Copyright ⓒ2016 KSII

R3: A Lightweight Reactive Ring based Routing Protocol for Wireless Sensor Networks with Mobile Sinks

Sheng Yu [1], Baoxian Zhang [1,2], Zheng Yao [1], and Cheng Li [3]
[1] Research Center of Ubiquitous Sensor Networks, University of Chinese Academy of Sciences
Beijing 100049, China
[e-mail: {bxzhang, yaozheng}@ucas.ac.cn]
[2] Jiangsu Internet-of-Things Research and Development Center
Wuxi, Jiangsu 214135, China
[3] Memorial University of Newfoundland
St. John's, NL A1B 3X5, Canada
[e-mail: licheng@mun.ca]
*Corresponding author: Zheng Yao

Received April 1, 2016; revised August 14, 2016; revised September 10, 2016; accepted October 10, 2016; published December 31, 2016

Abstract

Designing efficient routing protocols for a wireless sensor network with mobile sinks (mWSN) is a challenging task since the network topology and data paths change frequently as sink nodes move. In this paper, we design a novel lightweight reactive ring based routing protocol called R3, which removes the need of proactively maintaining data paths to mobile sinks as they move in the network. To achieve high packet delivery ratio and low transmission cost, R3 combines ring based forwarding and trail based forwarding together. To support efficient ring based forwarding, we build a ring based structure for a network in a way such that each node in the network can easily obtain its ring ID and virtual angle information. For this purpose, we artificially create a virtual hole in the central area of the network and accordingly find a shortest cycled path enclosing the hole, which serves as base ring and is used for generating the remaining ring based structure. We accordingly present the detailed design description for R3, which only requires each node to keep very limited routing information. We derive the communication overhead by ring based forwarding. Extensive simulation results show that R3 can achieve high routing performance as compared with existing work.

Keywords: Wireless sensor network, mobile sinks, and reactive routing

This work was supported by National Natural Science Foundation of China under Grant Nos. 61531006, 61173158, 61471339, the National Science & Technology R&D Program of China under Grant No. 2014BAK06B03, and the Natural Sciences and Engineering Research Council (NSERC) of Canada (Discovery Grant 293264-12 and Strategic Project Grant STPGP 397491-10).

http://dx.doi.org/10.3837/tiis.2016.12.017 **ISSN : 1976-7277**

1. Introduction

Wireless sensor networks with mobile sinks (mWSN) have attracted a lot of attention in recent years [1-4]. Such networks typically consist of many static sensor nodes for sensing and one or more mobile sink nodes (MSs) for data collection. Sensor nodes are low-cost and low-power devices which can measure surrounding physical phenomena, process the measurements, and transfer the data to other nodes through wireless communications. The mobile sink nodes roam over the sensing field and collect the sensing data from sensor nodes. Mobile sinks can be mounted on various moving objects like vehicles, robots, or people with different mobility patterns [5, 6].

Exploiting sink mobility has been widely regarded as an efficient way to alleviate the hot spot issue in WSNs so as to improve the network performance [8-21]. The hot spot issue is commonly known as the sensor nodes near static sinks consume energy much faster than sensor nodes in other regions of the network [7]. When the nodes near sinks run out of their power, the resting nodes are partitioned away from the sinks. In the context of mWSNs, however, sink mobility is often unpredictable and can cause unexpected changes in network topology and data routing paths. It may require extra signaling procedures to update the data paths to reach mobile sinks as they move. Therefore, it is highly desirable to design efficient and scalable routing protocols for mWSNs to achieve high routing performance.

In this paper, we propose a lightweight Reactive Ring based Routing protocol (referred to as R3) for mWSNs with sporadic traffic. An mWSN with sporadic traffic usually works in event driven mode and has low traffic load and the sensing data are generated occasionally at low speed from some sensor nodes detecting certain events in the network. For such a network, one key issue for designing efficient routing protocol is to maximally suppress the protocol overhead for route discovery. R3 is purely a distributed routing protocol in which each network node only needs to keep limited routing information. R3 integrates ring based forwarding and trail based forwarding. Specifically, each data packet is forwarded using ring based forwarding until it reaches a mobile sink or can be forwarded along a fresh trail to reaching a mobile sink. To support efficient ring based forwarding, we build a ring based infrastructure for a multihop wireless sensor network when it is initially deployed. For creating such a ring based structure, R3 first builds a shortest cycled path (measured in hop count), which tightly embraces a virtual topological hole that we artificially create in the central area of the network, as the *base ring*. Each remaining (outer) ring is formed by those nodes having the same hop distance to the base ring. When performing ring based forwarding, data packets are forwarded along nodes on a selected ring in a pre-determined direction (either clockwise or anticlockwise). To enable the packet forwarding to keep moving in the same direction on a particular ring and also minimize the path distance, virtual angle of each node in the ring structure is computed in the following way: Nodes on the base ring are first assigned with virtual angles based on their positions (more exactly, hop distances to a preselected reference node on the cycle, in the same direction) on the cycled path and non-base-ring nodes' virtual angles are iteratively computed based on the virtual angles of their father nodes on their shortest paths to the base ring. In this sense, we say a quasi-polar coordinate system is built on top of the multi-ring based structure such that each node on the ring based structure is assigned with a ring ID (representing also the hop distance away from the base ring) and a virtual angle, which can ease the ring based packet forwarding. We present the design details of the ring based forwarding and also how it is integrated with trail based forwarding. We also present

design details regarding how we handle the ring broken point bypassing issue. The performance of R3 is evaluated through extensive simulations and the results show that R3 can save average per-packet transmission cost by up to 70% while significantly improving the packet delivery ratio performance compared to the TRAIL protocol [16]. The major contributions in this paper are summarized as follows.

1) We design a lightweight reactive ring based routing protocol named R3 for mWSNs, which is purely distributed and only needs each network node to keep very limited routing information. R3 integrates ring based forwarding and trail based forwarding together to achieve high routing performance while keeping the protocol overhead to the minimum.
2) We propose a novel idea for building multi-ring based structure for a network such that each node can easily obtain its ring ID and virtual angle in the network for supporting efficient ring based packet forwarding. To the best of our knowledge, this is the first work that builds such multi-ring based structure for an mWSN without the assistance of location information.
3) We derive the communication overhead by R3 to be $O(|V|)$, where $|V|$ represents the number of nodes in the network.

The rest of this paper is organized as follows. Section 2 presents related work in the field of routing protocols for mWSNs. Section 3 presents the implementation details of the R3 protocol and discusses how it works. Section 4 presents the simulation results for performance evaluation. Section 5 concludes this paper.

2. Related Work

Many routing protocols for mWSNs have been proposed in recent years. Based on the mobility pattern of MSs, existing protocols in this area can be divided into two major categories, one is for MSs with controllable mobility [22-25], and the other is for MSs with random mobility [8-21]. The protocols in the first category usually need to jointly solve the routing problem and the movement planning problem. We are interested in scenarios where MSs independently and randomly move in the sensing field so our protocol in this paper falls into the second category.

Routing protocols for MSs with random mobility can be further divided into location based protocols and topology based protocols based on whether location information is available or not. Location based protocols include LBDD [8], ALURP [9], ER [10], ILSR [11], MPRP [12] and HLS [13][14]. Topology based protocols include AVRP [16], MDRP [17], λ-flooding [18], TRAIL [16], WARP [19], and DDRP [20,21]. Next, we shall briefly introduce how typical protocols in either type and also discuss their properties.

2.1 Location based protocols

Location based routing protocols usually adopt greedy geographical forwarding, according to which each packet holder makes a local decision on which is the best neighbor as the next hop to reach an MS by using the location information of the MS, its neighbors', and of itself. Design of a location based routing protocol in general includes two aspects: design of greedy geographical forwarding discipline based on certain optimization criteria, and use of a location service for providing the up-to-date location of a target MS. Next, we shall respectively introduce typical work in these two aspects.

In the Line Based Data Dissemination (LBDD) protocol [8], data packets are forwarded to

and then stored at nodes in a line area, and sink queries which include sink locations are also sent to the line area and search for interesting data stored there. The Adaptive Location Update based Routing Protocol (ALURP) [9] restricts the scope of the frequent location updates caused by sink mobility to a local area (called destination area) to reduce communication overhead. When an MS moves inside its destination area, it only broadcasts its location to nodes inside the destination area (usually with high frequency). Data packets are first forwarded to target MS via greedy geographical forwarding. Inside the destination area, topology based forwarding is used. The Elastic Routing (ER) protocol [10] enables a source sensor node to keep obtaining the up-to-date location information of a mobile sink during its continuous data reporting to the sink. When a sink node moves, its new location information is propagated backwards along the data path to the source sensor node via piggybacking the freshest destination location information in each data packet and promiscuous channel listening.

