Multicast Tree Construction and Flooding in Wireless Ad Hoc Networks

Hyojun Lim, Chongkwon Kim
Department of Computer Science and Engineering, Seoul National University, Korea

Abstract

In an ad hoc network, each host assumes the role of a router and relays packets toward final destinations. This paper studies efficient routing mechanisms for multicast and broadcast in ad hoc wireless networks. Because a packet is broadcast to all neighboring nodes, the optimality criteria of wireless network routing is different from that of wired network routing. In this paper, we point out that the number of packet forwarding is the more important cost factor than the number of links in the ad hoc network. After we show constructing minimum cost multicast tree is hard, we propose two new flooding methods, self pruning and dominant pruning. Both methods utilize neighbor information to reduce redundant transmissions. Performance analysis shows that both methods perform significantly better than blind flooding. Especially, dominant pruning performs close to the practically achievable best performance limit.

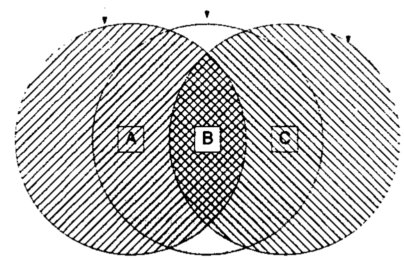

Figure 1. Packet forwarding in the ad hoc network

1 Introduction

An ad hoc network consists of wireless mobile hosts which form a temporary network without the aid of established infrastructure or centralized administration. An ad hoc network can be easily constructed with low cost since wireline infrastructures need not be installed. Ad hoc networks will be employed in areas such as emergency rescue sites, combat fields, etc.

Each mobile host in an ad hoc network acts as a router. If a source cannot send a packet directly to a final destination due to the limited transmission range, the source host sends the packet to intermediate hosts and intermediate hosts forward the packet toward the destination. Figure 1 shows the example of an ad hoc network where host B assumes the role of a router and relays packets from host A to host C.

One imminent problem in ad hoc networks is ad hoc network routing. Ad hoc network routing should be flexible and efficient because network topology changes dynamically and bandwidth is limited in ad hoc wireless networks. Several efficient point-to-point routing algorithms have been proposed [1, 3, 4, 6, 8]. These proposed algorithms try to avoid routing loops and reduce the overhead of routing information exchange.

In addition to point-to-point routing, several researchers studied multicast routing in ad hoc networks [11, 13]. These methods exploit the shared tree or DAG (Destination-oriented Acyclic Graph) to adapt to the dynamic changes of network topology.

Though the theoretical studies should be taken first in order to design the efficient multicast routing protocol in ad hoc networks, little research efforts were made. Before describing the theoretical bases on the multicast routing protocol, let's review the theoretical bases of the wireline multicast routing protocols. The two major methods used to construct multicast tree is the *shortest path tree* and *Steiner tree* [20]. In shortest path tree, packets can be sent in minimum hops since the paths between source and all the destinations are shortest paths. On the contrary, Steiner tree minimizes the total cost of multicast tree. It is known that constructing Steiner tree in arbitrary graph is NP-complete [18]. Also, the shape of Steiner tree changes

MSWIM 2000 Boston MA USA

so frequently when the multicast destination is added or deleted. Thus, the shortest path tree is most widely used as the multicast tree in wireline networks.

The important property of the ad-hoc network that should be considered in designing the multicast routing protocol, is the *broadcast property.* For example, the packet sent by node B is delivered to all nodes in the transmission range of B in figure 1. Thus, only one packet transmission is needed for node B to deliver the multicast packet to A and C simultaneously. In wireline networks, two transmissions are needed to deliver the packet to both A and C. In designing the multicast tree construction algorithm, the benefit by the broadcast property must be considered.

We also deal with *broadcast* which is the special case of multicast. Since broadcast is to send data from one source node to all other nodes in the network, we can regard broadcast as the multicast in which all nodes are destinations. Broadcast has extensive applicability in ad hoc wireless networks. For example, several point-to-point routing algorithms such as AODV [4] and DSR [8] rely on broadcast to obtain routing information. Furthermore, broadcast can be a method of choice over complicated ad hoc multicast routing.

