An Efficient Algorithm for Finding an Almost Connected Dominating Set of Small Size on Wireless Ad Hoc Networks

Yamin Li, Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan

Abstract— **In this paper, we propose an efficient, distributed and localized algorithm for finding an almost connected dominating set of small size on wireless ad hoc networks. Broadcasting and routing based on a connected dominating set (CDS) is a promising approach. A set is dominating if all the nodes of the network are either in the set or neighbors of nodes in the set. The efficiency of dominating-set-based broadcasting or routing mainly depends on the overhead in constructing the dominating set and the size of the dominating set. Our algorithm can find a CDS faster and the size of the found CDS is smaller than the previous algorithms proposed in the literature. Although our algorithm cannot guarantee the set found is actually a CDS but from our simulation results, the probabilities that the found set is a CDS are higher than 99.96% in all cases.**

I. Introduction

A wireless ad hoc network is an interconnection of mobile computing devices, where the link between two neighboring nodes is established via radio propagation. Neighboring nodes can communicate directly when they are within transmission range of each other and radio propagation condition in the vicinity of these nodes is adequate. Communication between non-neighboring nodes requires a multi-hop routing protocol. In a multi-hop wireless network, each node has a transmission radius and is able to send a packet to all its neighbors that are located within the radius. Wireless networks consist of static or mobile hosts that can communicate with each other over the wireless links without any static network interaction. Each mobile host has the capacity to communicate directly with other mobile hosts in its vicinity. They can also forward packets destined for other nodes. Example of such networks are ad hoc local area, packet radio, and sensor networks.

Design of efficient broadcasting and routing protocols is one of the challenging tasks in ad hoc networks. Among various existing routing and broadcasting protocols, the ones based on dominating set are very promising. A subset of vertices in a graph is a *dominating set* if every vertex not in the subset is adjacent to at least one vertex in the subset. The dominating set should be connected, called *CDS*, for ease of the broadcasting or routing within the induced graph of dominating vertices. The main advantage of dominating-set-based approach is that it simplifies the broadcasting or routing process to the one in a smaller subnetwork generated from the CDS. Only the dominating vertices, called *forwarding nodes*, need to be active.

The efficiency of dominating-set-based approach depends largely on the time complexity for finding and maintaining a CDS and the size of the corresponding subnetwork. It is desirable to find a small CDS without compromising the functionality, reliability, and efficiency of an ad hoc network. Moreover, the algorithm for constructing the CDS should be efficient, distributed, and based on local information only. Since finding a minimum CDS for most graphs is NP-complete, efficient approximation algorithms are used to find a CDS of small size.

There are many existing algorithms in the literature for broadcasting/routing in ad hoc networks using dominating-set-based approach or its extensions [1], [2], [3], [5], [6], [7], [8], [10]. These algorithms can be evaluated by the efficiency in terms of the number of forwarding nodes, reliability in terms of delivery ratio, and running time for selecting the set of forwarding nodes.

The algorithm ensures *full coverage* if the found dominating set is connected and the nodes that are not in

the set connect to at least one node in the set. Some of the previous algorithms do not ensure the full coverage, the *span* algorithm [1] for instance. On the other hand, even an algorithm ensures the full coverage, it cannot ensure 100% delivery rate practically due to the contention and collision [3]. In general, if the number of forwarding nodes is large, there will be a rather high probability to cause contention and collision. In order to increase the delivery rate, the algorithm should try to reduce the size of the set of forwarding nodes.

In this paper, we propose a new algorithm for finding an almost CDS on ad hoc wireless networks. Our algorithm generates a smaller number of forwarding nodes and the time for selecting the set of forwarding nodes is shorter compared to other algorithms. Although the full coverage of the set of forwarding nodes cannot be guaranteed, it is almost full coverage in the sense that the successful rate of broadcasting using our algorithm is higher than 99.96% in all cases in our simulations.