The Integrated Location Service and Routing (ILSR) protocol [11] integrates flexible location service and routing scheme to offer guaranteed packet delivery to an MS. It has two versions for predictable mobility and unpredictable mobility, respectively. In the Milestone based Predictive Routing Protocol (MPRP) [12], a sequence of milestone nodes, which are located in the vicinity of the trail of an MS, are selected to estimate the current location of the MS. Data packets are forwarded to the estimated location and, if the MS is not at the estimated location, packets are then forwarded to the most recent milestone node.

Hierarchical Location Service (HLS) [13-14] was designed to provide efficient location service for supporting greedy geographical forwarding in large scale sensor networks with multiple MSs. HLS first divides the network field into a non-overlapping grid structure and finds location servers in the grid structure in a hierarchical way to store the locations of their closest MS(s).

The main limitation of location based protocols is the assumption that every node needs to know its location information (and, in many cases, that of its neighbors as well), which may be impractical in many real applications due to the cost issue and environment restrictions.

2.2 Topology based protocols

Topology based routing protocols work in an active way to build a routing structure for the whole mWSN, in order to keep the route from each sensor to a nearby mobile sink. One big concern in designing efficient routing protocols in this aspect is how to greatly control the overhead for route discovery and maintenance. Next, we shall introduce how existing protocols in this aspect address this issue.

In the Anchor based Voronoi Routing Protocol (AVRP) [16], each sink selects a neighbor node with the best link quality as its anchor node, and then each anchor node broadcasts an interest message towards the network to build the Voronoi scope for the sink to which it is associated. Every sensor node chooses the closest anchor node as its target sink for data reporting. The Multistage Data Routing Protocol MDRP [17] enhances AVRP by dividing the Voronoi scope associated with each mobile sink into multiple layers based on the gradient information of sensor nodes, in order to further constrain the scope of local broadcasts. The λ-flooding protocol [18] aims to offer a bound on the worst-case path stretch ratio of end-to-end (E2E) packet delivery by partially updating a pre-built shortest path spanning tree (as necessary) each time the MS changes its anchor node.

A key issue in topology-based protocols is they need to work proactively to find a short path from every sensor node to a nearby mobile sink as mobile sink(s) move in the network no matter whether the discovered routes are actually used for data delivery or not. Such proactive routing

behavior can cause excessive overhead and is not suitable for mWSNs with sporadic traffic.

2.3 Reactive routing protocols

Reactive routing protocols work in a reactive/passive way for route learning and updating. It in general has low protocol overhead but has long initial route acquisition latency. Next, we shall introduce typical protocols belonging to this category.

The DDRP protocol [20,21] uses overheard data transmissions for route learning and it requires each data packet to carry an option recording its distance to target sink node from the current packet sender, which can assist the route learning. The route learning/updating capability of DDRP is largely affected by the amount of data traffic in the network and their distribution. In general, the more sporadic the data traffic is, the lower the route learning/updating capability will be. Thus, DDRP is not suitable for mWSNs with sporadic data traffic as we study in this paper.

In the TRAIL protocol [16], each mobile sink leaves a trail when it moves in the network by periodically broadcasting beacon messages to its one-hop neighbor sensor nodes. When a sensor node has data packets to report, it will adopt a forwarding strategy that combines random walk and trail based forwarding. Specifically, when no trail of any sink is known, random walk routing is used for packet forwarding; once the data packets reach a sensor node with fresh sink trail information, they will be forwarded along the trail. To take advantage of fresher route (when possible), new sink trail can intercept old sink trail. In TRAIL, the use of random walk routing, although reluctantly, may result in excessively long data path.

The Whirlpool Routing Protocol (WARP) [19] is a reactive routing protocol which first builds a gradient based routing structure rooted at the MS in the network. If the MS moves to other locations, some nodes lose their connections to the MS. In this case, these nodes will start the speculative routing and randomly forward data packets along a spiral trajectory around the last known position of the MS. The speculative routing terminates when data packets reach a neighbor node of the MS.

Our R3 protocol in this paper is a hybrid protocol and it differs from the above existing protocols in the following ways. It first builds a multi-ring based structure for a WSN without the assistance of location information. Ring ID and virtual angle information are assigned to nodes on the multi-ring based structure, which are then used for supporting ring based forwarding. Unlike existing toplogy based routing protocols, the multi-ring based structure does not need to be updated when the network is operating or can be updated at exetreme low freqency for acheving certain load balancing goal (if needed). The ring based forwarding process works in a reactive way to unicast a data packet to an MS by traveling along a particular ring in a pre-determined direction. Also, to improve the routing performance, R3 integrates ring based forwarding and trail based forwarding to accelerate the routing convergence when possible.

3. Proposed Protocol

In this section, we provide the detailed design description of the R3 protocol. We first give an overview of the network model and some basic concepts, and then elaborate the protocol in three parts: ring building algorithm, ring based forwarding algorithm, and trail based forwarding algorithm.

3.1 Protocol Overview

In this paper, we study a wireless multihop network, which can be modelled by $G(V,E)$. $V(G)$

consists of one or multiple mobile sink nodes and multiple static sensor nodes. $E(G)$ represents the set of links in the network, Sensor nodes and sink nodes have the same communication range. For each pair of nodes $u,v \in V(G)$, we say link $(u,v) \in E(G)$ if $d_{uv} \leq R$; otherwise $(u,v) \notin E(G)$. d_{uv} represents the geometrical distance between node u and node v and R represents the uniform transmission range of all nodes in the network. Each node is assumed to be equipped with a omni-directional antenna. Nodes are deployed in a two-dimensional sensing field. Each mobile sink node moves randomly in the sensing field. No node location information is assumed in this paper.

The R3 protocol proposed in this paper is a lightweight reactive routing protocol and it is purely distributed and only needs each node to keep very limited routing information including its node id, ring id, angle, gradient, and also its one-hop neighbor list (including each neighbor node's id, ring id, and angle). R3 integrates ring based forwarding and trail based forwarding. Conceptually, a ring is a group of nodes, all of which have the same hop distance to a base ring and are expected to form an annulus for forwarding packets. Forwarding data packets along a ring is called ring based forwarding. R3 also makes use of the most recent visit history of MSs on sensor nodes. When data packets arrive at a node with fresh visit history of an MS, they will be forwarded to a neighbor with fresher visit record, and so forth, and we call this forwarding method as trail based forwarding.

More specifically, R3 protocol consists of the following two major parts: One is the construction of a multi-ring based network structure when the network is initially deployed and another is the packet forwarding procedure. Next, we shall respectively introduce how either part works.

For creating a multi-ring based network structure, R3 first works to identify a shortest cycled path, which tightly embraces a virtual topological hole that we artificially create in the central area of the network, as the base ring. Each remaining (outer) ring is formed by those nodes with the same hop distance to the base ring. During the ring based network structure construction process, each node in the multi-ring structure are assigned a virtual angle in the following way: Nodes on the base ring are first assigned with virtual angles based on their positions (more exactly, hop distances to a preselected reference node on the cycle, always in the same direction) on the cycled path and non-base-ring nodes' virtual angles are iteratively computed based on the virtual angles of their father nodes on their shortest paths to the base ring. This multi-ring based structure construction process will be accomplished via three rounds of flooding operations.

For performing packet forwarding, R3 integrates ring based forwarding and trail based forwarding. Specifically, each data packet is forwarded using ring based forwarding until it reaches a mobile sink or can be forwarded along a fresh trail to reaching a mobile sink. When performing data packet forwarding along a target ring, either clockwise or anticlockwise direction is first chosen and then the packet will be forwarded along the selected direction. If searching for an MS on the selected ring fails, a new random ring will be tried until an MS (or an agent node of an MS) is found. We shall present the procedures for target ring selection, how to forward a packet to its target ring, how to forward a packet along a ring, and also how to handle the case when a ring broken point is met.

Next, we shall present the design details regarding R3 works.

3.2 Multi-Ring based Network Structure Building

The purpose of building a multi-ring based structure in R3 is to facilitate the ring based forwarding and also shorten the path length taken by ring based forwarding. Building a multi-ring based structure for a multihop wireless network while meeting the above routing

objective is not trivial when no location information is available. R3 uses three rounds of flooding operations to build a ring based structure in the network. The first round of flooding builds gradient information for nodes in the network. The second round of flooding builds a base ring for constructing other rings. The third round of flooding builds ring based structure for remaining nodes in the network by assigning appropriate ring id and virtual angle for each of them.