In spite of the importance of broadcast, little research efforts were made to find efficient broadcasting methods. Works in [2, 12, 15, 19, 22, 23] seek to find efficient broadcast methods in an ad hoc environment. They focus on finding time-slot assignment in TDMA-based wireless systems. Since these works rely on TDMA, they cannot be applied directly to MAC-based ad hoc networks. The broadcast storm problem in ad hoc networks was addressed in [21]. They proposed some methods that can improve the efficiency of broadcasting in ad hoc networks.

In this paper, we describe the multicast tree construction. We point out that the number of packet forwarding is the more important factor than the number of links in the ad hoc network. After we show constructing minimum cost multicast tree is hard, we propose two new flooding methods: *self pruning* and *dominant pruning* which improve the existing flooding method. Proposed methods adopt different approaches compared with [21] in that topological information is utilized to reduce redundant broadcasts. Self pruning uses direct neighborhood information while dominant pruning uses extended neighborhood information. Our methods do not rely on any link-layer assumptions and can be applied to ad hoc networks in parallel with other methods such as [21]. Our performance study shows that the new flooding methods can reduce the number of transmissions significantly.

This paper is organized as follows. We describe the multicast tree construction in section 2. Then we propose two efficient flooding methods in section 3. Simulation results are shown in section 4 and section 5 concludes the paper.

2 Optimal multicast tree in ad hoc networks

2.1 Graph model

Like a wired network, an ad hoc network can be modeled as a graph. Each mobile terminal is represented as a vertex and two nodes are connected by a link when they are within the transmission range of each other.

In a wireless environment, some links may be unidirectional because transmission ranges of two nodes may be different. However, these unidirectional links can be hidden to the network layer protocol by using link layer ACK packets or unidirectional link detection mechanisms. Therefore, we assume that all links are bidirectional, that is if node v_i can communicate with v_j, then v_j can also communicate with v_i.

We define $N(v)$ as the set of adjacent nodes of node v. $N(N(v))$ is defined as the set of nodes that is at most two-hops apart from node v.

2.2 Optimal multicast tree

We should first determine the parameters that we should optimize in order to construct optimal multicast tree. In this paper, we consider following 3 parameters.

- The delay to send a packet to each destination
- The number of nodes that is concerned in multicast
- The number of forwarding nodes

To minimize the delay to send a packet to each destination, we should construct shortest path tree. Shortest path tree can be constructed in polynomial time and implemented easily. Figure 2.(a) shows the example of the shortest path multicast tree. In the figure, A is the sender, and B, C, F are destinations. The bold lines represent the multicast tree.

In order to minimize the number of nodes that is concerned in multicast, we should construct Steiner tree. Steiner tree problem is to find the connected subtree T of a given graph $G(V, E)$ with the set $Z \subset V$, that has the minimum cost and includes all elements in Z. If we find the Steiner tree in the graph by defining Z as the set of source node and destination nodes, this

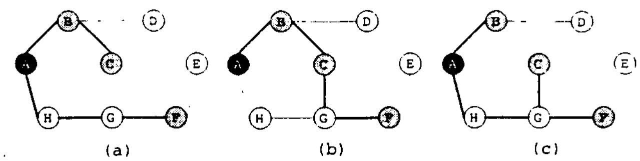

Figure 2. Optimal multicast trees in an ad hoc network

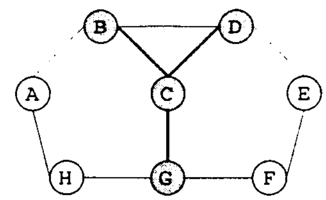

Figure 3. Minimum Connected Dominating Set

tree becomes multicast tree because it connects source node and all destination nodes. The cost of the Steiner tree is less than the number of nodes in Steiner tree by 1, since the cost of all nodes is equally 1. Therefore the Steiner tree becomes the tree that minimizes the number of nodes concerned in multicast. Figure 2.(b) shows the Steiner multicast tree. In the figure (b), the number of nodes concerned in multicast is 5(A, B, C, F, G). In the figure (a) and (c), the number of nodes concerned in multicast is 6 (A, B, C, F, G, H).