The rest of the paper is organized as follows. Section 2 reviews the previous algorithms for selecting the set of forwarding nodes. Section 3 presents the new algorithm. Section 4 gives simulation results on the performance of the new algorithm and compares these results to that of Rieck's algorithm. Section 5 concludes this paper.

II. Previous Work

We consider an ad hoc network as a graph $G = (V, E)$, where V is a set of nodes and E is a set of bidirectional links. For each node v, $N(v) = \{u|(u, v) \in E\}$ denotes its neighbor set. Let $F \subset V$. We say F is a CDS if F is connected and $V - F \subset N(F)$, where $N(F) = \cup_{v \in F} N(v)$. A broadcasting or routing algorithm is full coverage if the set of selected forwarding nodes is a CDS. The key issue on designing a distributed algorithm for broadcasting or routing on wireless ad hoc networks is to determine a set of forwarding nodes with its size as small as possible, based on affordable local information.

In previously known algorithms that select a set of forwarding nodes, for each node v in the network, all pairs of neighbors of v are checked in order to determine its forwarding status. Node v is marked as forwarding node if it has two neighbors that are not connected directly. They differ in the ways of pruning techniques that are used to reduce the number of forwarding nodes.

In Wu and Li's algorithm, two pruning rules are used to reduce the size of the resultant CDS [11]. In rule 1, a forwarding node becomes non-forwarding if all of its neighbors are also neighbors of another node that has higher priority value. In rule 2, a forwarding node can be nonforwarding if its neighbor set is covered by two other nodes that are directly connected and have higher priority values. Dai and Wu extended the Wu and Li's algorithm by using a more general rule called Rule k in which a forwarding node becomes non-forwarding if its neighbor set is covered by k other nodes that are connected and have higher priority values [2].

Three types of priority were defined in [10]: 0-hop priority (node id), 1-hop priority (node degree), and 2-hop priority (NCR - neighborhood connectivity ratio), and the authors concluded that sing node id as priority is more efficient and more reliable than node degree and NCR [3]. In this paper, we use node id as the node priority value.

Chen et al. proposed an algorithm, called *Span*, to construct a set of forwarding nodes, called *coordinators* [1]. A node v becomes a coordinator if it has two neighbors that cannot reach each other by either directly connected, indirectly connected via one intermediate coordinator, or indirectly connected via two intermediate coordinators. Span uses 3-hop information and cannot ensure a CDS.

Rieck et al. proposed an algorithm that can be viewed as the enhanced Span [7]. In Rieck's algorithm, a node v is a forwarding node if it has two neighbors that cannot reach each other by either directly connected or indirectly connected via one intermediate node with higher priority than v. Rieck's algorithm requires only 2-hop information. Checking every pair requires $O(d^2)$ running time, where d is the maximum node degree of a network. Rieck's algorithm also checks an intermediate node that needs $O(d)$ running time. Therefore, the time complexity of Rieck's algorithm is $O(d^3)$,

The algorithm proposed in this paper differs with all previous algorithms by that the algorithm doesn't check all pairs of its neighbors in order to determine the forwarding status. The algorithm only check certain pairs of neighbors. So the running time of the algorithm is shorter. Furthermore, the number of forwarding nodes found by our algorithm is significantly smaller than other algorithms. Although the algorithm cannot guarantee the full coverage but it is almost full coverage in the sense that the successful transmission rate is higher than 99.96% in all cases in our simulations.

III. The Proposed Algorithm

Full coverage of a broadcasting algorithm in ad hoc network can be achieved theoretically by selecting a CDS

as the set of forwarding nodes. However, practically, the delivery ratio (the percentage of nodes that correctly receive a data packet) in most of cases is lower than 100% due to collision, contention, and mobility. Therefore, it is desirable to design a distributed broadcasting algorithm that is efficient in selecting a small set of forwarding nodes and the running time for the selection is fast although the set of selected forwarding nodes might not be a CDS with a very small probability. This is especially important for real-time applications.

