Wireless Networks
https://doi.org/10.1007/s11276-018-1779-z

Efficient receiver-based flooding in mobile ad hoc networks

Xin Bai[1] · Xiaohui Wei[1] · Sen Bai[2]

Abstract
Flooding is one of the most important operations in *mobile ad hoc networks*. There are two strategies for existing flooding algorithms to make the forwarding decision. The first strategy is sender-based, where a forwarding node is in charge of determining its next-hop forwarding nodes. Existing sender-based algorithms can achieve linear computation overhead for each node, but this is at the expense of the transmission redundancy. The second strategy is receiver-based, where a node makes the forwarding decision of itself when it receives the flooding message. Receiver-based flooding algorithms perform well on reducing redundancy of transmissions, but each node requires at least $O(n^2)$ computations, where n is its number of neighbors. In this paper, we show that a receiver-based algorithm can achieve both good performance and low computation overhead at the same time. We first introduce an efficient $O(nlogn)$ algorithm, RBF. Second, two extensions of RBF, RBF-E1 and RBF-E2, were proposed. RBF-E1 has outstanding performance (1) in 3D wireless networks, and (2) when the defer time is short. RBF-E2 further reduces the computation overhead for each node to $O(n)$. The proposed algorithms require only one-hop neighbor location information to make the forwarding decision. Extensive simulations are carried out to evaluate the performance of our algorithms.

Keywords Flooding · Mobile ad hoc network (MANET) · Broadcasting

1 Introduction

We consider the flooding problem in *mobile ad hoc networks* (MANETs) in this paper. Since the topology of a MANET is dynamic, flooding enables to disseminate a message to the network without the knowledge of its topology in advance. Flooding plays a key role in route discovery for some routing protocols of ad hoc networks, such as AODV [24], DSR [12] and LAR [17]. The importance of flooding also rises because of its usefulness in the recent applications of IoT networks [7, 19]. Flooding is different from some broadcasting mechanisms in static wireless networks, such as [2, 5, 6, 26, 31, 34], which are used for transmission of large amount data or stream data. The broadcast mechanisms need to find an efficient route before actual transmission of data. However, flooding does not require routing beforehand, and is more appropriate to be used for dissemination of control messages in mobile networks.

The simplest flooding is pure flooding [11]. In pure flooding, a node transmits and only transmits the flooding message to its neighbors when it is the first time to receive it. In a collision-free and connected wireless network, pure flooding guarantees full delivery of the message. However, since all nodes in the network serve as forwarding nodes, pure flooding suffers from severe transmission redundancy and signal interference problem, which is referred to the "broadcast storm" problem [30]. Therefore, efficient flooding algorithms were demanded to reduce the redundant transmissions of flooding.

The flooding algorithms in MANETs can be classified into three categories based on the information each node maintains: (1) no need of neighbor information, (2) one-

✉ Xiaohui Wei
weixh@jlu.edu.cn

Xin Bai
baixin13@mails.jlu.edu.cn

Sen Bai
baisenbx@126.com

[1] College of Computer Science and Technology, Jilin University, No. 2699, Qianjin Street, Changchun 130012, China

[2] School of Software, Tsinghua University, No. 30, Shuangqing Street, Beijing 100084, China

Published online: 27 June 2018

hop neighbor information, (3) at least two-hops neighbor information. Pure flooding is a typical example of the first category. Some probabilistic-based flooding algorithms [29] also require no neighbor information. These algorithms in the first category cannot guarantee both fully delivery and efficiency. Though algorithms in the third category [20, 32] can both guarantee full delivery and reduce transmissions effectively, maintaining at least two-hops neighbor information incurs high overhead in networks with high mobility and high node density. Therefore, flooding algorithms in the second category, such as [14, 18], are considered more promising.

There are two strategies for selecting forwarding nodes of flooding: sender-based and receiver-based. A sender-based flooding algorithm selects a subset of neighbors for a forwarding node to be the next hop forwarding nodes. When a node decides to forward the flooding message, it attaches the list of its forwarding neighbors to the flooding message. Existing sender-based algorithms can be implemented in $O(n)$ [14] or $O(nlogn)$ [18] time, where n is the maximum number of neighbors of a node, but they cannot effectively reduce the redundant message transmissions.

A receiver-based flooding algorithm does not need to attach additional information to the flooding message. Using a receiver-based algorithm, a node makes its forwarding decision when it receives the message. The best-known 1-hop neighbor information based flooding algorithm in MANETs is proposed by Khabazian and Bhargavas [14]. Their algorithm is highly efficient on reducing redundant transmissions. However, their algorithm requires $O(n^2)$ computation overhead for each node.

In this paper, we introduce an efficient flooding algorithm requiring location information of only 1-hop neighbors, the RBF algorithm. RBF is highly efficient, and can be implemented in $O(nlogn)$ time. Two extensions of RBF, RBF-E1 and RBF-E2, were proposed: (1) RBF-E1 outperforms other algorithm outstandingly in 3D networks. It also performs well when defer time is short, where shorter defer time leads to lower latency of flooding; (2) RBF-E2 further reduces the computation overhead for each node to $O(n)$.

2 Related work

We briefly review the delivery-guaranteed flooding algorithms in MANETs in this section. For other flooding algorithms such as the probabilistic-based schemes, readers can refer to the surveys [27, 28].

2.1 Sender-based flooding algorithms in MANETs

The basic idea of most sender-based flooding algorithms is to compute a forwarding set of one-hop neighbors of a node, such that the transmission area of this forwarding set is able to cover its two-hops neighbors. Some previous works denote this problem as the Minimum Forwarding Set Problem (MFSP). The MFSP problem has been proven NP-complete in general graphs by Qayyum et al. [25]. Calinescu et al. [4] studied the MFSP problem in Unit Disk Graphs (UDGs), and they proposed two flooding algorithms based on the location information of neighbors. Their first algorithm has approximation ratio 6 and time complexity $O(nlogn)$. Their second algorithm has approximation ratio 3 and time complexity $O(nlog2n)$, where n denotes the number of 2-hops neighbors of a node. Baysan et al. [3] proposed a polynomial time complexity algorithm to compute the exact solution of the MFSP problem in UDGs. Wu et al. [33] studied an extended version of the MSFP problem where the three-hops neighbor information is considered. Ahn et al. [1] added additional MPR nodes as forwarding nodes to improve the robustness of the system.

For 1-hop information based algorithms, Liu et al. [18] studied how to solve the MSFP problem when only 1-hop neighbors' location information is available. They proposed an algorithm with time complexity $O(nlogn)$, where n is the number of 1-hop neighbors of a node. Khabbazian and Bhargava [14] proposed another subclass of sender-based flooding algorithms, where each node can decide whether or not to forward after each message reception. Whereby they proposed a 1-hop neighbors' location information based algorithm with $O(n)$ time complexity.

2.2 Receiver-based flooding algorithms in MANETs

For receiver-based flooding algorithms, Peng et al. [23] proposed a Scalable Broadcast Algorithm (SBA). The SBA algorithm require 2-hop neighbors' topology information, but it can be simply extended to the case where only one-hop neighbors' location information is available. Dai and Wu [8, 32] studied the flooding algorithms based on self-pruning. They showed that their algorithm is more efficient than SBA on reducing redundant message transmissions. However, their algorithm requires the topology information of at least two-hops neighbors of a node. Dai and Wu [9] studied the flooding problem in MANETs using direction

antennas. Khabbazian et al. [13] proposed a receiver-based flooding algorithm using location information of two-hops neighbors. They proved that the algorithm guarantees a constant approximation in UDGs. Khabbazian et al. [15] proposed a receiver-based flooding algorithm using 2-hops neighbor topology information. Khabbazian et al. extended their receiver-based algorithm in [14] when only 1-hop neighbor information is available. As aforementioned, this algorithm has the best-known efficiency but can only be implemented in $O(n^2)$ time. More recently, Papanikos et al. [21, 22] studied the flooding problem with the coding theory.

2.3 Summary

The sender-based algorithms achieve lower computation overhead than the receiver-based algorithms, but this is at the expense of their performance. Existing flooding algorithms cannot guarantee both good efficiency and low computation overhead at the same time.

3 System model

This paper uses the Protocol Model and Physical Model proposed in [10] as the system model. The same system model is also used [14, 18]. We consider a wireless network as a collection of wireless nodes equipped with omnidirectional antennas. The nodes in the network have transmission range R. A pair of nodes can communicate with each other if they are located in each other's cover area with radius R.

Our algorithms require each node to keep its 1-hop neighbor information, including their *IDs* and geographic locations. The location information of nodes can be obtained via some localization techniques such as Global Positioning System (GPS). The nodes in the network form a connected graph. Each node periodically broadcasts a *Hello* message including its *ID* and location. In the medium access control (MAC) layer, we assume that scheduling is done according to the p-persistent CSMA/CA protocol, which is based on IEEE 802.11 in the broadcast mode. That is to say, when a node has a message to transmit, it initiates a defer timer by a random number and starts listening to the channel. The node continues to listen until the channel is idle, then it starts decreasing the defer timer at the end of each time unit. The message is broadcasted when the defer timer expires.

The algorithms proposed in this paper follow the receiver-based flooding model as shown in Algorithm 1.

Algorithm 1: Receiver-based flooding

```
Extract information from the received message M
if M has been received before then
    drop the message
else
    set a defer timer
if the defer timer expires then
    decides whether or not to forward the message
```

4 The RBF algorithm

The RBF algorithm is proposed in this section. We first introduce the coverage condition of RBF, and prove that a receiver-based flooding as shown in Algorithm 1 guarantees full delivery under this coverage condition. Then, we introduce the implementation of RBF.

