Wireless Networks (2021) 27:3073–3089
https://doi.org/10.1007/s11276-021-02569-3

Energy-efficient routing in the proximity of a complicated hole in wireless sensor networks

Khanh-Van Nguyen[1] · Chi-Hieu Nguyen[1] · Phi Le Nguyen[1] · Tien Van Do[2] · Imrich Chlamtac[3]

Accepted: 8 February 2021 / Published online: 5 March 2021

Abstract
A quest for geographic routing schemes of wireless sensor networks when sensor nodes are deployed in areas with obstacles has resulted in numerous ingenious proposals and techniques. However, there is a lack of solutions for complicated cases wherein the source or the sink nodes are located close to a specific hole, especially in cavern-like regions of large complex-shaped holes. In this paper, we propose a geographic routing scheme to deal with the existence of complicated-shape holes in an effective manner. Our proposed routing scheme achieves routes around holes with the $(1+\epsilon)$-stretch. Experimental results show that our routing scheme yields the highest load balancing and the most extended network lifetime compared to other well-known routing algorithms as well.

Keywords Wireless sensor network · Geometric routing · Routing hole

1 Introduction

Wireless sensor nodes can be applied to build many real-life applications (medical and health, environmental monitoring, transportation and logistics, etc.) [1]. Sensor nodes for environmental monitoring may be deployed in areas with obstacles (holes), such as waterlogged areas, that poses a challenge for the design of geographic routing. Natural obstacles, such as ponds or lakes (for example Dai Lai lake in Vinh Phuc Vietnam, Balaton lake in Hungary, Boy Lake in Minnesota, Lago do Arroz Lake in Brazil, Ozero Lake in Russia, Laguna Yanauyac Lake in Peru, etc.), may contain significant concave parts. Figure 1 plots the deployment example of a wireless sensor network around a lake with cavern-like regions. In such cases a routing procedure should take into account a problem of traffic build-up in nodes near the cavern. Most existing hole-bypassing proposals that consider the load-balancing issues still could not solve the problem of traffic hotspots when either the senders or the recipients are in cavern-like regions. It worth emphasizing that greedy geographic routing algorithms [2–7] do not work well in the presence of holes. Recently, the problem of "routing in a cavern" has been tackled in [8] and [9]. However, these schemes [8, 9] could not achieve a small route stretch and suffer from possible congestion in hotspots.

In this paper, we propose a load-balancing, constant-stretch routing scheme to deal with caverns around a large hole of complicated shape. Source-destination pair s and t builds k almost-parallel lanes (a "strip" of width k, which is a small constant integer). Then, source node s randomly selects among these lanes to forward a packet, which disperses the traffic between s and t. Since the k-width strip is aligned with the shortest path between s and t, any almost-parallel lane to this k-strip would have a length not far-off from the shortest path. This technique is termed *k-multi-lane path routing*, or *k-MLP routing* in this paper. The routing scheme could achieve a good trade-off between shortening routes and load-balancing by applying

✉ Phi Le Nguyen
lenp@soict.hust.edu.vn

✉ Tien Van Do
do@hit.bme.hu

Khanh-Van Nguyen
vannk@soict.hust.edu.vn

Chi-Hieu Nguyen
20151331@student.hust.edu.vn

[1] Hanoi University of Science and Technology, Hanoi, Vietnam

[2] Department of Networked Systems and Services, Budapest University of Technology and Economics, Budapest, Hungary

[3] University of Trento, Trento, Italy

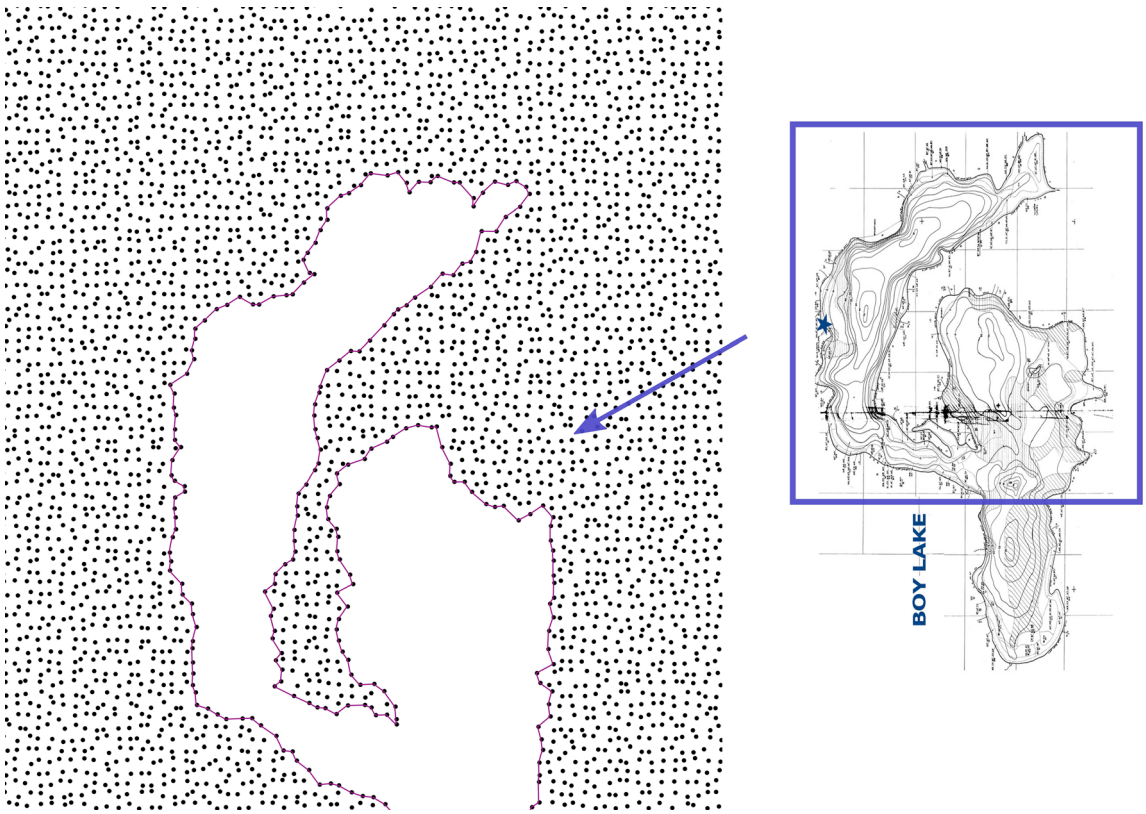


Fig. 1 An imaginary setup of a WSN around the Boy Lake in Minnesota. The blue shape is the real-life representation of the lake, the figure is taken from [10]

k-MLP whenever possible. However, for a cavern of complicated shape, it is not always possible to perform k-MLP routing from any given node (e.g., within a narrow alley of width less than k). Thus, for a given integer k, the algorithm must identify the area where k-MLP routing is possible.

To sum up, the main contributions of this paper are as follows.

- We propose *k-MLP routing* technique with the combination of small stretch routing and load-balancing to solve the issue of traffic hotspots completely.
- We rigorously prove that for any given $\epsilon > 0$ and a given network-hole scenario k can be chosen so that the *k-MLP routing* produces routes with $1 + \epsilon$-stretch.
- We compare the proposed *k-MLP routing* with 4 other existing routing schemes. Results show that the *k-MLP routing* achieves significantly longer network lifetime, especially, in problematic scenarios where caverns have a zig-zag turn at the entrance. Table 1 captures a quick look at these comparative results.

It is also worth emphasizing that our *k-MLP routing* proposal can be further developed to solve the problem of planning short paths with clearance through a crowded environment. This problem of planning short path with

Table 1 Summary of performance comparison

	k-MLP	ADAV	STABLE	GPSR	ALBA-R
Regular hole scenarios					
Network lifetime	85.87	70.73	38.22	68.72	78.67
Balancing index	0.20	0.13	0.15	0.04	0.42
Route stretch	1.21	1.22	1.18	4.76	4.29
G-shape hole					
Network lifetime	91.37	52.87	48.02	35.91	48.73
Balancing index	0.16	0.14	0.08	0.06	0.34
Route stretch	1.13	1.14	1.08	4.48	3.07

A quick capture of numerical results in Sect. 6 where our k-MLP is compared to 4 other existing routing schemes. The numerical results in the top half – i.e. with "Regular hole scenarios" – are averaged per multiple regular scenarios and test sizes while those in the bottom half are per the special case, namely the G-shape that has a zig-zag turn at the cavern entrance. *Network lifetime* is defined as the length of time that the network functions as a whole which ends when the first sensor node runs out of energy. *Balance index* (*BI*) [13] is to quantify how well the traffic load is balanced among the sensor nodes: $BI = \frac{(\sum_{i=1}^{M} p_i)^2}{M \sum_{i=1}^{M} p_i^2}$, where M is the number of sensor nodes and p_i is the total number of packets sent or forwarded by the ith node

clearance is crucial in applications for virtual environments such as robotics, computer graphics, simulations, geographic information systems (GIS), very-large-scale integration (VLSI) design, and games [11, 12].

The remainder of this paper is structured as follows. The related works are discussed in Sect. 2. Assumptions and definitions are presented in Sect. 3. Our proposed routing protocol is described in details in Sect. 4. The performance of the proposed protocol is theoretically analyzed in Sect. 5, and then evaluated by experiments in Sect. 6. Finally, the general conclusion and future work are stated in Sect. 7.

2 Related work

Nodes near *holes* should perform a specialised routing procedure to forward packets to their destination, which is a critical issue in the geographic routing. Otherwise, those nodes may not find 1-hop neighbour closer to the destination at the hole's surroundings.

2.1 Hole-bypassing approaches

The "Greedy Perimeter Stateless Routing for Wireless Networks" (GPSR) proposal was the pioneer work [2] to deal with the "local minimum phenomenon" where the simplified approach combines the greedy and perimeter routing modes. That is, nodes may alternatively switch between two modes to overcome the so-called "stuck nodes" problem. The GPSR variants, however, have severe shortcomings because the nodes located on the hole boundary should handle a large amount of traffic. Therefore, the battery of the boundary nodes quickly drain, and consequently, the hole is enlarged. Subramanian et al. [14] show that GPSR can cause the throughput capacity of the whole network to drop due to the traffic build-up significantly. Furthermore, perimeter routing may lead to long routing paths, especially for a hole with a complicated shape, and therefore, result in energy inefficiency for the whole network. In [3], Kuhn et al. prove that perimeter routing can produce path length quadratic of the optimal.

Several later efforts [15–18] follow the regular strategy to explore beforehand and acquire the core information of the holes' shapes. These approaches utilise modest core information to find routes that bypass a hole. Based on this hole-bypassing strategy, many proposals create some special *forbidden* areas that are often minimum coverage of the hole and a particular shape, e.g. a circle [19], a hexagon [20] or an ellipse [21]. Whenever a data packet encounters the forbidden areas, it is pushed back to follow a detour route to the destination.

On the one hand, the hole-bypassing approach decreases traffic on the hole boundary. On the other hand, it may cause traffic congestion around the perimeter of the forbidden area. Some proposals such as [22] introduce a variable distance between the detour and the forbidden area that can be randomly determined to overcome such a problem. Most of the hole-bypassing proposals, however, lead to routing paths that may be unnecessarily long due to the rerouting overhead. That is, the hole-bypassing routing algorithm overestimates the forbidden area compared to the original hole.