The existing algorithms for deciding forwarding or non-forwarding status for a node v need to check every pair of neighboring nodes of v. If there is any pair of neighboring nodes of v that are not directly connected then v will be included in the initial set of the forwarding nodes. Therefore, the initially selected CDS might contain too many redundant nodes for forwarding the message in broadcasting or routing. Although some pruning techniques are used to reduce the size of the selected CDS in many algorithms, the overhead is high, especially when the size of the initially selected set is large.

For deciding forwarding or non-forwarding status for a node v, our algorithm does not check all pairs of v's neighbors. The number of pairs checked by the algorithm is $O(d \log d)$, where d is the maximum degree of nodes in the network. Intuitively, if there is a cycle that connects all the neighbors of a node then removing that node from the set of forwarding nodes might be OK since other nodes are still connected after removal of the node. This idea leads to a very simple $O(d)$ time heuristic algorithm that checks whether the cycle exits or not for the set of its neighbors in a random order. However, the coverage rates of the networks from the simulations were not completely satisfied. For ad hoc networks with 40 - 200 nodes in 2000m $\times$ 2000m area, the coverage rates are between 97% and 99% in average.

To increase the coverage of the network, we should increase the connectivity among the neighbors. This leads to the proposed algorithm in which for a node v, every neighbor of v checks log r other neighbors, where $r = deg(v)$ is the degree of node v. Node v is removed from the set of forwarding nodes if for every neighbor of v, all the log r neighbors checked have direct links. Intuitively, the log r connectivity for the neighbor set of a node should provide very high coverage of the network after removing that node. The algorithm first provides a circular array of the set $N(v)$, and then the indices of the neighbors are selected in an exponentially increasing fashion. If all pairs of the selected neighbors have direct links then v is set as a non-forwarding node.

Our algorithm extends the direct links to 2-hop links as in Rieck's algorithm. It works as follows: For each node v that has more than one neighbor, the algorithm first arranges its neighboring nodes in a total order, for example, an increasing order of *node_id*s. Let the neighboring nodes of v listed in this order be $v_0,\ v_1,\ \ldots,\ v_{r-1}$, where $r = deg(v)$. The algorithm checks the pairs of nodes $(v_i, v_{(i+s) mod\ r})$, where $i = 0, 1, \ldots r-1$ and $s = 2^j,\ j = 0, 1, \ldots, \lfloor \log_2 r \rfloor$. If there exists a pair of nodes that are neither connected directly nor connected via a node u that has a higher priority than v then v is marked as forwarding node.

The distributed algorithm runs in $O(d \log d)$ time for 1-hop connectedness and $O(d^2 \log d)$ for 2-hop connectedness, respectively. Previous algorithms for 1-hop and 2-hop connectedness run in $O(d^2)$ and $O(d^3)$, respectively.

Algorithm 1
begin
  my_status = nonforward;
  r = my_degree;
  if $r > 1$
    $i = 0$;
    $s = 1$;
    while $(s \leq r)$
      while $(i < r)$ and (my_status = nonforwarding)
        $j = (i + s)$ mod r;
        x = my_neighbor_id[i];
        y = my_neighbor_id[j];
        if $((x, y) \notin E)$ and ($\nexists z$ s.t. z.id > my_id
          and $(x, z) \in E$ and $(z, y) \in E$)
          /* require 2-hop information */
          my_status = forwarding;
          exit;
        endif
        i++;
      endwhile
      $s = 2s$;
    endwhile
  endif
end

The proposed distributed algorithm for each node v is shown in Algorithm 1. We use my_id and my_degree to denote node v and $deg(v)$, respectively. In the algorithm,

