Highly Efficient Flooding in Mobile Ad Hoc Networks

Majid Khabbazian, *Student Member, IEEE*, Vijay K. Bhargava, *Fellow, IEEE*
Department of Electrical and Computer Engineering
University of British Columbia
2356 Main Mall, Vancouver, BC, Canada V6T 1Z4

***Abstract*— This paper presents two efficient flooding algorithms based on 1-hop information. In the first part of the paper, we consider sender-based flooding algorithms, specifically the algorithm proposed by Liu et al. In their paper, Liu et al. propose a sender-based flooding algorithm that can achieve local optimality by selecting the minimum number of forwarding nodes in the lowest computational time complexity $O(nlogn)$, where n is the number of neighbors. We show that this optimality only holds for a subclass of sender-based algorithms. We propose an efficient sender-based flooding algorithm based on 1-hop information that reduces the time complexity of computing forwarding nodes to $O(n)$. In Liu's algorithm, n nodes are selected to forward the message in the worst case, whereas in our proposed algorithm, the number of forwarding nodes in the worst case is 11. In the second part of the paper we propose a simple and highly efficient receiver-based flooding algorithm. When nodes are randomly distributed, we prove that the probability of two neighbor nodes broadcasting the same message exponentially decreases when the distance between them decreases or when the node density increases. Using simulation, we confirm these results and show that the number of broadcasts in our proposed receiver-based flooding algorithm can be even less than one of the best-known approximations for the minimum number of required broadcasts.**

I. Introduction

Flooding is a fundamental communication operation in which one node sends a message to all other nodes in the network. Flooding is widely used as a basic mechanism in many ad hoc network protocols. For example, ad hoc on demand routing protocols, such as AODV [1] and DSR [2], typically use flooding in their route discovery phase. Flooding is also used for topology updates, network maintenance or simply for sending a control or warning message. The simplest flooding algorithm is blind or pure flooding, in which every node broadcasts the message when it receives it for the first time. Using blind flooding, each node receives the message from all its neighbors in a collision-free network. Therefore, the broadcast redundancy significantly increases as the average number of neighbors increases. High broadcast redundancy can result in high power and bandwidth consumption in the network. Moreover, it increases packet collisions, which can lead to additional transmissions. This can cause severe network congestion or significant performance degradation, a phenomenon called the broadcast storm problem [3]. Consequently, it is crucial to design efficient flooding algorithms to reduce the number of required message broadcasts in the network.

A set of nodes is called a Dominating Set (DS) if any node in the network either belongs to the set or is a 1-hop neighbor of a node in the set. The set of broadcasting nodes forms a Connected Dominating Set (CDS). Therefore, the minimum number of required broadcasts is not less than the size of the minimum CDS. Unfortunately, finding the minimum CDS is NP-hard, even for the unit disk graphs [4], [5]. However, there are some distributed algorithms to find a minimum CDS with even constant approximations [6], [7]. These algorithms can be employed to find a small-sized CDS that can be used as a virtual backbone for flooding in ad hoc networks. However, this approach is not efficient in networks with frequent topology changes, as maintaining a CDS is often costly [8].

The main objective of efficient flooding algorithms is to reduce the number of broadcasts while keeping the bandwidth and computational overhead as low as possible. One approach to classify flooding algorithms is based on the neighbor information they use. Some flooding algorithms such as blind flooding and probabilistic flooding algorithms [9], [10] do not rely on neighborhood knowledge. These algorithms cannot typically guarantee 100% delivery and/or effectively reduce the number of broadcasts. Moreover, to decide whether or not to broadcast, they may use a threshold (such as probability of broadcast) which may not be easy to find for different network situations. In the second category, flooding algorithms require to have 2-hop or more neighbor information. The flooding algorithms in this category can reduce the number of broadcasts in the network and guarantee 100% delivery [11], [12], [13]. However, they may induce high overhead in highly dynamic networks as they need to maintain 2-hop network connectivity.

In this paper, we propose two flooding algorithms based on 1-hop neighbor information. The first proposed algorithm is a sender-based algorithm. In sender-based algorithms, the broadcasting nodes select a subset of their neighbors to forward the message. We compare our proposed flooding algorithm to one of the best sender-based flooding algorithms that use 1-hop information [8]. In [8], Liu et al. propose a flooding algorithm that reduces the number of broadcasts and achieves local optimality by selecting the minimum number of forwarding nodes with minimum time complexity $O(nlogn)$, where n is the number of neighbors. We show that this optimality only holds for a subclass of sender-based flooding

algorithms employing 1-hop information, and prove that our proposed sender-based algorithm can achieve 100% delivery with time complexity $O(n)$. Moreover, Liu's algorithm selects n forwarding nodes in the worst case, while our proposed algorithm selects 11 nodes in the worst case. Based on our simulation results, our sender-based algorithm results in fewer broadcasts that does Liu's algorithm.

We also propose a receiver-based flooding algorithm in this paper. In receiver-based algorithms, the receiver decides whether or not to broadcast the message. The proposed receiver-based algorithm is a novel flooding algorithm that can significantly reduce the number of broadcasts in the network. We show that using our proposed receiver-based algorithm, two close neighbors are not likely to broadcast the same message. In other words, we prove that the probability of broadcast for a node N_A exponentially decreases when the distance between N_A and its broadcasting neighbor decreases, or when the density of nodes increases. Based on our experimental results, the number of broadcasts using our proposed receiver-based algorithm is less than one of the best-known approximations for the minimum number of required broadcasts.

The rest of this paper is organized as follows. In Section II, we describe the system model and network assumptions. In Section III, we discuss our proposed sender-based flooding algorithm and its characteristics. We propose a simple and highly efficient receiver-based flooding algorithm in Section IV and prove a unique property of the algorithm. We also relax some system model assumption in this section. In Section V, we verify the theoretical results using simulation and compare the number of forwarding nodes of our proposed flooding algorithms with that of one of the best existing flooding algorithms and an approximated lower bound of optimal solution. Finally, we provide conclusions in Section VI.

II. System Model

Our system model is very similar to that used by Liu et al. [8]. We assume that all nodes are located in a 2-D plane and have a transmission range of R. Therefore, the topology of the network can be represented by a unit disk graph. We assume that the network is connected. Two nodes are considered neighbors if they are in transmission range of each other. We suppose that each node knows its location via a localization technique such as Global Positioning System (GPS) or the lightweight techniques summarized in [14]. Each node periodically broadcasts a very short *Hello* message, which includes its *id* and position. Thus, each node gets the position of its neighbors as well. In the Medium Access Control (MAC) layer, we assume that scheduling is done according to the p-persistent CSMA/CA protocol, which IEEE 802.11 is based on in the broadcast mode. In the p-persistent CSMA/CA protocol, when a node has a message to transmit, it initiates a defer timer by a random number and starts listening to the channel. If the channel is busy, it continues to listen until the channel becomes idle. When the channel is idle, it starts incrementing the defer timer at the end of each time unit.

III. An Efficient Sender-Based Flooding Algorithm

A. Algorithm Structure

Our first proposed flooding algorithm is a sender-based algorithm, i.e., each sender selects a subset of nodes to forward the message. Each message can be identified by its source *id* and a sequence number incremented for each message at the source node. Algorithm 1 is a general sender-based flooding algorithm and indicates the structure of our proposed sender-based flooding algorithm. As shown in Algorithm 1, each node schedules a broadcast for a received message if the node is selected by the sender and if it has not broadcast the same message before. In other words, the message will be dropped by a node if the node has broadcast the message or if the node is not in the forwarding list of the sender node. Clearly, each message is broadcast once at most by a node, which is similar to Liu's algorithm. However, in Liu's algorithm, each node may only schedule a broadcast when it receives a message for the first time. In contrast, in Algorithm 1, a broadcast schedule can be set at any time. For example, a message can be dropped after the first reception but scheduled for broadcast the second time. Clearly, the main design issue in Algorithm 1 is how to select the forwarding nodes.