It has been proved that Steiner tree problem is NP-complete even when costs of all links are same [18]. Therefore we can know that minimizing the number of nodes concerned in multicast is difficult.

The number of forwarding nodes is the most important parameter among the parameters that we want to optimize. As explained above, a packet transmitted by a node is broadcast to all its neighbor nodes. Because each broadcast consumes the same amount of wireless resources regardless of the number of neighbor nodes, it is crucial to reduce the number of send operations of the mobile hosts.

All the nodes but the leaf nodes in the multicast tree should forward the packet to other nodes. The leaf nodes need only receive the packet. So the number of nodes that should forward is equal to the number of non-leaf nodes in the multicast tree. That is, the problem of minimizing the number of packet forwarding is equivalent to the problem of finding the multicast tree that minimizes the number of non-leaf nodes.

Figure 2.(c) shows the multicast tree that optimizes the number of packet forwarding. The number of packet forwarding is 4 (A, B, H, G) in (a), 4 (A, B, C, G) in (b) and 3 (A, H, G) in (c).

2.3 Broadcast tree

Multicasting to all nodes in an ad hoc network is equivalent to broadcast. That is, broadcast is the special case of multicast. In this subsection, we show that constructing multicast tree minimizing the number of packet forwarding is hard, by showing that constructing broadcast tree minimizing the number of packet forwarding is hard.

The problem of constructing broadcast tree that minimizes the number of packet forwarding is very similar to the MCDS (Minimum Connected Dominating Set) problem [16]. MCDS problem is to find the minimum connected subset S of V in which all elements in $V - S$ is adjacent to at least one element of S, given graph $G(V, E)$. For example, MCDS of the graph in figure 3 is $\{B, C, D, G\}$. All the nodes but $\{B, C, D, G\}$ is linked to at least on element of $\{B, C, D, G\}$, and $\{B, C, D, G\}$ is the connected set. Since we can find no connected dominating set whose size is less than 4, $\{B, C, D, G\}$ is MCDS.

For a given graph, if MCDS includes the source node, the number of packet forwarding is equal to the size of MCDS. Otherwise, the number of packet forwarding is size of MCDS plus 1 since source node should send a packet to one node in MCDS. For example, if B is the source node, the four nodes B, C, D, G need to forward a packet to neighboring nodes. If A is the source node, the five nodes A, B, C, D, G should forward a packet since A is not in MCDS.

As described above, problem of finding optimal broadcast tree is very similar to the MCDS problem. The optimal broadcast tree problem can be solved in similar way as the MCDS problem. Let's assume that the optimal broadcast tree problem can be solved in polynomial time. For a given graph $G(V, E)$ in which $|V|$ is n, we can make n optimal broadcast trees $T_1, T_2, \ldots, T_n$ by regarding all $v \in V$ as the source node. Then, the source node will be in MCDS for at least one broadcast tree $T_k (1 \leq k \leq n)$. Therefore the broadcast tree whose size is smallest among all T_k is MCDS. It means that we can find MCDS in polynomial time if we can find optimal broadcast tree in polynomial time. In other words, we cannot find the optimal broadcast tree in polynomial time if we cannot find MCDS in polynomial time. Accordingly, we can say that the optimal broadcast tree problem is harder than MCDS problem.

We can say that the optimal broadcast tree problem is hard since the MCDS problem has been proved to be NP-complete [17]. The optimal multicast tree problem in the ad hoc network is also a hard problem, since the special case problem, the optimal broadcast tree problem is hard to solve.