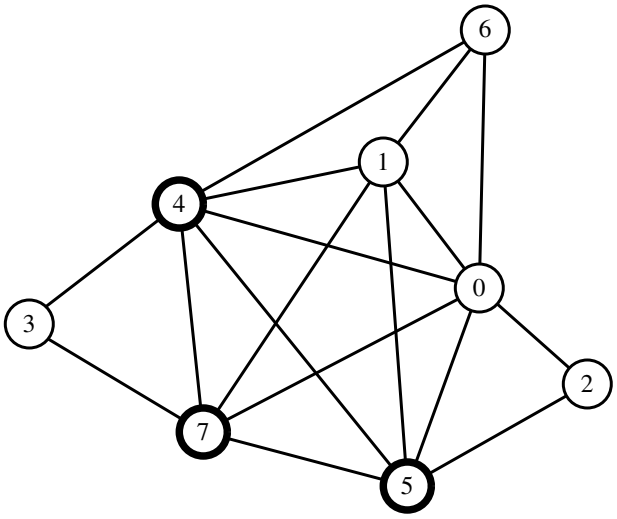


Fig. 1. Node 0: non-forwarding

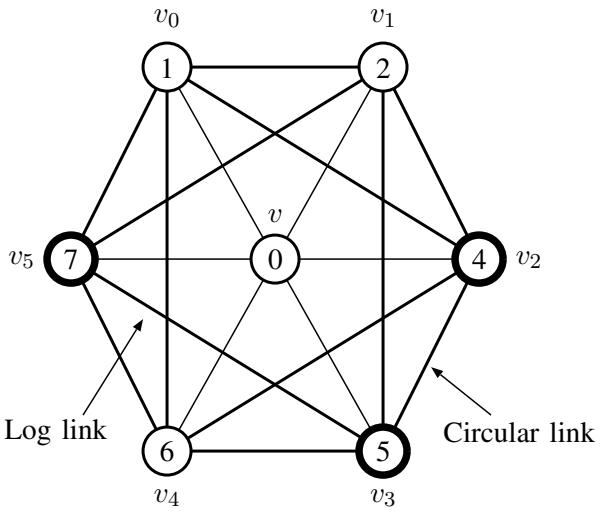


Fig. 2. A circular array and log links

my_neighbor_id, an array of length $deg(v)$, stores the *id*s of v's neighbors. The output of the algorithm is my_status that will be "forwarding" or "nonforwarding".

Figure 1 shows an example marked by our algorithm. The nodes with bold cycles, nodes 4, 5, and 7, are forwarding nodes; the rest are non-forwarding nodes. Our algorithm marks node 0 as a non-forwarding node: Node 0 has 6 neighbors: nodes 1, 2, 4, 5, 6, and 7. Our algorithm first checks whether these 6 nodes form a circular link (either 1-hop or 2-hop) in the increasing order of *node_id* or not. As shown as in Table I, it does.

TABLE I

A CIRCULAR ARRAY AND LINKS

Circular array		Log links	
Pair	Connected	Pair	Connected
(1,2)	Via node 5	(1,4)	Directly
		(1,6)	Directly
(2,4)	Via node 5	(2,5)	Directly
		(2,7)	Via node 5
(4,5)	Directly	(4,6)	Directly
		(4,1)	Directly
(5,6)	Via node 1	(5,7)	Directly
		(5,2)	Directly
(6,7)	Via node 1	(6,1)	Directly
		(6,4)	Directly
(7,1)	Directly	(7,2)	Via node 5
		(7,5)	Directly

Referring to Figure 2, in addition to the circular link, the algorithm also checks the log links (the links between two nodes of distance 2^j in the circular array). Since $r = deg(0) = 6$, only the nodes of distance 2 need to be checked. This is also listed in Table I. Since all log links exist, we mark node 0 as a nonforwarding node. Note that Rieck's algorithm marks node 0 as a forwarding node because nodes 2 and 6 are not connected.