Routing algorithms by [5–7] can guarantee short routes with a constant stretch. These algorithms require a lot of communication overhead to acquire and maintain knowledge of the whole network topology (e.g. a visibility graph in [5] or a global road map in [23]) in contrast with the approach based on only local information with limited node states GPSR [2]. Therefore, the proposed algorithms [5–7, 23] are not energy-efficient.

Routing procedures [7, 9, 24, 25] change the shape of the forbidden area of hole H to disperse the traffic on the hole border, reducing its concentration in the hotspots. The distance and position of each node influence the forbidden area with a randomised factor.

Heuristic methods [26–30] maintain load-balancing between all nodes. Here, every node utilises either the workload or the remained energy of the neighbour nodes to make routing decisions. Huang et al. [30] exploit energy to improve the load balance in that every source node sends two so-called burst packets towards the destination before sending data packets where one burst packet is forwarded along the right-hand side, and the other is sent along the left-hand side of the hole. Note that these proposals may lead to the *extra energy consumption* of the sensor nodes due to the *significant overhead* of exchanging burst packets between the nodes.

Recently, IoT technologies have been studied intensively. Thus more effort are made to create and control WSNs in harsh environments where overcoming routing holes is typical. Several protocols have been proposed to solve the void hole problem in underwater WSNs, which is due to frequent topology changes (nodes moving around because of water flows) and signal attenuation and long delay [31–34]. Also, in [35] a virtual force based routing strategy is proposed to handle the energy hole problem, while in [36] a routing algorithm is created to overcome dynamic holes.

Several graph-theoretic or geometric techniques have been introduced to capture the border of the hole (as a polygon) [37, 38]. A technique to bypass a hole in a 3D wireless sensor networks has also been proposed by [39].

2.2 Routing in close proximity to a hole

If a destination t is within the proximity of a specific hole, then the greedy perimeter routing is utilised. However, such a strategy can lead to a significant buildup of traffic on the perimeter of a hole, especially a large one with extensive data transmission to its surrounding area. Figure 2 illustrates such a scenario where sensor nodes at positions P_1 and P_3 collect 2 different important environmental factors in their respective localities and continuously transmit the collected data to the 2 respective base stations at $P4$ and P_5. Also, a sink node is placed at position P_2 to gather other environmental data. Severe hotspots can occur at end-points B and C of the cavern's gate and the convex vertices A and D. As a consequence, the battery of the sensor nodes at these hotspots quickly drains, which further enlarges the hole. Situations like this can lead to unnecessary shortening of the network lifetime.

In [8], a procedure is proposed to avoid creating hotspots around the hole boundary. However, the routing algorithm may compute the unnecessarily high route stretch. In [7, 9, 40], novel techniques are applied to achieve load-balancing and small route stretch. For load-balancing, the common idea is to produce Around a hole, the procedure randomly selects alternative pathways that are similar to the contour lines with a varying degree factor (reflecting the distance to the hole border).

For example, in [9], these alternative pathways can be created through a specific homothetic transformation of the shortest path. This mechanism, however, would cause a severe hotspot if the cavern has a zig-zag turn at the entrance (Fig. 3), e.g., the G-shape hole in Fig. 3. The shortest route from P_1 to P_2 (blue line) switches from the right wall to the left wall of the cavern (looking from the inside), and the lines created by homothetic transformation would converge at the centre point of the cavern mouth.

3 Assumptions and notations

3.1 Assumptions

Following the literature [2–7], wireless network sensors are assumed to be built from the homogeneous radio hardware with a transmission range normalized to 1. Also, each node knows its position (using GPS or other positioning services [41]), its 1-hop neighbours (through the neighbour notification packets), and the source node knows the position of the destination node. The energy consumption of the computation performed for one packet is negligible compared to the energy consumption of the transmission of one packet.

For theoretical analysis, we make a reasonable assumption that the considered network is sufficiently dense such that there are sensors everywhere apart from the considered hole. Given such an ideal situation, we can model the geographical greedy routing path between two given nodes s and t as the Euclidean line connecting s and t.

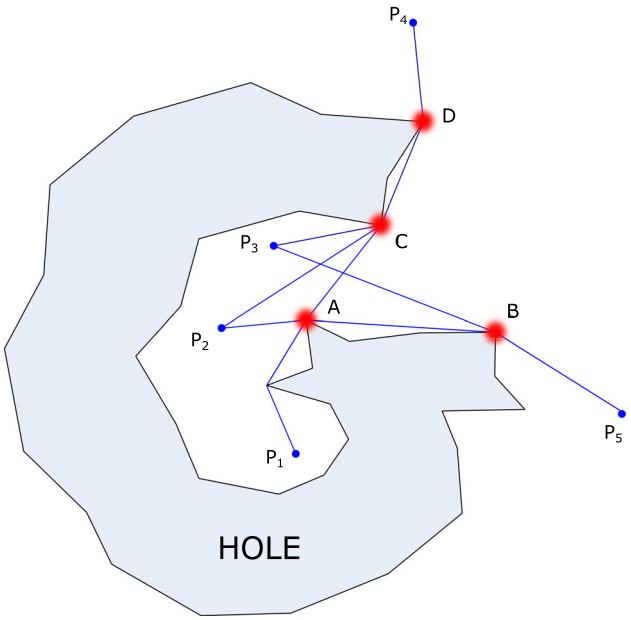


Fig. 2 Illustration of possible hot spots on the boundary of a large cavern

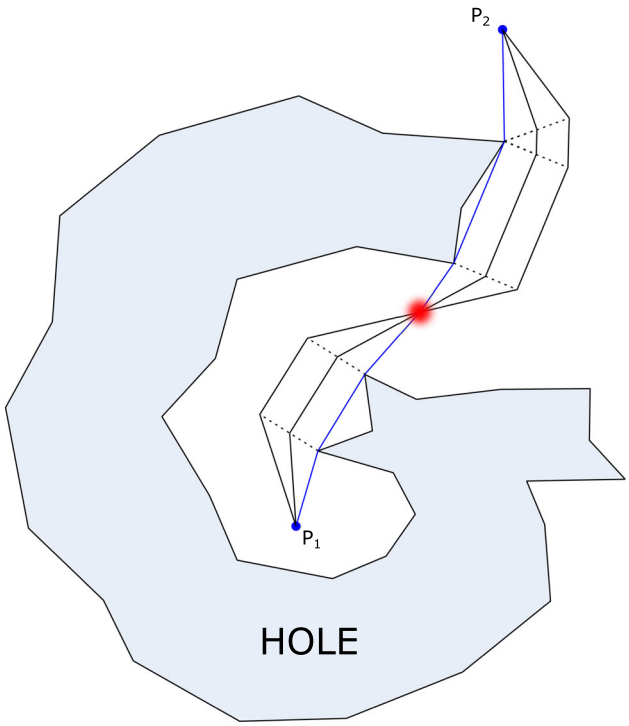


Fig. 3 The G-shape case: a possible hot spot at a zig-zag turn at the cavern entrance

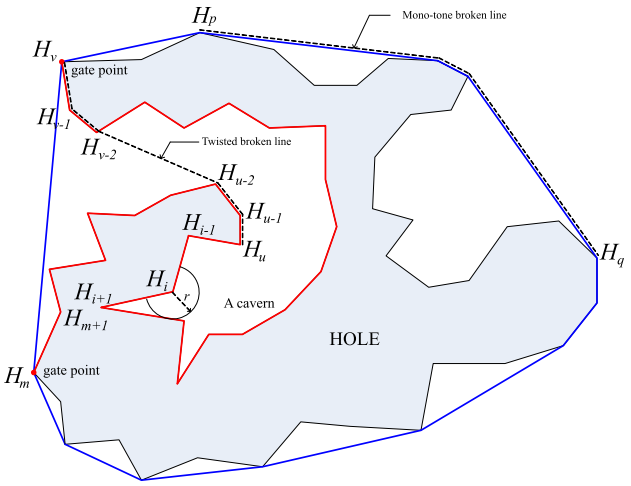


Fig. 4 Illustration of definitions

So, for two Euclidean parallel lines with distance 1 apart, the two actual routes (that can be created aligning with each Euclidean line by using only sensor nodes nearest to it) can be almost perfectly separate, i.e. using almost non node in common.[1]

3.2 Definitions and notations

In this section, we provide the definitions and notations used throughout this article.

The illustration of the definitions is presented in Fig. 4.

Definition 1 (*Routing hole and the boundary*) A *routing hole* is a non-self-intersecting with its vertices as sensor nodes $S_1, \ldots, S_n$ and satisfies the following conditions:

1. Its interior does not contain any sensor nodes.
2. The Euclidean distance between S_i and S_{i+1} is within the transmission range ($\forall i = \overline{1,n}; S_{n+1} \equiv S_1$).
3. S_i and S_{i+j} are out of the transmission range ($\forall i = \overline{1,n}; \forall j \geq 2; S_{n+k} \equiv S_k, \forall k \in \mathbb{N}$).

The sensor nodes at the boundary of a hole are called the *boundary nodes*.

Throughout this article, the boundary nodes, $H_1, H_2, \ldots, H_n$, of a hole $\mathbb{H}$ are sorted in an order such that the hole's interior stays on the right side of the boundary when going from H_1 to H_n.

The Euclidean length between two points A and B is denoted by $|AB|$, $|l|$ is the Euclidean length of the line l. $\widehat{H_i}$ denotes the interior angle of $\mathbb{H}$ at H_i, i.e., $\widehat{H_i} = \angle H_{i-1}H_iH_{i+1}$ ($\forall i = 1, 2, \ldots, n$, $H_0 \equiv H_n, H_{n+1} \equiv H_1$). $\widehat{H_i}$ is called a convex angle if $\widehat{H_i} < 180°$, otherwise a concave angle.

Definition 2 (*Convex boundary vertex and convex hull*) H_i is a convex boundary vertex if $\widehat{H_i}$ is a convex angle, and a concave boundary vertex, otherwise. The *convex hull* of $\mathbb{H}$ is defined as the convex polygon that has all vertices as vertices of $\mathcal{H}$. (A convex hull is illustrated as in blue in Fig. 4.)

Definition 3 (*Concave regions and related*) Let H_k, H_l be two consecutive vertices of the convex hull of $\mathbb{H}$ such that $k < l$.

1. Polygon $\mathbb{C} = \{H_k, H_{k+1}, \ldots, H_l\}$ is called a *concave region* – or a *cavern* – of $\mathbb{H}$ with the two nodes H_k, H_l being the *gate-points* of the cavern $\mathbb{C}$ (in red as illustrated in Fig. 4).
2. The depth of $\mathbb{C}$, denoted as $d_{\mathbb{C}}$, is defined as the maximum distance from its vertices to its gate, i.e., $d_{\mathbb{C}} = \max_{i=\overline{k+1,l-1}} |H_i, H_kH_l|$.
3. For a convex vertex H_i inside a cavern, let $\mathcal{A}(H_i, r)$ denote the circular arc centered at H_i with radius r sweeping from $\overrightarrow{H_ix}$ to $\overrightarrow{H_iy}$ where ray $\overrightarrow{H_ix} \perp \overrightarrow{H_iH_{i-1}}$ and ray $\overrightarrow{H_iy} \perp \overrightarrow{H_iH_{i+1}}$ (see Fig. 4 for illustration). Then, the accessibility level – or just *level*, in brief – of node H_i is defined as the maximum value of r such that $\mathcal{A}(H_i, r)$ does not intersect the hole's interior.