Algorithm 1: A general sender-based algorithm

```
1.  Extract the required information from the received message M;
2.  if M has been broadcast before or does not contain node's id
3.      drop the message;
4.  else
5.      set a defer time;
6.  When defer time expires
7.    Select a subset of neighbors to forward the message;
8.    Attach the list of forwarding node to the message;
9.    Broadcast the message;
```

B. Forwarding-Node Selection Algorithm

Let us consider point A as the node N_A and a circle $\mathcal{C}_{A,R}$ centered at A with a radius R as the transmission range of N_A. Before delving into the algorithm description and proofs, we need to define the following terms:

Definition 1 (Bulged Slice). *As illustrated in Figure 1, we define a bulged slice around A as the intersection area of three circles with radius R and centers A, M and N, where $\overline{AM} = R$, $\overline{AN} = R$, $\overline{MN} = R$ and $\overline{PQ}$ is the distance between two points P and Q. Note that in any bulged slice AMN we have $\angle MAN = \frac{\pi}{3}$.*

Definition 2 (Left/Right Bulged Slice). *As shown in Figure 2, let A and B be two points such that $0 < \overline{AB} \leqslant R$. A bulged slice AMN around A is called the left bulged slice of B around A if B is on the arc $\overset{\frown}{AM}$ and M is an $\frac{\pi}{3}$ clockwise rotation of N around A. Similarly, a bulged slice AMN around A is called the right bulged slice of B around A if B is on the arc $\overset{\frown}{AM}$ and M is a $\frac{\pi}{3}$ anti-clockwise rotation of N around A.*

Definition 3 (Bulged Angle). *Let B and C be two points inside a circle with radius R and center A. The bulged angle $\angle_A(B, C)$ is defined to be equal to $0 \leqslant \alpha < 2\pi$ if the left bulged slice of B is a α anti-clockwise rotation of the left bulged slice of C around A.*

Definition 4 (B-coverage Set). *A subset of neighbors of N_A is called a B-coverage set of N_A if any non-empty bulged slice around A contains at least one node from the set. A bulged slice is empty if there is no node inside it.*

Definition 5 (Slice-Based Selection Algorithm). *A forwarding-node selection algorithm is called a slice-based selection algorithm (or slice-based algorithm) if for any node N_A, it selects a B-coverage set of it.*

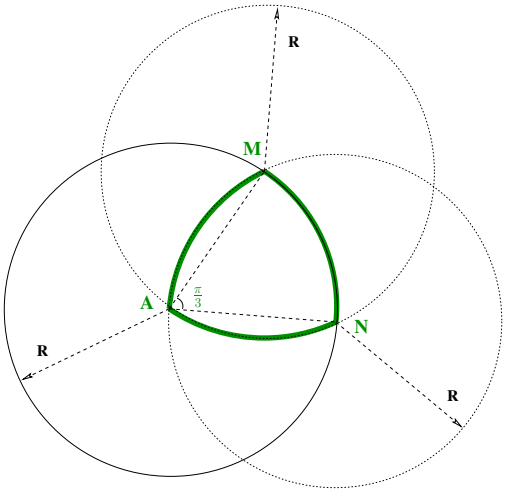

Fig. 1. A bulged slice around A.

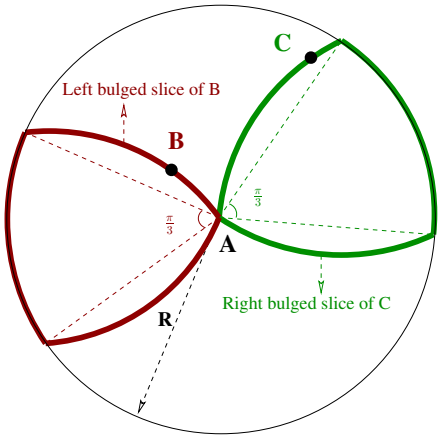

Fig. 2. Left bulged slice of B and right bulged slice of C around A.

A node can have several different B-coverage sets. Therefore, there are more than one slice-based selection algorithms. For example, a trivial slice-based selection algorithm would be one that selects all of the neighbors as the B-coverage set. Clearly, this algorithm will result in blind flooding if it is used as the forwarding-node selection scheme in Algorithm 1. In this section, we first show that Algorithm 1 can achieve 100% delivery if it uses any slice-based algorithm to select the forwarding nodes. We then present an efficient slice-based algorithm that selects 11 nodes in the worst case and has computational complexity $O(n)$, where n is the number of neighbors.

Lemma 1. *For any two points P_1 and P_2 inside a bulged slice, we have*

$$\overline{P_1P_2} \leqslant R.$$

Proof: As shown in Figure 3, the line passing through P_1 and P_2 intersects the bulged slice AMN at P_1' and P_2'. Clearly, $\overline{P_1P_2} \leqslant \overline{P_1'P_2'}$. Therefore, to prove the lemma it is sufficient to show that $\overline{P_1'P_2'} \leqslant R$. Without loss of generality, we can assume that P_1' and P_2' are on the arcs $\widehat{AM}$ and $\widehat{AN}$, respectively. Let us consider the perpendicular bisector of the line segment $P_1'M$ (Line L). Line L passes through N because $\overline{NM} = \overline{NP_1'} = R$. Since the point P_2' is on the arc $\widehat{AN}$, the line segment MP_2' will cross the line L at a point Q. Using triangle inequality, we have

$$\overline{P_1'P_2'} \leqslant (\overline{QP_2'} + \overline{QP_1'}) = (\overline{QP_2'} + \overline{QM}) = P_2'M = R.$$

Note that Q is on the line L, hence $\overline{QP_1'} = \overline{QM}$. ■

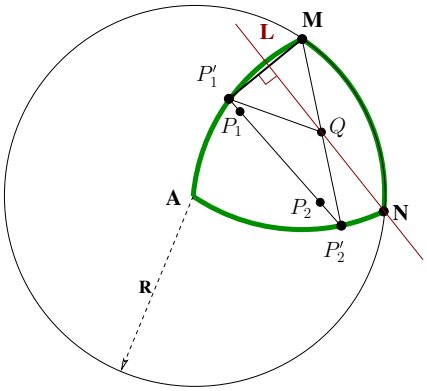

Fig. 3. Upper bound on the distance between two nodes inside a bulged slice.

Consider two points A and B such that $R < \overline{AB} \leqslant 2R$. As shown in Figure 4, the line segment AB intersects the circle $\mathcal{C}_{A,R}$ at point Q. Let AQM and AQN be the left and right bulged slices of Q around A, respectively. The following lemmas hold:

Lemma 2. *A point P is inside the bulged slice AQM or AQN if $\overline{AP} \leqslant R$ and $\overline{BP} \leqslant R$.*

Proof: It is easy to show that for any triangle $\triangle ABC$, $\overline{AM} \leqslant \overline{AB}$ or $\overline{AM} \leqslant \overline{AC}$, where M is a point on the line segment BC. Consequently, in the triangle $\triangle PAB$ (shown in Figure 4) we have

$$\overline{PQ} \leqslant \overline{AP} \leqslant R \quad \text{or} \quad \overline{PQ} \leqslant \overline{BP} \leqslant R.$$

Therefore, $\overline{PQ} \leqslant R$. Thus based on the bulged slice definition, the point P is inside the bulged slice AQM or AQN. ■

Lemma 3. *For any point $P \neq A$ inside the bulged slice AQM or AQN, we have*

$$\overline{BP} < \overline{BA}.$$

Proof: Using triangle inequality we get

$$\overline{BP} \leqslant \overline{BQ} + \overline{QP} \leqslant \overline{BQ} + R = \overline{BA}.$$

Note that this equality holds only when $\overline{BP} = \overline{BQ} + \overline{QP}$ and $\overline{QP} = R$, or simply when $P = A$.

■

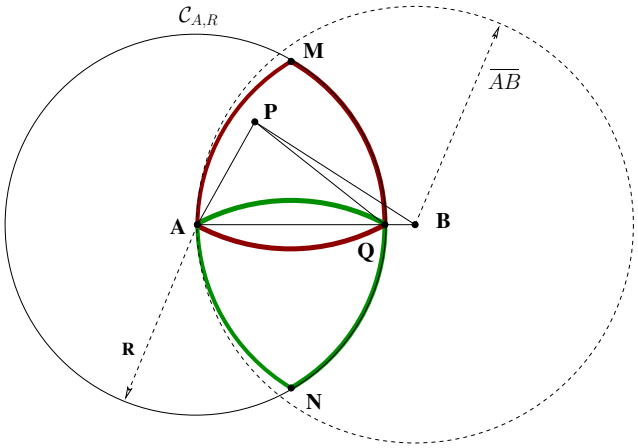

Fig. 4. Other properties of bulge slice from Lemmas 2 and 3.