1) Gradient Establishment

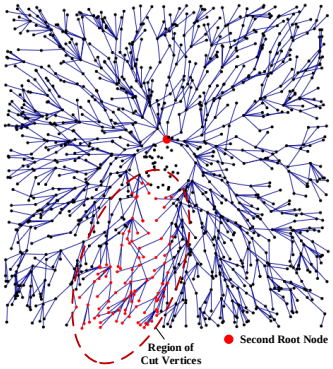
In here, the purpose of gradient establishment is to prepare necessary information for generating a base ring in the network, instead of supporting network-wide packet forwarding as in the famous Directed Diffusion protocol [26] and its variants. For this purpose, a designated root node, which can be a sensor node near the center of the network, is selected to initiate the first round flooding. The selection of such a node can be pre-programmed or artificially determined when the network is initially deployed. During the first round flooding, the root node composes a short signaling message and disseminates it across the whole network[1]. In such a flooding process, to enable nodes to learn their hop distances to the root node, upon receipt of such a message, an intermediate node intentionally defers its re-transmission for a sufficiently long period T plus certain jitter (to avoid transmission collision among neighbor nodes), where T is a network parameter. Each intermediate node only needs to re-transmit the signaling message once. In this way, nodes in the network can learn their hop distances to the designated root node. Those flooding processes used in the rest of this paper will also work similarly for shortest path discovery, unless otherwise stated. Moreover, for clarity, we call this designated root node as the first root node and its gradient is 0. The gradients of other nodes are their hop distances to the first root node.

2) Build the base ring

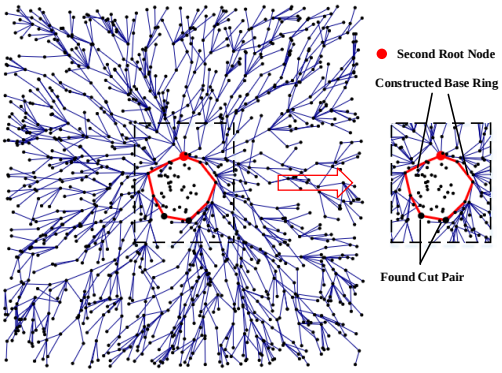
The second round flooding is to build a base ring for the network, which is the basis for generating other rings. To ease the presentation, the algorithm for this purpose is referred to as the base-ring-generating algorithm. Our idea for creating the base ring is motivated by the pioneering work in [27, 28], which, however, is for identifying whether there is natural topological hole(s) in the network and further determine boundary nodes of the holes (if exist). Our idea here is to identify the boundary nodes which can form a shortest cycled path surrounding a *virtual* topological hole that we artificially create in the central area of the network. Specifically, if all the nodes with gradients 1 and 0 were deleted, we say a virtual hole appears in the central area of the network. We hope to find a cycled path enclosing this hole and the cycled path will serve as the base ring. If multiple such cycled paths exist, we choose the shortest one. The base ring will be used for generating other rings and also the angle information of nodes in the network. The shortest property of the base ring can statistically make nodes more uniformly distributed on the cycled path, which helps reduce the expected inaccuracy of virtual angles assigned for nodes on the ring, when no *a priori* knowledge on node distribution is known, and hence reduces the inaccuracy of virtual angles assigned for the remaining nodes in the network. The availability of angle information of nodes is a prerequisite for performing ring based packet forwarding in a *pre-determined* direction (e.g., anticlockwise). Furthermore, the accuracy of such angle information also has big impact on the packet forwarding efficiency (e.g., maximally reducing the hop count distance taken by a packet to travel along a ring).

[1] Another choice is to just flood the signaling message to certain area, wherein a base ring can be surely discovered if this is possible based on certain *a priori* knowledge such as node distribution. For simplicity, hereafter, we will not discuss this choice any further.

To build the base ring, a node with gradient 2 is randomly selected as the initiator of the second round flooding and this node is accordingly called the second root node. All those nodes with gradients 1 or 0 will not participate in this round flooding such that a virtual hole naturally appears. The flooding of a signaling message from the second root node will generate a shortest path tree (denoted by T_2) rooted at the second root node and cover nodes with gradient 2 or above. Our approach of identifying the shortest cycled path surrounding the virtual hole is to inspect those *cut pair*s on T_2 in order to generate cycled paths. The term *cut pair* is borrowed from Ref. [27]. Informally, the "flow" of a shortest path tree is forked near a topological hole, continues along opposite sides of the hole, and then meets again past the hole (See **Figs. 1 and 2** for illustration). A cut pair is a pair of neighboring nodes, called *cut vertices* (e.g., the small red nodes in **Fig. 1(a)**), on two branches of the tree, which forks on one side of the hole and meets again on the other side. **Fig. 1(a)** shows the region where such cut vertices are located. As long as we can detect such cut pairs caused by the virtual hole, cycled paths surrounding the virtual hole can be easily formed. Each such cycled path is created by concatenating the link connecting a pair of cut vertices, and the paths from the two cut vertices to their lowest common ancestor (LCA) on T_2. When multiple such cycled paths exist, we choose the shortest one.



(a)



(b)

Fig. 1. (a) Illustration of cut vertices which may form cycled paths enclosing the virtual hole in the central area of the network. Note that those isolated black nodes in the central area are those nodes with gradients $\in\{0,1\}$. (b) Illustration of the base ring structure built for the network shown in subfigure (a).

In this paper, we modify the algorithm in Ref. [27] to detect cut pairs, and demonstrate that our modified algorithm can effectively return a de]sired cut pair with limited communication overhead. Intuitively, two nodes in a cut pair should be "far away" from their LCA and the

on-tree paths from the two nodes to their LCA should be "well separated" (See **Fig. 2**). The degree at which the two nodes in a cut pair are far away from their LCA can be easily checked by the distances that their respectively received signaling packets (for constructing T_2) have left their LCA. However, whether the two paths are well separated is not so easy to measure. The approach used in [27] is to check whether the maximum hop distance between each pair of nodes on the two paths exceeds a given threshold, which can cause a lot of communication overhead because flooding operation by each node is required for such purpose. In particular, there may exist many such cut pairs (especially, also including those candidates which only meet Condition 1 while violating Condition 2) to check in the network. To suppress the overhead for checking how well two paths are separated, we choose to check whether a pair of representative nodes selected from the two paths are properly separated. In this paper, for a neighboring node pair (p_i, q_i), let LCA(p_i, q_i) represent their LCA on T_2, their representative nodes are, respectively, chosen to be the third node on the path on T_2 from LCA(p_i, q_i) to p_i, denoted by $r_1(p_i, q_i)$, and the third node on the path on T_2 from LCA(p_i, q_i) to q_i, denoted by $r_2(p_i, q_i)$. The reason for making such a choice on representative node selection is because we found, via extensive simulations, that the hop lengths of base rings are mostly distributed between 10 and 14 for large random networks with average degree ≥ 8.

Based on the above discussions, in R3, for a neighbor node pair (p_i, q_i) in the network to be a cut pair qualified for the base ring generation, it must meet the following two conditions.

Condition 1: The hop distance from p_i or q_i to $LCA(p_i, q_i)$ on T_2 is no less than 3 hops.

Condition 2: The hop distance between two representative nodes $r_1(p_i, q_i)$ and $r_2(p_i, q_i)$, denoted by δ, is no less than 3 hops away on the graph excluding all those nodes in the virtual hole.

Obviously, the satisfaction of Condition 1 can easily be checked locally. Only those neighbor pairs satisfying Condition 1 will proceed to check whether they also meet Condition 2 or not. For this purpose, we let nodes with gradients ≥ 2 to learn their respective one- and two-hop neighbor list, without using any node in the virtual hole. If $r_1(p_i, q_i)$ sees that $r_2(p_i, q_i)$ is not in its two-hop neighborhood, a success notification (for meeting Condition 2) can be issued; or otherwise a failure notification should be issued. If multiple cut pairs satisfying Conditions 1 & 2 are found, the second root node will select the one leading to the shortest cycled path. **Fig. 2** gives an example for qualified cut pair selection. In **Fig. 2**, link (m, n) is a qualified cut pair of T_2 because it meets both Conditions 1 and 2. In contrast, links (x, m) and (n, y) do not meet either Condition 1 or Condition 2, so neither of they are cut pairs of T_2.