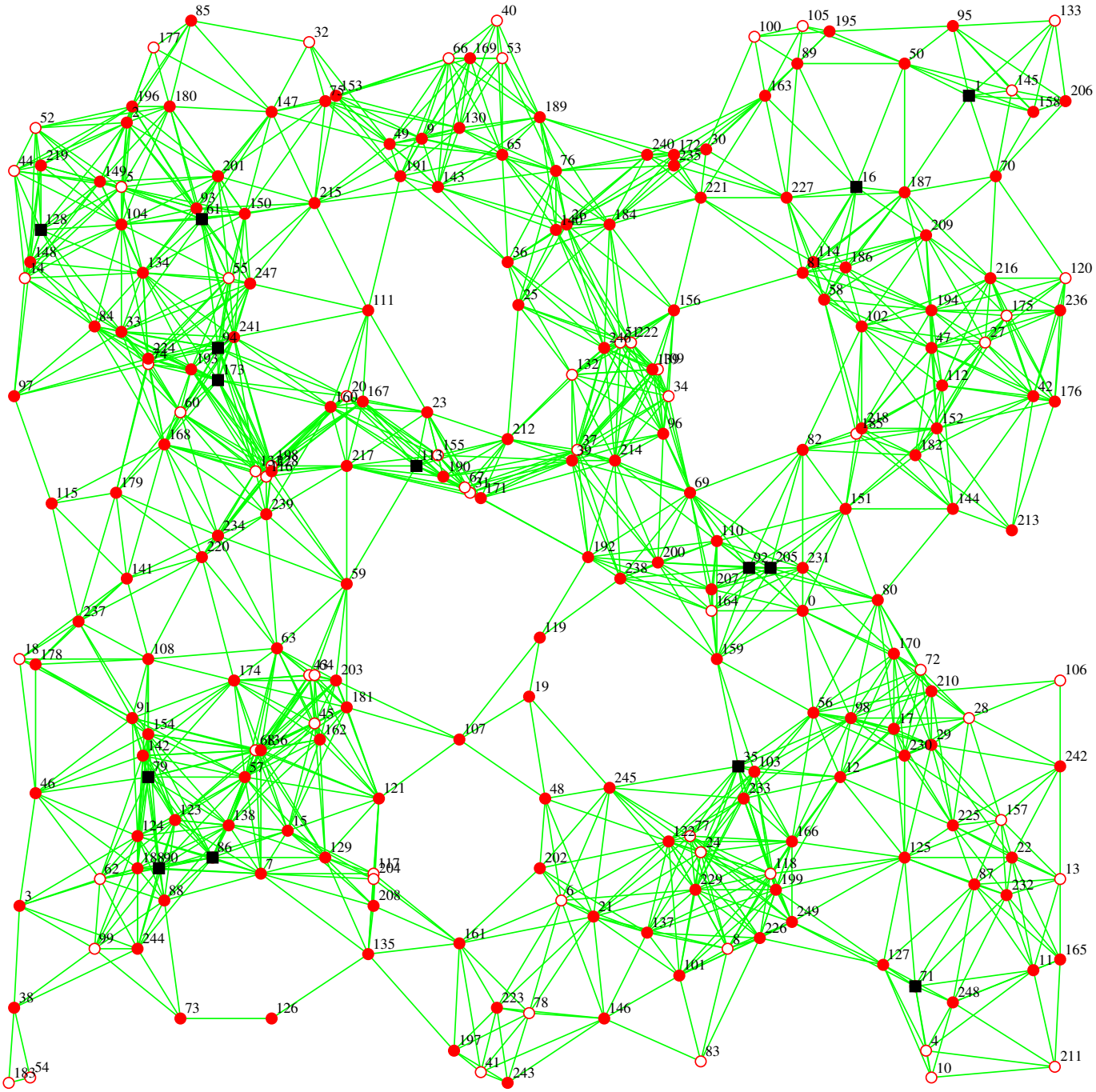
For 1-hop checking, since only up to $d \log d$ links are checked, the computing time is $O(d \log d)$. In practice, to reduce the size of the forwarding node set, we also check 2-hop connection between a pair of neighbors, that is, connected via an intermediate node. In this case, the computing time of the algorithm is $O(d^2 \log d)$.