4.1 The coverage condition of RBF

Consider a node v receives a flooding message. We use $N(v)$ to denote the set of neighbors of v. During the defer time, suppose that v receives the same message from its neighbor set $S(v) \subset N(v)$. The cover area of nodes in $S(v)$ is denoted as $C(S(v))$. We define two subsets of vs neighbors $P(v) \subset N(v)$ and $Q(v) \subset N(v)$ as (1):

$$\begin{aligned} P(v) &= \{a | a \in N(v), a \in C(S(v)) \text{ and } ID_a < ID_v\} \\ Q(v) &= \{a | a \in N(v), a \notin C(S(v))\} \end{aligned} \tag{1}$$

$P(v)$ includes the neighbors of v with smaller *ID*s than v and located in area $C(S(v))$. $Q(v)$ includes the neighbors of v out of area $C(S(v))$.

As shown in Fig. 1, suppose node N_5 receives the flooding message from N_2 and N_6 during the defer time. First, since $S(v)$ denotes the nodes that have received the

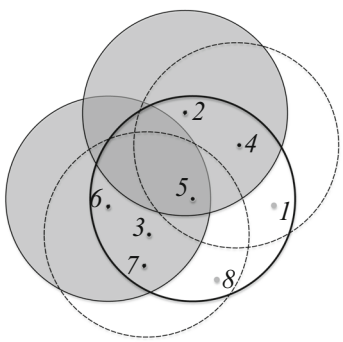


Fig. 1 An example illustrating the RBF algorithm. The numbers represent the *ID*s of the nodes

message, we have $S(N_5) = \{N_2, N_6\}$. Second, the shadowed area is $C(S(N_5))$, which denotes the cover area of N_2 and N_6. Note that N_3, N_4 and N_7 is in $C(S(N_5))$. Third, since $P(N_5)$ denotes the nodes in $C(S(N_5))$ with smaller *ID* than N_5, we have $P(N_5) = \{N_3, N_4\}$. At last, $Q(N_5)$ denotes the nodes that is not in $C(S(N_5))$. Therefore, $Q(N_5) = \{N_1, N_8\}$.

The main idea of RBF is that if any node in $Q(v)$ is covered by at least one node in $P(v)$, then node v is not required to forward the flooding message. Hereafter in this paper, we use $|uv|$ to denote the Euclidean distance between two nodes u and v. The idea is presented in Coverage Condition 1 and Rule 1.

Coverage Condition 1 (RBF) *For every node $u \in Q(v)$, there exists at least one node $w \in P(v)$ such that $|uw| \leq R$.*

Rule 1 (RBF) *If Coverage Condition 1 holds for a node v, then v does not forward the flooding message.*

As the example shown in Fig. 1, any node in $Q(N_5)$ is covered by at least one node in $P(N_5)$, so that node N_5 will not forward the flooding message according to Rule 1.

We prove that RBF can guarantee full delivery in collision-free networks in Theorem 1.

Theorem 1 *In a collision-free network, Algorithm 1 can achieve full delivery if each node uses Rule 1 to make the forwarding decision.*

Proof With Algorithm 1, each node forwards a message at most once. Therefore, the flooding will eventually terminate.

By contradiction, suppose that there is at least one node that has not received the message after the flooding termination. Let us consider the set

$$\varphi = \{(u, v) \mid u \text{ and } v \text{ are neighbors, } u \text{ has received the message, and } v \text{ has not received the message}\}.$$

First, we prove that $\varphi \neq \emptyset$. Suppose that r is the node that initiated flooding, and t is a node that does not receive the message. Consider a path between r and t. There must exist two adjacent nodes a and b along the path from r to t, such that node b has not received the message but node a has received. Hence, $\exists (a, b) \in \varphi$. That is, $\varphi \neq \emptyset$.

Since $\varphi \neq \emptyset$, we can consider one of its element (a', b') as follows, where a' has the smallest *ID* in φ.

$$\exists (a', b') \in \varphi \; s.t. \; \forall (u, v) \in \varphi, \text{ we have } ID_u > ID_{a'}. \quad (2)$$

Since (1) a' and b' are neighbors, and (2) b' has not received the message, we know that a' has not forwarded the message. Therefore, according to Rule 1, we have

1. If b' is not in $Q(a')$, then b' is in $C(S(a'))$, so b' will receive the message from a node in $S(a')$. This contradicts the fact that b' does not receive the message;
2. If b' is in $Q(a')$, then b' is covered by a node p in $P(a')$. Since $p \in C(S(a'))$, p receives the flooding message from a node in $S(a')$. Hence, we have $(p, b') \in \varphi$. However, $ID_p < ID_{a'}$ because $p \in P(a')$. This contradicts (2). □

The most relative algorithm to RBF is the SBA algorithm proposed in [23]. SBA adopts a stronger coverage condition which requires all the neighbors of the target node to be covered by the forwarded neighbors. For example, in the example shown in Fig. 1, the coverage condition of SBA requires N_4, N_3, N_7, N_1, and N_8 to be all covered by N_2 and N_6.

Remark The coverage condition of RBF is a necessary condition for the coverage condition of the SBA algorithm proposed in [23]. That is, RBF guarantees to outperform SBA.

Another comparable algorithm to RBF is Khabbazian's receiver-based algorithm in [14]. Instead of using identifiers as RBF, Khabbazian's algorithm makes the forwarding decision through measuring the distance among nodes. Though Khabbazian's algorithm can also achieve good performance, its coverage condition can only be checked within $O(n^2)$ time.

4.2 An *O(nlogn)* implementation of RBF

A trivial implementation of RBF is to compute the neighbor set of nodes in $P(v)$, and then check that whether all nodes in $Q(v)$ are in this neighbor set. This implementation has $O(n^2)$ time complexity. However, Liu et al. in [18] have shown that for a set of neighbors of a node, computing the boundary of their cover area requires only $O(nlogn)$ time, where n is the number of the neighboring nodes. With the help of Liu's approach, we show in this section that RBF can be implemented in $O(nlogn)$ time.

The $O(nlogn)$ implementation of RBF proposed in this paper has four phases. First, we compute the boundary of $C(S(v))$. Second, $P(v)$ and $Q(v)$ are computed according to the boundary of $C(S(v))$. Third, the boundary of $C(P(v))$ is computed. At last, we check that whether all nodes in $Q(v)$ are in the boundary of $C(P(v))$. The key issue of this implementation is to address the following problem: *Given a node v, for a set of its neighbors $\{a_1, a_2, \ldots, a_n\}$, how to compute boundary of the union of cover areas of $\{a_1, a_2,$*

$\cdots, a_n\}$? Liu et al. in [18] proposed an algorithm to address this problem in $O(nlogn)$ time. Since Liu's algorithm is complicated, we do not describe the details in this paper to save space. But we introduce the terminologies they used as follows to describe the implementation of RBF.

In [18], the boundary for the cover areas of a set of nodes is represented as the union of a set of arcs. Each arc is denoted as a triple (start angle, center, ending angle). For example, as shown in Fig. 2, suppose x, y, and z are neighbors of node v. As aforementioned, we use $C(\{x, y, z\})$ to denote the cover area of nodes in $\{x, y, z\}$, then the boundary of $C(\{x, y, z\})$ is represented as (3):

$$\begin{aligned}\{\overset{\frown}{od}, \overset{\frown}{db}, \overset{\frown}{ba}, \overset{\frown}{ac}, \overset{\frown}{co}\} = \{&(0, z, \angle ovd), (\angle ovd, y, \angle ovb),\\ &(\angle ovb, x, \angle ova), (\angle ova, y, \angle ovc),\\ &(\angle ovc, z, 2\pi)\}.\end{aligned} \tag{3}$$

The boundary divides the cover area inside the boundary into some sectors. For example, in Fig. 2, the cover area of the adjacent nodes of node v is divided into five sectors: ovd, dvb, bva, avc, and cvo. We use $sector_{arc}$ to denote the corresponding sector of an arc. For a neighbor u of v, its angle is defined as the angle from ov to uv according to the anticlockwise direction. For example, in Fig. 2, the angle of y is $\angle ovy$.

Details of the implementation is presented in Algorithm 2. In RBF, the boundary of $C(S(v))$ and $C(P(v))$ can be obtained in $O(nlogn)$ time using the Liu's algorithm in [18]. In the first half of Algorithm 2, we traverse $List_{N(v)}$ and $ARC_{S(v)}$ at the same time to compute $P(v)$ and $Q(v)$. In the second half of Algorithm 1, we traverse $List_{Q(v)}$ and $ARC_{P(v)}$ to check that whether every node in $Q(v)$ is covered by $C(P(v))$.

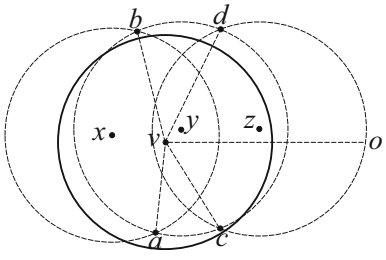


Fig. 2 The boundary for a set of nodes is represented as a union of arcs

Algorithm 2: The RBF algorithm

Input: $List_{N(v)}, List_{S(v)}$
Output: Output *true* if Coverage Condition 1 holds; Otherwise output *false*.