2.4 Approximation algorithm

Since it is difficult to find an optimal broadcast tree, we should use an approximation algorithm. One of efficient approximation algorithms for the MCDS problem is Berman's algorithm [5]. Berman's algorithm runs in two phases. In the first phase, the algorithm generates sets of connected nodes. In the second phase, the algorithm merges the sets of connected nodes into one connected dominating set. Guha et al. proved that the approximation ratio of Berman's algorithm is $\ln \Delta + 3$ where Δ is the maximum degree of the tree [5].

Using Berman's algorithm, we can find an efficient broadcast tree. A major drawback of Berman's algorithm is that it requires the global network topology information. In an ad hoc environment where nodes may move freely, it is impossible to gather global network topology information. Therefore it is difficult to utilize this algorithm in distributed systems. In this paper, we regard the performance of Berman's algorithm as the upper bound.

In the next section, we propose two new heuristic algorithms that can compute broadcast trees in a distributed manner.

3 Heuristic flooding algorithms

The most common broadcasting method is blind flooding where each node broadcasts a packet to its neighbors whenever it receives the packet along the shortest path from the source node. The broadcast tree constructed by this method becomes the shortest path tree. Blind flooding is simple and relatively efficient if it is used in wired networks. However, blind flooding could waste wireless resources considerably and may not be suitable in wireless network. Consider an example shown in figure 4 where node A floods a packet. Nodes B, C and D all receive a packet from A. If blind flooding is used, nodes B, C and D all forward the received packet unnecessarily and waste limited wireless bandwidth.

In this section, we propose two heuristic algorithms that may flood packets more efficiently than blind flooding in wireless networks. These algorithms are *self pruning* and *dominant pruning*. These algorithms are simple and easy to implement in a distributed manner. These methods reduce unnecessary transmissions by utilizing the neighborhood information exchanged between mobile nodes.

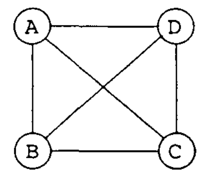

Figure 4. The example of the ad hoc network

3.1 Self pruning

In self pruning, each node exchanges the list of its adjacent nodes with neighbors. To use this technique, we should assume that all nodes know the adjacent nodes. This assumption is not unrealistic as most ad hoc routing protocols assume that every node knows the adjacent nodes [1, 3, 4, 8]. Each node emits 'Who I am' packet periodically to inform its existence to the neighbor nodes.

Node v_i, who wishes to forward a packet, piggybacks the adjacent node list $N(v_i)$ in the flooded packet. Node v_j who receives the packet checks whether the set $N(v_j) - N(v_i) - \{v_i\}$ is empty. If it is empty, node v_j refrains from forwarding the packet because it knows that all its adjacent nodes must have received the packet when node v_i forwarded the packet. Otherwise, node v_j forwards the packet. In figure 4 where all nodes are directly connected, only one transmission is required by self pruning, while blind flooding uses four transmissions.

To decide whether to forward a received packet or not, v_j should iterate for all $v \in N(v_i)$ finding and removing v from $N(v_j) - \{v_i\}$. If all elements are removed from $N(v_j) - \{v_i\}$, v_j doesn't forward the packet. Otherwise, v_j forwards the packet. Therefore, the time complexity of the self pruning method is $O(\Delta)$ where Δ is the maximum degree of the tree.

We expect that the effect of self pruning realizes most significantly in the perimeter of networks. The nodes in the center is more likely to have non-overlapping neighbor nodes than the nodes in the perimeter.

Self pruning requires extra transmission overhead of exchanging neighborhood information. To reduce the overhead, each node can store the received adjacent node list in their cache. A node does not piggyback the adjacent node list if the neighborhood information

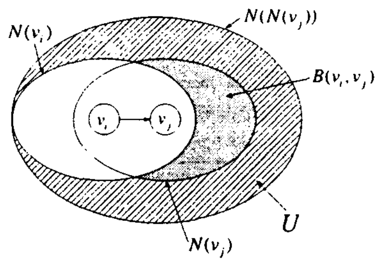

Figure 5. Dominant pruning method

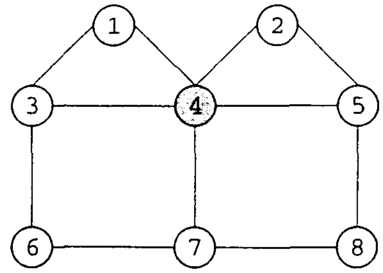

Figure 6. The example of flooding in the ad hoc network

does not change after the last transmission. The node that receives the packet without the adjacent node list uses the cached adjacent list. In addition to neighborhood information caching, we may be able to reduce the overhead further by reporting the difference of neighborhood information only.