Figure 3 shows a sample ad hoc network of 250 nodes located in a 2000m $\times$ 2000m area (the transmission range is set to 250m). The cycle nodes and the solid-circle nodes are forwarding and nonforwarding nodes, respectively, marked by our algorithm and Rieck's algorithm. The rectangle nodes (14 in total) are nonforwarding nodes in our algorithm but forwarding nodes in Rieck's algorithm. The sizes of the set of forwarding nodes in the two algorithms are 175 and 189, respectively.

IV. Performance Analysis and Simulations

We had done some simulations on our algorithm and Rieck's algorithm for broadcasting on wireless ad hoc networks. Our interests here are on evaluating efficiency (the number of forwarding nodes), coverage rate (the percentage of the forwarding nodes forming a CDS), and redundancy (the number of packets received per node).

All simulations were conducted on static networks



- Forwarding node marked by both algorithms
- Non-forwarding node marked by both algorithms
- Non-forwarding node marked by our algorithm but forwarding node marked by Rieck's algorithm

Fig. 3. The sets of forwarding nodes found by two algorithms on a sample ad hoc network

with a collision-free MAC layer. Each ad hoc network is generated by randomly placing n, $100 \leq n \leq 400$, nodes in a restricted 2000m $\times$ 2000m area. The transmission ranges are set to be 250m, 350m, and 450m. Both algorithms check 2-hop connectedness and use *node_id* as priority. For each configuration, we test

10,000 networks.

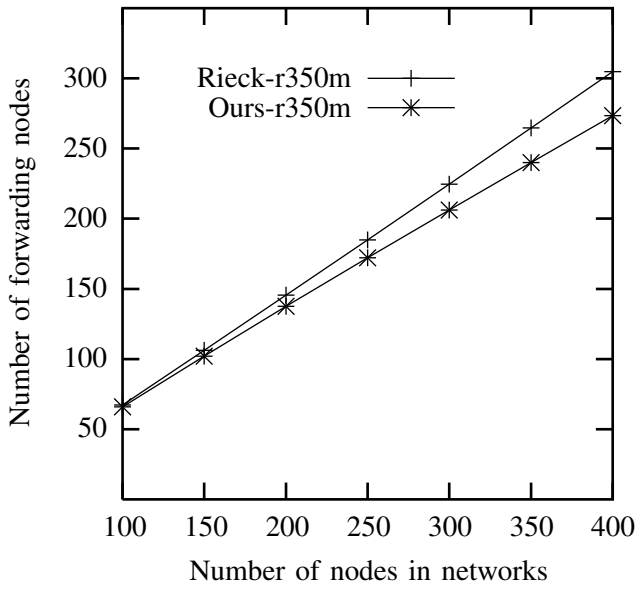


Fig. 4. The numbers of forwarding nodes

Figure 4 shows the number of forwarding nodes for randomly generated ad hoc networks of node ranges from 100 to 400, and the transmission range is set to be 350m. From the figure, it is clear that our algorithm outperforms Rieck's algorithm by reducing the number of forwarding nodes. For other transmission ranges (250m and 450m), the results are similar to that in Figure 4. Table II lists the details.

TABLE II
THE AVERAGE NUMBERS OF FORWARDING NODES

#nodes	r250m		r450m	
	Rieck	Ours	Rieck	Ours
100	64.3	64.0	67.4	64.47
150	102.0	100.6	104.1	96.93
200	142.9	139.6	141.0	128.2
250	184.3	178.2	178.2	159.2
300	225.7	216.3	215.5	188.9
350	267.4	254.2	253.2	218.2
400	309.2	291.7	290.7	247.1

Table III gives the coverage rate, the percentage of the forwarding nodes forming a CDS. These are obtained by dividing the number of full coverages by the total number of trials. The worst case is that, in 10000 trials, there are only 3 times in which the forwarding nodes do not forward packets to all nodes in the network. We can see clearly from the simulation results that the coverage rates are higher than 99.96% in all cases in our simulations. We conclude that the set of forwarding nodes generated by our algorithm is *almost* a CDS practically.