1 Compute the boundary of $C(S(v))$, output to $ARC_{S(v)}$
2 Sort nodes in $List_{N(v)}$ in an ascending order with their angles
3 Initiate two empty node lists $List_{P(v)}$ and $List_{Q(v)}$
4 $i \leftarrow 1, j \leftarrow 1$
5 **while** *true* **do**
6 **if** *the angle of* $List_{N(v)}[i]$ *is bigger than the end angle of* $ARC_{S(v)}[j]$ **then**
7 $j \leftarrow j + 1$
8 **continue**
9 **if** $List_{N(v)}[i]$ *is in* $sector_{ARC_{S(v)}[j]}$ *and* $ID_{List_{N(v)}[i]} < ID_v$ **then**
10 $List_{P(v)} \leftarrow List_{P(v)} + \{List_{N(v)}[i]\}$
11 **else if** $List_{N(v)}[i]$ *is not in* $sector_{ARC_{S(v)}[j]}$ **then**
12 $List_{Q(v)} \leftarrow List_{Q(v)} + \{List_{N(v)}[i]\}$
13 $i \leftarrow i + 1$
14 **if** $i > length(List_{N(v)})$ **then**
15 **break**
16 **if** $length(List_{Q(v)}) = 0$ **then**
17 **return** *true*
18 **else if** $length(List_{Q(v)}) > 0$ *and* $length(List_{P(v)}) = 0$ **then**
19 **return** *false*
20 Compute the boundary of $C(P(v))$, output to $ARC_{P(v)}$
21 Sort $List_{Q(v)}$ in an ascending order with their angles
22 $i \leftarrow 1, j \leftarrow 1$
23 **while** *true* **do**
24 **if** $List_{Q(v)}[i]$ *is in* $sector_{ARC_{P(v)}[j]}$ **then**
25 $i \leftarrow i + 1$
26 **else**
27 $j \leftarrow j + 1$
28 **if** $i > length(List_{Q(v)})$ **then**
29 **return** *true*
30 **else if** $j > length(ARC_{P(v)})$ **then**
31 **return** *false*

Theorem 2 *The time complexity of Algorithm* 2 *is* $O(nlogn)$, *where* n *is the number of neighbors of node* v.

Proof Step 1 and step 20 of Algorithm 2 respectively have time complexity $O(nlogn)$ and $O(mlogm)$ according to [18], where m is the number of nodes in $P(v)$. With the quick-sort algorithm, step 2 can be finished in $O(nlogn)$ time. Steps 3–19 have time complexity $O(n + l)$, where l is the number of nodes in $S(v)$. Step 21 has time complexity $O(klogk)$, where k is the number of nodes in $Q(v)$. Steps 22–31 have time complexity $O(k + m)$. Since $k \leq n, l \leq n, m \leq n$, the time complexity of Algorithm 2 is $O(nlogn)$. □

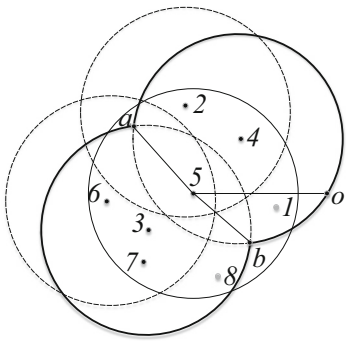


Fig. 3 An example illustrating the $O(nlogn)$ implementation of RBF. The numbers close to the nodes represent their IDs

We show an example in Fig. 3 to illustrate steps 22–31 of Algorithm 2. Steps 3–19 of Algorithm 2 are similar. Suppose node N_5 receives the flooding message from N_2 and N_6 during the defer time. Then $S(N_5) = \{N_2, N_6\}$. Hence, $P(N_5) = \{N_3, N_4\}$ and $Q(N_5) = \{N_1, N_8\}$. The bolded arcs are the arcs in $ARC_{P(v)}$ outputted by step 20. Thus, we have $ARC_{P(v)} = \{\overset{\frown}{oa}, \overset{\frown}{ab}, \overset{\frown}{bo}\}$ and $List_{Q(v)} = \{N_8, N_1\}$. We first consider N_8 and the sector of $\overset{\frown}{oa}$. Since N_8 is not in the sector of $\overset{\frown}{oa}$, we consider the next sector of $\overset{\frown}{ab}$. N_8 is in the sector of $\overset{\frown}{ab}$, so we consider the next node in $List_{Q(v)}$ which is N_1. N_1 is not in the sector of $\overset{\frown}{ab}$, so we consider the next sector $\overset{\frown}{bo}$. N_1 is not in the sector of $\overset{\frown}{bo}$, so the coverage condition of RBF holds.

5 The RBF-E1 algorithm

In fact, the coverage condition of RBF can be further improved, where nodes in $Q(v)$ only need to be path-covered by nodes in $P(v)$. This extension is denoted as RBF-E1 in this paper. In this section, we present descriptions of RBF-E1 and prove that RBF-E1 guarantees full-delivery in collision-free networks. Later with simulation, we will show that RBF-E1 has outstanding performance in two cases: (1) in 3D networks; and (2) when the defer time is short. A short defer time brings up low latency of flooding.

5.1 The coverage condition of RBF-E1

The main idea of RBF-E1 is that a node in $Q(v)$ is only required to be connected to a node in $P(v)$ by a path, instead of directly covered by a node in $P(v)$. In RBF-E1, the intermediate nodes along the path must have smaller *ID*s than v.

Coverage Condition 2 (RBF-E1) *For every node* $u_0 \in Q(v)$, *there exists at least one path* $(u_0, u_1, \ldots, u_k)$ *such that* $|u_i u_{i+1}| \le R (i = 0, 1, \ldots, k-1)$, $u_k \in P(v)$, *and* $ID_{u_i} < ID_v$ $(i = 1, 2, \ldots, k-1)$.

Rule 2 (RBF-E1) *If Coverage Condition 2 holds for a node v, then v does not forward the flooding message.*

Consider the example shown in Fig. 4. Suppose N_5 receives the flooding message from N_2 and N_6 during the defer time. Then $P(v) = \{N_3, N_4\}$ and $Q(v) = \{N_1, N_8, N_9\}$. The three paths (N_8, N_3), (N_1, N_4), and (N_9, N_1, N_4) make the coverage condition of RBF-E1 hold. Therefore, node N_5 is not required to forward the flooding message according to Rule 2.

Theorem 3 *In a collision-free network, Algorithm* 1 *can achieve full delivery if each node uses Rule* 2 *to make the forwarding decision.*

Proof By contradiction, suppose that there is at least one node that has not received the message after the flooding termination. Consider the set $\varphi = \{(u, v) \mid u$ and v are neighbors, u has received the message, and v has not received the message$\}$. Similar to the proof of Theorem 1, we have $\varphi \neq \emptyset$, and

$$\exists (a', b') \le \varphi \quad s.t.\ \forall (u, v) \in \varphi,\ we\ have\ ID_u > ID_{a'}. \qquad (4)$$

Since b' has not received the message, we know that a' has not forwarded the message. Therefore, according to Rule 2, we have (1) If b' is not in $Q(a')$, then b' is in $C(S(a'))$, so b' will receive the message from a node in $S(a')$. This contradicts the fact that b' does not receive the message; (2) If b' is in $Q(a')$, then there is a path $(b', u_1, \ldots, u_k)$, such that

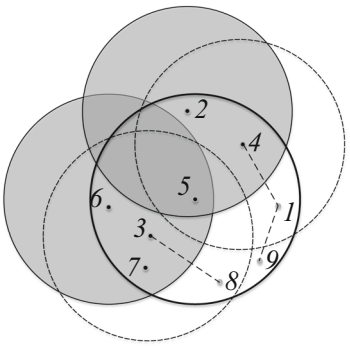


Fig. 4 An example illustrating the RBF-E1 algorithm. The numbers represent the IDs of the nodes

$u_k \in P(v)$, and $ID_{u_i} < ID_v (i = 1, 2, \ldots, k-1)$. u_k receives the flooding message because $u_k \in P(v)$. Therefore, there are at least two nodes (u_i, u_{i-1}) along the path such that u_i receives the message, but u_{i-1} does not receive the message, and $ID_{u_i} < ID_v$. This contradicts (4). □

5.2 Implementation of RBF-E1

RBF-E1 has a non-trivial $O(n^2)$ implementation. *Maximal independent set* plays a key role in this implementation. Maximal independent sets help us to devise implementation of RBF-E1 because (1) A maximal independent set can be found in $O(n)$ time; and (2) Under our system model, a maximal independent set in the neighborhood of a node is bounded by constant.

Definition 1 (*Independent Set*) An independent set of a graph is a subset of nodes in it, such that no two of which are adjacent.

Definition 2 (*Maximal Independent Set*) A maximal independent set of a graph is an independent set that is not a subset of any other independent set.

In the implementation of RBF-E1, the neighbor sets of v in (5) are used:

$$\begin{aligned} \breve{Q}(v) &= \{a \mid a \in Q(v), ID_a < ID_v\} \\ \hat{Q}(v) &= \{a \mid a \in Q(v), ID_a > ID_v\} \end{aligned} \tag{5}$$

As shown in Algorithm 3, we first build the node lists of $P(v)$, $Q(v)$, $\breve{Q}(v)$, and $\hat{Q}(v)$ in RBF-E1. Second, a maximal independent set of $\breve{Q}(v)$ is constructed, which is denoted as $MIS_{\breve{Q}(v)}$. Subsequently, we check for each node in $MIS_{\breve{Q}(v)}$ that whether there is a path from this node to a node in $P(v)$, such that all the intermediate nodes have smaller *ID*s than v. In fact, if every node in $MIS_{\breve{Q}(v)}$ has such a path, then every node in $\breve{Q}(v)$ has such a path. At last, we check that whether every node in $\hat{Q}(v)$ is adjacent to at least one node with smaller *ID* than v.

