An Efficient Distributed Broadcasting Algorithm for Wireless Ad Hoc Networks

Yamin Li, Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan
{yamin;speng}@k.hosei.ac.jp

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan
w-chu@u-aizu.ac.jp

Abstract

In this paper, we propose a distributed broadcasting algorithm for wireless ad hoc networks. In the algorithm, an efficient strategy is used to determine the forward status of a node by just checking whether there exists a ring that contains all its neighbors. The proposed algorithm is more efficient than the existing broadcasting algorithms in the literatures. That is, the size of the forwarding nodes found by our algorithm is smaller and the running time is faster than other broadcasting algorithms. Reducing the number of forwarding nodes will decrease the probability of transmission collision, and hence improve the packet delivery ratio. The algorithm runs in $O(d^2)$ time, where d is the maximum node degree. The full coverage is not guaranteed but as shown by the simulation results, the probability of full coverages can be over 99 percent when the network contains 100 or more nodes.

1. 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. In a broadcasting task, a source node sends the same packet to all the nodes in the network. It is an important task used for paging, alarming, location updates, route discoveries or even routing in highly mobile environments.

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.

For a broadcasting task, we assume the one-to-all model in which transmission by each node can reach all nodes that are within radius distance from it. *Blind flooding* is the most basic broadcasting approach, in which a source node broadcasts the packet to its neighborhood. Recursively, each node that receives the packet first time broadcasts the packet to its neighborhood. Blind flooding results in redundant transmissions: A node with d neighbors will receive the same packet d times. Blind flooding ensures the coverage. However, the redundant transmissions in blind flooding may cause contention and collision [4, 9]. In a collision, several nodes send packets simultaneously and packets are lost due to interference. In a contention, a node backs off when the channel is occupied.

Self-pruning is an effective method in reducing broadcast redundancy. In self-pruning-based broadcast protocols [1, 2, 3, 5, 6, 7, 8, 10], each node collects neighborhood topology information (static information) and extract broadcasting history information (dynamic information) from incoming packets. Each node decides its role in a specific broadcasting. It either becomes a forwarding node and forwards the broadcast packet or becomes a non-forwarding node and does nothing. If the decision is made based on only static information, the corresponding protocol is a static protocol. Otherwise, it is a dynamic protocol.

The forwarding node set derived by static protocols can be used in any broadcasting while the one derived from dynamic protocols is normally used in a specific broadcasting. In general, dynamic protocols can generate smaller forwarding node set. However, this is done at the cost of longer completion time of the broadcasting process.

A set of nodes is a dominating set if every node in the network is either in the set or a neighbor of a node in the set. If the forwarding nodes, including the source node, form a connected dominating set then the self-pruning protocol ensures the full coverage. Self-pruning protocols 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.

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

In this paper, we propose a new static self-pruning scheme that is more efficient than the algorithms proposed before. Our algorithm generates a smaller number of forwarding nodes and runs much faster for selecting the set of forwarding nodes though the full coverage of the broadcasting cannot be guaranteed. The simulations show that reliability of the broadcasting is 99% or higher for networks containing 100 or more nodes.

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

2. 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 be the set of forwarding nodes. A broadcasting is successful if every node in G receives the broadcast packet; that is, $V - F \subset N(F)$, where $N(F) = \cup_{v \in F} N(v)$. A broadcasting algorithm is full coverage if it guarantees successful broadcasting, providing that G is connected. The key issue in designing a broadcasting algorithm is to determine a small forwarding node set based on affordable local information.

A self-pruning is distributed if each node decides its own status (forwarding/non-forwarding) independently based on the local information. In the previous distributed self-pruning algorithms, for each node v, all pairs of neighbors of v are checked in order to determine its status. For each pair (u, u') of neighboring nodes of v, up to k-hops information (whether there exists a path of length k from u to u' or not) is checked. The value of k determines the time complexity of the algorithm.

In this section, we review some static self-pruning algorithms. In Wu and Li's algorithm [11], a marking process is proposed to determine a set of gateways (forwarding nodes) that forms a CDS (connected dominating set). A node is marked as a gateway if it has two neighbors that are not directly connected. Two pruning rules are used to reduce the size of the resultant CDS. In rule 1, a gateway v becomes a non-gateway if all of its neighbors are also neighbors of another node that has higher priority value. In rule 2, a marked node can be unmarked if its neighbor set is covered by two other nodes that are directly connected and have higher priority values.

Dai and Wu [2] extended the Wu and Li's algorithm by using a more general rule called Rule k. A gateway becomes a non-gateway if its neighbor set is covered by k other nodes that are connected and have higher priority values.