3.2 Dominant pruning

While self pruning exploits the knowledge of directly connected neighbors only, dominant pruning extends the range of neighborhood information into two-hop apart nodes. This two-hop neighborhood knowledge can be obtained by exchanging the adjacent node lists with neighbors. Dominant pruning should perform better than self pruning because it is based on extended knowledge.

Another point in which dominant pruning differs from self pruning is the routing decision point. In self pruning, a node that receives a packet decides by itself whether it forwards the packet or not. In dominant pruning, the sending node selects adjacent nodes that should relay the packet to complete broadcast. The IDs of selected adjacent nodes are recoded in the packet as a *forward list*. An adjacent node that is requested to relay the packet again determines the forward list. This process is iterated until broadcast is completed.

Let us examine how each node determines the forward list. Suppose node v_j receives a packet from v_i and v_j is in the forward list. Node v_j should determine its own forward list so that all nodes within two-hop distance from v_j receive the packet. The forward list should be minimized to decrease the number of transmissions. Among nodes in $N(N(v_j))$, v_i, v_j, $N(v_i)$ have already received the packet, and $N(v_j)$ will receive the packet when v_j forwards the packet. Therefore a node v_j determines its forward list so that all nodes in $U = N(N(v_j)) - N(v_i) - N(v_j)$ receive the packet[1]. Figure 5 shows the set U. Let $B(v_i, v_j) = N(v_j) - N(v_i)$. Then we select a set of nodes $F = \{f_1, f_2, \dots, f_m\} \subseteq B(v_i, v_j)$ such that $\bigcup_{f_i \in F} (N(f_i) \cap U) = U$.

[1] If v_j is the source node, then we let $N(v_i) = \emptyset$.

Finding a minimum F is the set cover problem which is NP-complete [9]. Thus, we use approximation algorithms to determine the forward list. In this paper, we use ***greedy set cover*** algorithm [14]. The algorithm which finds the set F is as follows:

1. Let $F = \emptyset$, $K = \{S_1, S_2, \cdots, S_n\}$ where $S_k = N(v_k) \cap U (1 \le k \le n)$, $Z = \emptyset$.

2. Find the set S_k whose size is maximal in a set K.

3. $F = F \cup \{v_k\}$, $Z = Z \cup S_k$, $K = K - \{S_k\}$, $S_l = S_l - S_k$ for all $S_l \in K$.

4. If $Z = U$, complete the algorithm.

5. Otherwise, repeat from 2 again.

This algorithm repeats selecting v_k in which the number of neighbor nodes that is not coverèd yet is maximum. It has been proved that this approximation algorithm has the approximation ratio of $(\ln |U| + 1)$ [14]. It has also been proved that this algorithm can be implemented to run in $O(|U||K| \min(|U|, |K|))$ time [10].

Let's explain self pruning and dominant pruning with an example shown in figure 6. In the figure, node 4 is the source node. Blind flooding needs eight packet forwarding because all the nodes that receive the packet should forward the packet. By using self pruning, the nodes 1 and 2 need not forward the packet. So total number of packet forward is six.

Now, consider the dominant pruning method. Node 4 should determine the forward list among the neighboring nodes. In the example, $N(N(4)) = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Among these nodes, $N(4) = \{1, 2, 3, 4, 5, 7\}$ receives the packet directly from node

4. We should determine the forward list such that $N(N(4)) - N(4) = \{6, 8\}$ are covered. The optimal forward list would be $\{7\}$. Based on the same method, we can determine the forward list of node 7 to be $\emptyset$. Consequently, only 2 nodes 4, 7 need to forward the packet in order that all the nodes get the packet.

