Broadcasting in Ad Hoc Networks Based on Self-Pruning

Jie Wu and Fei Dai
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
E-mail: {jie,fdai}@cse.fau.edu

***Abstract*— We propose a general framework for broadcasting in ad hoc networks through self-pruning. The approach is based on selecting a small subset of hosts (also called nodes) to form a forward node set to carry out a broadcast process. Each node, upon receiving a broadcast packet, determines whether to forward the packet based on two neighborhood coverage conditions proposed in this paper. These coverage conditions depend on neighbor connectivity and history of visited nodes, and in general, resort to global network information. Using local information such as k-hop neighborhood information, the forward node set is selected through a distributed and local pruning process. The forward node set can be constructed and maintained through either a proactive process (i.e., "up-to-date") or a reactive process (i.e., "on-the-fly"). Several existing broadcast algorithms can be viewed as special cases of the coverage conditions with k-hop neighborhood information. Simulation results show that new algorithms, which are more efficient than existing ones, can be derived from the coverage conditions, and self-pruning based on 2- or 3-hop neighborhood information is relatively cost-effective.**

Index Terms—**- Ad hoc networks, broadcasting, localized algorithms, pruning.** [1]

I. Introduction

Recent advances in technology have provided portable computers with wireless interfaces that allow network communication among mobile users. The resulting computing environment, which is often referred to as mobile computing, no longer requires users to maintain a fixed and universally known position in the network and enables almost non-restricted mobility. It is argued that future wireless computing will be converged to be more ad hoc and reconfigurable [1]. An *ad hoc wireless network* (or simply *ad hoc network*) is a special type of wireless mobile network in which a collection of mobile hosts with wireless network interfaces form a temporary network, without the aid of any established infrastructure (i.e., base stations) or centralized administration (i.e., mobile switching centers). The applications of ad hoc networks range from civilian use to disaster recovery (search-and-rescue) and military use (battlefield).

Broadcasting is more frequent in ad hoc networks than in wired networks, especially as the basic vehicle for on-demand route discovery. Broadcasting in ad hoc networks poses more challenges than the one in wired networks for two reasons: node mobility and scarce system resources. Because of the diversity in node movement patterns, there is no single optimal scheme for all situations in ad hoc networks. In a low mobility environment, tree-based schemes such as *minimal connected dominating set* (MCDS) [2] are better in reducing resource consumption. In a high mobility environment, simple flooding is the only way to achieve the full coverage; that is, the broadcast packet is guaranteed to be received by every node in the network, providing there is no packet loss caused by collision in the MAC layer. Williams and Camp [3] divided broadcast techniques into four categories: simple flooding, probability-based methods, area-based methods, and neighbor-knowledge-based methods. When a packet is broadcast via simple flooding, it is forwarded by every node in the network exactly once. Simple flooding ensures the coverage, but it also has the largest *forward node set* and may cause network congestion and collision. Probability- and area-based methods [4] are proposed to solve the so-called *broadcast storm problem*. In these schemes, each node will estimate its potential contribution to the overall broadcasting before forwarding a broadcast packet. If the estimated contribution is lower than a given threshold, it will not forward the packet. These methods generate smaller forward node sets than simple flooding. However, the estimation methods are inaccurate and cannot ensure the full coverage.

Neighbor-knowledge-based methods are based on the following idea: To avoid flooding the whole network, a small set of *forward nodes* is selected. Basically, the forward node set forms a *connected dominating set* (CDS). A node set is a dominating set if every node in the network is either in the set or the neighbor of a node in the set. The challenge is to select a small set of forward nodes in the absence of global network information. It has been proved that finding the smallest set of forward nodes with global network information is NP-hard. In the absence of global network information, this problem is even more challenging. Heuristic methods are normally used to balance cost (in collecting network information and in decision making) and effectiveness (in deriving a small connected dominating set).

Neighbor-knowledge-based algorithms can be further divided into *neighbor-designating* methods and *self-pruning*

[1]This work was supported in part by NSF grant CCR 9900646 and grant ANI 0073736.

methods. In neighbor-designating methods [5], [6], [7], [8], the forwarding status of each node is determined by its neighbors. Basically, the source node selects a subset of its 1-hop neighbors as forward nodes to cover its 2-hop neighbors. This forward node list is piggybacked in the broadcast packet. Each forward node in turn designates its own forward node list. Most neighbor-designating methods use similar heuristics. In *multipoint relaying* [8], the complete 2-hop neighbor set shall be covered, since it is independent of any particular broadcasting. In *dominant pruning* [5], only a partial 2-hop neighbor set shall be covered by taking the advantage of routing history information; nodes that are also the 1-hop neighbors of the last visited node are excluded in the current coverage. This is also the case in AHBP [7]. A more efficient algorithm is proposed recently by Lou and Wu [6], where not only the 1-hop neighbors but also some of the 2-hop neighbors of the last visited node are excluded from the current set to be covered. In self-pruning methods [9], [10], [11], [12], [13], [14], each node makes its local decision on forwarding status: forwarding or non-forwarding. Although these algorithms are based on similar ideas, this similarity is not recognized or discussed in depth. Fair comparison of these algorithms is complicated by the lack of in-depth understanding of the effect of the underlying mechanisms, such as neighborhood information collection, piggybacking routing history in broadcast packets, type of priority value to establish a total order among mobile hosts, etc.