Theorem 1. *In a collision-free network, Algorithm 1 can achieve* 100% *delivery if it uses a slice-based selection algorithm to select the forwarding nodes.*

Proof: Using Algorithm 1, each node broadcasts the message at most once. Therefore, flooding will eventually terminate. By contradiction, suppose there is at least one node that has not received the message after the flooding termination. Let us consider the following set:

$$\Lambda = \{(N_X, N_Y, N_Z) | N_X \text{ has broadcast the message, } N_Z \text{ has not received the message and } N_Y \text{ is the neighbor of both } N_X \text{ and } N_Z\}.$$

Suppose N_S is the node that initiated flooding and N_T is a node that has not received the message. The network is connected, thus there is a path between N_S and N_T. Clearly, we can find two neighbor nodes N_C, N_B along the path from N_T to N_S such that N_C has not received the message, while N_B has received it. Suppose that N_B has received the message from N_A. Consequently, $(N_A, N_B, N_C) \in \Lambda$, thus $\Lambda \neq \emptyset$. As a result,

$$\exists (N_{A'}, N_{B'}, N_{C'}) \in \Lambda \text{ s.t. } \forall (N_X, N_Y, N_Z) \in \Lambda : \overline{A'C'} \leqslant \overline{XZ}. \quad (1)$$

Obviously, $N_{A'}$ and $N_{C'}$ are not neighbors, because $N_{C'}$ has not received the message. Thus, $\overline{A'C'} > R$. Using Lemma 2, B' is inside the bulged slice $A'P_1P_2$ or $A'P_1P_3$, where P_1 is the intersection of line segment $A'C'$ and the circle $\mathcal{C}_{A',R}$ and $A'P_1P_2$ and $A'P_1P_3$ are the left and the right bulged slices of P_1 around A', respectively. Without loss of generality, assume that B' is inside the bulged slice $A'P_1P_2$. Since $N_{A'}$ has at least one neighbor (i.e., $N_{B'}$) in this slice, there must be a selected node N_D in the slice that has forwarded the message. Using Lemma 1, we get $\overline{B'D} \leqslant R$; hence, nodes N_D and $N_{B'}$ are neighbors. Therefore, $(N_D, N_{B'}, N_{C'}) \in \Lambda$. However, this contradicts to (1) because using Lemma 3 we have

$$\overline{DC'} < \overline{A'C'}.$$

■

Algorithm 2 shows our proposed slice-based selection algorithm. Suppose that node N_A uses the proposed algorithm to select the forwarding nodes from its neighbors. Let us assume that N_A stores all of its neighbors' *id*s and locations in an array of length n, where n is the number of neighbors. The algorithm selects the first node N_{S_1} randomly from the array. The first node can also be selected deterministically by, for example, selecting the node that is the farthest away from N_A. Suppose that N_{S_i} is the last node selected by the algorithm and $AM_{S_i}N_{S_i}$ is the left bulged slice of S_i around A. To select the next node, the algorithm iterates through the array and selects the node $N_{S_{i+1}}$ such that $N_{S_{i+1}}$ is inside the slice $AM_{S_i}N_{S_i}$, $\angle_A(S_{i+1}, S_i) \neq 0$ and

$$\forall N_B \text{ inside } AM_{S_i}N_{S_i} : \quad \angle_A(B, S_i) \leqslant \angle_A(S_{i+1}, S_i). \quad (2)$$

If there is no such node, the algorithm will select $N_{S_{i+1}}$ such that $\angle_A(S_{i+1}, S_i) \neq 0$ and

$$\forall N_B \text{ inside } \mathcal{C}_{A,R} : \quad \angle_A(B, S_i) \neq 0 \Longrightarrow \angle_A(S_{i+1}, S_i) \leqslant \angle_A(B, S_i). \quad (3)$$

The algorithm terminates by selecting the last node N_{S_m} if

$$\angle_A(S_1, S_m) \leqslant \angle_A(S_{m+1}, S_m).$$

Algorithm 2: A slice-based selection algorithm

Input: $List_A[1..n]$: List of all neighbors of N_A
output: A B-coverage set of N_A: $\{N_{S_i}\}$

```
1.  ind ← 1;   i ← 0;
2.  do
3.    ang_max ← 0;   ang_min ← 2π;
4.    i ← i + 1;
5.    N_{S_i} ← List_A[ind];
6.    chk ← false;
7.    for (j = 1; j ⩽ n; j++)
8.      if (List_A[j] is in the left bulged slice of N_{S_i})
9.        if (∠_A(List_A[j], N_{S_i}) > ang_max)
10.         chk ← true;
11.         ind_max ← j;
12.         ang_max ← ∠_A(List_A[j], N_{S_i});
13.     else
14.       if (∠_A(List_A[j], N_{S_i}) < ang_min)
15.         ind_min ← j;
16.         ang_min ← ∠_A(List_A[j], N_{S_i});
17.   if (chk)  ind ← ind_max;
18.   else  ind ← ind_min;
19. while (∠_A(List_A[ind], S_i) < ∠_A(S_1, S_i))
```

Lemma 4. *Suppose the proposed algorithm chooses* m *nodes* $\{N_{S_1}, N_{S_2}, ..., N_{S_m}\}$. *For any* $1 \leqslant i \leqslant m-2$, *we have*

$$\angle_A(S_{i+2}, S_i) > \frac{\pi}{3}.$$

Proof: Based on (2), (3) and the algorithm termination condition, we can show that

$$\angle_A(S_{i+2}, S_i) > \angle_A(S_{i+1}, S_i),$$

for any $1 \leqslant i \leqslant m-2$. By contradiction, assume that $\angle_A(S_{i+2}, S_i) \leqslant \frac{\pi}{3}$. Therefore, S_{i+2} is inside the left bulged slice of S_i. Thus, using (2) we have $\angle_A(S_{i+1}, S_i) \geqslant \angle_A(S_{i+2}, S_i)$, which is a contradiction. ■

Theorem 2. *The proposed slice-based selection algorithm will select at most 11 nodes.*

Proof: It is easy to see that

$$\angle_A(S_{i+2}, S_i) = \angle_A(S_{i+2}, S_{i+1}) + \angle_A(S_{i+1}, S_i). \quad (4)$$

Therefore,

$$\sum_{i=1}^{m} \angle_A(S_j, S_i) = 2\pi,$$

where $j = (i \mod m) + 1$. By contradiction, suppose $m > 11$. Using (4) and Lemma 4 we get

$$\begin{aligned}\angle_A(S_{11}, S_1) &= \sum_{i=1}^{10} \angle_A(S_{i+1}, S_i) \\ &= \sum_{i=1}^{5} (\angle_A(S_{2i+1}, S_{2i-1})) > 5 \times \frac{\pi}{3}.\end{aligned}$$

Therefore, $\angle_A(S_1, S_{11}) < (2\pi - 5 \times \frac{\pi}{3}) = \frac{\pi}{3}$. Consequently, S_1 is inside the left bulged slice of S_{11}, thus the proposed slice-based algorithm will terminate after selecting S_{11}. ■

The above theorem gives an upper bound on the number of nodes selected by the proposed selection algorithm. In Section V, using simulation we show that the average number of selected nodes (when the nodes are distributed uniformly) is less than 6.

Theorem 3. *Time complexity of the proposed slice-based selection algorithm is $O(n)$, where n is the number of neighbors.*

Proof: The algorithm selects the first node in $O(1)$. To select each of the other nodes, the algorithm performs $O(n)$ operations by checking all the neighbors in the array. Therefore, the complexity of the algorithm is $O(m \times n)$, where m is the number of selected nodes. Using Proposition (2) we have $m \leqslant 11$, thus the time complexity of algorithm is $O(n)$. ■

C. Reducing the Number of Forwarding Nodes

In the sender-based flooding algorithms, each broadcasting node attaches a list of its selected forwarding nodes to the message before broadcasting it. This procedure will increase the bandwidth and power required to broadcast the message. As shown earlier, our proposed slice-based selection algorithm reduces the number of selected forwarding nodes to 11 in the worst case. In this section, we show how to further reduce the number of selected nodes.

Recall that the proposed slice-base algorithm selects a subset of N_A's neighbors such that there is at least one selected node in any non-empty bulged slice around A. Suppose N_A extracts the list of the forwarding nodes from each message it receives. Let $\mathcal{L}_A$ be a subset of N_A's neighbors that has broadcast the message or been selected by other nodes to forward it. Since all of the selected forwarding nodes are required to broadcast the message, it is sufficient for N_A to find a subset of its neighbors $\mathcal{S}_A$ such that any non-empty bulged slice around A contains at least one node from $\mathcal{S}_A \cup \mathcal{L}_A$. Algorithm 3 shows how to find $\mathcal{S}_A$ when $\mathcal{L}_A$ is given. Note that Algorithm 2 is special case of Algorithm 3 in which $\mathcal{L}_A = \emptyset$. Using an approach similar to that used to prove the complexity of Algorithm 2, one can show that the time complexity of Algorithm 3 is $O(n)$.