Definition 4 (*Positive/negative segments, mono-tone broken lines and related*) Assume that the hole boundary nodes are indexed in the direction that the hole's interior stays on the right side of the boundary when one travels from H_1 to H_n.

1. Let H_i, H_j be two hole boundary vertices, then H_iH_j is called a positive segment if $H_{i+1} \ldots H_{j-1}$ stay on the same side concerning $\overrightarrow{H_iH_j}$, otherwise, H_iH_j is called a negative segment.
2. A broken line $\mathcal{H} = H_{k_1}H_{k_2}\ldots H_{k_m}$ is called a *mono-tone broken line* if $H_{k_j}H_{k_{j+1}}$ is a positive segment for $\forall i = \overline{1,m}$, otherwise, *a twisted broken line*. A mono-tone broken line $\mathcal{H}$ is said *right mono-tone* if the hole's interior stays on the right side of $\mathcal{H}$ when one travels in the direction of increasing node indexes, otherwise, *left mono-tone*. (As illustrated in Fig. 4, H_PH_Q is a right monotone broken-line while H_iH_l is a twisted broken-line.)
3. Suppose that $\mathcal{H} = H_{k_1}H_{k_2}\ldots H_{k_m}$ is a mono-tone broken line, the *angle* of $\mathcal{H}$, denoted as $\varphi(\mathcal{H})$, is defined by the

[1] The probability of these two actual routes having a node in common can be as small as we want by increasing the sensor density. In our theoretical analysis we assume this probability negligible. In reality, it can still be non-negligible but would have a very little impact as our numerical results by simulations strongly reflect our theoretical expectation.

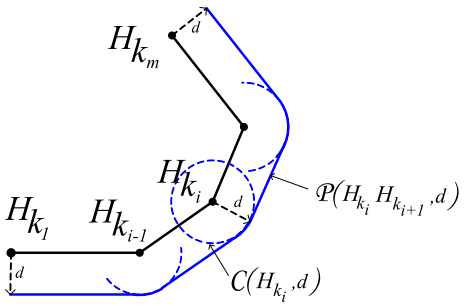


Fig. 5 Illustration of shifting transformation

total of the angles at every vertices of $\mathcal{H}$, i.e., $\varphi(\mathcal{H}) = \sum_{i=2}^{m-1} \angle H_{k_i}$.

Note that if $\mathcal{H} = H_{k_1} H_{k_2} ... H_{k_m}$ is a mono-tone broken line, then the hole's interior always stays on the same side (i.e., left side or right side) of $\mathcal{H}$. On the contrary, if $\mathcal{H}$ is a twisted broken line and $H_{k_j} H_{k_{j+1}}$ is a negative segment of $\mathcal{H}$, then the hole's interior stays on both sides of $H_{k_j} H_{k_{j+1}}$.

Definition 5 (*Shifting transformation*) For each convex boundary hole vertex H_i, let $\mathcal{C}(H_i, d)$ denote the circle with a radius of d and centered at H_i. For the positive segment $H_u H_v$, let $\mathcal{P}(H_u H_v, d)$ denote the parallel line with a distance of d to $H_u H_v$ which is opposite the hole's interior across $H_u H_v$. Suppose that $\mathcal{H} = H_{k_1} H_{k_2} ... H_{k_m}$ is a mono-tone broken line, let $H^1_{k_i}$ and $H^2_{k_i}$ be the tangent points of $\mathcal{C}(H_{k_i}, d)$, $\mathcal{P}(H_{k_{i-1}} H_{k_i}, d)$ and $\mathcal{P}(H_{k_i} H_{k_{i+1}}, d)$ respectively, where $H^1_{k_1} \equiv H^2_{k_m} \equiv null$. Then the shifting transformation with the shifting factor of d of $\mathcal{H}$, denoted as $\mathcal{F}(\mathcal{H}, d)$, produces a line which is obtained by concatenating all circular arcs $\overset{\frown}{H^1_{k_i} H^2_{k_i}}$ and line segments $H^2_{k_i} H^1_{k_{i+1}}$ $(i = \overline{1, m})$—see Fig. 5 for illustration.

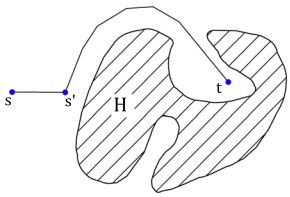
4 Proposed scheme

Our aim is to design a routing scheme that achieves a *short path with a given width* $k > 1$ between a given source s and destination t. In normal situations, short paths are aligned with the shortest route (as close to the shortest route as possible). Since the actual routes based on *these parallel lanes* have almost the similar length, the route stretch is $1 + \epsilon$ with $\epsilon > 0$ as small as possible. Besides, the width of the lane large enough is selected to ensure that these actual routes are almost separate from each other. Thus, each packet from s to t can be *randomly sent over a route amongst the set of routes based on the parallel lanes*. This random selection alleviates the hotspots phenomenon.

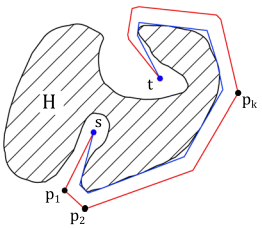
However, it is not simple to construct *these parallel lanes* in a situation with the G-shaped hole (see Sect. 2.2). In such a case, contour lines straightforwardly are to be used, which could still produce convergence points at a zig-zag turn (Fig. 6).

We propose a mechanism for constructing parallel lanes with three major points as follows:

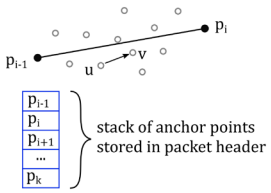
1. We use the shortest $s - t$ route as the basis to align the parallel lanes.



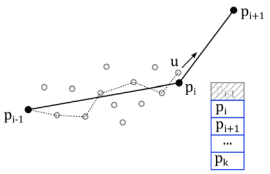
(1) If the source s is not aware of (far from) the hole, greedy forwarding from s towards t until reaching a sub-source node s' who is aware of hole H.



(2) At s' or s (if already within the hole proximity), determine shortest path (blue) then the Euclidean path (red) using k-MLP algorithm. Insert (the coordinates of) these vertices as anchor points $p_1, p_2, ..., p_k$ to packet header.



(3) *Loop*
For each forwarding step current node u performs: (i) find current anchor point p_i then determine next node v toward p_i, aligning uv with $p_{i-1}p_i$; (ii) update the stack of anchor points (as in next screen).



Update the stack of anchor points: Upon reaching node u closest to anchor p_i then remove p_{i-1} from the packet header (current anchor becomes p_{i+1}).

Fig. 6 Activity diagram of the routing mechanism: The three main steps of a given $s - t$ routing task

2. We introduce the concept of *accessibility level* of a boundary, a convex vertex node that helps to compute the maximum number of establishable lanes.
3. *We number lanes to deal with zig-zag turns*. If the hole is on the left of the lanes, the lanes are numbered from 1 to k. Otherwise, they are numbered from k to 1.

The purpose of the second point is to find the proper path width k for each $s-t$ pair and ensures that within a cavern, even the outermost lane does not cut into the other side of the cavern. The third point solves the mentioned convergence issue when packets travel from one side to the opposite of the cavern. If lane α inside lane β at one side of a cavern, α is outside of β and they *are not crossing each other at a zig-zag turn* (see an illustration in Fig. 7).

Our proposed routing scheme consists of an initialization phase and a working (routing) phase. During the first phase, sensor nodes on the hole's boundary cooperate to identify the hole boundary. Then, the information of the hole boundary is disseminated to all sensor nodes around the hole by a simple mini-broadcast. Such hole-awareness involves some communication overhead but greatly aid in making intelligent and energy-saving routing decisions. Moreover, to reduce the setup overhead, we do not broadcast the hole information all over the network but limit to only the nodes that are close enough to the hole.

In the routing phase, once the routed packet reaches the hole-aware region, the current node first determines k^{max}-MLP path to the destination whose width $k^{max}$is computed based on its route stretch upper-bounded by a given $1+\epsilon$. Then, the current node determines the projected path to the destination by randomly selecting a Euclidean routing path

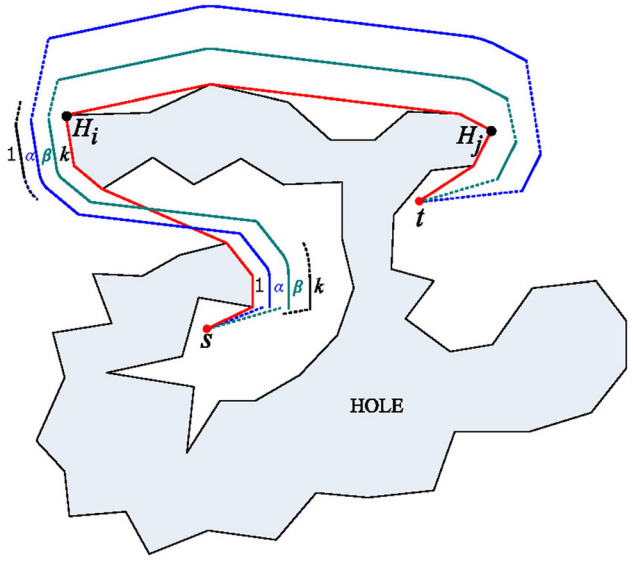


Fig. 7 Paths in the network: The red path is the shortest path, the green path and the blue path represent two Euclidean routing paths of two packets from s to t

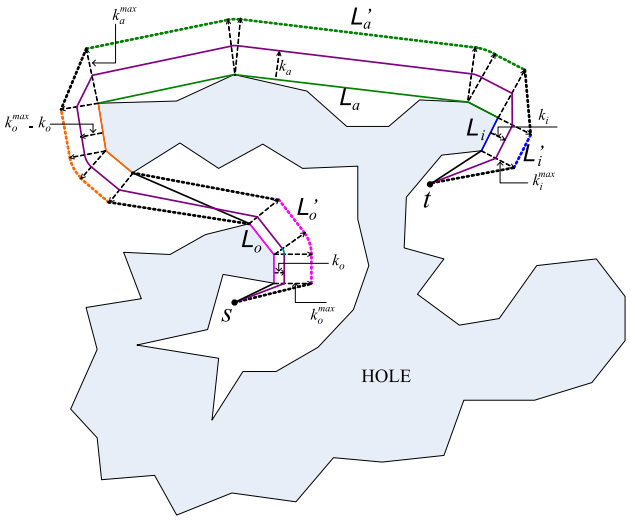


Fig. 8 Illustration of constructions of a k-MLP, the $s-t$ multi-lane path, and an Euclidean routing path

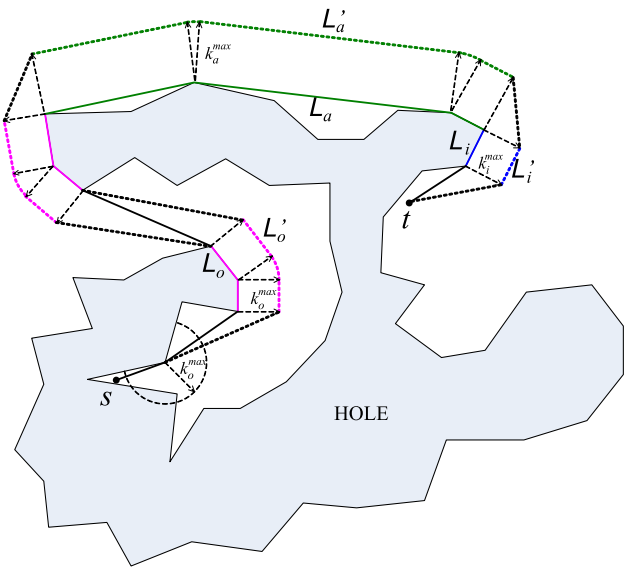


Fig. 9 An example of the multi-lane path when the source node stay in a deep cavern

from the set of *almost* parallel lanes of the k^{max}-MLP path. This selected Euclidean routing path is created by shift-transforming the shortest hole-bypassing path whose shifting factor d is randomly selected from $\{1..k^{max}\}$. Finally, the packet is greedily forwarded along the projected Euclidean routing path using the sensor nodes that are the nearest to it. Figures 8 and 9 illustrates our proposed scheme.