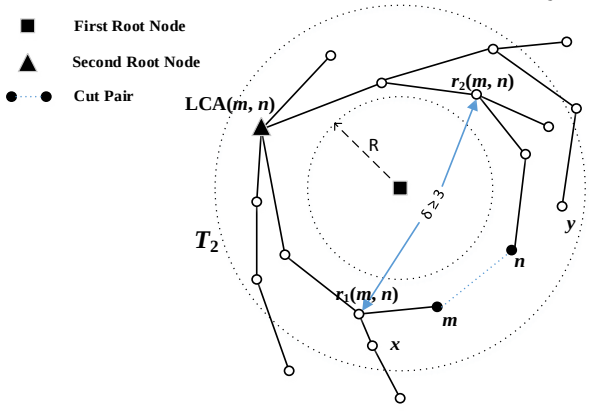


Fig. 2. Example for cut pair qualification examination. In this example, (m, n) is a qualified cut pair, which meets both Condition 1 and Condition 2, link (m, x) is not a qualified cut pair because it violates both Conditions 1 and 2, and link (n, y) is not a qualified cut pair either because it violates Condition 2.

In the above process, one concern is that there may exist natural topological hole(s) in other regions of the network, which may lead to some other cut pairs satisfying Conditions 1 & 2. Using a cycled path surrounding such a hole as the base ring may degrade the quality of the later generated ring based structure and affect the routing performance. Another concern is that if many cut pair candidates satisfying Condition 1 simultaneously run the process to check their satisfaction of Condition 2, many transmission collisions and a lot of communication overhead can be resulted. To address the above concerns, we have the following observation: The cut pair, which can form a shortest cycled path enclosing the artificial hole, usually includes two nodes with low gradients since the nodes inside the hole have the lowest gradients in the network. Based on this simple observation, we prefer to let cut pair candidates constituent of nodes with lower gradients to check the satisfaction of the two conditions (in particular, Condition 2) first. Accordingly, multiple rounds are used. In round 1, those cut pair candidates with lowest gradient sum, i.e., 2+2=4, will check their satisfaction of Condition 2, followed by those cut pair candidates with second lowest gradient sum, i.e., 5, and so on. The gradient sum associated with a cut pair candidate is the sum of the gradients of the two end point nodes of the cut pair candidate. This process continues until a qualified cut pair is found or no cut pair meeting both conditions can be found when all cut pair candidates have been exhausted. To avoid contention between neighboring rounds, one round should last enough long time. Once a qualified cut pair is found, a notification is sent to the second root node, which will then broadcast a short message to notify the success of finding a base ring to all nodes in the network.

Simulation results show that our base-ring-generating algorithm can find a cut pair using nodes with low gradients and form a cycled path enclosing the virtual hole is with very high probability. Specifically, our simulations (2000 trials) on randomly generated networks with average degree ranging from 10 to 16 in a disk area show that the possibility that the algorithm succeeds within the first round (for cut pair qualification examination) is greater than 0.90 and within the third round is greater than 0.99. That is, we say that nodes with gradients $\in\{2, 3\}$ are sufficient to return a qualified cut pair for most networks. This can also largely suppress the amount of nodes needed to involving in two-hop neighborhood knowledge disseminating and gathering as required for checking the satisfaction of Condition 2.

Let (n, m) denote the cut pair returned by our algorithm, the base ring is then formed by connecting the path from LCA(n, m) to n on T_2, the path from LCA(n, m) to n on T_2, and the edge connecting n and m. **Fig. 1(b)** shows the resulting base ring by our algorithm for the example in **Fig. 1(a)**.

If no qualified cut pair is found, the old second root node will randomly select another node with gradient 2 as the new second root node, which will re-start the second round flooding over again. This process continues until a qualified cycle is found or all nodes with gradient 2 (or a pre-determined number of such nodes) have been tried but failed to return a qualified cycle. Our simulation results (2000 trials) show that the average number of flooding operations required for finding a base ring is 1.67, which is quite small. In rare cases (actually not happened in our simulations), if all the searching processes from nodes with gradient 2 fail, nodes with gradient 3 can be selected as the second rood node and they repeat the above process for searching a qualified cut pair and consequently a cycled path as the base ring.

Once the base ring is formed, each node on the ring is assigned with a virtual angle value which represents its relative position on the ring. All virtual angles are within the interval $[0, 2\pi]$. The LCA node is numbered as the first node on the base ring and its virtual angle is set to 0. We let len_1 denote the length of the base ring (i.e., number of nodes on the ring), and $n_{1,i}$ denote the i^{th} node on the base ring (suppose nodes on a ring are numbered in the anticlockwise

direction from the LCA node, and $va(n_{1,i})$ denote the virtual angle of node $n_{1,i}$. We have

$$va(n_{1,i}) = \frac{i-1}{len_1} \times 2\pi. \quad (1)$$

Moreover, all the nodes on the base ring have a ring id of 1, and nodes with gradient 1 set their ring id to 0. The ring id of the first root node is set to -1.

3) Build ring information

The third round flooding is to determine the ring ids and virtual angles for all the nodes outside the base ring. After nodes on the base ring get their ring ids and virtual angles, each of them broadcasts a short signal message containing its own node id, ring id, and virtual angle to its neighbors at a predefined time plus a small jitter. The third round flooding is similar to the first two rounds because it is like adding a virtual root node which is the father node of all the nodes on the base ring as the flooding initiator. The nodes with gradient 1 or 0 do not participate in the third round flooding. Each network node outside the base ring treats its hop distance to the base ring plus one as its own ring id. Moreover, it calculates its virtual angle according to the following rule. Supposing a node is on the k^{th} (k>1) ring, i.e., with ring id k, it calculates its virtual angle by averaging the virtual angles of all its neighbor nodes on the $(k\text{-}1)^{th}$ ring. As long as the network is connected, the virtual angle of every node outside the base ring can be iteratively determined.

In this sense, we say a quasi-polar coordinate system is built on top of the multi-ring based structure. In this system, each node on the ring based structure is assigned with a ring ID,which is also its hop distance away from the base ring and also a virtual angle. This system can assist ring based packet forwarding. The reason we use the term “quasi-polar coordinate system” is due to the following two reasons: 1) we treat all the nodes on the base ring as the pole of the system, and 2) the virtual angle assigned to each node in this paper is obtained via iterative calculations instead of its absoluate angle (which is irrelevant to those of its farther nodes) in a regular polor coordinate system.

However, some outer rings near the field boundary may be partitioned with certain probability and might be unsuitable for ring based forwarding, we use a parameter M to control the maximum number of rings which may involve in the ring based forwarding. The value of M is chosen in a way such that the percentage of nodes belonging to the 1^{st}–M^{th} rings exceeds a threshold. Those nodes outside the M^{th} ring are expected to be opportunistically covered by using trail based forwarding. To obtain the value of M, each node needs to send a notification message containing its ring id back to the second root node (certain en-route aggregation can be performed to reduce the amount of such messages). Once the value of M is determined, another round of flooding is used to notify nodes in the network this value.

It is worth noting that a node on the k^{th} ring (k>1) may have neighbor nodes on the $(k\text{-}1)^{th}$ ring with virtual angles falling at low end and high end of the interval $[0,2\pi]$ at the same time. Simply averaging such neighbor nodes’ virtual angles will lead to wrong results. A simple way to solve this problem is to add 2π to each of those angle values at low end of the interval, then calculate the average value. If the result is larger than 2π, then minus the result by 2π. Also, although each of the three rounds of flooding requires network-wide flooding, they are only performed during the network initialization phase, so their cost can be amortized over the entire lifetime of the network.

3.3 Ring based Forwarding

R3 combines ring based forwarding and trail based forwarding for packet delivery. In general,

a data packet is first forwarded to and then travel along a pre-selected ring until it meets an MS, a sensor node with fresh visit history of an MS, or finishes the travel along the ring and then chooses another not-visited-yet ring (if any) for the packet delivery. Upon meeting a sensor node with fresh visit history of an MS, the data packet will be forwarded towards the MS using trail based forwarding (see later for details). When a source sensor node *s* has a data packet to report, if it has neither MS in its direct communication range nor recent visit history of an MS in its cache, it randomly chooses an integer $k \in [1, M]$ as the id of the initial target ring to travel, which is to be encapsulated in the *target_ring* field in the data packet, initial start position (i.e., start virtual angle position) on target ring as -1 (i.e., still unknown). The node *s* further sets the TTL field for the data packet. The data packet will be discarded if the TTL drops to 0.