Chen et al. [1] proposed the *Span* protocol to construct a set of forwarding nodes called *coordinators*. 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 a CDS 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, Rieck's algorithm's running time is $O(d^3)$,

The algorithm proposed in this paper, called *Ring* algorithm, is an effort continuing the trend of previous algorithms to generate the set of forwarding nodes with small size in an efficient way. The algorithm will be described in the next section.

3. Ring Algorithm

Full coverage of a broadcasting algorithm in ad hoc network can be achieved theoretically by selecting a CDS as the forwarding node set. However, practically, the delivery ratio in most of cases is lower than 100% due to collision, contention, and mobility. Therefore, it is desirable to design a distributed self-pruning broadcasting algorithm that is efficient in selecting a small forwarding node set and its running time is fast although a CDS cannot be ensured. This is especially important for real-time applications.

The existing algorithms for deciding forward or non-forward 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 nor connected via 2 or 3 hops, v will be included in the forwarding node set. Therefore, the size of the forwarding node set might become too large and the time to check forward status might also be quite long.

Our idea to reduce the size of the forwarding node set as well as the time for checking is to check whether there exists a cycle that contains all the neighboring nodes of v. If such a cycle exists then we mark v as a non-forwarding node. This idea is motivated by observing that if a cycle that contains all nodes in the set of neighboring nodes of v exists then node v will be very likely a redundant node for forwarding the broadcast packet.

Our algorithm works as follows. Initially, every node is marked as a non-forwarding node. For each node v that has more than one neighbor, the algorithm checks its neighboring nodes in a particular order, e.g., 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 $r = d(v)$. For each node pair $(v_i, v_{(i+1) mod\ r})$, the algorithm checks whether they are connected directly or connected via a node u that has a higher priority than v. If such path (cycle) does not exist then v is remarked as a forwarding node. Otherwise, there is a cycle of length between $d(v)$ and $2d(v)$ that includes all the neighboring nodes of v and the node v remains non-forward status.

The Ring algorithm is shown in Algorithm 1. We use my_id and my_degree to denote node v and $d(v)$, respectively. In the algorithm, my_neighbor_id, an array of length my_degree, stores the neighbor's *id*s of my_id. The output of the algorithm is my_status that will be "forward" or "non-forward".

```
Algorithm 1 (Ring)
begin
  my_status = nonforward;
  if my_degree > 1
    $i = 0$;
    while (i < my_degree) and (my_status = nonforward)
      $j = (i + 1)$ mod my_degree;
      $x$ = my_neighbor_id[$i$];
      $y$ = my_neighbor_id[$j$];
      if ($x$ and $y$ are not directly connected by a link)
        if (a node $z$ with higher id than my_id that
              connects to both $x$ and $y$ does not exist)
          my_status = forward;
        endif
      endif
      $i$++;
    endwhile
  endif
end
```

Figure 1 shows an example marked by our algorithm. The nodes with bold cycles are forwarding nodes; the rest are non-forwarding nodes. Node 1 has 4 neighbors: nodes 2, 3, 5, and 8. These neighbors form a ring in the increasing order of *node_id*s: Node 2 connects to node 3 directly; node 3 connects to node 5 directly; node 5 indirectly connects to node 8 via an intermediate node 3 that is higher than node 1; and node 8 indirectly connects to node 2 via node 3. Note that node 3 is also marked as a non-forwarding node by our algorithm, but it is marked as a forwarding node by Rieck's algorithm.

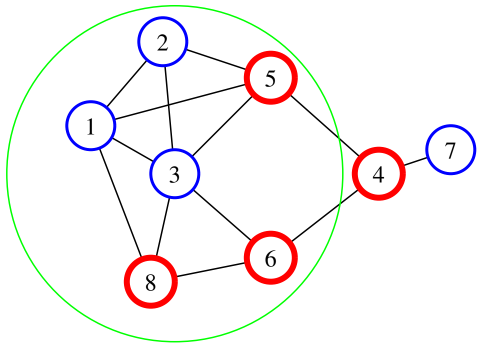

Figure 1. An example

For 1-hop checking, since only up to d links are checked, the computing time is $O(d)$. 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. Therefore, the computing time is $O(d^2)$ and the ring found is of the length between $d(v)$ and $2d(v)$.

4. Performance Analysis and Simulations

We perform some experiments on Rieck's algorithm and our Ring algorithm on ad hoc networks. Our interest here is on efficiency (the number of forwarding nodes), coverage rate (the percentage of the forwarding nodes forming a CDS), and speedup (the ratio of the running times of the two algorithms).