4.1 Initialization phase

In the hole boundary determination step, sensor nodes collaborate to formulate the hole's boundary polygon using

the technique introduced in GAR [38]. If a node is on the hole boundary, it initiates a hole-collecting packet that is repeatedly forwarded to the adjacent node on the hole boundary until the packet returns to the sending node. As the position information of nodes is appended to the info packet, the hole's boundary polygon is collected when the information packet returns to the sending node.

In the next step, the node broadcasts the information about the boundary polygon across the area within distance δ of the hole boundary. The value of δ will be discussed in Sect. 5. Each boundary node inside a cavern can compute the convex hull of the hole or the shape of the cavern based on the boundary polygon. Inside the cavern, each convex boundary node determines its accessibility level, then the proper number of lanes (k).

4.2 Routing phase

Note that nodes far from the hole forward packets with the greedy forwarding scheme. When a packet arrives at a hole-aware node (we call this node as a sub-source), the sub-source node then performs the following process of three stages:

- determining the shortest path to the destination,
- creating the projected path, i.e. the Euclidean routing path,
- forwarding packet along the Euclidean routing path (used as the projected path).

Table 2 Summary of notations

Notation	Description
$\mathbb{H} = H_1H_2...H_n$	Hole with boundary polygon vertices
$\|AB\|$	Length of segment AB
$\widehat{H_i} = \angle H_{i-1}H_iH_{i+1}$	Angle at vertex H_i
$\mathbb{C} = \{H_k, H_{k+1}, ..., H_l\}$	Concave region (cavern) with gate port H_k and H_l
$\mathcal{A}(H_i, r)$	Circular arc centered at H_i with radius r
$\mathcal{P}(H_uH_v, d)$	A line parallel to positive segment H_uH_v and has distance d to it
$\mathcal{C}(H_i, d)$	A circle centered at H_i with a radius of d
$\mathcal{L}_o, \mathcal{L}_a, \mathcal{L}_i$	The out-of-cavern section, hole-around section, into-cavern section, respectively
$\mathcal{F}(\mathcal{H}, d)$	The shifting transformation with the shifting factor of d of a mono-tone broken line $\mathcal{H}$
$\varphi(\mathcal{H}) = \sum_{i=2}^{m-1} \angle H_{k_i}$	Angle of a mono-tone broken line $\mathcal{H}$
k_o^{max}, k_a^{max}, k_i^{max}	The maximum k values (lane indexes) that still validates the route stretch upper bound $1+\epsilon$ per $\mathcal{L}_o, \mathcal{L}_a, \mathcal{L}_i$, respectively

Let s and t be the source and destination, respectively. Source node s is far from the hole, and x is the sub-source node. In what follows, we show how s (or x) can utilize the hole information to make a routing decision. Figure 6 summarizes all these main steps (Table 2).

4.2.1 Determining the shortest path

Generally, the shortest path is composed of three sections: an outward section from a cavern if the source node s is inside a cavern, a section around the hole, and an inward section into a cavern if the destination t is inside a cavern. Figure 7 illustrates an example of the shortest path between s and t. Nodes H_i and H_j are the two gate points of the caverns, respectively. The segment from s to H_i is *the out-of-cavern section*, the segment from H_i to H_j *the hole-around section*, and the segment from H_j to t is *the into-cavern path*, respectively.

The algorithm [42] is used to build a visibility graph whose vertices are boundary nodes inside the caverns. Then, either Dijkstra or A* algorithm can be applied to determine the shortest out-of-cavern section and the into-cavern section. The total computational complexity is $O(m^2 \log m)$, where m is the number of boundary nodes in the caverns. Let O and I be the other endpoints of the shortest out-of-cavern segment and the into-cavern section, (apart from s and t), respectively. Then the hole-around path is the shortest path from O to I that goes through the vertices of the convex hull of the hole.

4.2.2 Determining the Euclidean routing path

Let $\mathcal{L}_o, \mathcal{L}_a$ and $\mathcal{L}_i$ denote the out-of-cavern section, hole-around section, and the into-cavern section, respectively.

On the one hand, a k-MLP (k-Multi-lane) path based on monotone broken line $\mathcal{L}$ is a set of k (almost) parallel lanes, called the j-lanes for $j = 1..k$, where the j-Lane can be obtained by the shifting transformation of $\mathcal{L}$ (see Definition 5) with an appropriate shifting factor.

On the other hand, each twisted broken line $\mathcal{L}$ can be divided into mono-tone segments, denoted as $\mathcal{L}_1, ..., \mathcal{L}_u$. Thus, the j-lane $\mathcal{L}'$ is the concatenation of $\mathcal{L}'_i$ ($i = 1, ..., u$), which are obtained by the appropriate shifting transformations of the $\mathcal{L}_i$ ($i = 1, ..., u$).

Following the lane-numbering rule (Sect. 4), the shifting transformation $\mathcal{F}(\mathcal{L}_i, j)$ or $\mathcal{F}(\mathcal{L}_i, k-j)$ is performed depending on whether $\mathcal{L}_i$ is left or right monotone, respectively. So, the obtained $\mathcal{L}'_i$ is parallel to $\mathcal{L}_i$ and has a distance of j or $k-j$ to $\mathcal{L}_i$. The lane-numbering rule

ensures that these j-lanes are not crisscrossing but instead, (almost) parallel. Figure 8 illustrates a multi-lane out-of-cavern path from s to H_i, with some mono-tone segments (the starting one in pink and the ending one in orange). This multi-lane path is stuck between 0-lane and k_o^{max}-lane, where k_o^{max} is the maximum value that still validates the route stretch upper bound $1+\epsilon$. For a given shifting factor k_o, $k_o = 1, \ldots, k_o^{max}$, the k_o-lane is depicted as a solid line in purple, which is the concatenation of multiple segments obtained by shifting transformations $\mathcal{F}(H_iH_{i-1}H_{i-2}, k_o)$ and $\mathcal{F}(H_{l-2}H_{l-1}H_l, k_o^{max} - k_o)$.

Now, a Euclidean path with the stretch upper bound $1+\epsilon$, from a source s to a destination t, is constructed as follows. Let $\mathcal{S}(s,t)$ be the shortest hole-bypassing path from s to t, which consists of the out-of-cavern section $\mathcal{L}_o$, the hole-around section $\mathcal{L}_a$, and the into-cavern section $\mathcal{L}_i$.

Source node s carries out separate MLP constructions for each sub-path (performed at the) to determine the k_o^{max}-MLP, the k_a^{max}-MLP and the k_i^{max}-MLP. Intuitively, the selection of these MLP (sub-)paths guarantees that the route stretch of the whole path does not exceed $(1+\epsilon)$ and that the created Euclidean routing path does not intersect the hole's interior. The detailed procedure on determining such k_o^{max}, k_a^{max} and k_i^{max} will be presented in Sect. 5.1.

Then k_o^{max}-MLP, k_a^{max}-MLP, k_i^{max}-MLP along $\mathcal{L}_o$, $\mathcal{L}_a$ and $\mathcal{L}_i$ are established, respectively. The combination of k_o^{max}-MLP is called a *multi-lane* path from s to t and denoted by $\mathbb{P}(s,t)$. Node s can randomly select one from the (almost) parallel lanes within $\mathbb{P}(s,t)$ to forward packets.

Specifically, node s randomly chooses three shifting factors $k_o \in 1..k_o^{max}$, $k_a \in 1..k_a^{max}$, and $k_i \in 1..k_i^{max}$ and then constructs the Euclidean routing path as follows. First, node s divides $\mathcal{L}_o$, $\mathcal{L}_a$ and $\mathcal{L}_i$ into mono-tone broken lines then performs shifting transformation of these mono-tone broken lines using the aforementioned lane-numbering rule. The left mono-tone broken lines of $\mathcal{L}_o$, $\mathcal{L}_a$ and $\mathcal{L}_i$ are shifted horizontally by k_o, k_a and k_i, respectively, and the right mono-tone broken lines of $\mathcal{L}_o$, $\mathcal{L}_a$ and $\mathcal{L}_i$ are shifted horizontally $k_o^{max} - k_o$, $k_a^{max} - k_a$ and $k_i^{max} - k_i$, respectively. The Euclidean routing path is the concatenation of all the shifted lines. To guarantee that all Euclidean routing paths from s to t parallel to each other, k_o, k_a, k_i are chosen such that $\frac{k_o}{k_o^{max}} = \frac{k_a}{k_a^{max}} = \frac{k_i}{k_i^{max}}$. In Fig. 8, the dotted lines represent the margin of the multi-lane path, and the violet line depicts a Euclidean routing path.

If either source node s or destination node t stay inside a deep caverns (Fig. 9), the k_o^{max}-MLP and k_i^{max}-MLP of $\mathcal{L}_o$ and $\mathcal{L}_i$ may intersect the hole's interior. A procedure to cope with such a situation as follows. Source node s determines points s' and t' that are on the shortest path $\mathcal{S}(s,t)$ and the nearest to s and t. The levels of s' and t' are greater than k_o^{max} and k_i^{max}. Then, node s creates the multi-lane path $\mathbb{P}(s',t')$ corresponding to the shortest path from s' to t'. The Euclidean routing path for every packet from s to t is determined as the concatenation of $\mathcal{S}(s,s')$, a shifted image of $\mathcal{S}(s',t')$ which stays inside $\mathbb{P}(s',t')$, and $\mathcal{S}(t',t)$. Note that $\mathcal{S}(s,s')$ and $\mathcal{S}(t',t)$ are identical for all the packets from s to t, while the shifted image of $\mathcal{S}(s',t')$ is varied for every packet.