R3 works as follows: Upon generation or receipt of a data packet, if the current node has an MS in its direct transmission range, it will directly forward the packet to the MS; else if it has a fresh sink trail in its cache, it will perform trail based forwarding; otherwise, it will perform ring based forwarding for delivering the data packet; The node receiving the packet will repeat the above procedure until the packet is successfully delivered to an MS or timed out and thus dropped. Therefore, we can see that, in the R3 protocol, trail based forwarding has priority to be taken than ring based forwarding because the enforcement of the former means the discovery of a fresh sink trail which leads to an opportunity for accelerated data packet delivery in limited hops. To ease the presentation, in the following, we shall first focus on introducing how the ring based forwarding works without considering the opportunity for a packet holder to change to the trail based forwarding mode.

Fig. 3 presents the pseudo codes describing how the ring based forwarding works in details. The pseudo codes contain one main function *RingBasedForwarding*(), which further contains three sub-fuctions: *ForwardToTargetRing*(), *BypassBrokenPoint*(), and *FindNextHopOnRing*(). The *ForwardToTargetRing*() sub-fuction is to forward a packet *m* to a selected target ring. The *BypassBrokenPoint*() sub-fuction is to bypass a ring broken point to resume regular ring based forarding when a packet encounters such a situation. The *FindNextHopOnRing*() sub-fuction (see **Fig. 4**) returns a next hop (if available) which leads to the largest virtual angle progress on a pre-determined direction (say anticlockwise direction). Next, we shall respectively introduce how each of these fuctions (and also the sub-fuctions) work and also how they work together to support efficient ring based forwarding.

1) Target ring selection

When a source node generates a data packet, we need a ring selection algorithm to select the initial target ring for the packet or selecting a new target ring after finishing the searching on one ring without finding an MS or a fresh sink trail. In R3, we randomly choose a not-selected-yet ring from [1, *M*] as the new target ring. Besides this selection algorithm, we had also implemented other selection algorithms including inner ring first, middle ring first, and maximum distance to already-selected rings first. The simulation results show no obvious difference among these algorithms. So we adopt the random selection algorithm in this paper.

2) Packet forwarding to target ring

The lines 1-3 in **Fig. 3** show how to forward a packet to its target ring. It is executed when a sensor node *i* receives or creates a data packet (as the source node). In this case, the node first checks whether the packet has reached or has already been on its target ring. Each data packet has a *start_va* field recording its start virtual angle where actual forwarding is made on a particular ring, whose default value is set to -1 (i.e., unknown initially). In Line 1, node *i* checks the value of *start_va* in the data packet. If *m.start_va* equals to -1, which means the packet has not reached its target ring yet, then *i* needs to forward the packet to its target ring.

The *ForwardToTargetRing*() sub-function in **Fig. 3** describes how this procedure works. More specifically, *i* compares its own ring id with the *target_ring* carried in the packet. If *ring_id*(*i*) > *m.target_ring*, which means *i* is outside the target ring, then *i* randomly selects a neighbor node with ring id *ring_id*(*i*)-1 as the next hop node (see Lines 18-19). Similarly, if *ring_id*(*i*) < *m.target_ring*, *i* will forward the data packet to a neighbor node with *ring_id*(*i*) + 1 (see Lines 20-21). And if *ring_id*(*i*) is equal to *m.target_ring*, which means *i* is the first node that the packet reaches on its target ring, node *i* will record its own virtual angle into the *start_va* field of the data packet, and then start the packet forwarding along that ring (see lines 22-24).

```
/* The procedure below is for node i to find a next hop for packet m and also a proper forwarding mode.*/
Procedure RingBasedForwarding(Node i, Packet m)
1     if (m.start_va == -1) // packet m has not reached its target ring yet.
2       ForwardToTargetRing(i, m)
3       return
4     elseif (i has a neighbor node j such that j.va == m.start_va and j is on anticlockwise side of i)
          // m has finished its trip on target ring without finding MS information.
5       if (all rings ∈[1, M] have already been tried)
6         drop m
7         return
8       else randomly choose one from the not-selected-yet rings ∈[1, M] as m.target_ring
9         m.start_va ⇐ -1
10        ForwardToTargetRing(i, m)
11        return
12    elseif (i.next_hop_on_ring ≠ NIL) // forward m on the current ring.
13      forward m to node i.next_hop_on_ring
14      return
15    else  // broken point bypassing is to be triggered.
16      BypassBrokenPoint(i, m)
17    return
/* sub functions*/

Procedure ForwardToTargetRing(Node i, Packet m)
18  if (i.ring_id > m.target_ring)
19          forward m to a random neighbor x such that x.ring_id == i.ring_id -1
20  elseif (i.ring_id < m.target_ring)
21    forward m to a random neighbor x such that x.ring_id == i.ring_id +1
22  else // i is the first node that m reaches on its target ring.
23    m.start_va ⇐ i.va            // mark packet m's start position on the current ring.
24    RingBasedForwarding(i, m)            // forward m on the current ring.
25  return

Procedure BypassBrokenPoint(Node i, Packet m)
26          m.BYPASS_MODE ⇐ ON
27    u ⇐ i.next_hop_for_bypass  // u is i's inner ring neighbor for bypassing
28    node i forwards packet m to node u
29        do{
30      if (u has a neighbor v such that v.ring_id == u.ring_id+1 and v is on the anticlockwise
            side of u and v leads to the max angle progress if multiple choices exist)
31                u forwards m to v and u ⇐ v  // m is sent to outer neighbor ring.
32        if (v.ring_id == i.ring_id)
33          m.BYPASS_MODE ⇐ OFF and return //bypass successfully
34              elseif (u.next_hop_on_ring ≠ NIL) // m has to keep traveling along the ring to which
                                                    u belongs
35                u forwards m to u.next_hop_on_ring and u ⇐ u.next_hop_on_ring
36            else  // u is another broken point and it will perform another local bypassing
37            u forwards m to u.next_hop_for_bypass and u ⇐ u.next_hop_for_bypass
38    }while(1)
39 return
```

Fig. 3. Pseudo code for ring based forwarding at a node *i* for it to forward a data packet *m*.

3) Packet forwarding on a ring

After the packet reaches its target ring, it will be forwarded along the ring until it is going to return back to its start position on that ring, in which case a new ring is to be selected for the forwarding if rings $\in [1, M]$ have not been exhausted. This on-ring forwarding process corresponds to Lines 4-16 in the *RingBasedForwarding*() procedure in **Fig. 3**. If the current packet holder *i* has a neighbor node whose virtual angle is equal to *start_va* in a data packet and further the neighbor node is on the anticlockwise side of *i* (as determined by the virtual angles of the two nodes), *i* considers that the data packet *m* has finished its travel around the current ring without discovering any MS information and there is no need to continue forwarding on the ring. In this case, node *i* either drops the packet if all rings $\in [1, M]$ have been exhausted (see Lines 5-7) or randomly selects a new target ring for the data packet if there still exist not-selected-yet rings (see Lines 8-11). Otherwise, if node *i* has neighbor nodes (all in the anticlockwise direction) on the current ring, it will choose the one leading to the largest angle progress among them as the next hop, denoted by *i.next_hop_on_ring* (see Lines 12-14), else a ring broken point bypassing process needs to be triggered (see Lines 15-16).

```
/* This algorithm returns the next hop of node i on the same ring with i, denoted by i.next_hop_on_ring */
/* nbr_on_ring[i]: list of neighbor nodes of i, each of which has the same ring id with i*/
Procedure FindNextHopOnRing(Node i)
1       if (i.ring_id ∉ [1, M]) //we do not care about those nodes with IDs ∉[1, M]
2               i.next_hop_on_ring ⇐ NIL
3               return
4       for each node x in nbr_on_ring[i]
5               define inc[x] ⇐ x.va – i.va.
6   if (inc[x] < 0)
7       inc[x] ⇐ inc[x] + 2π  //inc[x]∈(0, π) means x is on the anticlockwise side of i
                              inc[x]∈(π,2π) means x is on the clockwise side of i
8   find the node n such that inc[n] ∈ (0, π) and inc[n] is the largest among all inc[x] ∈ (0, π).
9       if (n ≠ NIL)
10              i.next_hop_on_ring ⇐ n
11      else
12        i.next_hop_on_ring ⇐ NIL
13      return
```

Fig. 4. Pseudo code for a node *i* to find its next hop on the same ring.