Routing decision should balance the overhead of information collection and the efficiency of smart routing. A node can make better routing decision if it has more extensive network information. However, network information is obtained at the expense of transmission overhead. Also, in case of broadcast in a wireless network, the complexity of routing decision increases rapidly as more extensive information is available. We limit our study to maximum two-hop apart neighbor information.

4 Simulation results

We measure the performance of the proposed flooding methods using computer simulation. We assume that wireless hosts are scattered randomly in a unit square. There are n nodes in the simulated network and nodes within r distance apart can communicate each other. Among those n nodes, one node is randomly chosen as the source node which initiates flooding. In the performance study, we vary n and r to analyze how the network size and connectivity affect the performance. We performed 10000 simulations and computed average number of packet forwarding and packet arrivals required to complete broadcast for each n and r.

To show the relative performance advantage of two proposed algorithms more clearly, we first study the worst and best performance that can be achieved practically. The proposed methods should perform better than blind flooding which does not use network topology information. Therefore, we use the performance of blind flooding as the lower performance bound. We obtain the best performance when we find the optimal broadcast tree based on complete network topology information. However, because it is very difficult to achieve the theoretical best performance even if complete network information is available, the performance of best heuristic algorithm that is run on complete network information can be regarded as the best performance. In this paper, the performance of Berman's algorithm is used as the upper bound.

Figure 7 shows the number of packet transmissions required to complete broadcast by blind flooding, self pruning, dominant pruning and Berman's algorithm when n is 10. X-axis represents the value of r and Y-axis represents the average number of packet forwarding required to complete broadcast. We omit the simulation result when r is smaller than 0.65 because the generated graph is partitioned when r is smaller than 0.65. Both proposed methods perform better than blind flooding. Dominant pruning outperforms self pruning and shows the performance near that of Berman's algorithm. Except for the blind flooding, the number of packet forwarding decreases as r increases. We can conclude that the performance improvement by the proposed algorithm becomes more significant as the network connectivity increases.

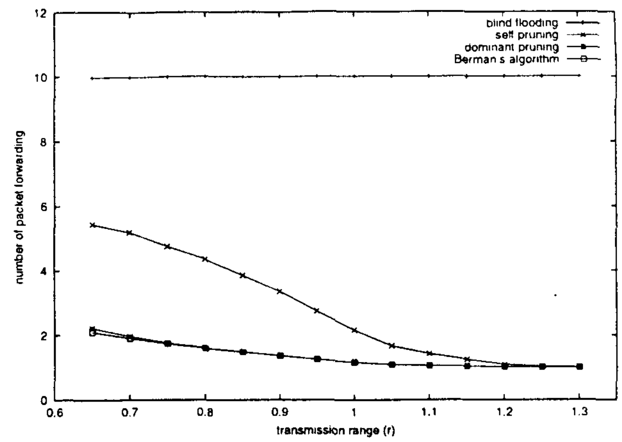

Figure 7. The number of packet forwarding when n is 10

Figure 8 is the graph that shows the average number of packet forwarding when n is 100. We observe the similar performance pattern as the $n = 10$ case. However, the performance difference of dominant pruning and self pruning is larger than $n = 10$ case. We can also observe that the performance of the dominant pruning method is near that of the Berman's algorithm even when the network size is large.

Figure 9 shows the total number of packets that nodes in an ad hoc network receive. This performance measure is important because each packet arrival requires packet processing overhead at mobile terminals. In blind flooding, the number of packet arrivals increase as r increases and finally reaches 90. In this figure, we can again observe that the dominant pruning performs significantly better than self pruning and its performance is near that of Berman's algorithm.

Figure 10 shows the average number of packet arrivals when n is 100. This graph is similar to figure 9. We can observe that the performance improvement of the dominant pruning method increases as the network becomes larger.