Algorithm 3: The RBF-E1 algorithm

Input: $List_{N(v)}, List_{S(v)}$
Output: Output *true* if Coverage Condition 2 holds; Otherwise output *false*.

Initiate empty node lists $List_{P(v)}$, $List_{Q(v)}$, $List_{\breve{Q}(v)}$ and $List_{\hat{Q}(v)}$.
for $i = 0; i \leq length(List_{N(v)}); i++$ **do**
  for $j = 1; j \leq length(List_{S(v)}); i++$ **do**
    if $dist(List_{N(v)}[i], List_{S(v)}[j]) \leq R$ **then**
      $List_{P(v)} \leftarrow List_{P(v)} + \{List_{N(v)}[i]\}$
      break
$List_{Q(v)} \leftarrow List_{N(v)} - List_{P(v)}$
for $i = 1; i \leq length(List_{P(v)}); i++$ **do**
  if $ID_{List_{P(v)}[i]} > ID_v$ **then**
    $List_{P(v)} \leftarrow List_{P(v)} - \{List_{P(v)}[i]\}$
for $i = 1; i \leq length(List_{Q(v)}); i++$ **do**
  if $ID_{List_{Q(v)}[i]} > ID_v$ **then**
    $List_{\breve{Q}(v)} \leftarrow List_{\breve{Q}(v)} - \{List_{Q(v)}[i]\}$
  else
    $List_{\hat{Q}(v)} \leftarrow List_{\hat{Q}(v)} - \{List_{Q(v)}[i]\}$
Initiate an empty node list $MIS_{\breve{Q}(v)}$
$L \leftarrow List_{\breve{Q}(v)}$
while $L \neq \emptyset$ **do**
  add an arbitrary node w in L to $MIS_{\breve{Q}(v)}$
  $L \leftarrow L - \{w\}$
  for $i = 1; i \leq lengthL; i++$ **do**
    if $dist(L[i], w) \leq R$ **then**
      $L \leftarrow L - \{L[i]\}$
build a graph G' with nodes in $(List_{P(v)} + List_{\breve{Q}(v)})$
for $i = 1; i \leq length(MIS_{\breve{Q}(v)}); i++$ **do**
  $check \leftarrow true$
  run breadth first search in G' with root $MIS_{\breve{Q}(v)}[i]$
  if *a node in* $List_{P(v)}$ *is traversed* **then**
    $check \leftarrow false$
    stop the breadth first search
  if *check* **then**
    return *false*
$L \leftarrow List_{\breve{Q}(v)} + List_{P(v)}$
for $i = 1; i \leq length(List_{\hat{Q}(v)}); i++$ **do**
  $check \leftarrow true$
  for $j = 1; j \leq length(L); j++$ **do**
    if $dist(L[j], List_{\hat{Q}(v)}[i]) \leq R$ **then**
      $check \leftarrow false$
      break
  if *check* **then**
    return *false*
return *true*

To analyze the complexity of Algorithm 3, we first introduce the following observation.

Observation *In unit disk graphs, a node has at most 5 independent neighbors. In unit ball graphs, a node has at most 12 independent neighbors.*

Proof In unit disk graphs, suppose u and v are two independent neighbors of node s. Clearly, we have $\angle usv > \pi/3$. Therefore, the number of independent neighbors of a node is less than $2\pi/(\pi/3) = 6$. In unit ball graphs, the lemma holds according to the kissing number problem [16]. □

Theorem 4 *The time complexity of Algorithm* 3 *is* $O(n^2)$, *where n is the number of neighbors of node v.*

Proof Steps 1–15 of Algorithm 3 have time complexity $O(n^2)$. Steps 16–23 of Algorithm 3 have time complexity $O(m^2)$, where m is the number of nodes in $\breve{Q}(v)$. Since there are constant nodes in $MIS_{\breve{Q}(v)}$, so the time complexity of steps 24–32 is $O(m^2 + k^2)$, where k is the number of nodes in $P(v)$. In addition, the time complexity of steps 33–42 is $O(l(m + k))$, where l is the number of nodes in $\hat{Q}(v)$. Since $k \le n$, $l \le n$, $m \le n$, the time complexity of Algorithm 3 is $O(n^2)$. □

6 The RBF-E2 algorithm

In this section we introduce the RBF-E2 algorithm, which weakens the coverage condition of RBF to reduce its computation overhead to $O(n)$. The main idea of RBF-E2 is to reduce the coverage condition of RBF from "the coverage of nodes" to "the coverage of nodes' projections".

6.1 The coverage condition of RBF-E2

We first E2. Suppose $S(v) = \{s_1, s_2, \ldots, s_m\}$ is the forwarded neighbors of v, λ is a constant tunable parameter, we define $S'(v)$, $P'(v)$, $Q'(v)$ as (6).

$$\begin{aligned} S'(v) &= \begin{cases} S(v), & \text{if } m \le \lambda \\ \{s_1, s_2, \ldots, s_\lambda\}, & \text{if } m > \lambda \end{cases} \\ P'(v) &= \{a \mid a \in N(v), a \in C(S'(v) \text{ and } ID_a < ID_v\} \\ Q'(v) &= \{a \mid a \in N(v), a \in C(S'(v))\} \end{aligned} \tag{6}$$

RBF-E2 modifies RBF-E1 from two aspects: (1) RBF-E2 only considers λ nodes in $S(v)$; (2) For every node in $Q'(v)$, we check that whether its projection on the boundary of v's disk is covered by any node in $P'(v)$. We will show by simulation that when $\lambda \ge 3$, RBF-E2 performs closely to RBF.

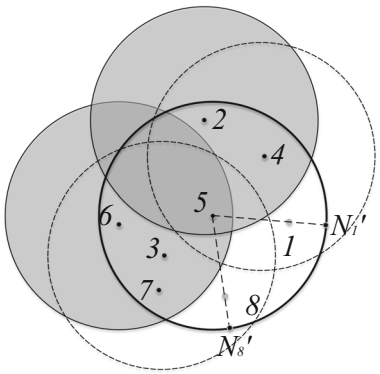


Fig. 5 An example illustrating Rule 3

Coverage Condition 3 (RBF-E2) *For every node* $u \in Q'(v)$, *its projection on the boundary of v's disk is covered by at least one node in* $P'(v)$.

Rule 3 (RBF-E2) *If coverage condition* 3 *holds for a node v, then v does not forward the flooding message.*

Consider the example shown in Fig. 5. Suppose N_5 receives the flooding message from N_2 and N_6 during the defer time, and $\lambda = 3$. Then $P'(v) = \{N_3, N_4\}$ and $Q'(v) = \{N_1, N_8\}$. The projections of nodes in $Q'(v)$ is $\{N_1', N_8'\}$. Since N_1' is covered by N_4 and N_8' is covered by N_3, Coverage Condition 3 holds.

Observation*Coverage Condition* 3 *is a sufficient condition of Coverage Condition 1.*

Proof As shown in Fig. 5, for any node $u \in Q'(v)$, if its projection on the boundary of v's disk is covered by $w \in P'(v)$, then u is also covered by w. Moreover, $P'(v) \subset P(v)$, and $Q'(v) \supset Q(v)$. Therefore, the theorem holds. □

Corollary 1 *In a collision-free network, Algorithm* 1 *can achieve full delivery if each node uses Rule* 3 *to determine whether or not to forward the message.*

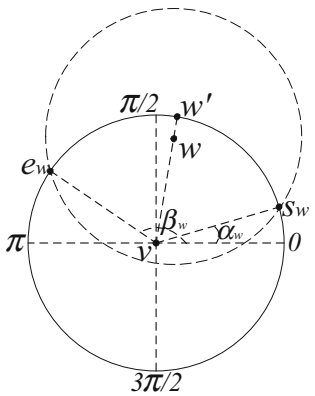


Fig. 6 s_w and e_w are defined as the start point and ending point of w

6.2 Implementation of RBF-E2

For a neighbor w of v and its projection w', as shown in Fig. 6, we define $\angle ovw'$ as its angle, s_w as its start point, e_w as its ending point, α_w as its start angle, β_w as its ending angle, and the arc from s_w to e_w as its arc. Notice that the angle of the arc of a neighbor ranges from $2\pi/3$ to π. Addressing following problem plays the key role on checking whether Coverage Condition 3 holds:

Problem *On a unit circle, given a set of arcs with central angles* $(2\pi/3, \pi)$, *and a set of points*, *how to determine that whether the points are covered by the arcs*?

In fact, this problem can be solved in linear time. The basic idea is to iteratively erase the covered points from the point set, and it can be proved that the iteration ends in constant rounds. In Algorithm 4, we first build the arc list $ARC_{P'(v)}$ for nodes in $P'(v)$, and the angle list $ANGLE_{Q'(v)}$ for nodes in $Q'(v)$. Then, the iteration begins. In each round of the iteration, we first compute the smallest angle α in $ANGLE_{Q'(v)}$. Then we find the arc in $ARC_{P'(v)}$ that (1) covers α, and (2) whose start angle is the closest to α. In each round, the angles in $ANGLE_{Q'(v)}$ that are covered by the selected arc are erased from $ANGLE_{Q'(v)}$.