We propose a generic scheme for broadcasting based on self-pruning. In this approach, each node, upon receiving a broadcast packet, determines whether to forward the packet based on a neighborhood coverage condition. Two novel coverage conditions are proposed in this paper and both will generate a connected dominating set. One condition will generate a smaller set than the other, but has a higher computation cost. Using local information such as k-hop neighborhood information for a small k, the forward node set is selected through a distributed and local pruning process. The forward node set can be constructed and maintained through either a proactive process (i.e., "up-to-date") or a reactive process (i.e., "on-the-fly"). Note that in a reactive process, the decision at each node can be postponed so that it has higher chance of becoming a non-forward node by overhearing its neighbors' forwarding activities. Different implementations of self-pruning based on k-hop neighborhood information are discussed, and their performances are compared through simulation. The proposed scheme provides a general framework that includes several existing broadcast protocols. In addition, the proposed framework is more powerful than any of these protocols, which is confirmed by simulation results. The simulation study also shows that the coverage conditions achieve good balance between performance and overhead with 2- or 3-hop neighborhood information, which happens to be the settings of most existing algorithms. In this paper, we assume target networks with moderate mobility, where near-to-accurate normal k-hop information can be maintained with affordable cost, especially for a small k.

The rest of this paper is organized as follows: Section 2 introduces our general self-pruning algorithm based on two neighborhood coverage conditions. Section 3 reviews several existing broadcast algorithms as the special cases of this general algorithm. Section 4 compares the performance of different broadcast algorithms via simulation, and Section 5 concludes the paper.

II. Broadcasting Through Self-Pruning

For broadcasting based on self-pruning, each node may determine its own status as a forward node or non-forward node (1) before a broadcast packet is received [9], [10], [14], (2) after the first copy of a broadcast packet is received [13], or (3) after several copies of a broadcast packet are received [11], [12]. Algorithms in category (1) produce a relatively stable forward node set and can also be used in unicasting and multicasting. However, by neglecting the routing history, they also produce the largest forward node set among the three. Algorithms in category (2) can produce a smaller forward node set than algorithms in category (1) by considering the routing history. Algorithms in category (3) can further reduce the size of a forward node set at the expense of longer end-to-end delay. In the following discussion, we assume that each node can determine its own status at any time.

A. Neighbor set coverage and coverage conditions

Here we propose a simple distributed heuristic approach to determine a small connected dominating set used as the forward node set. Two approaches can be adopted: In the static approach, a connected dominating set is constructed based on the network topology, but irrelative to any broadcasting. In the dynamic approach, a connected dominating set is constructed for a particular broadcast request, and it is dependent on the *location of the source and the progress of the broadcast process*. We assume that in the dynamic approach, each node v determines its status "on-the-fly" when the broadcast packet arrives at the node. We also assume that the broadcast packet that arrives at v carries information of h most recently visited nodes for a small h and the corresponding node set is denoted as $D(v)$. This assumption does not lose any generality, since we can assume h to be 0 when the packet does not carry any routing history information.

In the proposed self-pruning scheme, each node decides its own status (forward-ing/non-forwarding) independently based on the following condition in the static approach.

Coverage Condition I (static):

Node v has a non-forward node status if for any two neighbors u and w, a *replacement path* exists that connects u and w via several intermediate nodes (if any) with higher priority values than the priority value of v.

Note that "replacement" can be applied iteratively. To avoid possible "cyclic dependency" situations, a total order is defined a priori among nodes. A simple solution is to use node id to define the total order, although other measures such as

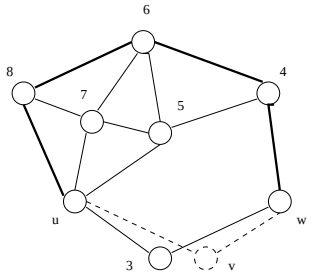


Fig. 1. A sample maximal replacement path.