Algorithm 1 Determine routing path

Input: Source s, destination t, route stretch ϵ
Output: Routing path $\mathcal{L}'$

1: $(s\mathcal{L}_o, \mathcal{L}_a, \mathcal{L}_i t) = \mathcal{S}(s,t)$
2: Compute k_o^{max}, k_a^{max}, k_i^{max}
3: $r = random(0,1)$
4: $\mathcal{L}'_o = \text{EUCLIDEANPATH}(\mathcal{L}_o, rk_o^{max}, k_o^{max})$
5: $\mathcal{L}'_a = \text{EUCLIDEANPATH}(\mathcal{L}_a, rk_a^{max}, k_a^{max})$
6: $\mathcal{L}'_i = \text{EUCLIDEANPATH}(\mathcal{L}_i, rk_i^{max}, k_i^{max})$
7: $\mathcal{L}' = s\mathcal{L}'_o \cup \mathcal{L}'_a \cup \mathcal{L}'_i t$
8: **return** $\mathcal{L}'$

9: **procedure** EUCLIDEANPATH($\mathcal{L}_i, k, k_{max}$)
10: $\mathcal{L}'_i = \{\}$
11: $monoSeg = \{\}$
12: $monoSeg.append(\mathcal{L}_i[0])$
13: **for** $j = 1$ to $\mathcal{L}_i.size() - 1$ **do**
14: **if** $\mathcal{L}_i[j-1]\mathcal{L}_i[j]$ is negative segment **then**
15: **if** $monoSeg$ is left mono-tone **then**
16: $shiftedSeg = \mathcal{F}(monoSeg, k)$
17: **else**
18: $shiftedSeg = \mathcal{F}(monoSeg, k_{max} - k)$
19: **end if**
20: $\mathcal{L}'_i.append(shiftedSeg)$
21: $monoSeg = \{\}$
22: **end if**
23: $monoSeg.append(\mathcal{L}_i[j])$
24: **end for**
25: **return** $\mathcal{L}'_i$
26: **end procedure**

4.2.3 Forwarding along the Euclidean routing path

In the final stage, a procedure to forward packets along the Euclidean routing path $\mathcal{L}'(s,t)$ is presented. Suppose $\mathcal{L}'(s,t) = sH'_{i_1}H'_{i_2}...H'_{i_m}t$ $(1 \le i_1, i_2, ..., i_m \le n)$. In the beginning, node s inserts $\mathcal{L}'(s,t)$ into the packet header as a list of anchor points. Then, the packet is routed towards $H'_{i_1}, H'_{i_2}, ..., H'_{i_m}, t$ sequentially using a forwarding algorithm given in Algorithm 2. Specifically, after receiving the packet, a current node u extracts the anchor list LA and determines the previous anchor a_p and the current anchor a_c as the first and second elements in LA. Node u then finds the next node v amongst u's neighbors. Next node v must satisfy that v is the closest node to the current anchor a_c while it does not deviate too far from the Euclidean routing path. Namely, in lines 3-6, only the neighbour w whose distance to the determined routing path, $d(w, a_pa_c)$, is less than value δ, is considered and the nearest node to the anchor a_c is selected as the next

Algorithm 2 Forward packet

Input: Packet pkt, current node u
1: Extract list of anchors LA from pkt
2: $a_p = LA[0], a_c = LA[1]$ /* the previous and current anchor points
3: $\mathcal{C} = \{w \in u.neighbor \mid d(w, A) < d(u, A)\}$
4: $\delta = \max(1/\rho, \min_{w\in\mathcal{C}} d(w, a_p a_c))$ /* Suppose network density is ρ nodes per m^2
5: $\mathcal{C}' = \{w \in \mathcal{C} \mid d(w, a_p a_c) \le \delta\}$
6: Find v in $\mathcal{C}'$ s.t. $d(v, a_c) = \min_{w\in\mathcal{C}'} d(w, a_c)$
7: **If** $d(v, a_c) < R$ **then** remove first element from LA
8: Send pkt to v

node. The value of δ is chosen based on the spatial density of the sensor nodes, wherein it must be large enough to ensure that there is always at least one point within that distance (line 4). Finally, before forwarding the packet to the next node v, node u removes the first element from the list of anchors (i.e. a_p) if v reaches the current anchor a_c, that is $d(v, a_c) < R$.

5 Theoretical analysis

5.1 Shifting factors

Let l_o, l_a and l_i denote the lengths of $s\mathcal{L}_o$, $\mathcal{L}_a$, and $\mathcal{L}_i t$, respectively. And l is the total length of the shortest path from s to t, i.e., $l = l_o + l_a + l_i$. Let $\mathcal{L}'_o$, $\mathcal{L}'_a$, and $\mathcal{L}'_i$ be the images of $\mathcal{L}_o$, $\mathcal{L}_a$, and $\mathcal{L}_i$ through the shifting transformation with the shifting factors of k_o, k_a, and k_i, respectively. Let l'_o, l'_a and l'_i represent the lengths of the images of $s\mathcal{L}'_o$, $\mathcal{L}'_a$, and $\mathcal{L}'_i t$, respectively. As will be proved in Sect. 5, we have

$$l'_x \le l_x + k_x(2\varphi(\mathcal{L}_x) + 2m_x - 1),\ x \in \{o, a, i\}, \tag{1}$$

where the m_x's, $x \in \{o, a, i\}$, are the number of mono-tone lines of the $\mathcal{L}_x$'s, respectively. Therefore, to guarantee the stretch upper bound of $(1 + \epsilon)$, we should ensure that:

$$k_x \le \frac{\epsilon l_x}{2\varphi(\mathcal{L}_x) + 2m_x - 1},\ x \in \{o, a, i\}. \tag{2}$$

Moreover, to ensure that the k_o-MLP of $\mathcal{L}_o$ and the k_i-MLP of $\mathcal{L}_i$ do not intersect the hole's interior, the values of k_o and k_i are limited by $\frac{\Delta(\mathbf{C}_s)}{2}$ and $\frac{\Delta(\mathbf{C}_t)}{2}$, where $\mathbf{C}_s$, $\mathbf{C}_t$ are the caverns containing s and t, and $\Delta(\mathbf{C}_s)$, $\Delta(\mathbf{C}_t)$ are their widths, respectively. Following the analysis described above, the widths of the multi-lane path $\mathbb{P}(s, t)$ are determined as

$$k_o^{max} = \min\left(\frac{\epsilon l_o}{2(\varphi(\mathcal{L}_o) + m_o)}, \frac{\Delta(\mathbf{C}_s)}{2}\right), \tag{3}$$

$$k_i^{max} = \min\left(\frac{\epsilon l_i}{2(\varphi(\mathcal{L}_i) + m_i)}, \frac{\Delta(\mathbf{C}_t)}{2}\right), \tag{4}$$

$$k_a^{max} = \frac{\epsilon l_a}{2(\varphi(\mathcal{L}_a) + m_a)}. \tag{5}$$

Note that for a common hole, the out-of-cavern, around-hole and into-cavern paths are usually mono-tone paths; thus, m_o, m_i and m_a are usually equal to 1. Moreover, it can be proved that $\varphi(\mathcal{L}_o), \varphi(\mathcal{L}_a)$ and $\varphi((\mathcal{L}_i)$ do not exceed 2π. Therefore, k_o^{max}, k_a^{max} and k_i^{max} can be approximated as follows

$$k_o^{max} = \min\left(\frac{\epsilon l_o}{2(2\pi + 1)}, \frac{\Delta(\mathbf{C}_s)}{2}\right), \tag{6}$$

$$k_i^{max} = \min\left(\frac{\epsilon l_i}{2(2\pi + 1)}, \frac{\Delta(\mathbf{C}_t)}{2}\right), \tag{7}$$

$$k_a^{max} = \frac{\epsilon l_a}{2(2\pi + 1)}. \tag{8}$$

5.2 Routing stretch

Let $H_1H_2....H_n$ be all the vertices of the hole boundary that are sorted in an order such that the hole interior stays on the right side of $\overrightarrow{H_iH_{i+1}}$ $(\forall i = 1, ..., n-1)$.

Lemma 1 *Let $\mathcal{L}$ be a mono-tone broken line and d is a positive number. Suppose that $\mathcal{L}'$ is the image of $\mathcal{L}$ through*

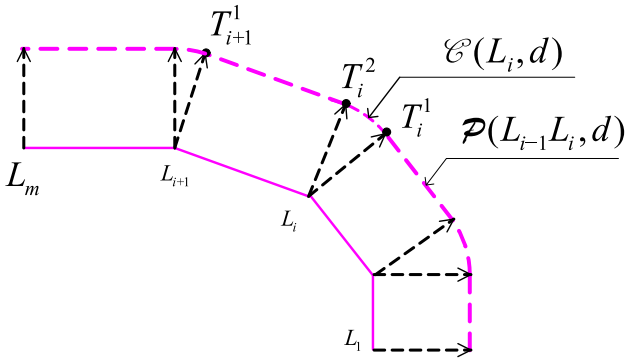


Fig. 10 Illustration of Lemma 1

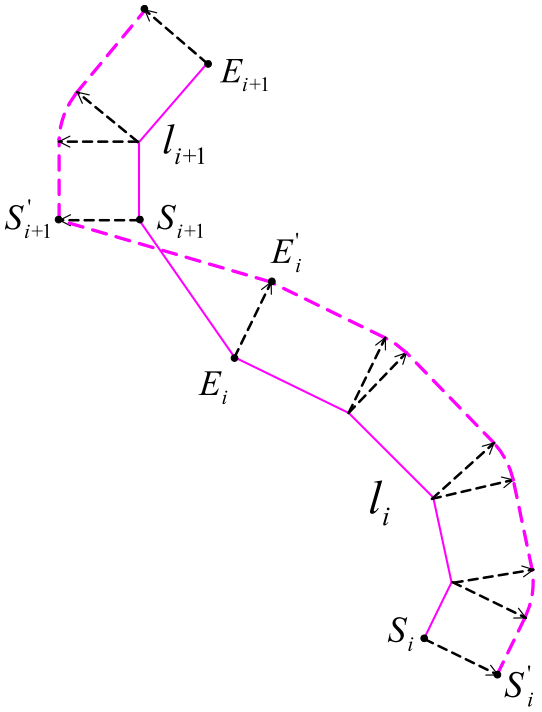


Fig. 11 Illustration of Lemma 2

the shifting transformation with the shifting factor of d, then $|\mathcal{L}'| = |\mathcal{L}| + d\varphi(\mathcal{L})$.