Fig. 4 presents the pseudo code for the procedure *FindNextHopOnRing*() which returns a next hop node for the current packet holder. The basic idea here is to choose a neighbor node on the same ring but with the largest progress in virtual angle along a pre-determined direction (e.g., anticlockwise direction as used in this paper). In other words, each data packet is expected to be forwarded as quickly as possible along a ring. Accordingly, each selected next hop is expected to cover more not-visited-yet area and thus have better chance of discovering an MS or MS visit information. Each node in the network runs the procedure after it obtains the ring ids and virtual angles of its neighbor nodes during the network initialization phase and records the returned value in its local entry *next_hop_on_ring*. Note that Lines 4-7 in **Fig. 4** are used to determine the relative direction (clockwise or anticlockwise) of a neighbor. Lines 6-7 are to avoid discontinuity in virtual angles near 0 for nodes on rings with IDs >1. To ease the understanding, **Table 1** gives the information requried for assisting packet forwarding in R3.

Table 1. Information required to be kept at a node for assisting packet forwarding in R3.

Information Type	Required Information
Node related	Node id, ring id, and virtual angle of the node
Ring related	Father node(s) on neighbor inner ring, children nodes on outer neighbor ring
Sink related	A flag indicating if the current node is an agent node or not and its next hop to anchor node; a flag indicating if the current node is an anchor node or not and next hop to sink
Forwarding related	Next hop
Sink Trail related	TrailExpirationTime, next hop on the trail
Neighbor related	One-hop neighbor list (including each neighbor node's id, ring id, and angle)

Table 2. Probability for a node to be a broken point on different rings.

Ring ID	Probability
1	0
2	0.122
3	0.166
4	0.183
5	0.231

However, some nodes may not have any neighbor nodes on the same ring in the anticlockwise direction and in this case their respective next hops will be null. We call these nodes as broken points and accordingly present a method for bypassing them by temporarily forwarding packets onto inner rings (see below).

4) Broken point bypassing

In some cases, ring based forwarding may encounter broken points. **Table 2** shows the probability for a node to be a broken point on different rings based on our simulation results. The simulations for generating the results in **Table 2** were carried out on 100 random networks, which have network size = 600 nodes and average degree ≈ 16. The results show that the probability of broken point occurrence on a ring increases with ring id.

Our solution for bypassing a broken point is to temporarily forward data packets onto inner ring(s) and return back to the ring where the bypassing was originally triggered as quickly as possible. The *BypassBrokenPoint*() sub-function in **Fig. 3** gives the pseudo codes for the bypassing when the current packet holder i is a broken point. In this function, the broken point i forwards packet m to a neighbor node, which has a ring id of $i.ring\_id - 1$ and further leads to the largest non-negative virtual angle progress (also in the anticlockwise direction) among all such neighbor nodes of i, and the selected next hop is denoted by $i.next\_hop\_bypass$. The detailed procedure for determining *next_hop_bypass* (not shown) is quite similar to that for determining *next_hop_on_ring* (see **Fig. 4**). To ease the presentation, let u represent $i.next\_hop\_bypass$. When u receives the packet, it will try to forward the packet in the following orders: (1) forward it back to neighbor node on a ring with a larger ID and also leading to positive angle progress (called outward-forwarding), or (2) forward it along the same ring (i.e., to $u.next\_hop\_on\_ring$) if $u.next\_hop\_on\_ring \neq$ NIL (called same-ring forwarding), or (3) deepen the bypassing by forwarding the packet to a next hop on u's inner neighbor ring if u itself is also a ring broken point (called inward-forwarding). Moreover, to avoid the creation of loops during the bypassing process, a neighbor node of the current packet holder is qualified to serve as a next hop candidate only if it leads to positive progress for outward-forwarding and same-ring forwarding, or non-negative progress for inward forwarding for packets in bypassing mode, as compared to the current packet holder. Once the packet returns back to the ring where the bypassing was triggered, the bypassing operation is said to have terminated successfully. Our simulation results show that most bypassing

operations can be finished within three hops.

3.4 Trail based Forwarding

When a data packet reaches a node which has fresh visit history of an MS, it will then be forwarded via trail based forwarding instead of ring based forwarding. For this case, trail based forwarding similar to that in [12] is used, which is briefly introduced as follows.

Trail information is considered as visit history that MSs leave on sensor nodes as they move in the network. For this to be applicable, each MS needs to periodically broadcast HELLO messages to its one-hop neighbor sensor nodes to notify its existence. Each sensor node keeps trail records containing the IDs of most recent visited MSs and their respective visit times. The trail records at a sensor node are ranked by their freshness when multiple sink trails are known. Furthermore, we use a timer to specify the time when a sink trail is going to expire, which is the time when the record was created (or updated the last time) plus a parameter *TrailExpirationTime*. If timed out, the entry will be purged.

When having a data packet to forward, if an MS visit entry exists in local cache, a sensor node will perform trail based forwarding. It will first broadcast a Query message containing its freshest trail to its direct neighbor nodes. Upon receipt of the Query message, those neighbors with fresher trail of *any* MSs will try to respond with Reply messages. To suppress unnecessary replies, each neighbor node with a fresher trail sets a reply timer whose length is proportional to the time difference between its trail creation time (indicating the freshness of a sink trail) and that of the querying node's freshest trail creation time. Other neighbor nodes overhearing a Reply message will suppress their transmissions. Upon receipt of the Reply containing the latest sink visit, the querying node locally records the id of the Reply sending node and will use it for packet forwarding before the current sink trail expires. If no neighbor sends a Reply, then the sensor node will broadcast another Query message containing its next freshest trail (if any) and also delete the freshest one. The process is repeated if no Reply is received until no trail record is available, then the trail based forwarding fails and the node changes to use ring based forwarding and it will forward the data packet(s) to their nearest having-not-been-selected-yet ring. The above process continues until a data packet reaches an MS, or its TTL drops to 0 and thus dropped.

4. Performance Evaluation

In this section, we present the simulation results of the R3 protocol and compare it with the TRAIL protocol. Both protocols bring little protocol overhead for route learning and maintenance and further both of them are suitable for mWSNs with sporadic traffic. Moreover, no location information is required in the implementation of both protocols. We evaluate three main performance metrics which are widely adopted in the literature to evaluate routing efficiency in WSNs. The first metric is Packet Delivery Ratio (PDR). We calculate PDR by dividing the number of data packets that are successfully received by MSs with the number of generated data packets. The second metric is the average transmission cost per data packet, which is the average number of transmissions required to deliver one data packet, no matter the data packet is successfully delivered to an MS or not. The value of this metric is calculated by dividing the overall number of transmissions (including both data packets and control packets) with the number of data packets. Note that the protocol overhead for the ring structure building has also been considered. The third metric is the average data path length from source nodes to mobile sinks. The average path length is measured by the average hop distance taken by each successfully delivered data packet from its source node to its finally reached MS. The

simulator was developed using C++ and works in an event-driven manner.

In our simulations, sensor nodes are randomly deployed in a disk area with uniform distribution. The diameter of the disk area is 180m and the uniform transmission range of sensor nodes and MSs is set to 30m. The MAC layer is considered to be ideal such that no transmission loss/collision/corruption can occur. We use the random waypoint (RWP) movement model for every MS. Each MS moves at a constant velocity towards its destination and changes to another destination once it arrives at one destination. The destination positions are randomly and uniformly distributed in the monitoring area. Each simulation lasts 11000s, and the data traffic starts after 1000s for the purpose of trail training. The time for network initialization is simply considered to be zero and not counted here. A random sensor node is selected to generate a data packet to report every 2 second. We choose $M = 4$ because we found that, based on our simulation settings, nearly 70% of the network nodes are with ring ids ≤ 4. Further, our simulation results show that this percentage has been high enough to ensure a good routing performance. The expiration time of the visit history of an MS is 150s and the TTL of all data packets is set to 100. We study the impact of three important parameters: 1) Network size N, i.e., the number of deployed sensor nodes. 2) MS Number. 3) Average MS velocity. Each result reported below is the average result of 10 independent trials.