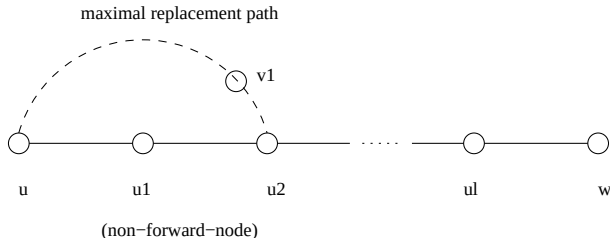


Fig. 2. Maximal replacement path for u_1.

node degree can also be adopted. Note that intermediate nodes may not exist. In this case, u and w are directly connected. In a formal term, assume that v is a non-forward node. Let $N(v)$ be the neighbor set of node v, then for any $u, w \in N(v)$, a replacement path $(u, u_1, u_2, ..., u_l, w)$ exists such that $id(u_i) > id(v)$ for $1 \leq i \leq l$.

Next we define a special replacement path, called *maximal replacement path*, such that all intermediate nodes (if any) are forward nodes. That is, none of the nodes in the maximal replacement path can be replaced.

Definition 1: Max-min node for (u, w, v): A *minimum node* in a path is a node with the lowest priority. Assume $\{P_i\}$ is the set of replacement paths for node v that connect u and w. A *max-min node* in $\{P_i\}$ is a node with the highest priority among all the minimum nodes in $\{P_i\}$.

Next we define a procedure called MAXMIN to construct a maximal replacement path for v that connects u and w.

MAXMIN(u, w, v):

1: **if** u and w are directly connected **then return** $\emptyset$.
2: Find the max-min node x for (u, w, v).
3: **return** path $(\text{MAXMIN}(u, x, v), x, \text{MAXMIN}(x, w, v))$.

Lemma 1: The procedure MAXMIN(u, w, v) will complete in a finite number of steps and generate a replacement path. In addition, any node in the path cannot be further replaced.

Proof: If nodes u and x (and nodes x and w) are not directly connected, the max-min node for (u, x, v) (and the one for (x, w, v)) has a higher priority than the priority of x, because there exists at least one path from u to x (and from x to w) via intermediate nodes with higher priorities than the priority of x. Similarly, all nodes selected by MAXMIN(u, x, v) (and MAXMIN(x, w, v)) have higher priorities than the priority of x. That is, x does not appear in either MAXMIN(u, x, v) or MAXMIN(x, w, v).

To show that MAXMIN(u, x, v) and MAXMIN(x, w, v) have no common element, we assume that MAXMIN$(u, x, v) = u_1, u_2, ..., u_m$ and MAXMIN$(x, w, v) = x_1, x_2, ..., x_n$. Suppose $u_i = x_j$, then $(u, u_1, ..., u_i, x_{j+1}, ..., x_n, w)$ is a replacement path for v that connects u and w. The fact that all nodes in this path have a higher priority than x contradicts to the fact that x is a max-min node. Since each recursive call of the max-min procedure selects a distinct node, this process will complete in finite steps.

Next we show that x cannot be replaced. If x is is replaced by path P, then (MAXMIN$(u, x, v), P,$ MAXMIN(x, w, v)) is another replacement path for v that connects u and w (if it is a walk with multiple occurrences of a node, multiple occurrences can be easily removed to form a path). Clearly, all the nodes in this path has a higher priority than x which contradicts to the fact that x is a max-min node. ■

Figure 1 shows a sample maximal replacement path constructed from the MAX-MIN procedure by including u and w at the two ends. In this example, v has a priority of 2. We use node id's as node priorities. Nodes with priorities lower than v are not shown. Node 4 is the max-min node for (u, w, v). Node 6 is the max-min node for $(u, 4, v)$ and node 8 is the max-min node for $(u, 6, v)$. Therefore, the maximal replacement path is $(u, 8, 6, 4, v)$.

Theorem 1: Given a graph $G = (V, E)$ that is connected but not a complete graph, the vertex subset V', derived based on coverage condition I, forms a connected dominating set of G.

Proof: We first show that V' forms a dominating set. Randomly select a vertex v in V. We show that v is either in V' or adjacent to a vertex in V'. If v is a forward node, the theorem holds. For the remaining case, we will show that there exists a neighboring forward node. Since v is a non-forward node, for any two neighbors of v, there is a replacement path for v that connects these two neighbors. There exists at least one neighbor u of v such that there is $w \in N(u)$, but $w \notin N(v) \cup \{v\}$ (otherwise, G is a complete graph). Let u be such a neighbor with the highest priority. Clearly, there is no replacement path for u that connects v and w and, hence, u is a forward node.