Proof Let us denote $L_1, L_2, ..., L_m$ as the vertices of $\mathcal{L}$ (Fig. 10). According to the definition 5, $\mathcal{L}'$ is the concatenation of $\mathcal{C}(L_i, d)$ and $\mathcal{P}(L_{i-1}L_i, d)$, where $\mathcal{C}(L_i, d)$ is a circle with a radius of d and centered at L_i, and $\mathcal{P}(L_{i-1}L_i, d)$ is the parallel line whose distance to $L_{i-1}L_i$ is d. Suppose that T_i^1 is the tangent point of $\mathcal{C}(L_i, d)$ and $\mathcal{P}(L_{i-1}L_i, d)$, while T_i^2 is the tangent point of $\mathcal{C}(L_i, d)$ and $\mathcal{P}(L_iL_{i+1}, d)$. Then, $\mathcal{L}'$ is the concatenation of all line segments $T_i^2T_{i+1}^1$ $(i = \overline{1, m-1})$, circular arcs $\overset{\frown}{T_j^1T_j^2}$ $(j = \overline{1, m})$ and line segments $T_j^2T_{j+1}^1$ $(j = \overline{2, m-1})$. Clearly, we have

$$|\mathcal{L}'| = \sum_{i=1}^{m-1} |T_i^2T_{i+1}^1| + \sum_{i=2}^{m-1} |\overset{\frown}{T_i^1T_i^2}|$$
$$= \sum_{j=2}^{m-1} |L_jL_{j+1}| + d\varphi(\mathcal{L}) = |\mathcal{L}| + d\varphi(\mathcal{L}).$$

Lemma 2 *Given a broken line* $\mathcal{L}$ *which is comprised of m mono-tone broken lines,* $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_m$. *Let* $\mathcal{L}'_1, \mathcal{L}'_2, ..., \mathcal{L}'_m$ *be the images of* $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_m$ *through the shifting transformations with the shifting factors of* $d_1, d_2, ..., d_m$, *respectively, where* $d_1, d_2, ..., d_m$ *are positive numbers. Let* l' *be the concatenation of* $\mathcal{L}'_1, \mathcal{L}'_2, ..., \mathcal{L}'_m$, *then* $|\mathcal{L}'| \le \mathcal{L} + 2\sum_{i=1}^{m} d_i\varphi(\mathcal{L}_i) + (2m-1)d$, *where* $d = \max_{i=\overline{1,m}} k_i$.

Proof Let us denote S_i and E_i as the two endpoints of l_i; S'_i and E'_i are the two endpoints of l'_i as shown in Fig. 11. We have

$$|\mathcal{L}| = \sum_{i=1}^{m} |\mathcal{L}_i| + \sum_{i=1}^{m-1} |S_iE_i|, \tag{9}$$

$$|\mathcal{L}'| = \sum_{i=1}^{m} |\mathcal{L}'_i| + \sum_{i=1}^{m-1} |S'_iE'_i|. \tag{10}$$

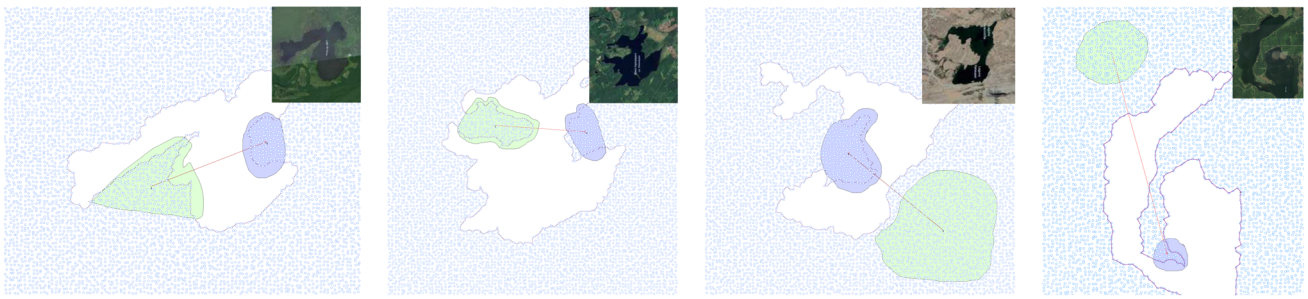
According to Lemma 1 we get

$$|\mathcal{L}'_i| = |\mathcal{L}_i| + 2d_i\varphi(\mathcal{L}_i). \tag{11}$$

Moreover, by using the triangular inequality, we have

$$|S'_iE'_i| \le |S_iE_i| + d_i + d_{i+1}. \tag{12}$$

From (9, 10, 11, 12), it is deduced that



Hole 1 *(Lago do Arroz Lake, Amazon, Brazil)*

Hole 2 *(Ozero Yaymyayarvi Lake, Karelia, Russia)*

Hole 3 *(Laguna Yanauyac Lake, Lima, Peru)*

G-shape Hole *(Boy Lake, Minnesota, USA)*

Fig. 12 Simulation scenarios

$$|\mathcal{L}'| \le \sum_{i=1}^{m}(\mathcal{L}_i + 2d_i\varphi(\mathcal{L}_i)) + \sum_{i=1}^{m-1}(E_iS_{i+1} + d_i + d_{i+1}),$$
$$\Rightarrow |\mathcal{L}'| \le \mathcal{L} + 2\sum_{i=1}^{m} d_i\varphi(\mathcal{L}_i) + (2m-1)d. \tag{13}$$

Theorem 1 *The stretch of all Euclidean routing paths is bounded by* $1+\epsilon$.

Proof Let $\mathcal{S}(s,t)$ and $\mathcal{R}(s,t)$ be the shortest path and routing path from the source node s to the destination node t, respectively. Without loss of generality, we assume that $\mathcal{S}(s,t)$ is comprised of an out-of-cavern path $\mathcal{L}_o$, a hole-around path $\mathcal{L}_a$, and an into-cavern path $\mathcal{L}_i$. Moreover, suppose that $\mathcal{R}(s,t)$ is obtained by shifting $\mathcal{L}_o$, $\mathcal{L}_a$ and $\mathcal{L}_i$ with the shifting factors of k_o, k_a and k_i. Let us denote $\mathcal{L}'_o$, $\mathcal{L}'_a$ and $\mathcal{L}'_i$ as the images of $\mathcal{L}_o$, $\mathcal{L}_a$ and $\mathcal{L}_i$ through the shifting transformation. According to Lemma 2, we have

Table 3 Simulation parameters

Parameter type	Parameter value
Model UDG	(Unit disk graph)
Simulation time	500 s
Transmission range	40 m
Mac protocol	CSMA
Energy capacity	18J
Traffic type	Constant bit rate
Data rate	4 KB/s
Size of data packet	256 byte
Communication pairs	Hole 1–3: 15
	G-shape hole: 10, 15, 20, 25, 30

$$|\mathcal{L}'_x| \le |\mathcal{L}_x| + 2k_o\varphi(\mathcal{L}_x) + (2m_o - 1)k_x \tag{14}$$

for each subscript $x \in \{o, a, i\}$. Moreover, we have

$$|\mathcal{L}'| \le |\mathcal{L}'_o| + |\mathcal{L}'_a| + |\mathcal{L}'_i| + k_o + k_a + k_i. \tag{15}$$

Therefore, we obtain

$$\begin{aligned}|\mathcal{L}'| \le |\mathcal{L}| &+ 2k_o(\varphi(\mathcal{L}_o) + m_o) \\ &+ 2k_a(\varphi(\mathcal{L}_a) + m_a) + 2k_i(\varphi(\mathcal{L}_i) + m_i).\end{aligned} \tag{16}$$

As k_o, k_a and k_i are smaller than k_o^{max}, k_a^{max} and k_i^{max}, respectively, from (6), (7), (8) and (16), we have:

$$|\mathcal{L}'| \le |\mathcal{L}| + \epsilon|\mathcal{L}| = (1+\epsilon)|\mathcal{L}|. \tag{17}$$

6 Numerical results

In our simulation experiments, we assume that sensors are deployed in 20×20 square grids of four network scenarios, as shown in Fig. 12. One sensor is placed at a random position in each square. In the first three scenarios, Fig. 12a–c, holes emulate the shapes of 3 natural lakes. We construct an artificial G-shape hole to test the behaviour of our proposed solution in a scenario (Fig. 12d) with a cavern that has a zig-zag turn at the entrance. Traffic is generated between sources in the green region and destinations in the blue region. We use the Castalia simulator v.3.3 [43] to conduct our experiments with parameters listed in Table 3.

We compare our scheme with 4 existing protocols: GPSR [2], ALBA-R [27], ADAV [9], and STABLE [40]. GPSR is the landmark geographic routing proposal that combines the greedy routing and the perimeter routing to bypass a hole. ALBA-R is focused on load-balancing. ADAV and STABLE are two different approaches that attempt at combining small route stretch with high load-balancing.

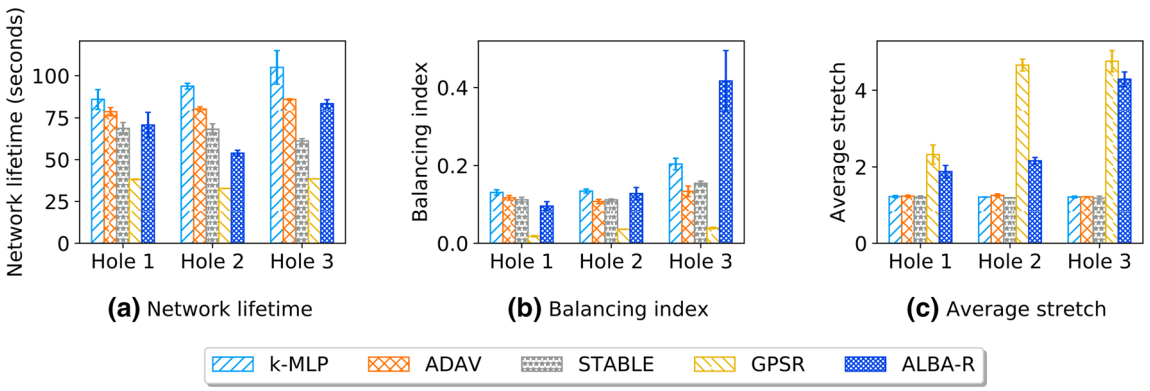


Fig. 13 Simulation results with three holes based on nature lakes

Table 4 Simulation results

	k-MLP	ADAV	STABLE	GPSR	ALBA-R
Hole 1					
Network lifetime	104.98	85.87	61.24	38.54	83.26
Balancing index	0.13	0.12	0.11	0.02	0.10
Route stretch	1.22	1.23	1.21	2.32	1.88
Hole 2					
Network lifetime	93.95	80.28	68.22	32.78	53.71
Balancing index	0.13	0.11	0.11	0.04	0.13
Route stretch	1.21	1.26	1.19	4.66	2.16
Hole 3					
Network lifetime	85.87	78.67	68.72	38.22	70.73
Balancing index	0.20	0.13	0.15	0.04	0.42
Route stretch	1.21	1.22	1.18	4.76	4.29
G-shape hole					
Network lifetime	91.37	52.87	48.02	35.91	48.73
Balancing index	0.16	0.14	0.08	0.06	0.34
Route stretch	1.13	1.14	1.08	4.48	3.07

6.1 Simulations with holes based on natural lakes

Figure 13 presents results on the following performance metrics:

- The average *routing path stretch* is the ratio between the hop count of the routing path using this routing algorithm and the theoretical shortest path.
- *Balance index* (*BI*) [13] is to quantify how the traffic load is balanced among the sensor nodes: $BI = \frac{(\sum_{i=1}^{M} p_i)^2}{M \sum_{i=1}^{M} p_i^2}$, where M is the number of sensor nodes and p_i is the total number of packets sent or forwarded by the ith node. It is easy to see that $\frac{1}{M} \leq BI \leq 1$ and a larger BI indicates the better balanced realisation of traffic.
- *Network lifetime* is defined as the length of time that the network functions as a whole, which ends when the first sensor node runs out of energy. Note that this choice corresponds to the most popular approach regarding the definition of the network lifetime [1].