Algorithm 3: A generalization of the proposed slice-based algorithm
Input: $List_A[1..n]$: A list of all neighbors of N_A s.t. $List_A[1] \in \mathcal{L}_A$
output: A B-coverage set of N_A: $\{N_{S_i}\}$

```
1.  ind ← 1;   i ← 0;   j ← 0;
2.  do
3.      ang_max1 ← 0;   ang_max2 ← 0;   ang_min ← 2π;
4.      j ← j + 1;   N_{T_j} ← List_A[ind];
5.      if (N_{T_j} ∉ L_A)
6.          i ← i + 1;   N_{S_i} ← N_{T_j};
7.      chk1 ← false;   chk2 ← false;
8.      for (k = 1; k ⩽ n; k++) ;
9.          if ((List_A[k] is in the left bulged slice of N_{T_j})
                and (∠_A(List_A[k], N_{T_j}) > 0))
10.             chk2 ← true;
11.             if (List_A[k] ∈ L_A)
12.                 chk1 ← true;
13.                 if (∠_A(List_A[k], N_{T_j}) > ang_max1) ;
14.                     ind_max1 ← k;
15.                     ang_max1 ← ∠_A(List_A[k], N_{S_i});
16.             else
17.                 if (∠_A(List_A[k], N_{T_j}) > ang_max2) ;
18.                     ind_max2 ← k;
19.                     ang_max2 ← ∠_A(List_A[k], N_{S_i});
20.         else
21.             if (∠_A(List_A[k], N_{T_j}) < ang_min) ;
22.                 ind_min ← k;
23.                 ang_min ← ∠_A(List_A[k], N_{T_j});
24.     if (chk1)   ind ← ind_max1;
25.     else if (chk2)   ind ← ind_max2;
26.     else   ind ← ind_min;
27. while (∠_A(List_A[ind], T_j) < ∠_A(S_1, T_j))
```

D. Maximizing the Minimum Node Weight of B-coverage Set

Suppose node N_A assigns a weight to each of its neighbors. The weight can represent the neighbor's battery life time, its distance to N_A, the average delay of the node, the level of trust or a combination of them. In some scenarios, we may desire to find a B-coverage set such that its minimum node weight is maximum or its maximum node weight is minimum among that of all B-coverage sets. For example, assume that the weight of each node represents its battery life time in a wireless sensor network. It may be desirable to select the nodes with a higher battery life time to forward the message in order to keep the nodes with a lower battery life time alive. Algorithm 4 shows how to find a B-coverage set such that its minimum node weight is maximum among that of all B-coverage sets. A similar approach can be used to find a B-coverage set such that its maximum node weight is minimum.

Algorithm 4 first sorts the nodes by their weights. Then, in each step it passes m nodes with the highest weights to Algorithm 2 as input and gets a set of (at most 11) nodes as output, where $1 \leqslant m \leqslant n$ is an integer initially set to $\lceil \frac{n}{2} \rceil$. If the output set is a B-coverage set, Algorithm 4 sets H to m and decreases m to $\lceil \frac{T+m}{2} \rceil$, where T and H are variables initially set to 1 and n, respectively. Otherwise, it sets T to m and increases m to $\lceil \frac{H+m}{2} \rceil$. After a finite number of steps we get $H = T + 1$. Algorithm 4 then returns the output of Algorithm_2($SList[1..H]$).

Algorithm 4: Maximizing the minimum node weight
Input: $List_A[1..n]$: List of all neighbors of N_A
output: A B-coverage set of N_A with highest minimum node weight.

// Sort the neighbor nodes by their weights.
1. $SList_A \leftarrow sort(List_A)$;
   $//SList[i] \geqslant SList[j] \Leftrightarrow i \leqslant j$.
2. $H \leftarrow n$; $T \leftarrow 1$; $m \leftarrow \lceil \frac{n}{2} \rceil$;
3. while $(H > T + 1)$
   // Pass m nodes with the highest weighs
   // to Algorithm 2 as the input.
4. $St \leftarrow Algorithm_2(SList[1..m])$;
5. if (St is a B-coverage set for N_A)
6. $H \leftarrow m$; $m \leftarrow \lceil \frac{T+m}{2} \rceil$;
7. else
8. $T \leftarrow m$; $m \leftarrow \lceil \frac{H+m}{2} \rceil$;
9. return (Algorithm_2($SList[1..H]$));

Corollary 1. *Algorithm 4 will select at most 11 nodes.*

Proof: The proof is clear, as Algorithm 4 returns an output of Algorithm 2 (Line 9). ■

Theorem 4. *Time complexity of Algorithm 4 is $O(nlogn)$.*

Proof: Algorithm 4 requires $O(nlogn)$ operations to sort the list of neighbors $List_A[1..n]$. The computational complexity of Algorithm 2 is $O(n)$. Therefore, Algorithm 4 performs $O(n)$ operations in each iteration of the while loop. The while loop terminates after $O(logn)$ iterations because it uses a binary search approach to find the minimum value of H. Consequently, the order of Algorithm 4 is $O(nlogn+logn\times n)$. ■

Theorem 5. *Minimum weight of nodes of the B-coverage set selected by Algorithm 4 is maximum among that of all B-coverage sets.*

Proof: Suppose that St_{min} is a B-coverage set such that the minimum weight of nodes in St_{min} is greater than or equal to that of other B-coverage sets. Let $N_X \in St_{min}$ be the node with the minimum weight in St_{min}. Assume N_A has K neighbors with weights greater than or equal to the weight of N_X. Therefore, the output of Algorithm_2($SList[1..K]$) is a B-coverage set. Note that Algorithm 4 finds the minimum H such that the output of Algorithm_2($SList[1..H]$) is a B-coverage set for N_A. Therefore, $H \leqslant K$ and thus the minimum weight of nodes of the B-coverage set selected by Algorithm 4 is greater than or equal to the weight of N_X. ■

IV. A Highly Efficient Receiver-Based Flooding Algorithm

In this section, we propose a novel receiver-based flooding algorithm that can significantly reduce redundant broadcasts in the network. As mentioned earlier, in receiver-based flooding algorithms the receiver of the message decides whether or not to broadcast the message. Therefore, a potential advantage of receiver-based flooding algorithms over sender-based ones is that they do not increase the size of the message by adding a list of forwarding nodes.

A. Algorithm Structure

Algorithm 5 shows a general approach used in several receiver-based flooding algorithms [13], [15]. Our proposed receiver-based flooding algorithm employs this approach. Clearly, the main design issue of Algorithm 5 is to determine whether or not to broadcast a received message. A trivial algorithm is to refrain broadcasting if and only if all the neighbors have received the message during the defer period. Although this algorithm is simple to implement, it has limited effect in reducing the number of redundant broadcasts. Suppose N_A's defer time expires at t_0. Using the above strategy, node N_A will broadcast if some of its neighbors (at least one) have not received the message by t_0. However, this broadcast is redundant if all such neighbors receive the message from other nodes after time t_0. This scenario typically occurs when t_0 is small compared to the maximum defer time. In the next section, we introduce a responsibility-based scheme that further reduces the redundant broadcasts without any changes in the MAC layer defer time design.

Algorithm 5: A general receiver-based algorithm
1. Extract the required information from the received message M.
2. if M has been received before
3. drop the message;
4. else
5. set a defer time;
6. When defer time expires, decide whether or not to broadcast;

B. A Responsibility-Based Scheme

Algorithm 6 shows the proposed Responsibility-Based Scheme (RBS). The main idea of Algorithm 6 is that a node avoids broadcasting if it is not responsible for any of its neighbors. A node N_A is not responsible for a neighbor N_B if N_B has received the message or if there is another neighbor N_C such that N_C has received the message and N_B is closer to N_C than it is to N_A. Suppose N_A stores *id*s of all its neighbors that have broadcast the message during the defer period. When executed by a node N_A, Algorithm 6 first uses this information to determine which neighbors have received the message (Lines 1–5 of Algorithm 6). It then returns *false* if and only if it finds a neighbor N_B that has not received the message and

$$\overline{AC} \leqslant \overline{BC}$$

for any N_A's neighbor N_C that has received the message. The output of RBS determines whether or not the broadcast is redundant.

Example 1. *As shown in Figure 5, N_A has five neighbors. Suppose that N_A has received a message from N_F. Note that N_A has the position of all its neighbors. Therefore, it can easily find that N_E and N_D have received the message but N_B and N_C have not. As shown in Figure 5, N_A is not required to broadcast because*

$$\overline{BE} < \overline{BA} \quad and \quad \overline{CD} < \overline{CA}.$$

Algorithm 6: A responsibility-based scheme (RBS)

Input: $List_A$: List of all neighbors of N_A, and
$List_B$: List of broadcasting neighbors.
output: True or False.

```
1.   List_C ← List_A.
2.   for (i = 1; i ⩽ length(List_C); i++ )
3.     for (j = 1; j ⩽ length(List_B); j++ )
4.       if (dist(List_C[i], List_B[j]) ⩽ R)
5.           removeElement(List_C[i], List_C); break;

6.   List_D ← List_A − List_C.
7.   for (i = 1; i ⩽ length(List_D); i++ )
8.     check ← true;
9.     for (j = 1; j ⩽ length(List_C); j++ )
10.      if (dist(List_D[i], List_C[j]) ⩽ dist(List_D[i], N_A))
11.          check ← false; break;
12.    if (check) return (false);
13.  return (true);
```

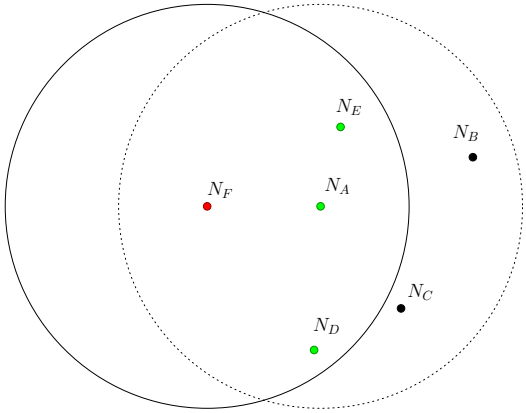

Fig. 5. An example of an RBS decision.