Next we show that V' is connected. Randomly select two nodes u and w in V'. Assume that $(u, u_1, u_2, \ldots, u_l, w)$ is a path in G that connects u and w. If u_1 is a non-forward node, find a maximal replacement path for u_1 that connects u to u_2. Assume that v_1 is the last intermediate node of the maximal replacement path (if u and w are directly connected, v_1 is u itself). Repeat the above process on $(v_1, u_2, ..., u_l, w)$ to replace u_2 (see Figure 2). If u_2 is a forward node, u_2 is skipped and repeat the above process on $(u_2, ..., u_l, w)$. Eventually, $u_1, u_2, ...,$ and u_l are all replaced or skipped and the resultant path connects u and w with forward nodes only (if it is a walk with multiple occurrences of a node, multiple occurrences can be easily removed to form a path). ■

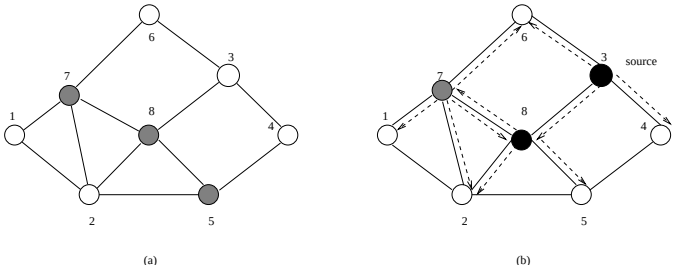


Fig. 3. (a) Forward node set without routing history (static). (b) Forward node set with routing history (dynamic) with node 3 being the source (visited node). Black nodes are visited nodes and gray nodes are forward nodes.

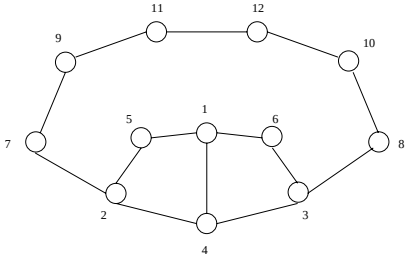


Fig. 4. Node 4 satisfies coverage condition I but not coverage condition II.

When the network is a complete graph, there is no need of forwarding node. One transmission from the source reaches all the nodes. We can extend the static version of coverage condition I to the dynamic version by including visited nodes (i.e., nodes that have forwarded the broadcast packet). By treating all visited nodes as a regular forward node with the highest priority (i.e., higher than all forward nodes), all the results for the static condition still holds for the dynamic condition. Note that the static condition is a special case of the dynamic condition (i.e., one without any visited node).

Coverage Condition I (dynamic):

Node v has a non-forward node status if for any two neighbors u and w, a *replacement path* exists that connects u and w via several intermediate nodes (if any) with either higher priority values than the priority value of v or with visited node status.

Figure 3 shows two examples of forward node set on the same network: one without routing history based on the static coverage condition 1 (Figure 3 (a)) and one with routing history based on the dynamic coverage condition 1 (Figure 3 (b)). In the example with routing history, it is assumed that the up-stream routing history is piggybacked with the broadcast packet. Because node 3 is a visited node, node 5 can conclude that it should be a non-forward node since any two neighbors can be connected using nodes 3 and 8. The forward node set derived from Figure 3 (a) can also be interpreted as one for any broadcasting without considering the location of source.

To check coverage condition I, each node needs to check every pair of its neighbors. There are $\Theta(\Delta^2)$ such pairs, where Δ is the maximum vertex degree in the network. In order to reduce the computation complexity at each node, we consider in the following another pruning method, called coverage condition II, and show that coverage condition I covers coverage condition II (i.e., coverage condition II is stronger than coverage condition I). Simulation results show that these two conditions are very close in reducing the number of forward nodes.

A set $C(v)$ is called a *coverage set* of v if the neighbor set of v can be "covered" by nodes in $C(v)$, i.e., $N(v) - C(v) \subseteq \cup_{u \in C(v)} N(u)$. In addition, nodes in $C(v)$ are either visited nodes or nodes with higher priorities than v's priority. Clearly, all nodes in $C(v)$ are within two hops of v. Note that $C(v)$ may include some neighbors of v.

Coverage Condition II (dynamic):

Node v has a non-forward node status if it has a coverage set. In addition, the coverage set belongs to a connected component of the subgraph induced from visited nodes and nodes with higher priority values than the priority value of v.

The above procedure can be formalized as follows: Denote $N_k(v)$ as the k-hop neighbor set of node v and $N_1(v)$, simply $N(v)$, is the neighbor set of v. $C(v)$ is the coverage set for v such that $C(v) \subseteq N_2(v)$ and for any $u \in C(v)$, either $id(u) > id(v)$ or $u \in D(v)$. For any $u, w \in C(v)$, there exists a path $(u, u_1, u_2, ..., u_l, w)$ such that either $id(u_i) > id(v)$ or $u_i \in D(v)$. Note that if $C(v)$ is a coverage node set then $C(v) \cup \{u\}$ is also a coverage node set provided $id(v) < id(u)$ or u is a visited node. Therefore, connecting nodes can be part of the coverage set and many coverage sets exist. Extending the coverage set beyond $N_2(v)$, we have a connected coverage set.

Theorem 2: Coverage condition II is stronger than coverage condition I.