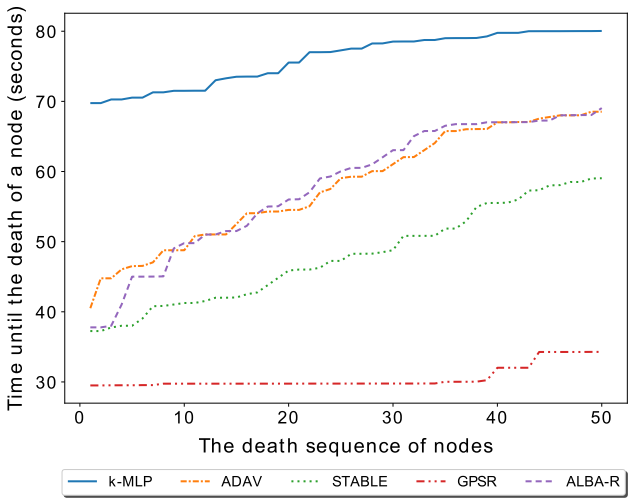


Fig. 15 Time to the death of a node

Simulation results are obtained with the 95%-confidence interval. For the *k*-MLP algorithm and the ADAV algorithm, we set the upper-bound stretch $\epsilon = .3$, so the average routing stretch is always less than 1.3. As shown in Fig. 13c, the average stretch of *k*-MLP, ADAV and STABLE is quite similar, ranging from 1.20 to 1.26. Meanwhile, the average stretch of GPSR and ALBA-R schemes is significantly larger, from 1.5 to 4 times larger than *k*-MLP, ADAV and STABLE to long detour paths around the hole.

From Fig. 13b), the *k*-MLP has the largest balancing index *BI*, which shows that both ADAV and STABLE algorithms cannot solve the problem of load-balancing

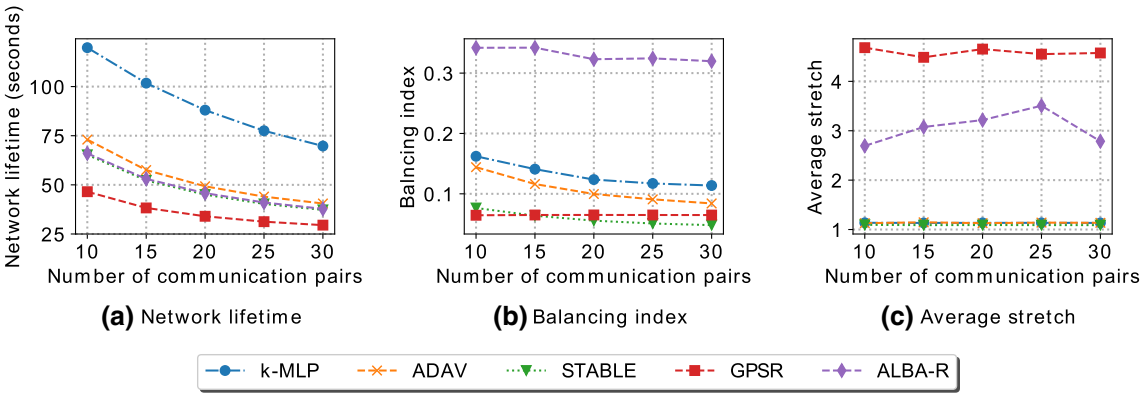


Fig. 14 Simulation results with Boy Lake

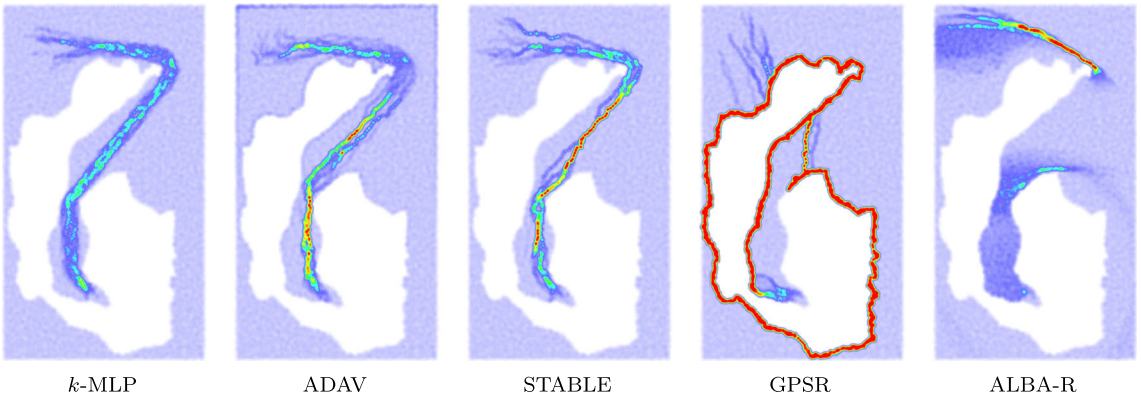


Fig. 16 Heatmaps in the scenario with the G-shape hole

completely as *k*-MLP. In the ADAV protocol, the routing paths created by using homothetic transformation can intersect each other, resulting in traffic build-up at these points. In STABLE, the routing paths are parallel in the case of simple holes (without a zig-zag turn), but the number of parallel paths depends on the shape and size of the hole, thus, reducing the load balancing capabilities. It can be observed that GPSR is the worst protocol in terms of load-balancing as it often creates routes along the hole's perimeter. In contrast, the ALBA-R achieves a competitively high *BI* as it mainly focuses on supporting load balancing when making routing decisions.

Among routing schemes, our proposal *k*-MLP always achieves the highest network lifetime as shown in Fig. 13a, namely higher than ADAV, STABLE, GPSR and ALBA-R respectively 16%, 43%, 160% and 37% in average. The three *k*-MLP, ADAV and STABLE are efficient in terms of network lifetime thanks to the implementation of adaptive and small stretch routing paths. The ALBA-R protocol, though produces a high *BI*, has an inferior network lifetime because of large routing stretch and high energy consumption for nodes located at the boundary between two different coloured regions. Table 4 sums up the numerical results so far.

With the assumption, the network continuously operate with the respective routing protocol after the death of each node, we plot the time to the death instance of the first 50 nodes in Fig. 15 (axis x represents the death sequence of nodes in the network). The results strongly confirm the superiority of our proposal *k*-MLP.

6.2 Scenario with the G-shape hole

In a particular scenario with an artificial G-shape hole (a zig-zag turn in the cavern entrance), the performance gap is high between our *k*-MLP scheme and the others. As shown in Fig. 14, the *k*-MLP scheme always achieves the most extended network lifetime while it still keeps a small average route stretch (less than 1.3). In particular, the average network lifetime in *k*-MLP is 1.7, 1.9, 2.5, or 1.8 times greater than in ADAV, STABLE, GPSR or ALBA-R, respectively. Regarding the balancing index, our *k*-MLP has *BI* 1.2, 2.2 or 2.0 times larger than ADAV's, STABLE's, GPSR's, and less than ALBA-R's - the topmost. This performance gap increases because the existing proposals ADAV and STABLE cannot deal efficiently with complex holes. It can be observed in the heat-maps (Fig. 16) that the routing paths created in STABLE can converge at a specific point in the cavern, inflicting the hotspot problem. Meanwhile, in the ADAV scheme, the routing paths usually intersect the hole boundary; thus, individual packets can be either dropped or routed along the perimeter of the hole, creating hotspots on the hole boundary.

7 Conclusion and future works

We have addressed a challenging problem that caused by a large hole of complicated shape in wireless sensor networks. We have proposed a load-balancing, $(1+\epsilon)$-stretch routing scheme *k-MLP routing* that achieves k almost-parallel routes with stretch at most $1+\epsilon$ from the shortest routes for any constant $\epsilon > 0$ and a given network-hole scenario, The main idea is to construct k almost parallel lanes between source s and a destination d such that this k-width strip always aligns itself with the shortest path between s and t. Thus, routing scheme *k-MLP routing* can achieve a good trade-off between shortening routes and load balancing (by choosing a random "lane" for each packet to be sent through). The derivation of the algorithm requires the combination of two supporting techniques. One is on the accessibility level of each convex vertex inside a cavern, and the other is on the lane-numbering rule. While the latter is for avoiding lane crisscrossing at a zig-zag turn, the former is for computing the maximum

width (in the number of parallel lanes, i.e. k in k-MLP) of the path segment that is possible to have outside any given convex vertex.

We have provided a comparison between our *k-MLP routing* algorithm and four other existing routing schemes that represent different approaches and strategies in dealing with holes. Our simulation results show that our *k-MLP routing* scheme outperforms the existing algorithm. In regular hole-scenarios, our network lifetime is about 17% longer than the second-best candidate's and nearly three times greater than the traditional GPSR's. In the challenging scenario of a cavern that has a zig-zag turn at the entrance, our algorithm performs exceptionally well with a network lifetime several times longer than all the other candidates (twice better than the second-top performer).

7.1 Possible applications and future works

Our *k-MLP routing* algorithm is most suited to WSN scenarios, that need intensive communication between locations surrounded by a massive obstacle of a complicated shape (such as a large building). For example, the organizer of a significant street event would want to deploy several reading devices to collect information at essential locations and stream data sets to different places. In another possible application, a WSN can be deployed in a building's carpark to provide short, intensive communication sessions between any corners and the park gate.

For future work, we aim to explore if we can further develop our mechanism of constructing a path with a given width (i.e. in k lanes) and other relevant techniques to help in solving the problem of planning short paths with clearance through crowded environment.[2] Our mechanism for planning paths using WSNs can be a potential approach to solve problems related to moving objects in physical environments, such as guiding autonomous vehicles in parking, where obstacles could be both static and dynamic.

Funding Open access funding provided by Budapest University of Technology and Economics. This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant Number 102.02-2017.316.

[2] As already mentioned, this is a common and significant problem in several fields such as in robotics, geographic information systems (GIS), VLSI design, computer graphics, and games.