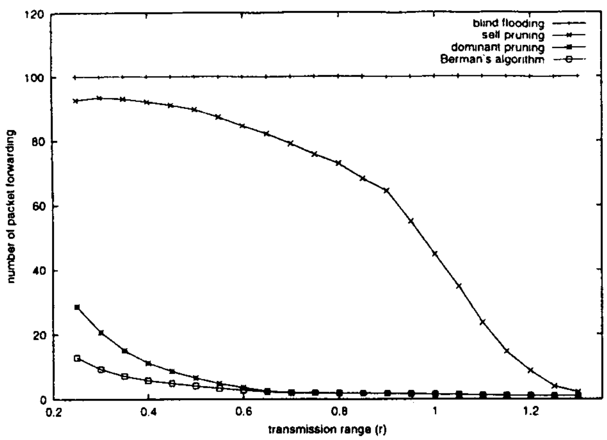

Figure 8. The number of packet forwarding when n is 100

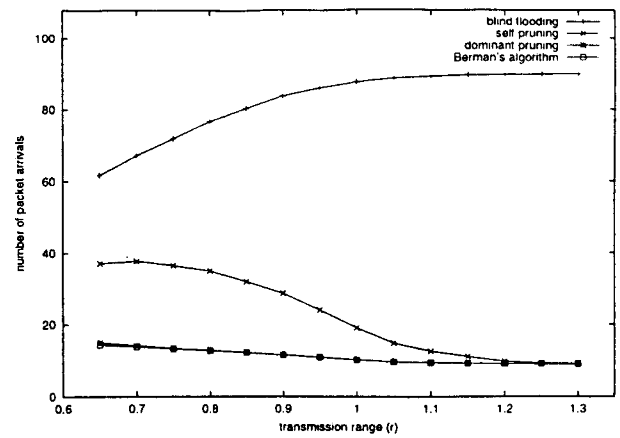

Figure 9. The number of packet arrivals when n is 10

Based on the simulation results, we can conclude that the performance of both proposed methods is good when the graph is near complete. However, performance of self pruning is far worse than that of Berman's algorithm in moderately connected large networks. The reason is that the performance gain of the self pruning can be realized only in the perimeter of the ad hoc network. The dominant pruning shows the performance near that of Berman's approximation algorithm independent of the size and the connectivity of network. Even though dominant pruning uses only limited network information, it achieves the near-optimal performance.

5 Conclusions

In this paper, we have described the optimal multicast tree in the ad hoc network and proposed two new flooding methods that can improve the performance of the classic flooding method. We have shown that we can construct various multicast trees depending on the parameter that we want to optimize, and that constructing multicast tree which minimizes the total number of packet forwarding is hard. We have also proposed two heuristic flooding methods that may reduce the number of transmissions.

Self pruning tries to reduce the flooding cost, utilizing neighborhood information. The neighborhood information can be piggybacked in packets exchanged between neighbor nodes. Simulation results show that this method performs better than blind flooding.

Dominant pruning uses extended neighborhood information. While self pruning uses direct neighbor information only, dominant pruning uses neighborhood information up to two hops apart. Based on extended neighborhood information, each node decides the forward list for the next transmissions on the broadcast tree. The performance gain of dominant pruning is greater than that of self pruning. However, dominant pruning has larger overhead than self pruning and the overhead increases as the host mobility increases. Thus, self pruning method could be more appropriate when the mobility of the host is high and the network is small. In contrast, the dominant pruning method could be a method of choice when the mobility is moderate and the network is large.

We plan to identify the effect of mobility on the proposed flooding methods in future works. We will also generalize the heuristic flooding methods so that they can be applied to multicast in ad hoc wireless networks.

Acknowledgments

This work was supported by the Brain Korea 21 Project.

References

[1] V. Park, S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, INFOCOM 97 (1997) 1407-1415.