All simulations are conducted on static networks with a collision-free MAC layer. Each ad hoc network is generated by randomly placing n, $40 \leq n \leq 200$, nodes in a restricted 200×200 area. The transmission ranges are set to be 60, 80, and 100. Both algorithms use 2-hop information and *node_id* as priority. For each configuration, we test 10,000 times to get the coverage rate and the average efficiency and speedup.

Figures 2 and 3 show the efficiency of Rieck's and Ring algorithms. Figure 2 shows the number of forwarding nodes for randomly generated ad hoc networks of node ranges

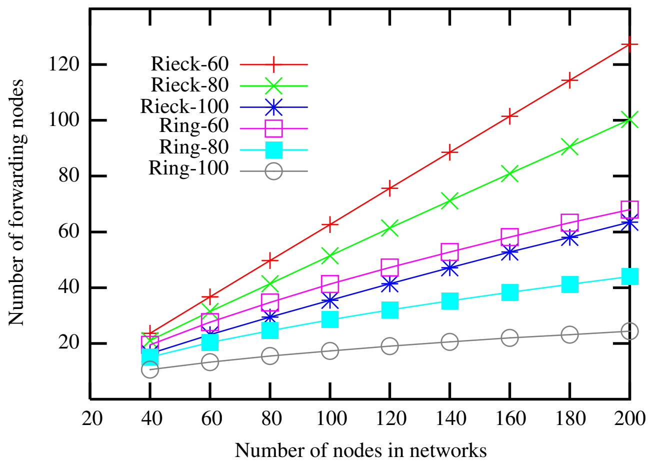

Figure 2. The numbers of forwarding nodes

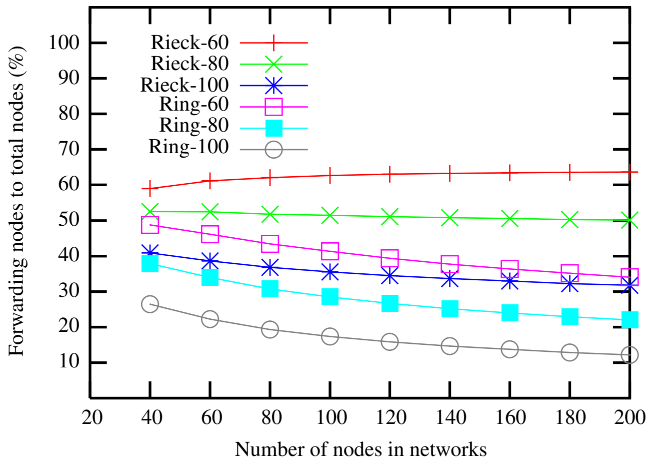

Figure 3. Ratio of the Numbers of forwarding nodes

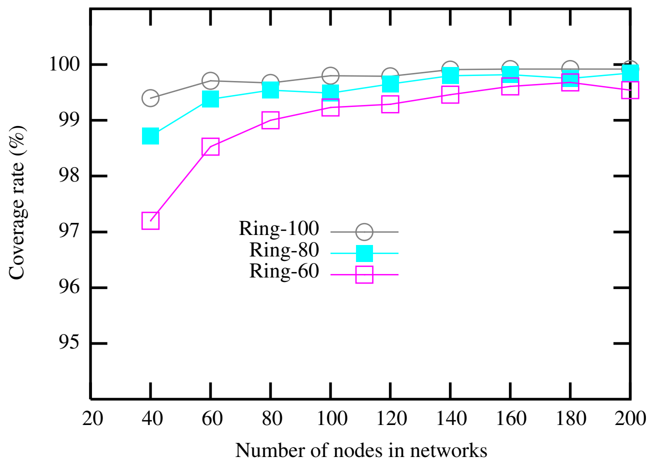

Figure 4. Rate of successful broadcasting

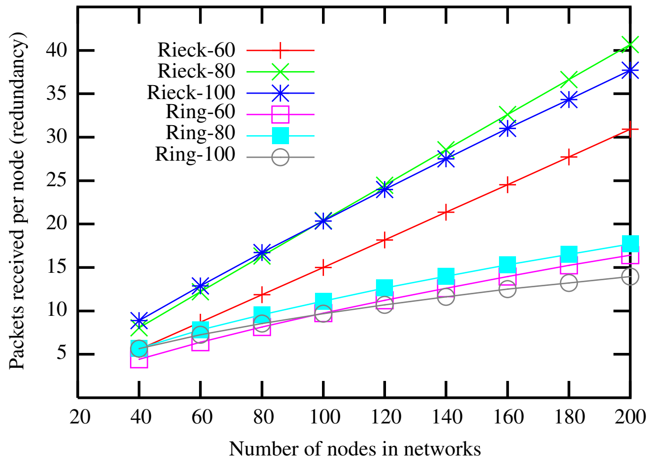

Figure 5. Redundancies of the two algorithms

from 40 to 200, and the transmission ranges are set to be 60, 80, and 100. From the figure, it is clear that Ring algorithm outperforms Rieck's algorithm by reducing the number of forwarding nodes to nearly half for dense ($n \geq 100$) graphs. Figure 3 shows the ratio of the numbers of forwarding nodes generated by the two algorithms to the total number of nodes in networks. We can see that the efficiency becomes better as the transmission radius increases.

Figure 4 shows the rate of successful broadcasting using Ring algorithm. We generated 10,000 networks for each network configuration, and obtained the successful rate by dividing the number of successful broadcastings by 10,000. From the figure, we can see that the successful rate increases as the network becomes more dense. The rate is over 99% for $n \geq 100$.