Proof: When a node v satisfies coverage condition II, it also satisfies coverage condition I. Because the existence of a connected coverage set implies the existence of a replacement path for any two neighbors. ■

However, the reverse situation normally does not hold as shown in the example of Figure 4, where there is no connected coverage set of node 4. The following theorem shows that coverage condition I is more costly than coverage condition II.

Theorem 3: The computation complexity of coverage condition I is $O(n\Delta^2)$ at node v and that of coverage condition II is $O(n\Delta)$, where n is the number of visited nodes and nodes with higher priority values than v.

Proof: Let v be the current node, and $G^{'}(v)$ be the subgraph of G induced from visited nodes and nodes with higher priority values than v. For coverage condition I, first

decompose $G^{'}(v)$ into connected components $V_1, V_2, \ldots, V_l$ (via depth-first search with a cost of $O(n\Delta)$). For each component V_i, compute the set of covered neighbors $N(V_i) = \bigcup_{w \in V_i}(N(w) \cap N(v))$ ($O(n\Delta)$). After the construction of the new graph $G^{''}(v) = (V^{''}(v), E^{''}(v))$, where $V^{''}(v) = N(v)$ and $E^{''}(v) = \bigcup_{i=1}^{l}(N(V_i) \times N(V_i))$ ($O(n\Delta^2)$), since $(u, w) \in E^{''}(v)$ iff a replacement path exists that connects nodes u and w, coverage condition I is equivalent to "$G^{''}(v)$ is a complete graph" ($O(\Delta^2)$). The overall complexity is $O(n\Delta^2)$.

For coverage condition II, decompose $G^{'}(v)$ and compute $N(V_i)$ $(i = 1, 2, \ldots, l)$ in the same way as for coverage condition I ($O(n\Delta)$), then check if there exists a V_i such that $N(v) \subseteq N(V_i)$ ($O(n\Delta)$). The overall complexity is $O(n\Delta)$. ■

B. Neighborhood information

$N_k(v)$, k-hop neighbor set of node v, deserves more discussion. We assume that v knows neighbor set $N_1(v)$ (or $N(v)$), but not connections of nodes in the set. k-hop neighbor set ($N_k(v)$) is collected by neighbor set distribution from nodes within $k-1$ hops. In this case, node v knows connections of all nodes within $k-1$ hops, but only partial knowledge of connections of nodes in k hops. In fact, only the connections between nodes in $k-1$ hops and nodes in k hops are known at v.

Coverage Condition I (k-hop approximation):

Node v has a non-forward node status if for any two neighbors u and w, a *replacement path* exists that connects u and w via several intermediate nodes (if any) in $N_k(v)$ with either higher priorities than the priority of v or with the visited node status.

Coverage Condition II (k-hop approximation):

Node v has a non-forward node status if it has a coverage set. In addition, the coverage set belongs to a connected component of the subgraph induced from visited nodes and nodes with higher priorities than v's priority in $N_k(v)$.

Consider again the implementation of coverage condition II, we argue that $N_2(v)$ is sufficient to decide a coverage set. Although the complete neighbor set of a coverage node that is 2-hop away is missing, $N_2(v)$ contains all connections that exist between v's 1-hop and 2-hop neighbors. Partial neighbor set is sufficient in this case for neighborhood coverage. Obviously, $N_2(v)$ is not sufficient to determine the connectivity condition for coverage set, and we will resort to $N_k(v)$, k-hop neighbor set as an approximation for global connectivity checking. For coverage condition I, each replacement path is constructed within $N_k(v)$ and $D(v)$.

Figure 5 shows self-pruning based on coverage condition II (k-hop approximation), where black nodes are coverage nodes for pruning candidate v. It is assumed that all coverage nodes and connecting nodes are either visited nodes or nodes with higher priorities than v's priority.

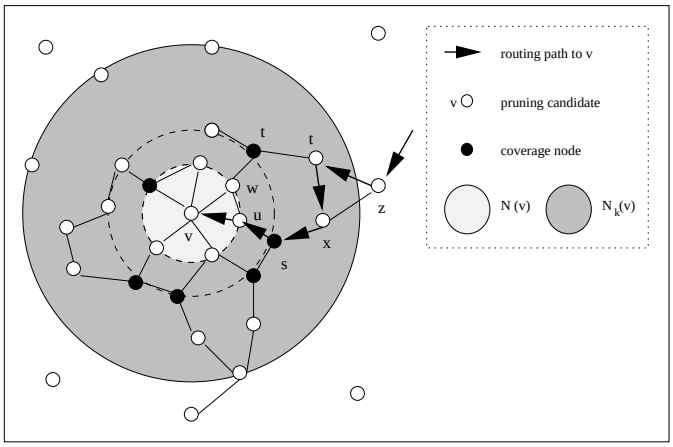


Fig. 5. Self-pruning based on k-hop neighborhood information.

