherali. 2012. Making sensor networks immortal: An energy-renewal approach with wireless power transfer. *IEEE/ACM Transactions on Networking* 20, 6, 1748–1761.

K. Yang. 2014. *Wireless Sensor Networks*. Springer.

F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. 2002. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the ACM MobiCom Conference (MobiCom'02)*. 148–159.

H. Zhang and H. Shen. 2009. Balancing energy consumption to maximize network lifetime in data-gathering sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 20, 10, 1526–1539.

H. Zhou, N. Ding, M. Jin, S. Xia, and H. Wu. 2011. Distributed algorithms for bottleneck identification and segmentation in 3D wireless sensor networks. In *Proceedings of the IEEE SECON Conference (SECON'11)*. 494–502.

M. Zhu, S. Ding, Q. Wu, R. R. Brooks, N. S. V. Rao, and S. S. Iyengar. 2010. Fusion of threshold rules for target detection in wireless sensor networks. *ACM Transactions on Sensor Networks* 6, 2, 18:1–18:7.

Received April 2015; revised January 2016; accepted June 2016