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 can see that Ring 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 Ring algorithm dividing by that of Rieck's algorithm. On avarage, the relative redundancy of Ring algorithm to Rieck's algorithm is 54.7%. Lower redundancy will has lower collision rate and hence improve the delivery ratio.

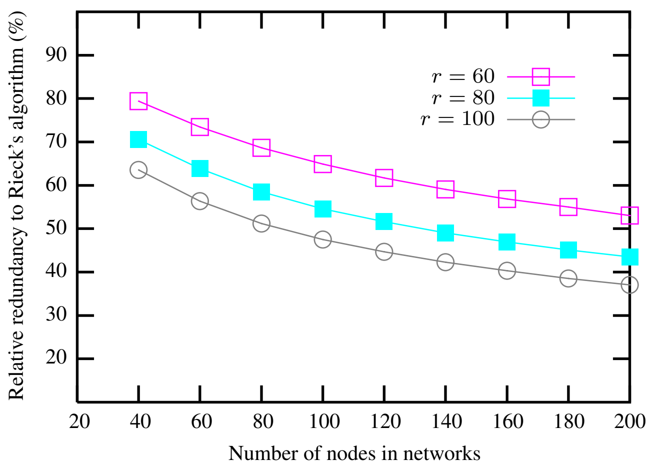

Figure 6. Relative traffic to Rieck's algorithm

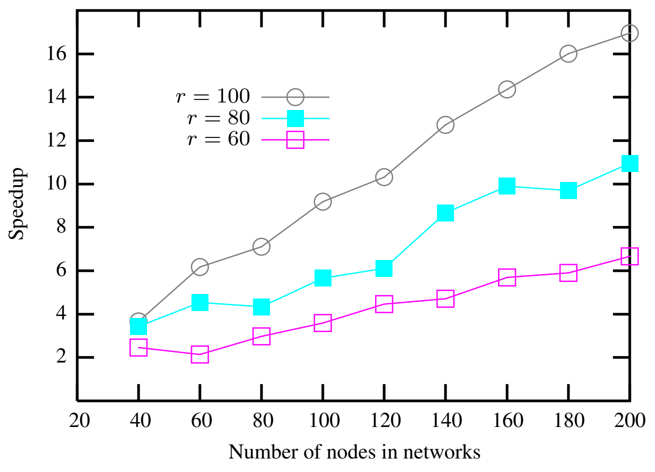

Figure 7. Speedup of Ring algorithm over Rieck's algorithm

Figure 7 shows the speedup of Ring algorithm over Rieck's algorithm. The speedup is defined as

$$\text{Speedup} = \frac{\text{Execution time of Rieck's algorithm}}{\text{Execution time of Ring algorithm}}$$

The speedup gets higher as the size of the networks grows up. The range of the speedup is approximately 2 to 16 for n from 40 to 200.

The simulation results demonstrated that Ring algorithm has better efficiency and shorter running time than Rieck's algorithm.

5. Concluding Remarks

A new distributed, self-pruning broadcasting algorithm 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 forwarding node set than the other existing broadcasting algorithms under the same requirement of neighborhood information.

Our future work includes the extension of our new algorithm to other models, e.g., the dynamic model in which the graph topology will change through time. We are also interested in the performance evaluation of the new algorithm in realistic environments with packet collision and node mobility.

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. 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 Aisa 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. N. 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.