By applying Theorem 3 on k-hop neighborhood, it is straightforward to prove the following theorem on the computation complexity of coverage conditions I and II with k-hop approximation.

Theorem 4: The computation complexity of coverage condition I with k-hop approximation is $O(k^2D^3)$ and that of coverage condition II with k-hop approximation is $O(k^2D^2)$, where D is the density of the network; that is, maximum number of nodes per unit area.

Proof: Considering the two parameters in Theorem 3, $\Delta \leq \pi r^2 D = cD$ and $n \leq \pi(kr)^2 D = ck^2 D$, where r is the transmitter range and $c = \pi r^2$ is a constant. That is, for coverage condition I, the complexity is $O(n\Delta^2) = O((ck^2D) \cdot (cD)^2) = O(k^2D^3)$ and for coverage condition II is $O(n\Delta) = O((ck^2D) \cdot (cD)) = O(k^2D^2)$. ■

Similarly, we conclude that the size of each control packet that is used to exchange $(k-1)$-hop information among neighbors is $O(n_{k-1}\Delta) = O(k^2D^2)$, where n_{k-1} is the number of nodes within $k-1$ hops. Obviously, the overhead is higher with larger D. Although appropriate density is necessary for network connectivity and redundancy, a very dense network is inefficient in a shared media access scheme because each node needs to contend with $O(D)$ neighbors for the limited bandwidth. On the other hand, the high density problem can be avoided by techniques such as adjustable transmitter range or clustering [15], [16] where a sparse graph is derived consists of cluster heads and selected connectors. Therefore, both computation time and packet size will be reasonably small.

III. Special Cases

Here we consider several existing algorithms in the general framework.

Simple flooding: Simple flooding can be viewed as a special case of either coverage condition I or condition II with 0-hop neighborhood information. Because each node has no

neighborhood information, both conditions I and II fail and no pruning can be accomplished. That is, every node is a forward node.

Wu and Li's algorithm: Wu and Li [14] proposed a *marking process* to determine a set of forward nodes (called *gateways*) that form a CDS: a node is marked as a gateway if it has two neighbors that are not directly connected. These gateways can be used as forward nodes in a broadcast process. For example, node v in Figure 6 (a) will not be marked as a gateway, because all its neighbors are directly connected with each other. Obviously, when a neighbor of v (e.g., node u) forwards a broadcast packet, it will be received by all other neighbors (e.g., node w). However, in Figures 6 (b) and 6 (c), node v will be marked as a gateway, because nodes u and w are not directly connected.

In Wu and Li's algorithm, two pruning rules are used together with the marking process to reduce the size of the resultant CDS. According to pruning Rule 1, a marked node (i.e., gateway) can be unmarked (i.e., become a non-gateway) if all of its neighbors are also neighbors of a coverage node, which can be a neighbor or a neighbor's neighbor, that has a higher priority value. For example, node v in Figure 6 (b) can be unmarked, because nodes w and u are covered by a gray node with higher priority than v. According to pruning Rule 2, a marked node can be unmarked if all of its neighbors are also neighbors of two connected coverage nodes that have higher priority values. For example, node v in Figure 6 (c) can be unmarked, because nodes u and w are covered by two gray nodes with higher priorities. Note that two types of priority can be used: node id and the combination of node degree and node id.

In order to implement the marking process and restricted versions of pruning rules where the coverage nodes are neighbors only, 2-hop information is collected at each node. That is, each node knows which nodes are its neighbors and neighbors' neighbors. The computation complexity is $\Theta(\Delta^2)$ for the marking process and $\Theta(\Delta^3)$ for pruning Rules 1 and 2. If the coverage nodes are neighbors' neighbors (the corresponding implementation is non-restricted), 3-hop information is collected at each node v, and the computation complexity of pruning Rules 1 and 2 becomes $\Theta(n_2\Delta^2)$, where n_2 is the number of nodes within 2 hops and have higher priorities than v. Note that this computation happens only once as long as the neighborhood topology remains the same. There is no extra computation for each incoming broadcast packet.

Wu and Li's algorithm is a special case of coverage condition II with 2- or 3-hop neighborhood information and no (i.e., 0-hop) routing history. Furthermore, the size of the coverage set is restricted to be less than or equal to 2.