Theorem 6. *In a collision-free network, Algorithm 5 can achieve* 100% *delivery if it uses the proposed responsibility-based scheme to determine whether or not to broadcast.*

Proof: Using Algorithm 6, each node broadcasts a message at most once. Therefore, flooding will eventually terminate. By contradiction, suppose there is at least one node that has not received the message after the flooding termination. Let us consider the set

$$\Lambda = \{(N_X, N_Y) | N_X \text{ and } N_Y \text{ are neighbors,} \\ N_X \text{ has received the message and} \\ N_Y \text{ has not received the message}\}$$

Suppose N_S is the node that initiated flooding and N_T is a node that has not received the message. The network is connected, thus there is a path between N_S and N_T. Clearly, we can find two neighbor nodes N_B and N_A along the path from N_T to N_S such that N_B has not received the message, while N_A has. Consequently, $(N_A, N_B) \in \Lambda$, thus $\Lambda \neq \emptyset$. As a result,

$$\exists (N_{A'}, N_{B'}) \in \Lambda \quad \text{s.t.} \quad \forall (N_X, N_Y) \in \Lambda : \quad \overline{A'B'} \leqslant \overline{XZ}. \tag{5}$$

Clearly, $N_{A'}$ has not broadcast since $N_{B'}$ has not received the message. Therefore, there must be a node $N_{C'}$ such that $N_{C'}$ has received the message and $\overline{C'B'} < \overline{A'B'} \leqslant R$. This result contradicts to (5), since $(N_{C'}, N_{B'}) \in \Lambda$. ■

Theorem 7. *Time complexity of the proposed RBS is $O(n^2)$, where n is the number of neighbors.*

Proof: Algorithm 6 consists of two parts. In the first part (Lines 1–5), the algorithm generates a list of neighbors that have received the message ($List_C$). Clearly, the time complexity of this part is $O(kn)$, where $1 < k \leqslant n$ is the number of broadcasting neighbors. In the second part, the algorithm checks whether there is a node N_B such that N_B has not received the message and $\overline{BA} \leqslant \overline{BC}$ for any neighbor $N_C \in List_C$. The time complexity of this part is $O(lm)$, where $0 \leqslant l \leqslant n$ is the number of neighbors that have not received the message and $1 \leqslant m \leqslant n$ is the number of neighbors that have received it. Therefore, the complexity of the algorithm is $O(lm + kn)$. ■

C. A Unique Property of the Proposed RBS

In the simulation section (Section V), we show that the proposed RBS can significantly reduce the number of broadcasts in the network. In particular, our simulation shows that using RBS, the average number of broadcasts is less than one of the best-known approximations for the minimum number of required broadcasts. To justify this, we prove a unique property of the proposed RBS.

Assume that nodes are placed randomly inside a square area of size $L \times L$ using a homogeneous planar Poisson distribution. Therefore, nodes are independently and uniformly distributed in the area. Moreover, we have

$$Prob(\text{number of nodes in area } \tau = k) = \frac{(\delta\tau)^k e^{-\delta\tau}}{k!},$$

where δ is the density of nodes [16], [17]. Suppose node N_B receives the message from N_A for the first time. For simplicity, assume that circle $\mathcal{C}_{A,2R}$ is completely inside the square area. Corollary 2 shows that the probability that N_B broadcasts the message exponentially decreases when the distance $\overline{AB}$ decreases or when the node density δ increases. This result is further confirmed by simulation in Section V.

Example 2. *Suppose $R = 250$ m, $L = 1000$ m and $\delta L^2 = 300$ (i.e. there are about 300 nodes in the network). Let $Prb(Brd_B)$ be the probability that N_B broadcasts the message after receiving it from N_A. Using Theorem 8 we get $Prb(Brd_B) \leqslant 1.26 \times 10^{-2}$, $Prb(Brd_B) \leqslant 1.4 \times 10^{-3}$ and $Prb(Brd_B) \leqslant 10^{-4}$ when $\overline{AB} = 100$ m, $\overline{AB} = 80$ m and $\overline{AB} = 60$ m, respectively.*

Let $\mathcal{R}_1, \mathcal{R}_2, \ldots \mathcal{R}_k$ be k non-overlapping regions inside the network. Suppose that $\zeta_{\mathcal{R}}$ is the event,

$$\zeta_{\mathcal{R}} = \{\text{at least one node inside } \mathcal{R}\}.$$

Since the nodes are placed by homogeneous planar Poisson distribution, the events $\zeta_{\mathcal{R}_i}$ are independent [17]. Consequently, we have

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_K}) = Prb(\zeta_{\mathcal{R}_1}) Prb(\zeta_{\mathcal{R}_2}) \ldots Prb(\zeta_{\mathcal{R}_K}). \tag{6}$$

Lemma 5 generalizes (6) to the case where the regions may overlap each other.

Lemma 5. *Let $\mathcal{R}_1, \mathcal{R}_2, \dots \mathcal{R}_k$ be k regions inside the network. We have*

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \dots, \zeta_{\mathcal{R}_K}) \geqslant Prb(\zeta_{\mathcal{R}_1}) Prb(\zeta_{\mathcal{R}_2}) \dots Prb(\zeta_{\mathcal{R}_K}).$$

Proof: See the Appendix. ■

Lemma 6. *Let $\mathcal{D}_{A,R}$ and $\mathcal{D}_{B,R}$ be two disks with radius R and centers A and B, respectively. Suppose $\overline{AB} \leqslant R$. As shown in Figure 6, consider a point Q such that $R < \overline{QA}$ and $\overline{QB} \leqslant R$. Let $\mathcal{D}_{Q,\overline{QB}}$ be a disk with radius $\overline{QB}$. We have*

$$\Delta(\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})) \geqslant \frac{\pi(R - \overline{AB})^2}{3},$$

where $\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})$ is the intersection of disks $\mathcal{D}_{A,R}$, $\mathcal{D}_{B,R}$ and $\mathcal{D}_{Q,\overline{QB}}$ and $\Delta(R)$ is the area of region R.

Proof: See the Appendix. ■

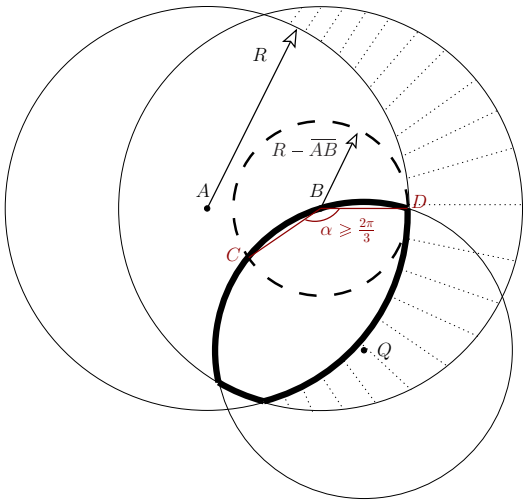

Fig. 6. Finding a lower bound for $\Delta(I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}}))$.

Theorem 8. *Suppose $d \leqslant R$ is the distance between two nodes N_A and N_B. We have*

$$Prb(Brd) \leqslant 1 - e^{-\delta\gamma e^{-\delta \frac{\pi(R-d)^2}{3}}},$$

where $Prb(Brd)$ is the probability that N_B broadcasts the message after receiving it from N_A, and

$$\gamma = \Delta(\mathcal{D}_{B,R}) - \Delta(\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}))$$

is the area of the hatched crescent shown in Figure 6.