[2] S. Basagni, D. Bruschi, I. Chlamtac, A mobility-transparent deterministic broadcast mechanism

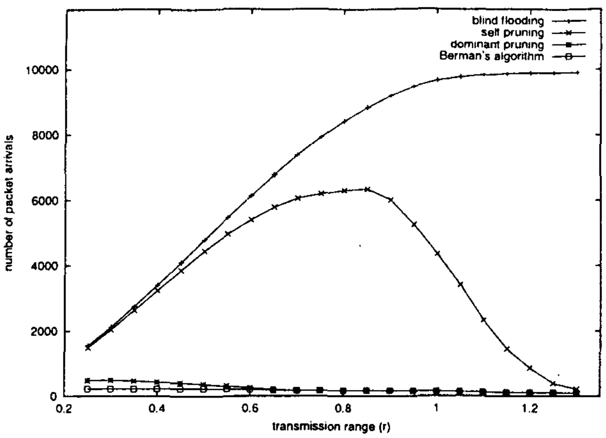

Figure 10. The number of packet arrivals when n is 100

for ad hoc network, IEEE/ACM Transactions on Networking 7 (6) (1999) 799-807.

[3] Z. J. Haas, A new routing protocol for the reconfigurable wireless networks, ICUPC 97 (1997) 562-566.

[4] C. Perkins, E. M. Royer, Ad-hoc on-demand distance vector routing, 2nd IEEE Workshop on Mobile Computing Systems and Applications (1999) 3-12.

[5] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (4) (1998) 374-387.

[6] P. Sinha, R. Sivakumar, V. Bharghavan, CEDAR: A core-extraction distributed ad hoc routing algorithm, INFOCOM 99 (1999) 202-209.

[7] A. S. Tanenbaum, Computer Networks: Third edition (Prentice Hall, 1996) 351-352.

[8] D. B. Johnson, D. A. Maltz, Dynamic source routing in ad hoc wireless networks, Mobile Computing, Academic Publishers (1996) 153-181.

[9] S. Even, Graph algorithms (Computer Science Press, 1979) 204-209.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, Introduction to Algorithms (The MIT Press, 1990) 974-978.

[11] E. M. Royer, C. Perkins, Multicast operation of the ad-hoc on-demand distance vector routing protocol, Mobicom 99 (1999) 207-218.

[12] I. Chlamtac, S. Kutten, On broadcasting in radio networks-problem analysis and protocol design, IEEE Transactions on Communications 33 (12) (1985) 1240-1246.

[13] S. Lee, M. Gerla, C.-C. Chiang, On-demand multicast routing protocol, WCNC 99 (1999) 1298-1302.

[14] L. Lovasz, On the ratio of optimal integral and fractional covers, Discrete mathematics 13 (1975) 383-390.

[15] R. Bar-Yehyda, O. Goldreich, A. Itali, On the time-complexity of broadcast in multihop radio networks, SIAM Journal on Computing 22 (4) (1993) 876-887.

[16] C. J. Colbourn, L. K. Stewart, Permutation graphs: connected domination and Steiner trees, Discrete Mathematics 86 (1990), 179-189.

[17] D. Lichtenstein, Planar formulae and their uses, SIAM Journal on Computing 11 (2) (1982) 329-343.

[18] R. M. Karp, Reducibility among combinatorial problems, Complexity of computer computations (Plenum Press, 1972) 85-104.

[19] E. Pagani, G. P. Rossi, Reliable broadcast in mobile multihop packet networks, Mobicom 97 (1997) 34-42.

[20] K. Bharath-Kumar, J. M. Jaffe, Routing to multiple destinations in computer networks, IEEE Transactions on Communications 31 (3) (1983) 343-351.

[21] S. Ni, Y. Tseung, Y. Chen, J. Sheu, The broadcast storm problem in a mobile ad hoc network, Mobicom 99 (1999) 151-162.

[22] I. Chlamtac, O. Weinstein, The wave expansion approach to broadcasting in multihop radio networks, IEEE Transactions on Communications 39 (3) (1991) 426-433.

[23] I. Chlamtac, S. Kutten, Tree-based broadcasting in multihop radio networks, IEEE Transactions on Computing 36 (10) (1987) 1209-1223.