Stojmenovic's algorithm: Stojmenovic et al [12] improved the Wu and Li's marking process in two ways: (1) By using geographic information, only 1-hop information is used to implement the marking process and Rules 1 and 2. That is, each node only maintains a list of its neighbors and their geographic positions. (2) The number of forward nodes

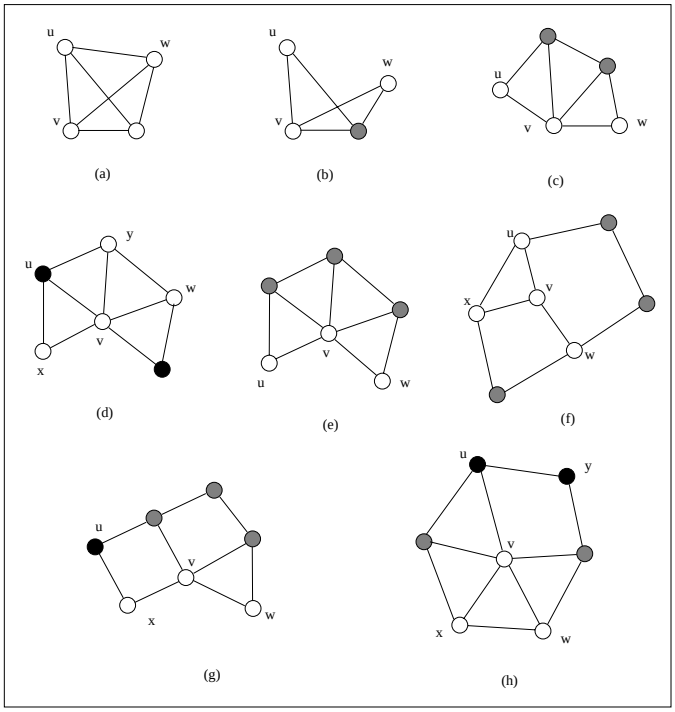


Fig. 6. Scenarios that different self-pruning methods apply: (a) the marking process, (b) Rule 1, (c) Rule 2, (d) neighbor elimination, (e) Rule k, (f) the enhanced Span, (g) LENWB, and (h) an example that the coverage conditions are more powerful than the existing methods. The black nodes are visited nodes and the gray nodes have higher priorities than v.

are further reduced by a *neighbor elimination* algorithm as follows: When a forward node v receives a broadcast packet, instead of forwarding the packet immediately, v will wait for a backoff delay and monitor the forwarding activities of its neighbors. For each neighbor u that has forwarded the broadcast packet, node v removes $N(u)$ from $N(v)$. If $N(v)$ is not empty after the delay period, node v forwards the broadcast packet; otherwise, node v becomes a non-forward node. For example, node v in Figure 6 (d) can be eliminated, because all its neighbors are covered by two visited nodes. Note that v cannot be unmarked by Rule 1 or Rule 2, since there is no coverage node.

The Stojmenovic's algorithm can still be viewed as a special case of coverage condition II, since all neighbors are covered by visited nodes and all visited nodes are connected (to the source).

Dai and Wu's algorithm: Dai and Wu [10] extended the marking process by using a more general pruning rule. According to this pruning Rule k, a marked node v can be unmarked if all of its neighbors are also neighbors of k connected coverage nodes that have higher priority values. Here k is not a exact value: it can be any positive integer. Rules 1 and 2 are special cases of Rule k where k is restricted to 1 and 2, respectively. For example, node v in Figure 6 (e) can be unmarked according to Rule k, because nodes u and v are covered by three nodes with higher priorities. Node v cannot

be unmarked by Rule 1 or 2, or the neighbor elimination process. An efficient algorithm based on depth-first search was proposed in [10] to implement the restricted version of Rule k, where coverage nodes must be neighbors. The computation complexity of the restricted Rules k is $\Theta(\Delta^2)$. Simulation results show that the restricted Rule k is almost as efficient as the non-restricted one in reducing the forward node set.

Clearly, Dai and Wu's algorithm is a special case of coverage condition II. The restricted version of Dai and Wu's algorithm is also a special case of coverage condition II with 2-hop neighborhood information and 0-hop routing history, as Wu and Li's algorithm is, except that there is no restriction on the size of coverage set.

Span: Chen et al [9] from MIT proposed the *Span* protocol to construct a set of forward nodes (also called *coordinators*). A node v becomes a coordinator if it has two neighbors that are not directly connected, indirectly connected via one intermediate coordinator, or indirectly connected via two intermediate coordinators. Before a node changes its status from non-coordinator to coordinator, it waits for a backoff delay which is computed from its energy level, node degree, and the number of pairs of its neighbors that are not directly connected.

Span cannot ensure the coverage since two coordinators may simultaneously change back to non-coordinators and the remaining coordinators may not form a CDS. To provide a fair comparison of Span and other broadcast algorithms that guarantee the full coverage, we use in this paper an enhanced version of Span. That is, a node becomes a coordinator if it has two neighbors that are not directly connected or indirectly connected via one or two intermediate nodes with higher priority values. For example, node v in Figure 6 (f) is a non-coordinator, because the three neighbors of v are either directly connected with each other, or connected via a node with a higher priority. The computation complexity for the enhanced algorithm is $\Theta(n_3\Delta^2)$ at node v, where n_3 is the number of nodes within 3 hops that have higher priority values than v. Note that node v in Figure 6 (f) cannot be pruned by pruning Rule k, because nodes u, w and x cannot be covered by a single connected coverage node set.