Proof: Node N_B is not required to broadcast if and only if

$$\zeta^* : \forall N_Q \in \Lambda : \quad \exists N_P \text{ inside } I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}}),$$

where

$$\Lambda = \{N_X | \overline{AX} > R \text{ and } \overline{BX} \leqslant R\}.$$

Note that the nodes' positions have Poisson distribution. Therefore, using Lemma 6 we get

$$Prb(\exists N_P \text{ inside } I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})) = 1 - e^{-\delta\Delta(I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})} \geqslant 1 - e^{-\delta \frac{\pi(R-d)^2}{3}}. \quad (7)$$

Thus

$$Prb(Brd) = 1 - Prb(\zeta^*) = 1 - \sum_{k=0}^{\infty} Prob(|\Lambda| = k) Prb(\zeta^* | (|\Lambda| = k)).$$

where $|\Lambda|$ is the cardinality of the set Λ. Therefore, using (7) and Lemma 5 we get

$$Prb(Brd) \leqslant 1 - \sum_{k=0}^{\infty} \frac{(\delta\gamma)^k e^{-\delta\gamma}}{k!} (1 - e^{-\delta \frac{\pi(R-d)^2}{3}})^k = 1 - e^{-\delta\gamma e^{-\delta \frac{\pi(R-d)^2}{3}}},$$

where γ is the area of the hatched crescent in Figure 6 (collection of all points Q, $QA > R$ and $QB \leqslant R$)

$$\gamma = \Delta(\mathcal{D}_{B,R}) - \Delta(\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R})) = R^2(\pi - 2\arccos(\frac{d}{2R})) + d\sqrt{R^2 - (\frac{d}{2})^2}.$$

■

Corollary 2. *Using Theorem 8, we get*

$$Prb(Brd) \leqslant \delta\gamma e^{-\delta \frac{\pi(R-\overline{AB})^2}{3}},$$

Proof: consider the function

$$f(x) = x - e^{-x} + 1.$$

It is easy to show that $f(x)$ has a global minimum at $x = 0$. Therefore, we have

$$1 - e^{-x} \leqslant x$$

for any real number x. As a result, we get

$$Prb(Brd) \leqslant 1 - e^{-\delta\gamma e^{-\delta \frac{\pi(R-d)^2}{3}}} \leqslant \delta\gamma e^{-\delta \frac{\pi(R-d)^2}{3}}.$$

■

It is also possible that node N_B receives the message from more than one neighbor in its defer period. In this case, the number of N_B's neighbors that have received the message increases and the number of that have not received the message decreases. Consequently, the probability that N_B is required to broadcast the message further decreases compared to the case where N_B receives the message from only one neighbor. It is worth mentioning that RBS can guarantee that the number of forwarding nodes is within a constant factor of the optimal solution (minimum CDS), if it is provided with 2-hop neighbor information [18]. To the best of the authors' knowledge, this is the first flooding algorithm that constructs a CDS "on-the-fly" and can guarantee both full delivery and a constant approximation ratio to the optimal solution.

D. Relaxing Some System Model Assumptions

We assumed in Section II that the nodes are placed in a 2-D plane. However, this assumption is not used in the proof of Theorem 6. Therefore, the proposed receiver-based algorithm can also achieve full delivery when the nodes are distributed in a 3-D space. Note that in this case RBS uses 3-D node positions.

We can also relax the assumption that all the nodes have the same transmission range R. When the nodes' transmission ranges are different, the topology graph should be defined as a directed graph. In this case, a node N_B is defined to be a neighbor of N_A if N_B is in the transmission range of N_A. Note that the neighborhood relation is not symmetric. Let us assume that nodes not only put their *id* and position but also their transmission range (or perhaps their transmission power) into the hello messages that they periodically broadcast. Therefore, the neighbors of a node know both its position and transmission range. Suppose that node N_S initiates a flooding and there is a path from N_S to any node N_D in the network. In this case, nodes can use Algorithm 7 to decide whether or not to broadcast.

Algorithm 7 is a modified version of RBS. When executed by a node N_A, Algorithm 7 uses the position and transmission range of the broadcasting nodes to determine which neighbors have received the message. It then returns *false* if and only if it finds a neighbor N_B that has not received the message and

$$\overline{AC} \leqslant \overline{BC} \quad or \quad \overline{BC} > \text{transmission range of } N_C$$

for any N_A's neighbor N_C that has received the message.

Algorithm 7: Modified RBS

Input: $List_A$: List of all neighbors of N_A, and
$List_B$: List of broadcasting nodes.
output: True or False.

```
1.  List_C ← List_A.
2.  for (i = 1; i ⩽ length(List_C); i++ )
3.    for (j = 1; j ⩽ length(List_B); j++ )
4.      if (dist(List_C[i], List_B[j]) ⩽ R_{List_B[j]})
        //R_{List_B[j]}: Transmission range of node List_B[j]
5.          removeElement(List_C[i], List_C); break;

6.  List_D ← List_A − List_C.
7.  for (i = 1; i ⩽ length(List_D); i++ )
8.    check ← true;
9.    for (j = 1; j ⩽ length(List_C); j++ )
10.     if (dist(List_D[i], List_C[j]) < dist(List_D[i], N_A))
11.         if (dist(List_D[i], List_C[j]) ⩽ R_{List_C[j]})
12.             check ← false; break;
13.   if (check) return (false);
14. return (true)
```

Corollary 3. *In a collision-free network, Algorithm 5 can achieve* 100% *delivery if it uses the modified responsibility-based scheme to determine whether or not to broadcast.*

Proof: The proof is similar to the proof of Theorem 6. By contradiction, suppose there is at least one node that has not received the message after the flooding termination. Let us consider the set

$$\Lambda = \{(N_X, N_Y) | N_Y \text{ is a neighbor of } N_X, \\ N_X \text{ has received the message and} \\ N_Y \text{ has not received the message}\}.$$

Suppose N_S is the node that initiated flooding and N_T is a node that has not received the message. There is a path from N_S to N_T, thus we can find two nodes N_A and N_B along the path from N_S to N_T such that N_B is a neighbor of N_A, N_B has not received the message and N_A has received it. Consequently, $(N_A, N_B) \in \Lambda$, thus $\Lambda \neq \emptyset$. As a result

$$\exists (N_{A'}, N_{B'}) \in \Lambda \quad \text{s.t.} \quad \forall (N_X, N_Y) \in \Lambda : \quad \overline{A'B'} \leqslant \overline{XZ}. \tag{8}$$

Clearly, $N_{A'}$ has not broadcast, since $N_{B'}$ has not received the message. Therefore, there must be a node $N_{C'}$ such that $N_{C'}$ has received the message and

$$\overline{C'B'} < \overline{A'B'} \quad \text{and} \quad \overline{C'B'} \leqslant \text{ Transmission range of } N_{C'}.$$

This result contradicts (8) because $(N_{C'}, N_{B'}) \in \Lambda$.

■

V. Simulation

A. Average Number of Nodes Selected by the Proposed Sliced-Based Algorithm

In Section III, we proved that the proposed forwarding-node selection algorithm selects 11 nodes in the worst case. In practice, the number of selected nodes is typically less than 11. To avoid the complexity of mathematical analysis, we used a simulation to find the average number of selected nodes. For a given number of neighbors $1 \leqslant n \leqslant 160$, we randomly put n points inside a circle with radius R. We then ran the proposed selection algorithm and obtained the number of selected nodes. To get the average number of selected nodes, we ran simulation 10^6 times for each given n. As shown in Figure 7, the average number of selected nodes is less than 6 and approaches 5 when n increases. Note that the proposed sliced-based selection algorithm does not necessarily select a B-coverage with a minimum number of nodes. However, there is a sliced-based selection algorithm that can find a B-coverage with a minimum number of nodes in $O(nlogn)$ and can consequently reduce the average number of selected nodes.

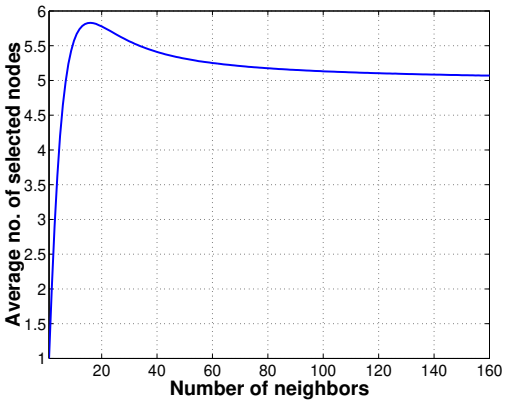

Fig. 7. Average number of nodes selected by the proposed slice-based algorithm.

B. Probability of Broadcast using the Proposed RBS

Suppose that the proposed receiver-based algorithm is used for flooding in the network. Assume that node N_B receives a message from N_A for the first time. It has been proven that the probability of N_B broadcasting the message ($Prb(Brd_B)$) exponentially decreases when the distance $\overline{AB}$ decreases or when the node density δ increases. We used simulation to

confirm this theoretical result. For the simulation, we considered two nodes N_A and N_B with distance $0 < d \leqslant R$ from each other. We uniformly placed nodes with density δ inside the network and checked whether or not N_B was required to broadcast the message. We ran simulation 10^6 times for a given δ and R. We then estimated $Prb(Brd_B)$ by the ratio of the number of times N_B was required to broadcast over total number of runs.