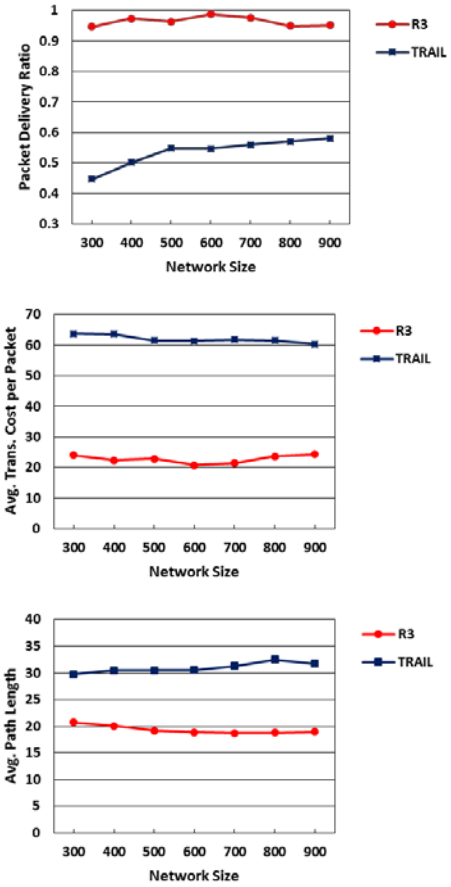


Fig. 5. Performance comparison by different protocols versus network size (i.e., number of sensor nodes in the network).

1) Impact of network size

To study the impact of network size on routing performance, we fixed the number of mobile sink to 1, the velocity of the mobile sink to 1 m/s, and varied the network size from 300 to 900 (with the average degrees varying from about 8 to about 25, resp.).

Fig. 5 compares the performance of different algorithms versus varying network size. The first observation is that the R3 protocol shows significant improvement comparing to the TRAIL protocol in all simulation scenarios. When the network size is 300, the PDR by R3 is 94.6 percent while that by TRAIL is just 44.6 percent. At the same time, the average transmission cost by R3 is only 37.6 percent of that by TRAIL, and the average E2E path length by R3 is about one third less than that by TRAIL. The results demonstrate the advantage of using ring based routing. The main reasons are two folds. First, in R3, a data packet is forwarded along parallel rings (in a one by one manner), which can increase its chance to meet an MS or fresh sink trail. In contrast, in TRAIL, before meeting a fresh sink trail, a data packet is forwarded using random walk routing, which may cause frequent revisit of previously met nodes, which reduces the likelihood for the packet to meet an MS. Second, in R3, each data packet is forwarded with the maximum possible progress in virtual angle at each hop when traveling along a ring, which can help reduce the number of hops to be taken to search the same size of area for an MS. While in TRAIL, data packets may be forwarded iteratively to randomly selected neighbors, which causes low searching efficiency. The results also show R3 works reasonably well in both low and high density networks.

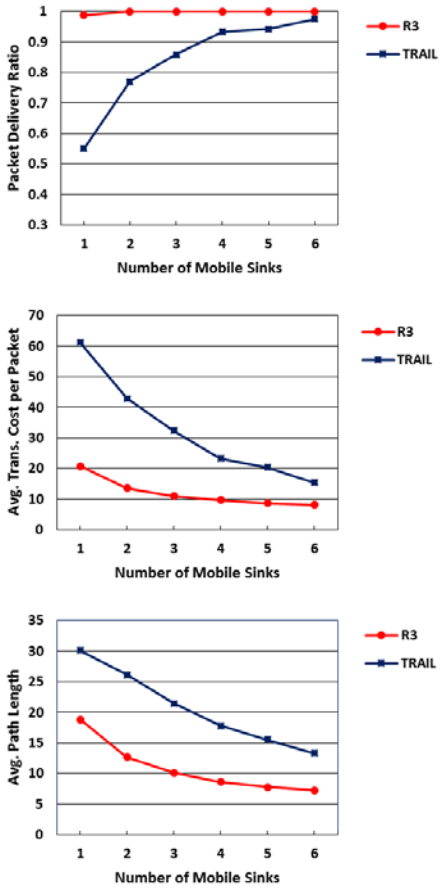


Fig. 6. Performance comparison by different protocols versus number of mobile sinks.

2) Impact of MS Number

To study the impact of the number of MSs on routing performance, we fixed network size to 600, the velocity of mobile sinks to 1 m/s, and varied the number of mobile sinks from 1 to 6. The initial positions and movement of the MSs are randomly and independently distributed.

Fig. 6 compares the performance of different algorithms versus varying number of MSs. Intuitively, more mobile sinks would make it easier to find a path to a mobile sink for data packets in both protocols, and the results conform to this intuition. The PDR by R3 stays near 100 percent when the number of MSs is more than 1, and the PDR by TRAIL grows fast with the number of MSs increasing. In both protocols, the average transmission cost and path length decrease quickly with the number of MSs increasing. In R3, when six MSs are available, the average end-to-end data path length is only 38.4 percent of that in the one-MS scenarios. The results show that deploying more MSs is an effective method for efficiently reducing the packet transmission cost.

3) Impact of MS Velocity

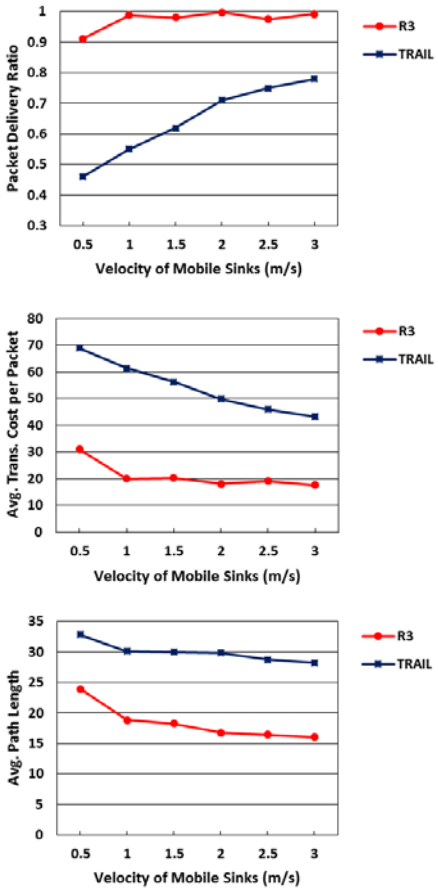


Fig. 7. Performance comparison by different protocols versus velocity of mobile sinks.

To study the impact of MS velocity on routing performance, we fixed the network size to 600, the number of MS to 1, and gradually varied the velocity of the MS from 0.5 m/s to 3 m/s.

Fig. 7 compares the routing performance of different algorithms versus the velocity of mobile sinks. In **Fig. 7**, the results show that higher MS velocities lead to higher performance

for both protocols. This is because MS with a higher velocity can enable more sensor nodes in the network to learn fresh sink trail in the same period of time since such an MS can meet more sensor nodes when it has higher velocity. As a result, more sensor nodes may change to perform trail based routing to forward data packets to the MS, which help reduce the path length and thus lead to improved routing performance. Also, in **Fig. 7**, it is seen that R3 significantly outperforms TRAIL in terms of each of the concerned measures due to the introduction of ring based forwarding.

5. Conclusion

In this paper, we proposed a lightweight reactive ring based routing protocol R3 for mWSNs. To support ring based packet forwarding, R3 introduces a novel strategy for constructing a base ring structure in the network with low protocol overhead, based on which more rings with accurate node angle information can be easily built to ease the routing in the network. To achieve improved performance, R3 integrates ring based forwarding and trail based forwarding. We present the detailed design description of R3 in the paper. Simulation results show that R3 can provide high data packet delivery and greatly reduced transmission cost as compared with existing work.

References

[1] M. D. Francesco, S. K. Das, G. Anastasi, "Data collection in wireless sensor networks with mobile elements: A survey," *ACM Transactions on Sensor Networks*, vol. 8, no. 1, pp. 1-31, Aug. 2011. Article (CrossRef Link)

[2] X. Li, A. Nayak, and I. Stojmenovic, "Sink mobility in wireless sensor networks," in *Proc. of 'Wireless Sensor and Actuator Networks: Algorithms and protocols for scalable coordination and data communication,'* Chapter 6, A. Nayak and I. Stojmenovic (eds.), pp. 153-184, *John Wiley & Sons, Inc.*, Hoboken, New Jersey, June 2010. Article (CrossRef Link)

[3] E. B. Hamida and G. Chelius, "Strategies for data dissemination to mobile sinks in wireless sensor networks," *IEEE Wireless Communications*, vol. 15, no. 6, pp. 31-37, Dec. 2008. Article (CrossRef Link)

[4] Z. Liu, J. Ma, Y. Park, and S. Xiang, "Data security in unattended wireless sensor networks with mobile sinks," *Wireless Communications & Mobile Computing*, vol. 12, no. 13, pp. 1131-1146, Sept. 2012. Article (CrossRef Link)

[5] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme, "Robomote: enabling mobility in sensor networks," in *Proc. of IEEE IPSN 2005*, pp. 55-66, Apr. 2005. Article (CrossRef Link)