Suppose θ_1 and θ_2 are the angles of two neighbors of v. As shown in Fig. 7, the difference between θ_1 and θ_2 is denoted as $diff(\theta_2, \theta_1)$, and defined as (7).

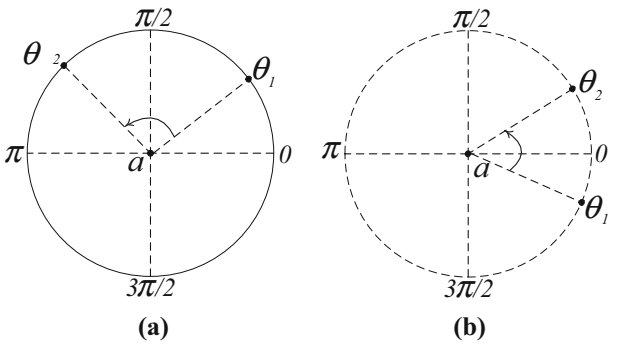


Fig. 7 **a** $diff(\theta_2, \theta_1)$ when $\theta_2 > \theta_1$; **b** $diff(\theta_2, \theta_1)$ when $\theta_2 < \theta_1$

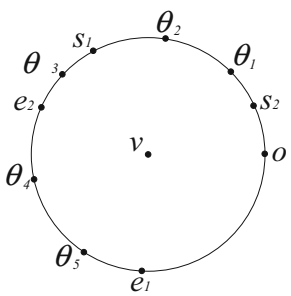


Fig. 8 An example illustrating Algorithm 3

$$diff(\theta_2, \theta_1) = \begin{cases} \theta_2 - \theta_1, & if\ \theta_2 > \theta_1 \\ \theta_2 - \theta_1 + 2\pi, & if\ \theta_1 > \theta_2 \end{cases} \quad (7)$$

Algorithm 4: The RBF-E2 algorithm

Input: $List_{N(v)}, List_{S'(v)}$
Output: Output *true* if Coverage Condition 3 holds; Otherwise output *false*.

Initialize $List_{P'(v)}$, $List_{Q'(v)}$, $ARC_{P'(v)}$, $ANGLE_{Q'(v)}$ to empty
for $i = 1; i \leq length(List_{N(v)}); i++$ **do**
  $check \leftarrow false$
  for $j = 1; j \leq \lambda; j++$ **do**
    if $dist(List_{N(v)}[i], List_{S'(v)}[j]) \leq R$ **then**
      $check \leq true$
      if $ID_{List_{N(v)}[i]} > ID_v$ **then**
        $List_{P'(v)} \leftarrow List_{P'(v)} + \{List_{N(v)}[i]\}$
      break
  if $check = false$ **then**
    $List_{Q'(v)} \leftarrow List_{Q'(v)} + \{List_{N(v)}[i]\}$
for $i = 1; i \leq length(List_{P'(v)}); i++$ **do**
  add the arc of $List_{P'(v)}[i]$ to $ARC_{P'(v)}$
for $i = 1; i \leq length(List_{Q'(v)}); i++$ **do**
  add the angle of $List_{Q'(v)}[i]$ to $ANGLE_{Q'(v)}$
while $ANGLE_{Q'(v)} \neq \emptyset$ **do**
  $\alpha \leftarrow 2\pi$
  for $i = 1; i \leq length(ANGLE_{Q'(v)}); i++$ **do**
    if $ANGLE_{Q'(v)}[i] < \alpha$ **then**
      $\alpha \leftarrow ANGLE_{Q'(v)}[i]$
  $\beta \leftarrow 2\pi$
  for $i = 1; i \leq length(ARC_{P'(v)}); i++$ **do**
    if $ARC_{P'(v)[i]}$ *covers* α *and* $diff(\alpha, ARC_{P'(v)}[i].startangle) < \beta$ **then**
      $arc \leftarrow ARC_{P'(v)}[i]$
      $\beta \leftarrow diff(\alpha, ARC_{P'(v)}[i].startangle)$
  if $\beta = 2\pi$ **then**
    return *false*
  else
    for $i = 1; i \leq length(ANGLE_{Q'(v)}); i++$ **do**
      if $ANGLE_{Q'(v)}[i]$ *is covered by arc* **then**
        erase $ANGLE_{Q'(v)}[i]$ from $ANGLE_{Q'(v)}$
return *true*

An example of the iteration in Algorithm 3 is shown in Fig. 8. Suppose there are two arcs in $ARC_{P'(v)}$ which are s_1e_1 and s_2e_2, and there are five angles in $ANGLE_{Q'(v)}$ which are $\{\theta_i\}(i = 1, 2, \ldots, 5)$. For the first round of the iteration, note that θ_1 is the smallest in $ANGLE_{Q'(v)}$. Hence, we search for the arc covers θ_1, and whose start angle is the closest to θ_1. This arc is s_2e_2. We erase the angles that are covered by s_2e_2 from $ANGLE_{Q'(v)}$, which are $\{\theta_1, \theta_2, \theta_3\}$. Thus completes the first round. In the second round, the smallest angle in the rest angles of $ANGLE_{Q'(v)}$ is θ_4. Note that s_1e_1 is the arc covers θ_4, and whose start angle is the closest to θ_1. Hence, we remove $\{\theta_4, \theta_5\}$ from $ANGLE_{Q'(v)}$.

The iteration ends after the second round and the algorithm outputs *true*.

Lemma 1 *Steps* 16–31 *of Algorithm* 3 *end in at most 6 rounds.*

Proof Let $\{arc_i\}(i = 1, 2, \ldots, k)$ denote the selected arcs in each round of the iteration. It can be easily proved that the central angle of arc_i ranges from $2\pi/3$ to π.

Let $s_i, e_i(i = 1, 2, \ldots, k)$ denote the start angle and ending angle of arc_i. Let $w_i(i = 1, 2, \ldots, k)$ denote the smallest angle in $ANGLE_{Q(v)}$ for each round. Now we prove that for any $i = \{1, 2, \ldots, k-2\}$, $diff(s_{i+2}, s_i) > 2\pi/3$. We have $diff(w_{i+1}, s_i) > diff(e_i, s_i) \geq 2\pi/3$ because all angles in $ANGLE_{Q'(v)}$ that are covered by arc_i have been erased in the ith round. Moreover, we have $diff(s_{i+2}, s_i) > diff(w_{i+1}, s_i)$, because otherwise the arc arc_{i+2} will be selected in the $(i+1)$th round. Therefore, for any $i = \{1, 2, \ldots, k-2\}$, $diff(s_{i+2}, s_i) > 2\pi/3$. Therefore, the iteration in Algorithm 3 ends in at most $2\pi/(2\pi/3)2 = 6$ rounds. □

Theorem 5 *The time complexity of Algorithm* 3 *is* $O(n)$, *where n is the number of neighbors of a node.*

Proof The first half of Algorithm 3 has time complexity $O(\lambda n)$. For the second half, each round of iteration spends $O(n)$ time. According to Lemma 1, Algorithm 3 has running time $O(n)$. □

7 Simulation

We carried out extensive simulations to compare our algorithms with existing 1-hop neighbor information based algorithms including the sender-based algorithms in [18] (denoted as LSA) and [14] (denoted as KSA), the receiver-based algorithms in [14] (denoted as KRA) and the one-hop SBA. Among them and the proposed algorithms, SBA, KRA, RBF, and RBF-E1 can be used and were tested in 3D networks. The algorithms are summarized in Table 1.

Remark RBF can be trivially extended to use 2-hops topology information instead of 1-hop neighbor position information for each node. This extension is denoted as RBF-E3 which is also evaluated in this simulation.

The ns-3 simulator was used for the simulation. The simulation parameters are shown in Table 2. The MAC layer follows the IEEE 802.11 MAC specification. In our simulation, a certain number of nodes, from 100 to 1000, are randomly placed on a $1000 \times 1000\,m^2$ area to represent a wireless network. The nodes form a connected graph.

The performance of the algorithms was investigated against two parameters: number of nodes, and transmission range. For each selected combination of parameters, 100 connected graphs were randomly generated, and the results were averaged. We define forwarding ratio and delivery ratio as (8) and (9) to evaluate the efficiency of the algorithms:

$$forwarding\ ratio = \frac{the\ number\ of\ forwarding\ nodes}{the\ number\ of\ total\ nodes} \quad (8)$$

$$delivery\ ratio = \frac{the\ number\ of\ nodes\ receiving\ the\ message}{the\ number\ of\ total\ nodes} \quad (9)$$

Table 1 The flooding algorithms in MANETs

Algorithm	Time complexity	System model
LSA	$O(nlogn)$	Sender-based
KSA	$O(n)$	Sender-based
SBA	$O(n^2)$	Receiver-based
KRA	$O(n^2)$	Receiver-based
RBF	$O(nlogn)$	Receiver-based
RBF-E1	$O(n^2)$	Receiver-based
RBF-E2	$O(n)$	Receiver-based

7.1 Performance versus number of nodes

To test the effects of number of nodes, we fix the transmission range at 250 m as shown in Fig. 9. For the receiver-based algorithms, the defer time is randomly selected from 0 to 1.0 s. For RBF-E2, λ is set to 3.

As shown in Fig. 9, when the number of nodes grows, the forwarding ratio decreases for all the algorithms. This is reasonable because the density of the nodes rises with the growing of node number. Then less percentage of nodes are required to forward the flooding message.