Figures 8–11 show the simulation results for several values of δ, d and R. As shown in these figures, the probability of broadcast exponentially decreases when d decreases or when δ increases. For example, when $R = 300$ m and $\delta = 4 \times 10^{-4}$, the probability of broadcast is 0.1 for $d = 250$ m and reduces to 10^{-4} for $d = 200$ m. This property can justify why the proposed receiver-based algorithm can significantly reduce the number of broadcasts in the network. Figure 12 illustrates an instance of using RBS for the case where $R = 300$ m, $\delta = 4 \times 10^{-4}$ and nodes are placed in a square area of 1000×1000 m^2. As shown in Figure 12, only 9 nodes (represented by stars) among 400 nodes broadcast the message.

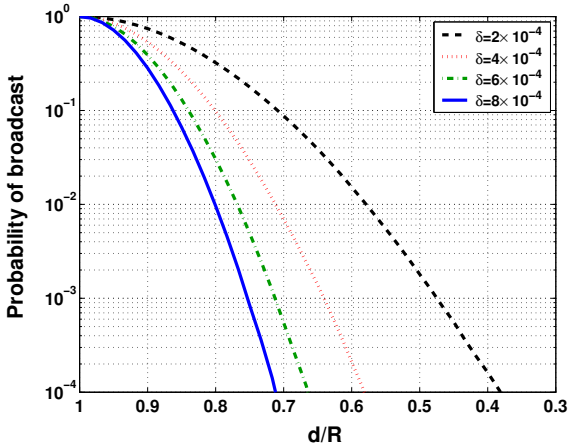

Fig. 8. Probability of broadcast for $R = 300$ m.

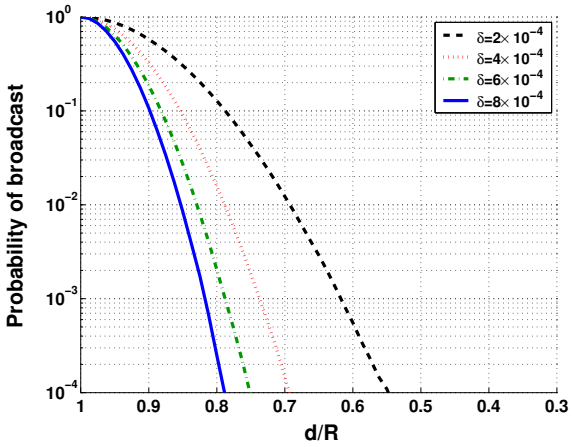

Fig. 9. Probability of broadcast for $R = 400$ m.

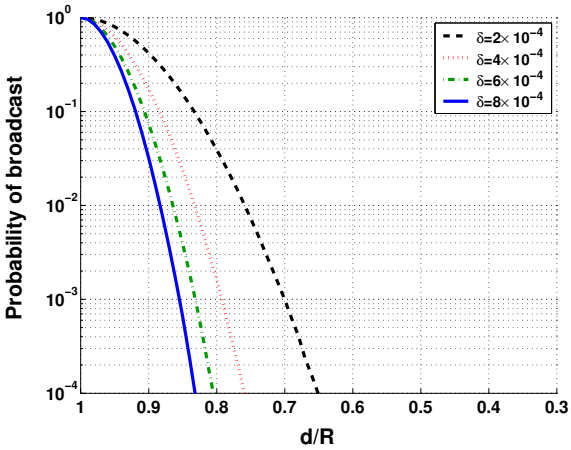

Fig. 10. Probability of broadcast for $R = 500$ m.

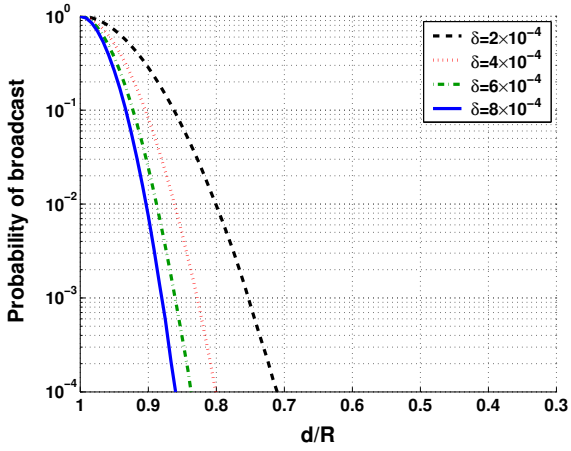

Fig. 11. Probability of broadcast for $R = 600$ m.

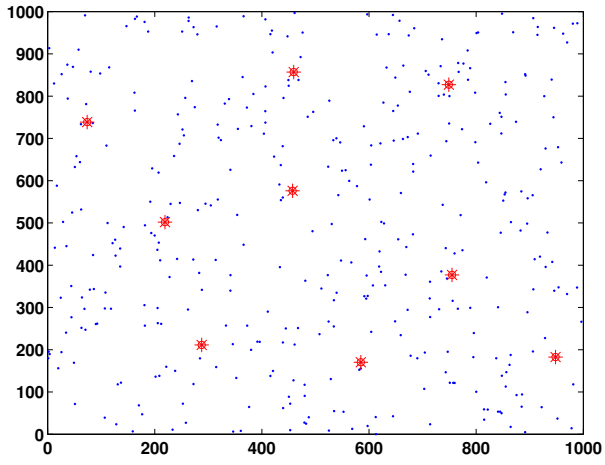

Fig. 12. Broadcasting nodes in a 1000×1000 m^2 square area with 400 nodes.

C. Performance of Proposed Sender-Based and Receiver-Based Algorithms

The main objective of efficient flooding algorithms is to reduce the number of broadcasts. Therefore, we considered the ratio of broadcasting nodes over the total number of nodes as the metric to evaluate the performance of proposed flooding algorithms. We evaluate this metric in the simulation against two parameters: transmission range and node density. We implemented a C++ simulator that uses a simplified version of IEEE 802.11 to simulate CSMA/CA broadcasting mode. Table I summarizes some of the parameters used in the simulator. The total size of the data packet was 256 bytes and the bandwidth of the wireless channel was set to 2 Mb/s. The signal propagation speed was assumed to be $3 \times 10^8\, m/s$ and the maximum defer time was set to 0.01 sec. In each simulation run, we randomly placed N nodes in a $1000 \times 1000\, m^2$ square area, where $N = \delta \times$ (network area). Only one flooding occurred in each simulation run by a randomly selected node.

Figures 13 and 14 show the average ratio of broadcasting nodes for over 1000 runs. The performance of our proposed algorithm is compared with the performance of the flooding algorithm proposed in [8]. In [8], Liu et al. show that the number of redundant broadcasts using their proposed flooding algorithm is significantly lower than that of previous notable flooding algorithms [19], [20]. As proved earlier, our proposed sender-based algorithm has lower computational complexity and selects fewer forwarding nodes than Liu's algorithm. The simulation results, shown in Figures 13 and 14, indicate two interesting facts. First, our proposed sender-based algorithm does not require more broadcasts than Liu's proposed flooding algorithm. Secondly, the number of broadcasts using our proposed receiver-based flooding algorithm is significantly lower than the number associated with Liu's algorithm. In fact, as shown in Figures 13 and 14, this number is even less than one of the best-known approximations (ratio 8 approximation) for the minimum number of required broadcasts [6].

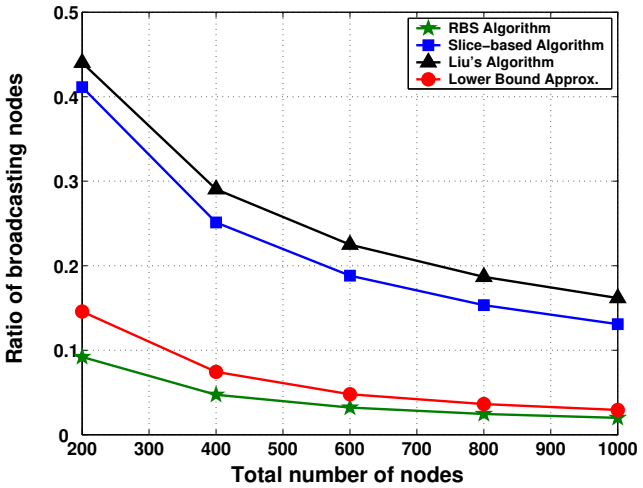

Fig. 13. Ratio of broadcasting nodes VS. total number of nodes.