[6] U. Park and J. Heidemann, "Data muling with mobile phones for sensornets," in *Proc. of ACM Sensys 2011*, pp. 162-175, Nov. 2011. Article (CrossRef Link)

[7] J.N. Al-Karaki and A.E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6-28, Dec. 2004. Article (CrossRef Link)

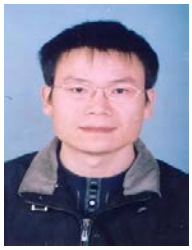
[8] E. B. Hamida and G. Chelius, "A line-based data dissemination protocol for wireless sensor networks with mobile sink," in *Proc. of IEEE ICC 2008*, pp. 2201-2205, May 2008. Article (CrossRef Link)

[9] G. Wang, T. Wang, W. J. Jia, M. Guo, and J. Li, "Adaptive location updates for mobile sinks in wireless sensor networks," *The Journal of Supercomputing*, vol. 47, no. 2, pp. 127–145, Feb. 2009. Article (CrossRef Link)

[10] F. Yu, S. Park, E. Lee, and S.-H. Kim, "Elastic routing: a novel geographic routing for mobile sinks in wireless sensor networks," *IET Communications*, vol. 4, no. 6, pp. 716-727, June 2010. Article (CrossRef Link)

[11] X. Li, J. Yang, A. Nayak, and I. Stojmenovic, "Localized geographic routing to a mobile sink with guaranteed delivery in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 9, pp. 1719-1729, Oct. 2012. Article (CrossRef Link)
[12] K. Shin and S. Kim, "Predictive routing for mobile sinks in wireless sensor networks: a milestone-based approach," *Journal of Supercomputing*, vol. 62, no. 3, pp. 1519-1536, Dec. 2012. Article (CrossRef Link)
[13] Y. Yan, B. Zhang, J. Zheng, and J. Ma, "Hierarchical Location Service for Wireless Sensor Networks with Mobile Sinks," *Wireless Communications and Mobile Computing*, vol. 10, no. 7, pp. 899-911, July 2010. Article (CrossRef Link)
[14] Y. Yan, B. Zhang, H. Mouftah, and J. Ma, "Hierarchical location service for large scale wireless sensor networks with mobile sinks," in *Proc. of IEEE Globecom 2007*, Washington, DC, USA, pp. 1222-1226, Nov. 2007. Article (CrossRef Link)
[15] S. Yu, B. Zhang, C. Li, and H. Mouftah, "Routing Protocols for Wireless Sensor Networks with Mobile Sinks: A Survey," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 150-157, July 2014. Article (CrossRef Link)
[16] K. Tian, B. Zhang, K. Huang, and J. Ma, "Data gathering protocols for wireless sensor networks with mobile sinks," in *Proc. of IEEE GLOBECOM 2010*, pp. 1-6, Dec. 2010. Article (CrossRef Link)
[17] L. Shi, B. Zhang, Z. Yao, K. Huang, and J. Ma, "An efficient multi-stage data routing protocol for wireless sensor networks with mobile sinks," in *Proc. of IEEE GLOBECOM 2011*, pp. 1-5, Dec. 2011. Article (CrossRef Link)
[18] Z. Li, M. Li, J. Wang, and Z. Cao, "Ubiquitous data collection for mobile users in wireless sensor networks," in *Proc. of IEEE INFOCOM 2011*, pp. 2246-2254, Apr. 2011. Article (CrossRef Link)
[19] J. W. Lee, B. Kusy, T. Azim, B. Shihada, and P. Levis, "Whirlpool routing for mobility," in *Proc. of ACM MOBIHOC 2010*, pp. 131-140, Sept. 2010. Article (CrossRef Link)
[20] L. Shi, B. Zhang, H. Mouftah, and J. Ma, "DDRP: An efficient data-driven routing protocol for wireless sensor networks with mobile sinks," *International Journal of Communication Systems*, vol. 26, no. 10, pp. 1341–1355, Oct. 2013. Article (CrossRef Link)
[21] L. Shi, B. Zhang, K. Huang, J. Ma, "An Efficient Data-Driven Routing Protocol for Wireless Sensor Networks with Mobile Sinks," in *Proc. of IEEE ICC 2011*, Kyoto, Japan, pp. 1-5, June 2011. Article (CrossRef Link)
[22] J. Luo and J.-P. Hubaux, "Joint sink mobility and routing to increase the lifetime of wireless sensor networks: The case of constrained mobility," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 871-884, June 2010. Article (CrossRef Link)
[23] Y. Shi and Y. T. Hou, "Theoretical results on base station movement problem for sensor network," in *Proc. of IEEE INFOCOM 2008*, pp. 1-5, Apr. 2008. Article (CrossRef Link)
[24] L. He, Z. Yang, J. Pan, L. Cai, and J. Xu, "Evaluating service disciplines for mobile elements in wireless ad hoc sensor networks," in *Proc. of IEEE INFOCOM 2012*, pp. 576-584, Mar. 2012. Article (CrossRef Link)
[25] W. Liang, J. Luo, and X. Xu, "Network lifetime maximization for time-sensitive data gathering in wireless sensor networks with a mobile sink," *Wireless Communications and Mobile Computing*, vol. 13, no. 14, pp. 1263-1280, Oct. 2013. Article (CrossRef Link)
[26] C. Intanagonwiwat, R. Govindan, D. Estrin, and J. Heidemann, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2-16, Feb. 2003. Article (CrossRef Link)
[27] Y. Wang, J. Gao, and J. S. B. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proc. of ACM MOBICOM 2006*, pp. 122-133, Sept. 2006. Article (CrossRef Link)
[28] S. Funke, "Topological hole detection in wireless sensor networks and its applications," in *Proc. of ACM DIALM-POMC 2005*, pp. 44-53, Sept. 2005. Article (CrossRef Link)

Sheng Yu received his BS degree in computer science and technology from Tsinghua University in 2008. He is currently a Ph.D. candidate in computer science at Research Center of Ubiquitous Sensor Networks at the University of Chinese Academy of Science, Beijing, China. His main research interests are routing and scheduling problems in mobile wireless sensor networks and ad hoc networks.



Baoxian Zhang received his PhD degree in Electrical Engineering from Northern Jiaotong University, China, in 2000. He is currently a Full Professor with the Research Center of Ubiquitous Sensor Networks at the University of Chinese Academy of Sciences (UCAS), Beijing, China. Prior joining UCAS, he was a Research Scientist with School of Information Technology and Engineering, University of Ottawa, Canada from 2002 to 2005. From 2001 to 2002, he was a Postdoctoral Fellow with Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada. He is currently an Associate Editor of *IEEE Systems Journal* and has served as Guest Editors of several special issues including *IEEE Journal on Selected Areas in Communications* and *Elsevier Ad Hoc Networks Journal***.** He has served on technical program committees for many international conferences and symposia. He has published over 150 refereed technical papers in archival journals and conference proceedings. His research interests cover network protocol and algorithm design, wireless ad hoc and sensor networks, Internet of Things, IP networks. Dr. Zhang is a senior member of the IEEE (2012).

Zheng Yao received his B.S. degree in Computer Science and Engineering from Xi'an Jiaotong University in 1991, and received his MS and Ph.D degree in Computer Application from Harbin Institute of Technology, China, in 1994 and 1997, respectively. He is currently a Full Professor with Research Center of Ubiquitous Sensor Networks, University of Chinese Academy of Sciences. He has published over 20 papers in journals and conference proceedings. His research interests covers software engineering, open source software, wireless sensor networks, and internet of things.

Cheng Li received the B. Eng. and M. Eng. degrees from Harbin Institute of Technology, Harbin, P. R. China, in 1992 and 1995, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Memorial University, St. John's, Canada, in 2004. He is currently a Full Professor at the Faculty of Engineering and Applied Science of Memorial University, St. John's, Canada. His research interests include mobile ad hoc and wireless sensor networks, wireless communications and mobile computing, switching and routing, and broadband communication networks. He is an editorial board member of *Wiley Wireless Communications and Mobile Computing* and *Journal of Networks*, and an associate editor of *Wiley Security and Communication Networks*. He has served as a co-chair for various technical symposia of many international conferences, including the IEEE GLOBECOM and ICC. He has served as the TPC member for many international conferences, including the IEEE ICC, GLOBECOM, and WCNC. Dr. Li is a registered Professional Engineer (P. Eng.) in Canada and is a Senior Member of the IEEE and a member of the IEEE Communication Society, Computer Society, Vehicular Technology Society, and Ocean Engineering Society.