References

1. Yetgin, H., Cheung, K. T. K., El-Hajjar, M., & Hanzo, L. H. (2017). A survey of network lifetime maximization techniques in wireless sensor networks. *IEEE Communications Surveys Tutorials, 19*(2), 828–854.
2. Karp, B., & Kung, H. T. (2000). GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of MOBICOM'00* (pp. 243–254).
3. Kuhn, F., Wattenhofer, R., & Zollinger, A. (2002). Asymptotically optimal geometric mobile ad-hoc routing. In *DIALM-POMC '02 proceedings of the 2002 joint workshop on foundations of mobile computing*.
4. Kuhn, F., Wattenhofer, R., & Zollinger, A. (2003). Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of ACM MobiHoc*.
5. Tan, G., Bertier, M., & Kermarrec, A.-M. (2009). Visibility-graph-based shortest-path geographic routing in sensor networks. *Proceedings of IEEE INFOCOM* (pp. 1719–1727).
6. Won, M., & Stoleru, R. (2014). A low-stretch-guaranteed and lightweight geographic routing protocol for large-scale wireless sensor networks. *ACM Transactions on Sensor Networks, 18,* 1–22.
7. Nguyen, L., Phi, J., Yusheng, L., Zhi, H., Vu, H., & Nguyen, K. V. (2017). Distributed hole-bypassing protocol in WSNS with constant stretch and load balancing. *Computer Networks, 129,* 232–250.
8. Vu, H., Nguyen, T. D., Nguyen, C. Q., & Nguyen, V. K. (2016). An efficient geographic algorithm for routing in the proximity of a large hole in wireless sensor networks. In *Proceedings of the seventh symposium on information and communication technology* (SoICT '16, pp. 286–293). ACM.
9. Le Nguyen, P., Ji, Y., Le, K., & Nguyen, T. H. (2018). Load balanced and constant stretch routing in the vicinity of holes in WSNS. In *2018 15th IEEE annual consumer communications networking conference (CCNC)* (pp. 1–6).
10. Boy lake-cass county lake located near longville mn=https://www.longville.com/lake-maps/boy-lake/
11. Bhattacharya, P., & Gavrilova, M. L. (2008). Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. *IEEE Robotics Automation Magazine, 15*(2), 58–66.
12. Anish, P., Shalini, E. P., & Parhi, D. R. (2017). Mobile robot navigation and obstacle avoidance techniques: A review. In *ICRA 2017* .
13. Hsiao, P.-H., Hwang, A., Kung, H. T., & Vlah, D. (2001). Load-balancing routing for wireless access networks. In *Proceedings IEEE INFOCOM 2001. Conference on computer communications. Twentieth annual joint conference of the IEEE computer and communications society (Cat. No.01CH37213)* (Vol. 2, pp. 986–995).
14. Subramatian, S., Shakkottai, S., & Gupta, P. (2007). On optimal geographical routing in wireless networks with holes and non-uniform traffic. In *Proceedings of IEEE INFOCOM*.
15. Gupta, N. K., Shankar Yadav, R., & Nagaria, R. K. (2018). Void handling in 3d wireless sensor networks. In *2018 15th IEEE India Council international conference (INDICON)* (pp. 1–6).

16. Jayashri, S., & Lakshmi Joshitha, K. (2019). A novel redundant hole identification and healing algorithm for a homogeneous distributed wireless sensor network. *Wireless Personal Communications, 104,* 1261–1282.
17. Li, W., & Yuwei, W. (2016). Tree-based coverage hole detection and healing method in wireless sensor networks. *Computer Networks, 103,* 33–43.
18. Koriem, S. M., & Bayoumi, M. A. (2018). Detecting and measuring holes in wireless sensor network. *Journal of King Saud University-Computer and Information Sciences*.
19. Yu, F., et al. (2008). Efficient hole detour scheme for geographic routing in wireless sensor networks. In *Proceedings of the 67th IEEE vehicular technology conference, VTC'08* (pp. 153–157).
20. Choi, M., & Choo, H. (2011). Bypassing hole scheme using observer packets for geographic routing in WSNs. In *Proceedings of international conference on information networking, ICOIN'11* (pp. 435–440).
21. Tian, Y., et al. (2008). Energy-efficient data dissemination protocol for detouring routing holes in wireless sensor networks. In *Proceedings of IEEE international conference on communications, ICC'08* (pp. 2322–2326).
22. Trajcevski, G., Zhou, F., Tamassia, R., Avci, B., Scheuermann, P., & Khokhar, A. A. (2011). Bypassing holes in sensor networks: Load-balance versus latency. In *GLOBECOM* (pp. 1–5).
23. Huang, H., Yin, H., Min, G., Zhang, X., Zhu, W., & Wu, Y. (2017). Coordinate-assisted routing approach to bypass routing holes in wireless sensor networks. *IEEE Communications Magazine, 55*(7), 180–185.
24. Nguyen, K.-V., Nguyen, L., Phi, Q. H., & Vu, Q. H., & Van Do, T. (2017). An energy efficient and load balanced distributed routing scheme for wireless sensor networks with holes. *Journal of Systems and Software, 123,* 92–105.
25. Zhou, F., Trajcevski, G., Tamassia, R., Avci, B., Khokhar, A., & Scheuermann, P. (2017). Bypassing holes in sensor networks: Load-balance versus latency. *Ad Hoc Networks, 61,* 16–32.
26. Yu, Y., Govindan, R., & Estrin, D. (2001) Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical report.
27. Petrioli, C., Nati, M., Casari, P., Zorzi, M., & Basagni, S. (2014). Alba-r: Load-balancing geographic routing around connectivity holes in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems, 25*(3), 529–539.
28. Le, K., Nguyen, T. H., Nguyen, K., & Nguyen, P. L. (2019). Exploiting q-learning in extending the network lifetime of wireless sensor networks with holes. In *2019 IEEE 25th international conference on parallel and distributed systems (ICPADS)* (pp. 602–609).
29. Lipare, A., Edla, D. R., & Kuppili, V. (2019). Energy efficient load balancing approach for avoiding energy hole problem in WSN using grey wolf optimizer with novel fitness function. *Applied Soft Computing, 84,* 105706.
30. Huang, H., Yin, H., Min, G., Zhang, J., Wu, Y., & Zhang, X. (2018). Energy-aware dual-path geographic routing to bypass routing holes in wireless sensor networks. *IEEE Transactions on Mobile Computing, 17*(6), 1339–1352.
31. Khan, Z. A., Awais, M., Alghamdi, T. A., Khalid, A., Fatima, A., Akbar, M., & Javaid, N. (2019). Region aware proactive routing approaches exploiting energy efficient paths for void hole avoidance in underwater WSNS. *IEEE Access, 7,* 140703–140722.
32. Awais, M., Ali, I., Alghamdi, T. A., Ramzan, M., Tahir, M., Akbar, M., & Javaid, N. (2020). Towards void hole alleviation: Enhanced geographic and opportunistic routing protocols in harsh underwater WSNS. *IEEE Access, 8,* 96592–96605.
33. Javaid, N. (2019). NADEEM: Neighbor node approaching distinct energy-efficient mates for reliable data delivery in underwater WSNS. *Transactions on Emerging Telecommunications Technologies*, e3805.
34. Khan, Z. A., Latif, G., Sher, A., Usman, I., Ashraf, M., Ilahi, M., & Javaid, N. (2019). Efficient routing for corona based underwater wireless sensor networks. *Computing, 101*(7), 831–856.
35. Sha, C., Ren, C., Malekian, R., Wu, M., Huang, H., & Ye, N. (2020). A type of virtual force-based energy-hole mitigation strategy for sensor networks. *IEEE Sensors Journal, 20*(2), 1105–1119.
36. Hadikhani, P., Eslaminejad, M., Yari, M., & Mahani, E. A. (2020). An energy-aware and load balanced distributed geographic routing algorithm for wireless sensor networks with dynamic hole. *Wireless Networks, 26*(1), 507–519.
37. Fang, Q., Gao, J., & Guibas, L. J. (2004). Locating and bypassing routing holes in sensor networks. In *IEEE INFOCOM 2004* (Vol. 4, pp. 2458–2468).
38. Liu, W. J., & Feng, K. T. (2009). Greedy routing with anti-void traversal for wireless sensor networks. *IEEE Transactions on Mobile Computing, 8*(7), 910–922.
39. Gupta, R. K., Kumar, N. N., & Yadav, R. S. (2020). 3D geographical routing protocols in wireless ad hoc and sensor networks: An overview. *Wireless Networks, 26,* 2549–2566.
40. Thai, N. Q. D., & Nguyen, K. (2018). Stable low-stretch routing scheme for wireless sensor networks with a large hole of complicated shape. In *2018 5th NAFOSTED conference on information and computer science (NICS)* (pp. 17–23).
41. Bulusu, N., Heidemann, J., & Estrin, D. (2000). Gps-less low-cost outdoor localization for very small devices. *IEEE Personal Communications, 7*(5), 28–34.
42. Lee, D. T. (1983). Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing, 22*(2), 207–221.
43. Castalia, simulator for wireless sensor networks, body area networks and generally networks of low-power embedded devices. https://github.com/boulis/Castalia/

Khanh-Van Nguyen received the BE degree from Hanoi University of Science and Technology (HUST), Vietnam in 1992, and received the MS degree from University of Wollongong, Australia in 1995, and received the PhD degree from the University of California - Davis in 2006. He is currently an Associate Professor in the School of Information and Communication Technologies, HUST. His main research domain is algorithms and theoretical models for computer networks and distributed computing.

Chi-Hieu Nguyen received his B.E. degree in information system from the School of Information and Communication Technology at the Hanoi University of Science and Technology. He is currently an M.S. student at the School of Information and Communication Technology at the Hanoi University of Science and Technology. His research interests include optimization algorithms, wireless sensor networks and unmanned aerial vehicle (UAV) communications and networking.

Phi Le Nguyen received her B.E. and M.S. degrees from the University of Tokyo in 2007 and 2010, respectively. She received her Ph.D. in Informatics from The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan in 2019. Currently, she is an assistant professor at the School of Information and Communication, Hanoi University of Science and Technology, Vietnam. Her research interests include network architecture, optimization, and artificial intelligence enabled networking.

Tien Van Do received the M.Sc. and Ph.D. degrees in telecommunications engineering from the Technical University of Budapest, Hungary, in 1991 and 1996, respectively. He is a professor in the department of Networked Systems and Services, the Budapest University of Technology and Economics. He habilitated at BME, and received the DSc from the Hungarian Academy of Sciences in 2011. He led various projects on network planning and software implementations that results are directly used for industry such ATM & IP network planning software for Hungarian Telekom, GGSN tester for Nokia, performance testing program for the performance testing of the NOKIA's IMS product, automatic software testing framework for Nokia Siemens Networks. His research interests are queuing theory, telecommunication networks, cloud computing, performance evaluation and planning of ICT Systems and machine learning.

Imrich Chlamtac (M'86–SM'86–F'93) is the President of CREATE-NET, the Bruno Kessler Professor with the University of Trento, Italy, and the President of the European Alliance for Innovation. He has held various honorary and chaired professorships in the United States and Europe, including the Distinguished Chair in Telecommunications Professorship at the University of Texas at Dallas, Sackler Professorship at Tel Aviv University, and University Professorship at the Technical University of Budapest. He was with Technion and UMass, Amherst, DEC Research. He has made significant contribution to various networking technologies as a Scientist, an Educator, and an Entrepreneur. He published around 400 refereed journal, book, and conference articles and is listed among ISI's Highly Cited Researchers in Computer Science. He has co-author four books, including the first book on Local Area Networks in 1980 and the Amazon.com best seller and IEEE Editor's Choice entitled Wireless and Mobile Network Architectures (Wiley, 2000). He was a recipient of multiple awards and recognitions including a fellow of the ACM, Fulbright Scholar, the ACM Award for Outstanding Contributions to Research on Mobility, and the IEEE Award for Outstanding Technical Contributions to Wireless Personal Communications. He has widely contributed to the scientific community as a Founder and the Chair of ACM Sigmobile, a Founder and the Steering Committee Chair of some of the leading conferences in networking, including ACM Mobicom, IEEE/SPIE/ACM OptiComm, CreateNet Mobiquitous, CreateNet WiOpt, IEEE/CreateNet Broadnet, IEEE/CreateNet Tridentcom, and IEEE/CreateNet Securecomm conferences. He also serves as the Founding Editor-in-Chief of Wireless Networks (Springer) and the Journal on Special Topics in Mobile Networks and Applications (Springer).