In general, the receiver-based algorithms outperform the sender-based algorithms, especially with small number of nodes. For the receiver-based algorithms, KRA and our algorithms perform closely, and they outperform SBA. For

Table 2 Simulation parameters

Parameter	Value
Simulator	ns-3(version 3.26)
MAC layer	IEEE 802.11
Propagation model	Two-ray Ground
Data packet size	64 KB
Bandwidth	2 Mb/s
Side length of area	1000 m
Transmission range	100–300 m
Number of nodes	100–1000

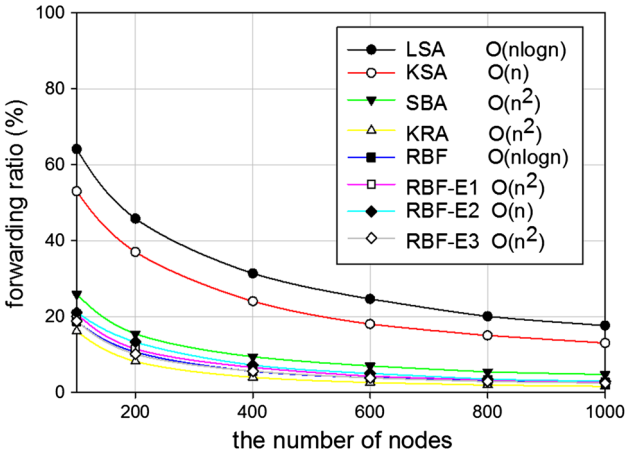


Fig. 9 Performance comparison with different number of nodes. The transmission range is fixed at 250 m. The defer time is set to 0–1.0 s

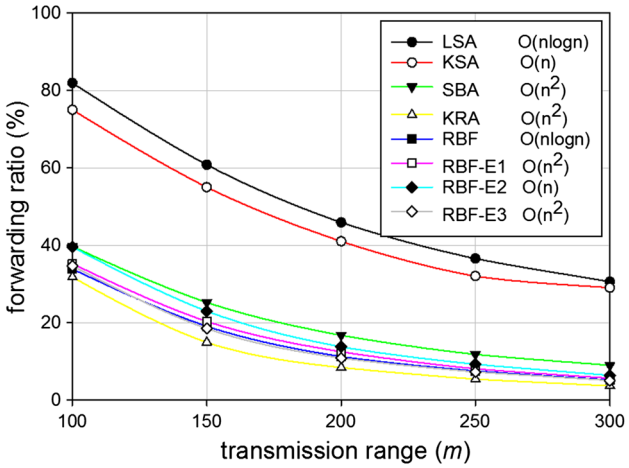


Fig. 10 Performance comparison with different transmission ranges. The number of nodes is fixed at 300. The defer time is set to 0–1.0 s

algorithms with linear time complexity, RBF-E2 greatly outperforms KSA. For algorithms with $O(nlogn)$ time complexity, RBF outperforms LSA, but RBF does not clearly outperform RBF-E2. For the $O(n^2)$ algorithms, KRA has the best forwarding ratio, but it is close to RBF-E2. Overall, RBF-E2 has both good computation complexity and low forwarding ratio, which leads to less computation overhead and more efficiency of the system.

7.2 Performance versus transmission range

The algorithms are tested under different transmission ranges as shown in Fig. 10. In the simulation, 300 nodes are randomly placed on a $1000 \times 1000\,\mathrm{m}^2$ area. The transmission range rises from 100 to 300 m. The defer time is randomly chosen from 0 to 1.0 s. For RBF-E2, λ is set to 3.

As shown in Fig. 10, the algorithms perform better when the transmission range of nodes increases. This is because the network density increases with the transmission range. KRA and our algorithms still outperform other algorithms.

7.3 Performance versus defer time

In the simulation introduced in previous subsections, the defer time of the receiver-based algorithms is set to be long enough from 0 to 1.0 s. However, a longer defer time leads to higher latency of flooding. We investigate the effect of defer time to the performance of the receiver-based algorithms in this subsection.

In the simulation, the defer time of nodes follows a uniform distribution from 0 to T_{defer}, where T_{defer} ranges from 0 to 100 ms. The number of nodes is fixed at 300, and the transmission range is fixed at 250 m. For RBF-E2, λ is set to 3. The results are presented in Fig. 11.

As shown in Fig. 11, the performance of SBA rapidly decreases when defer time becomes shorter. When T_{defer} is bigger than 40 ms, KRA and our algorithms perform closely. However, when T_{defer} is smaller than 40 ms, RBF-E2 relies more on shorter defer times for better performance compared with RBF, KRA, and RBF-E1. When T_{defer} becomes very small, only RBF-E1 still keeps high performance.

7.4 Collisions

We tested the number of collisions with the algorithms. The results are shown in Fig. 12. To make the receiver-based algorithms comparable to the sender-based algorithms, we set short defer times ranging form 0 to 1.0 ms.

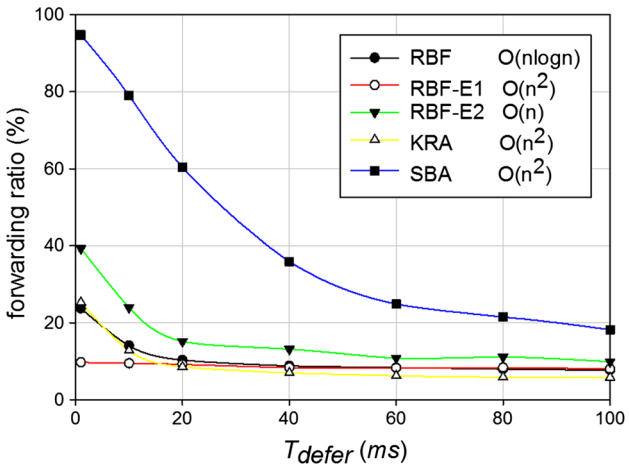


Fig. 11 The effect of defer time for the receiver-based algorithms. The number of nodes is fixed at 300. The transmission range is fixed at 250 m. The defer time ranges from 0 to T_{defer}

As we discussed in the last subsection, SBA has high forwarding ratio with short defer times. As a results, as shown in Fig. 12, the collision number for SBA is big and close to pure flooding. By comparison, other algorithms result in small number of collisions because they require less number of forwarding nodes. Specially, the collision number of RBF-E1 is the smallest since it requires the least number of forwarding nodes with short defer times.

Though collisions occur during the flooding and full delivery cannot be guaranteed theoretically, we found that for all the proposed algorithms the delivery ratio can always reach 100% in our simulation. This is because when a collision occurs, a node still has a chance to receive the flooding message from another node.

7.5 Latency

To test the running time of the algorithms, we define the latency of flooding as the time of the last node in the network to receive the message. To reduce the effect caused by the defer times, we set the defer times to 0–1.0 ms for the receiver-based algorithms. The results are shown in Fig. 13. We found that though KSA and RBF-E2 have linear computation complexities, they cannot achieve low latency because they incur more retransmissions. On the contrary, the algorithms with lower forwarding ratio, especially RBF-E1, also perform well in term of latency since they require less retransmisisons.

7.6 The effects of λ to RBF-E2

The performance of RBF-E2 was tested with different λ. The results are shown in Fig. 14. The node number, transmission range, and T_{defer} are set to 300, 250 m, and 1.0 s. The simulation results show that when λ is more than 3, RBF-E2 performs very closely to RBF.

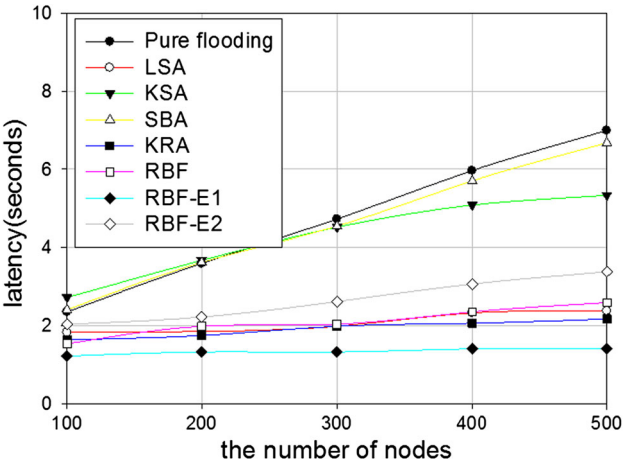


Fig. 13 The time of the last node to receive the flooding message. The transmission range is fixed at 250 m. For the receiver-based algorithms, the defer time is set to 0–1.0 ms

7.7 Performance comparison in 3D networks

For 3D wireless networks, the performance of SBA, KRA, RBF, and RBF-E1 is tested. RBF can be trivially implemented in $O(n^2)$ time in 3D networks. To represent a 3D network, a certain number of nodes, from 200 to 1000, are randomly placed on a $1000 \times 1000 \times 1000\,\mathrm{m}^3$ area. In the simulation, the transmission range is fixed at 250 m. The defer time ranges from 0 to 1.0 s. The results are shown in Fig. 15.