TABLE III
RATE OF SUCCESSFUL BROADCASTING

#nodes	r250m	r350m	r450m
100	99.97%	100.00%	100.00%
150	99.97%	99.98%	100.00%
200	99.99%	100.00%	100.00%
250	99.99%	100.00%	100.00%
300	100.00%	100.00%	100.00%
350	99.99%	100.00%	100.00%
400	100.00%	100.00%	100.00%

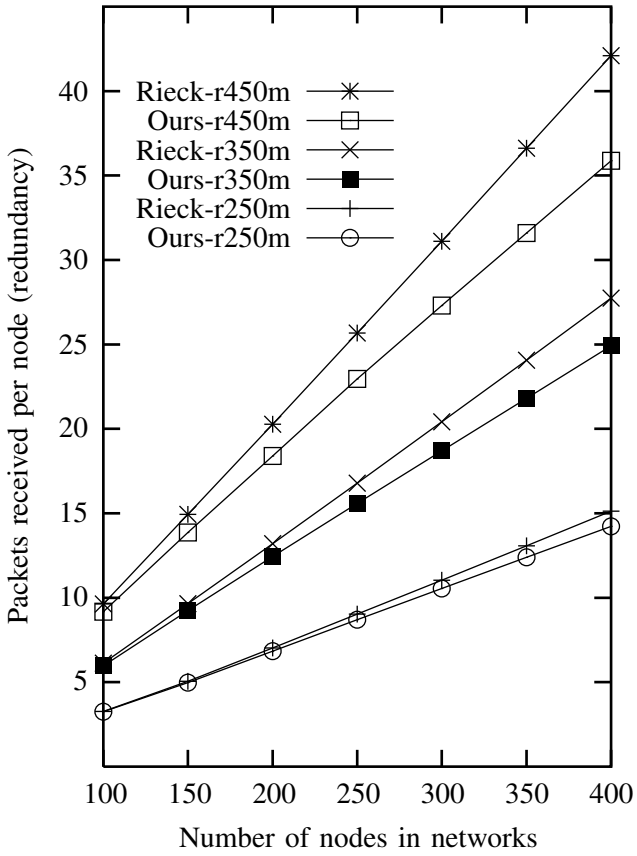


Fig. 5. Redundancies of the two algorithms

Figure 5 shows the broadcast redundancy, which is defined as the average number of duplicated packets received at each node when a node broadcasts a packet

to all the other nodes. We only test the broadcast redundancy when the forwarding nodes form a CDS. In such a case, any node can act as the initial node to broadcast a packet to all the other nodes and selecting different initial node does not affect the broadcast redundancy. Node 0 was assigned as the initial node in this simulation. We can see that our algorithm has lower redundancy (higher efficiency) than Rieck's algorithm.

Figure 6 shows the relative redundancy to Rieck's algorithm. The relative redundancy is defined as the total number of packets received by all nodes of our algorithm dividing by that of Rieck's algorithm. On average, the relative redundancy of our algorithm to Rieck's algorithm is 88.68%. Lower redundancy will has lower collision rate and hence improve the delivery ratio.