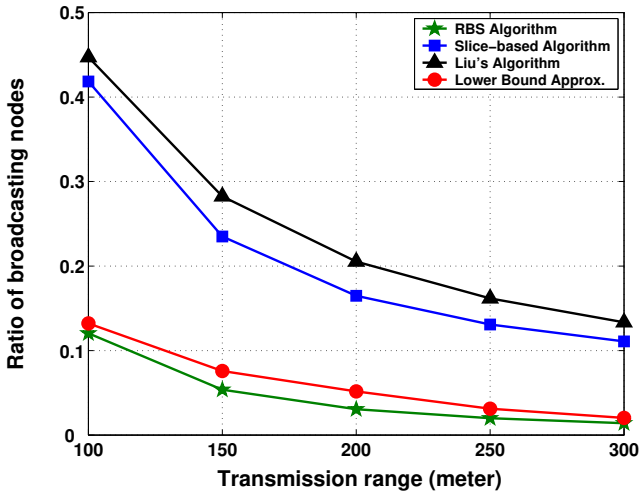

Fig. 14. Ratio of broadcasting nodes VS. transmission range.

Parameter	Value
Packet Size	256 bytes
Bandwidth	2 Mb/sec
Size of Square Area	10^6 m^2
Transmission Range	100–300 m
Interference Range	2 × Transmission Range
Signal Propagation Speed	10^8 meter/sec
Number of Nodes	200–1000
Maximum Defer Time	0.01 sec

TABLE I
SIMULATION PARAMETERS

VI. CONCLUSION AND FUTURE WORK

In the first part of this paper, we proposed a forwarding-node selection algorithm that selects at most 11 nodes in $O(n)$, where n is the number of neighbors. This limited number of nodes is an improvement over Liu's algorithm, which selects n nodes in the worst case and has time complexity $O(nlogn)$. Moreover, we showed that our proposed forwarding-node selection algorithm results in fewer broadcasts in the network. In the second part of the paper, we proposed an efficient receiver-based algorithm and showed why it significantly reduces the number of forwarding nodes in the network. We also relaxed some system model assumptions that are typically used in the flooding algorithms. Interestingly, the 2-hop-based version of our proposed receiver-based algorithm can guarantee constant approximation to the optimal solution (minimum CDS). As far as the authors know, this is the first flooding algorithm that constructs a connected dominating set (CDS) "on-the-fly" and can guarantee both full delivery and a constant approximation ratio to the optimal solution. As part of our future work, we will investigate the necessary conditions to guarantee both full delivery and constant approximation ratio to the minimum CDS.

APPENDIX

Proof of Lemma 5: The proof is by induction on the number of regions. The lemma is true if the number of regions is one (i.e. $k = 1$). Suppose that the inequality holds for $k = d$ regions. We have

$$\begin{aligned} &Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \bar{\zeta}_{\mathcal{R}_{d+1}}) = \\ &Prb(\zeta_{\mathcal{R}_1 - \mathcal{R}_{d+1}}, \zeta_{\mathcal{R}_2 - \mathcal{R}_{d+1}}, \ldots, \zeta_{\mathcal{R}_d - \mathcal{R}_{d+1}}), \end{aligned} \tag{9}$$

where $\bar{\zeta}_{\mathcal{R}_i}$ is the complement of $\zeta_{\mathcal{R}_i}$ and $R_i - R_j$ is the collection of all points inside R_i and outside R_j. Note that

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \zeta_{\mathcal{R}_1 - \mathcal{R}_{d+1}}, \zeta_{\mathcal{R}_2 - \mathcal{R}_{d+1}}, \ldots, \zeta_{\mathcal{R}_d - \mathcal{R}_{d+1}}) = 1.$$

Thus

$$\begin{aligned} &Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}) \geqslant \\ &Prb(\zeta_{\mathcal{R}_1 - \mathcal{R}_{d+1}}, \zeta_{\mathcal{R}_2 - \mathcal{R}_{d+1}}, \ldots, \zeta_{\mathcal{R}_d - \mathcal{R}_{d+1}}). \end{aligned} \tag{10}$$

It follows from (9) and (10) that

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}) \geqslant Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \bar{\zeta}_{\mathcal{R}_{d+1}}). \tag{11}$$

We have

$$\begin{aligned} &Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}) = \\ &Prb(\zeta_{\mathcal{R}_{d+1}}) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \zeta_{\mathcal{R}_{d+1}}) + \\ &Prb(\bar{\zeta}_{\mathcal{R}_{d+1}}) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \bar{\zeta}_{\mathcal{R}_{d+1}}). \end{aligned}$$

Therefore, using (11) we get

$$\begin{aligned} &Prb(\zeta_{\mathcal{R}_{d+1}}) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \zeta_{\mathcal{R}_{d+1}}) \geqslant \\ &(1 - Prb(\bar{\zeta}_{\mathcal{R}_{d+1}})) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}). \end{aligned}$$

Thus, using induction hypothesis we get

$$\begin{aligned} &Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}, \zeta_{\mathcal{R}_{d+1}}) \geqslant \\ &Prb(\zeta_{\mathcal{R}_{d+1}}) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}) \geqslant \\ &Prb(\zeta_{\mathcal{R}_1}) Prb(\zeta_{\mathcal{R}_2}) \ldots Prb(\zeta_{\mathcal{R}_d}) Prb(\zeta_{\mathcal{R}_{d+1}}). \end{aligned}$$

■

Proof of Lemma 6: For any point P on the circle $\mathcal{C}_{A,R}$ we have

$$\overline{BP} \leqslant \overline{AP} - \overline{AB} = R - \overline{AB}.$$

Therefore, as shown in Figure 6, the disk $\mathcal{D}_{B,(R-\overline{AB})}$ is inside $\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R})$. Consequently, we have

$$\Delta(I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})) \geqslant \Delta(I(\mathcal{D}_{B,(R-\overline{AB})}, \mathcal{D}_{Q,\overline{QB}})).$$

Since $\overline{QA} \geqslant R$, using triangle inequality we get

$$\overline{QB} \geqslant \overline{QA} - \overline{AB} \geqslant R - \overline{AB}.$$

Therefore, we have

$$\angle QBC \geqslant \frac{\pi}{3} \quad \text{and} \quad \angle QBD \geqslant \frac{\pi}{3}$$

and hence $\angle CBD \geqslant \frac{2\pi}{3}$. Therefore,

$$\Delta(I(\mathcal{D}_{B,(R-\overline{AB})}, \mathcal{D}_{Q,\overline{QB}})) \geqslant \frac{\Delta(\mathcal{D}_{B,(R-\overline{AB})})}{3} = \frac{\pi(R - \overline{AB})^2}{3}.$$

■

REFERENCES

[1] C. Perkins, "Ad hoc on demand distance vector (AODV) routing (internet-draft)," 1997, internet Draft, draft-ietf-manet-aodv-00.txt.

[2] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996.

[3] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," *In Proc. ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 151–162, 1999.

[4] M. Garey and D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[5] B. Clark, C. Colbourn, and D. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, pp. 165–177, 1990.

[6] P. Wan, K. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," *In Proc. IEEE INFOCOM*, vol. 3, pp. 1597–1604, 2002.

[7] S. Funke, A. Kesselman, U. Meyer, and M. Segal, "A simple improved distributed algorithm for minimum cds in unit disk graphs," *ACM Trans. Sensor Networks (TOSN)*, vol. 2, no. 3, pp. 444–453, 2006.

[8] H. Liu, P. Wan, X. Jia, X. Liu, and F. Yao, "Efficient flooding scheme based on 1-hop information in mobile ad hoc networks," *In Proc. IEEE INFOCOM*, 2006.

[9] Y. Tseng, S. Ni, and E. Shih, "Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc networks," *In Proc. Inernational Conference on Distributed Computing Systems (ICDCS)*, pp. 481–488, 2001.

[10] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," *In Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1124–1130, 2003.

[11] W. Lou and J. Wu, "Double-covered broadcast (DCB): a simple reliable broadcast algorithm in manets," *In Proc. IEEE INFOCOM*, pp. 2084–2095, 2004.

[12] J. Wu and F. Dai, "Broadcasting in ad hoc networks based on self-pruning," *In Proc. IEEE INFOCOM*, pp. 2240–2250, 2003.

[13] W. Peng and X. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," *In Proc. ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 129–130, 2000.

[14] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes in large scale sensor networks," *In Proc. ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 81–95, 2003.

[15] M. Sun, W. Feng, and T. Lai, "Broadcasting in ad hoc networks based on self-pruning," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 2842—2846, 2001.

[16] A. Papoulis, *Probability and statistics*. Prentice-Hall, Inc., 1990.

[17] R. Chang and R. Lee, "On the average length of delaunay triangulations," *BIT Numerical Mathematics*, vol. 24, pp. 269–273, 1984.

[18] M. Khabbazian and V. Bhargava, "On the number of forwarding nodes using reactive broadcasting algorithms in ad hoc networks," *In preperation*.

[19] Y. Cai, K. Hua, and A. Phillips, "Leveraging 1-hop neighborhood knowledge for efficient flooding in wireless ad hoc networks," *In Proc. IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 347–354, 2005.

[20] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," *In Proc. International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DiaLM)*, pp. 7–14, 1999.