The simulation results show that RBF-E1 significantly outperforms other algorithms in 3D networks. The results indicate that RBF or KRA may be close to the optimal

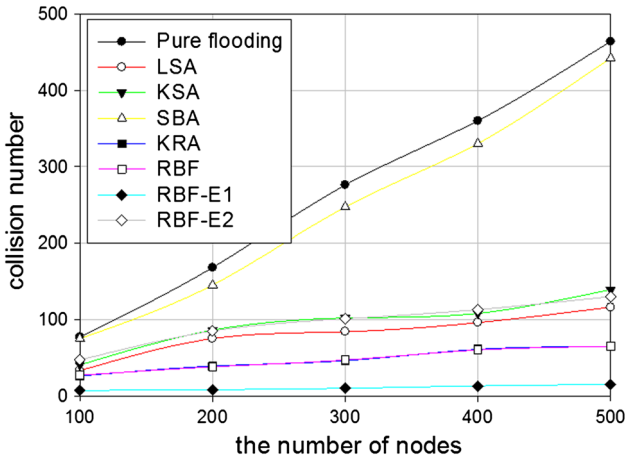


Fig. 12 The number of collisions with the algorithms. The transmission range is fixed at 250 m. For the receiver-based algorithms, the defer time is set to 0–1.0 ms

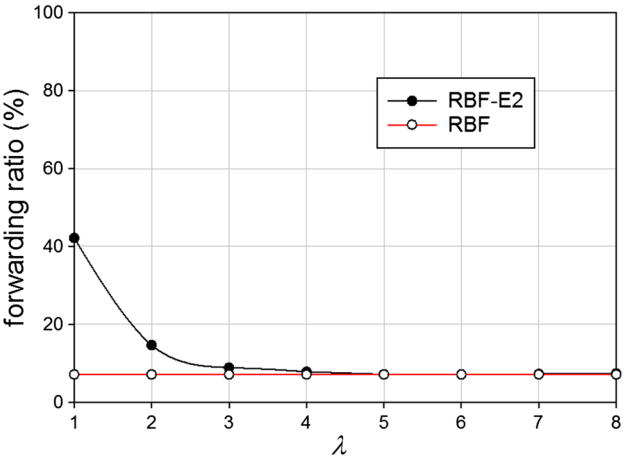


Fig. 14 Effects of λ to RBF-E2. The transmission range is fixed at 250 m. The node number is 300. T_{defer} is 1.0 s. The forwarding ratio of RBF is also shown for comparison

solution in 2D networks, but they are far from the optimal solution in 3D networks. With a better flooding rule, RBF-E1 gets closer to the optimal solution in 3D networks.

7.8 Mobility and message reception failures

Several reasons may cause the failures of message reception, such as mobility, localization errors, and signal collisions. We tested the delivery ratios to evaluate the effects of message reception failures to the performance of the algorithms.

In our simulation, we tested the performance of the algorithms under the random walk mobility model. The maximum velocity is set to 10 m a second. The transmission range is fixed at 250 m and the number of nodes ranges from 100 to 1000. The results show that all the proposed algorithms can still achieve at least 95% delivery ratio. A static and a mobile wireless network achieve almost the same forwarding ratio as the results shown in Fig. 9.

We also considered the effect of message-reception failure to the performance of the algorithms. We fixed the transmission range at 250 m, and tested delivery ratio of the algorithms with different probabilities of message-reception failure when the number of nodes is 100 and 400. The results are shown in Fig. 16(a) and (b). The results show that SBA is the most robust when there exist message-reception failures. RBF-E1 and KSA achieve obviously worse delivery ratio with message-reception failure, especially when the number of nodes is big. RBF-E2 guarantees better delivery ratio compared to RBF and KRA. This is because RBF-E2 uses the projection of a node to check the coverage condition, which has better chances to hold when nodes are mobile. However, the performance gap between RBF, KRA, and RBF-E2 becomes smaller when the number of nodes increases.

Furthermore, we considered the case that most of the links between nodes are reliable, but the rest ones are unreliable with certain probability of failure to receive messages. The results are shown in Fig. 17(a) and (b). In Fig. 17(a), 10% of the links are unreliable. We found that all the algorithms except RBF-E1 can guarantee nearly full delivery. For RBF-E1, when the probability of message-reception failure is above 80%, the delivery ratio will decrease but still be more than 95%. In Fig. 17(b), 30% of the links are unreliable. Most of the algorithms except RBF-E1 can still guarantee delivery ratios above 95%.

8 Conclusions

In this paper, we proposed an efficient flooding algorithm for MANETs based on location information of 1-hop neighbors, the RBF algorithm. RBF achieves both good

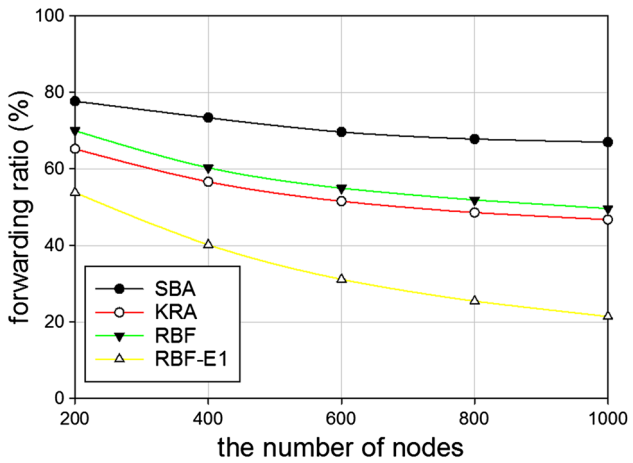


Fig. 15 Algorithm performance in 3D networks. The transmission range is fixed at 250 m. The defer time ranges from 0 to 1.0 s

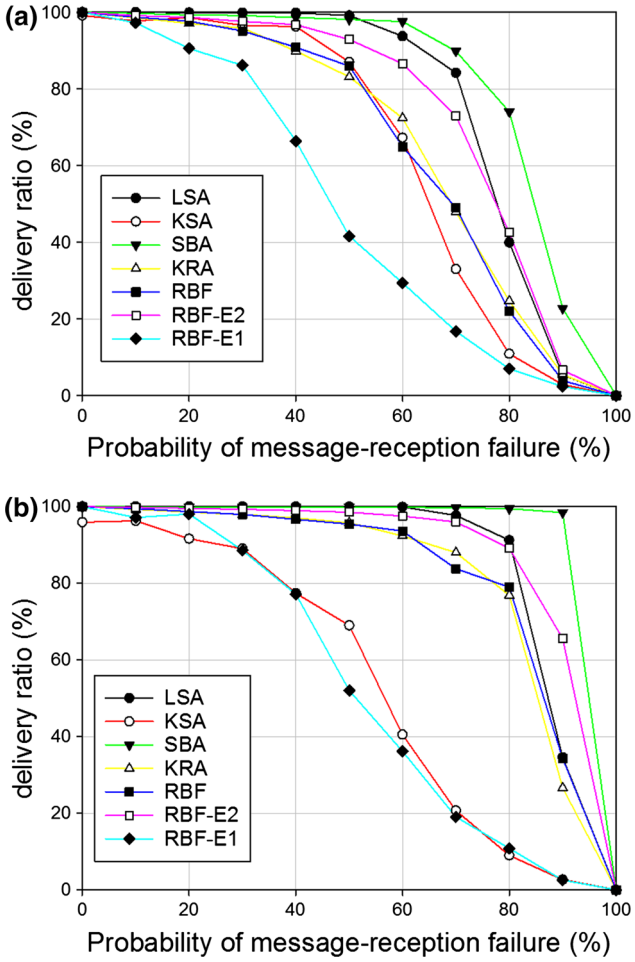


Fig. 16 Delivery ratio versus probability of message-reception failure. **a** The number of nodes is 100. **b** The number of nodes is 400

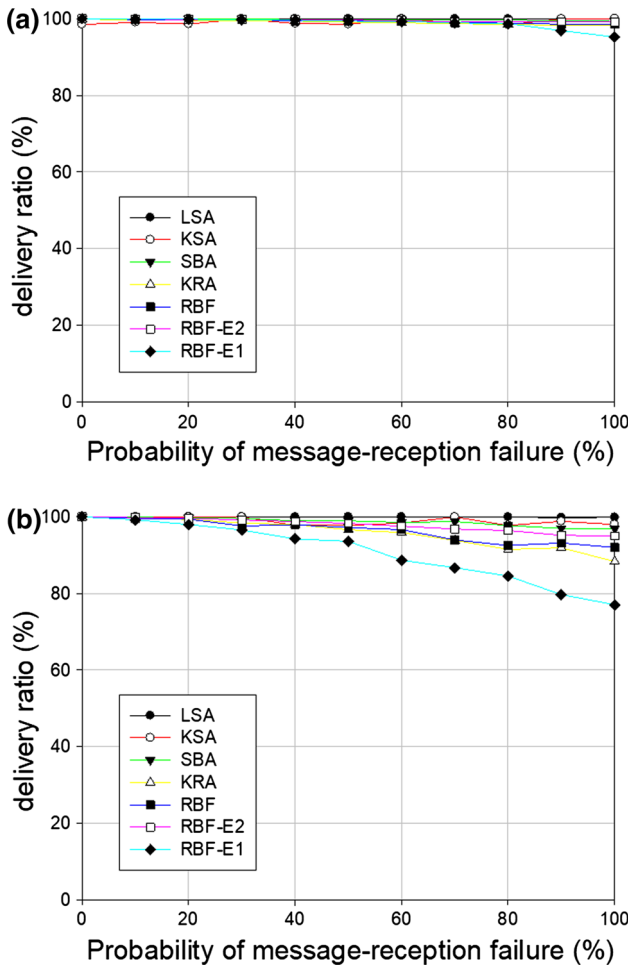


Fig. 17 Delivery ratio when partial links are unreliable. **a** 10% of the links are unreliable. **b** 30% of the links are unreliable

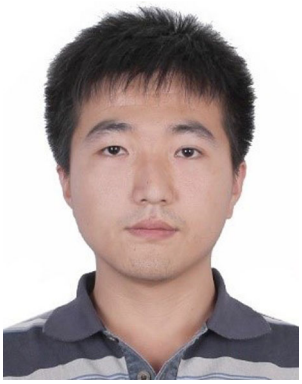
efficiency and low computation overhead at the same time. Two extensions of RBF, RBF-E1 and RBF-E2, are proposed. RBF-E1 further reduces the forwarding ratio, and RBF-E2 can be implemented in linear time without the expense of efficiency. We carried out extensive simulations to demonstrate the performance of the proposed algorithms, The results show that the proposed algorithms, especially RBF-E1, can outperform existing algorithms in many aspects, such as forwarding ratio, latency, and in 3D networks.