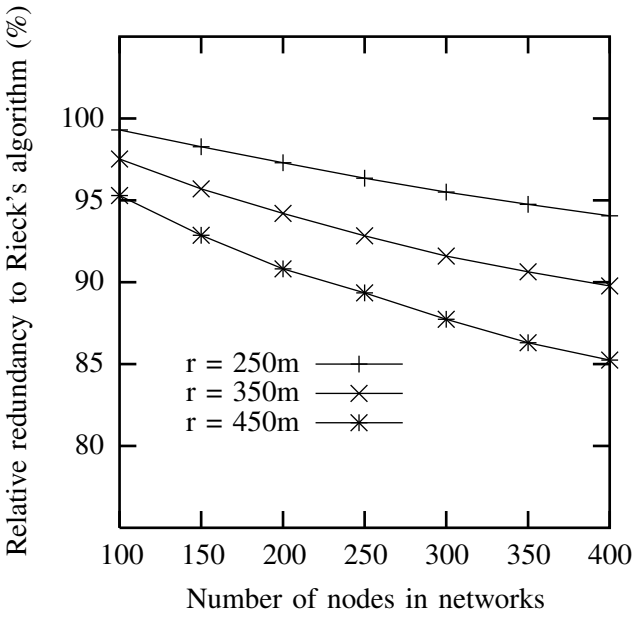


Fig. 6. Relative traffic to Rieck's algorithm

V. Concluding Remarks

A new distributed algorithm for finding an almost connected dominating set on ad hoc network was proposed and the performance was evaluated through simulations. Although the performance is compared only to Rieck's algorithm, it is foreseeable that our algorithm will produce smaller set of forwarding nodes than the other CDS algorithms under the same requirement of neighborhood information.

We did not perform pruning techniques on the generated set of forwarding nodes in our algorithm. Theoretically, we can apply any self-pruning technique to improve the size of the set of forwarding nodes; just replace the initial-set generated from checking all pairs by the set generated from our algorithm. It is quite obvious that the size of the resulting forwarding set will be smaller than using the original initial-set. If the self-pruning technique used guarantees full coverage then the coverage of the resulting set should remain the same; 99.97% at least. Our future work includes combining some self-pruning techniques in our algorithm to reduce furthermore the size of the forwarding set.

It is interesting to investigate the performance of the proposed algorithm theoretically, for example, to derive a good lower bound of the coverage rate of the proposed algorithm under certain conditions. For application consideration, to investigate the performance of the proposed algorithm under dynamic environment with packet collision and node mobility is also worth further research.

References

[1] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5):481–494, Sept. 2002.

[2] F. Dai and J. Wu. Distributed dominant pruning in ad hoc wireless networks. In *Proc. of IEEE International Conf. on Communications*, pages 353–357, 2003.

[3] F. Dai and J. Wu. Performance analysis of broadcast protocols in ad hoc networks based on self-pruning. *IEEE Trans. Parallel and Distributed Systems*, 15(11):1–13, 2004.

[4] H. Lim and C. Kim. Flooding in wireless ad hoc networks. *Computer Comm. J.*, 24(3-4):353–363, 2001.

[5] W. Lou and J. Wu. On reducing broadcasting redundancy in ad hoc wireless networks. *IEEE Trans. Mobile Computing*, 1(2):111–123, 2002.

[6] E. Pagani and G.P. Rossi. Providing reliable and fault-tolerant broadcasting delivery in mobile ad hoc networks. *Mobile Networks and Applications*, 4:175–192, 1999.

[7] M. Q. Rieck, S. Pai, and S. Dhar. Distributed routing algorithms for wireless ad hoc networks using d-hop connected dominating sets. In *Proc. Int'l Conf. High Performance Computing in Asia Pacific Region*, Dec. 2002.

[8] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks. *IEEE Trans. Parallel and Distributed Systems*, 13(1):14–25, 2002.

[9] Y.-C. Tseng, S.-Y. Ni ans Y.-S. Chen, and J.-P Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2/3):153–167, 2002.

[10] J. Wu and F. Dai. A generic distributed broadcast scheme in ad hoc wireless networks. *IEEE Trans. Computers*, 53(10):1343–1354, 2004.

[11] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 7–14, 1999.