Acknowledgements The corresponding author of this paper is Xiaohui Wei. This work is supported by the National Natural Science Foundation of China (NSFC) (Grants Nos. 61772228, 61702298, 61602205, 51627805, 61170004), National key research and development program of China (Grant Nos. 2017YFC1502306, 2016YFB0201503, 2016YFB0701101), Specialized Research Fund for the Doctoral Program of Higher Education (20130061110052), Major Special Research Project of Science and Technology Department of Jilin Province (20160203008GX), Key Science and Technology Research Project of Science and Technology Department of Jilin Province (20140204013GX), Graduate Innovation Fund of Jilin University.

References

1. Ahn, J. H., & Lee, T. J. (2014). Multipoint relay selection for robust broadcast in ad hoc networks. *Ad Hoc Networks*, *17*, 82–97.
2. Baghaie, M., & Krishnamachari, B. (2011). Delay constrained minimum energy broadcast in cooperative wireless networks. In *IEEE international conference on computer communications (INFOCOM)* (pp. 864–872). IEEE.
3. Baysan, M., Sarac, K., Chandrasekaran, R., & Bereg, S. (2009). A polynomial time solution to minimum forwarding set problem in wireless networks under unit disk coverage model. *IEEE Transactions on Parallel and Distributed Systems*, *20*(7), 913–924.
4. Calinescu, G., Mandoiu, I. I., Wan, P. J., & Zelikovsky, A. Z. (2004). Selecting forwarding neighbors in wireless ad hoc networks. *Mobile Networks and Applications*, *9*(2), 101–111.
5. Cheng, D., Mao, Y., Wang, Y., & Wang, X. (2015). Improving energy adaptivity of constructive interference-based flooding for WSN-AF. *International Journal of Distributed Sensor Networks*. https://doi.org/10.1155/2015/538145.
6. Cheng, D., Wang, Y., Han, J., Mao, X., & Wang, X. (2016). E2f: Achieving energy-efficient flooding with constructive interference in WSNs. *Ad Hoc and Sensor Wireless Networks*, *30*, 241–259.
7. Cheng, L., Niu, J., Luo, C., Shu, L., Kong, L., Zhao, Z., et al. (2018). Towards minimum-delay and energy-efficient flooding in low-duty-cycle wireless sensor networks. *Computer Networks*, *134*, 66–77.
8. Dai, F., & Wu, J. (2004). Performance analysis of broadcast protocols in ad hoc networks based on self-pruning. *IEEE Transactions on Parallel and Distributed Systems*, *15*(11), 1027–1040.
9. Dai, F., & Wu, J. (2006). Efficient broadcasting in ad hoc wireless networks using directional antennas. *IEEE Transactions on Parallel and Distributed Systems*, *17*(4), 335–347.
10. Gupta, P., & Kumar, P. R. (2002). The capacity of wireless networks. *IEEE Transactions on Information Theory*, *46*(2), 388–404.
11. Ho, C., Obraczka, K., Tsudik, G., & Viswanath, K. (1999). Flooding for reliable multicast in multi-hop ad hoc networks. In *International workshop on discrete algorithms and methods for mobile computing and communications* (pp. 64–71). ACM.
12. Johnson, D. B., & Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. In T. Imielinski & H. F. Korth (Eds.), *Mobile computing* (pp. 153–181). Boston: Springer.
13. Khabbazian, M., & Bhargava, V. K. (2008). Localized broadcasting with guaranteed delivery and bounded transmission redundancy. *IEEE Transactions on Computers*, *57*(8), 1072–1086.
14. Khabbazian, M., & Bhargava, V. K. (2009). Efficient broadcasting in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, *8*(2), 231–245.
15. Khabbazian, M., Blake, I. F., & Bhargava, V. K. (2012). Local broadcast algorithms in wireless ad hoc networks: Reducing the number of transmissions. *IEEE Transactions on Mobile Computing*, *11*(3), 402–413.
16. Kim, D., Zhang, Z., Li, X., Wang, W., Wu, W., & Du, D. Z. (2010). A better approximation algorithm for computing connected dominating sets in unit ball graphs. *IEEE Transactions on Mobile Computing*, *9*(8), 1108–1118.
17. Ko, Y. B., & Vaidya, N. H. (2000). Location-aided routing (lar) in mobile ad hoc networks. *Wireless Networks*, *6*(4), 307–321.
18. Liu, H., Jia, X., Wan, P. J., Liu, X., & Yao, F. F. (2007). A distributed and efficient flooding scheme using 1-hop information

in mobile ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, *18*(5), 658–671.
19. Liu, W., Nakauchi, K., & Shoji, Y. (2017). A location-aided flooding mechanism in community-based IoT networks. In *IEEE annual computing and communication workshop and conference (CCWC)* (pp. 1–6). IEEE.
20. Lou, W., & Wu, J. (2004). Double-covered broadcast (DCB): A simple reliable broadcast algorithm in manets. In *IEEE international conference on computer communications (INFOCOM)* (Vol. 3, pp. 2084–2095). IEEE.
21. Papanikos, N., & Papapetrou, E. (2016). Revisiting XOR-based network coding for energy efficient broadcasting in mobile ad hoc networks. *Computer Communications*, *96*, 1–16.
22. Papanikos, N., & Papapetrou, E. (2017). Deterministic broadcasting and random linear network coding in mobile ad hoc networks. *IEEE/ACM Transactions on Networking*, *25*(3), 1540–1554.
23. Peng, W., & Lu, X. C. (2000). On the reduction of broadcast redundancy in mobile ad hoc networks. In *ACM international symposium on mobile ad hoc networking and computing (MobiHoc)* (pp. 129–130). IEEE Press.
24. Perkins, C., & Belding, E. (1999). Ad-hoc on-demand distance vector routing. In *IEEE workshop on mobile computing systems and applications (WMCSA)* (Vol. 25, pp. 90–100). IEEE.
25. Qayyum, A., Viennot, L., & Laouiti, A. (2002). Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *Annual Hawaii international conference on system sciences* (pp. 3866–3875). IEEE.
26. Qiu, C., Shen, H., & Yu, L. (2014). Energy-efficient cooperative broadcast in fading wireless networks. In *IEEE international conference on computer communications (INFOCOM)* (pp. 1114–1122). IEEE.
27. Reina, D., Toral, S., Johnson, P., & Barrero, F. (2015). A survey on probabilistic broadcast schemes for wireless ad hoc networks. *Ad Hoc Networks*, *25*, 263–292.
28. Ruiz, P., & Bouvry, P. (2015). Survey on broadcast algorithms for mobile ad hoc networks. *ACM Computing Surveys*, *48*(1), 8.
29. Tseng, Y. C., Ni, S. Y., & Shih, E. Y. (2003). Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *IEEE Transactions on Computers*, *52*(5), 545–557.
30. Wang, Q., Zhu, Y., & Cheng, L. (2006). Reprogramming wireless sensor networks: Challenges and approaches. *IEEE Network*, *20*(3), 48–55.
31. Wang, Y., He, Y., Mao, X.F., & Liu, Y. (2012). Exploiting constructive interference for scalable flooding in wireless networks. In *IEEE international conference on computer communications (INFOCOM)* (pp. 2104–2112).
32. Wu, J., & Dai, F. (2003). Broadcasting in ad hoc networks based on self-pruning. *International Journal of Foundations of Computer Science*, *14*(02), 201–221.
33. Wu, J., Lou, W., & Dai, F. (2006). Extended multipoint relays to determine connected dominating sets in manets. *IEEE Transactions on Computers*, *55*(3), 334–347.
34. Yu, S., Wu, X., Wu, P., Wu, D., Dai, H., & Chen, G. (2014). Cirf: Constructive interference-based reliable flooding in asynchronous duty-cycle wireless sensor networks. In *Wireless communications and networking conference* (pp. 2734–2738).



Xin Bai received the B.S. degree from the Department of Mathematics, Shanghai Jiao Tong University, China, in 2008. He received the M.S. degree from Department of Computer Science, Jilin University, China, in 2013, where he is currently pursuing the Ph.D. degree. His research interests include wireless networks and intelligent transportation systems.

Xiaohui Wei is a Professor and Dean of the College of Computer Science and Technology (CCST), Jilin University, China. He is currently the Director of High Performance Computing Center of Jilin University. His current major research interests includes resource scheduling for large distributed systems, infrastructure level virtualization, large scale data processing system and fault-tolerant computing. He has published more than 50 journal and conference papers in the above areas.

Sen Bai received the BS degree from the School of Software Engineering, Huazhong University of Science and Technology, China, in 2008. He received the M.S. and Ph.D. degrees from Department of Computer Science, Jilin University, China, in 2016. He is currently working at school of software, Tsinghua University. His research interests include wireless networks and intelligent transportation systems.