The enhanced Span algorithm is a special case of coverage condition I with 3-hop neighborhood information and 0-hop routing history.

LENWB: Sucec and Marsic [13] from Rutgers proposed the *lightweight and efficient network-wide broadcast* (LENWB) protocol, which computes the forward node status on-the-fly. Whenever node v receives a broadcast packet from a neighbor u, it computes the set C of nodes that are connected to u via nodes that have higher priority values than v. If $N(v)$ is contained in C, node v is a non-forward node; otherwise, it is a forward node. For example, node v in Figure 6 (g) is a non-forward node, because from the sender u, the other two neighbors x and w can be reached directly or via a node with a higher priority. Note that node v is a forward node in all previous algorithms. LENWB uses 2-hop neighborhood information and its computation complexity is $\Theta(n_2\Delta)$.

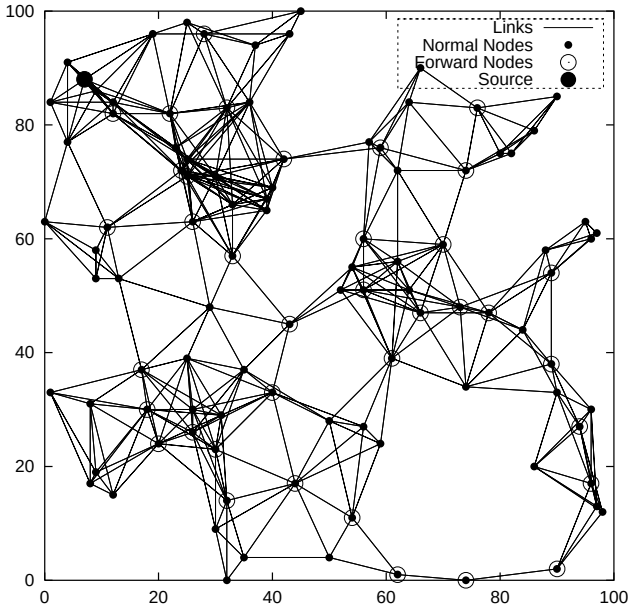


Fig. 7. A sample broadcasting generated by *ds* with 38 forward nodes and 62 non-forward nodes.

LENWB is a special case of coverage condition II with 2-hop neighborhood information. Unlike other algorithms, LENWB uses 1-hop routing history (i.e., $D(v)$ contains exactly one node where the broadcast packet comes from).

Although all the above algorithms can be viewed as special cases of coverage condition I or II, none of these algorithms exhausts the potential capability of these two conditions. For example, node v in Figure 6 (h) can be pruned according to coverage condition II, if 2-hop history information is piggybacked in the broadcast packet. In this case, visited node u carries the visited node status of y (assuming y has a lower priority than v). The two visited nodes and two nodes with higher priorities form a connected component that covers the other two neighbors x and w. Note that node v cannot be pruned by Rule k, enhanced Span or LENWB, since nodes u and w are not connected via nodes with higher priorities. The neighbor elimination process does not apply either, since nodes x and w are not covered by visited nodes.

IV. Simulation

Our simulation study focuses on two aspects:

Efficiency: First we evaluate the performance of the coverage conditions in reducing the number of forward nodes. Besides several special cases of the coverage conditions, MCDS and a neighbor-designating algorithm are also simulated and compared.

Parameters: The general framework contains several configuration parameters, such as type of coverage condition, size of neighborhood information that is collected at each

node (k), size of routing history that is piggybacked in each broadcast packet (h), and type of priority value. Obviously, a weak coverage condition, large k and large h will produce a relatively small forward node set. On the other hand, they also produce relatively high overhead. These parameters shall be carefully tuned to balance the pruning efficiency and the overhead.

The simulation is conducted with a custom simulator *ds* [17], which simulates several broadcast algorithms on random ad hoc networks, including coverage conditions I and II, marking process enhanced by pruning Rules 1 and 2, marking process enhanced by pruning Rule k, enhanced Span, and LENWB. Data for the enhanced neighbor designating algorithm is obtained from another simulator used in [6]. Unlike ns-2, where the entire network protocol stack is considered, ds considers only functions in the network layer, assuming an ideal MAC layer without contention or collision. Simulations that cover the entire network protocol stack can be found in [3]. To generate a random ad hoc network, n hosts are randomly placed in a restricted 100×100 area. To study the behaviors of different algorithms under a given average node degree d, the transmitter range r is adjusted to produce exactly $\frac{nd}{2}$ links in the corresponding unit disk graph. Networks that cannot form a strongly connected graph are discarded. Figure 7 shows a sample network generated by *ds* with 38 forward nodes and 62 non-forward nodes. Every simulation is repeated until the 90% confidence intervals of all average results are within $\pm 5\%$.

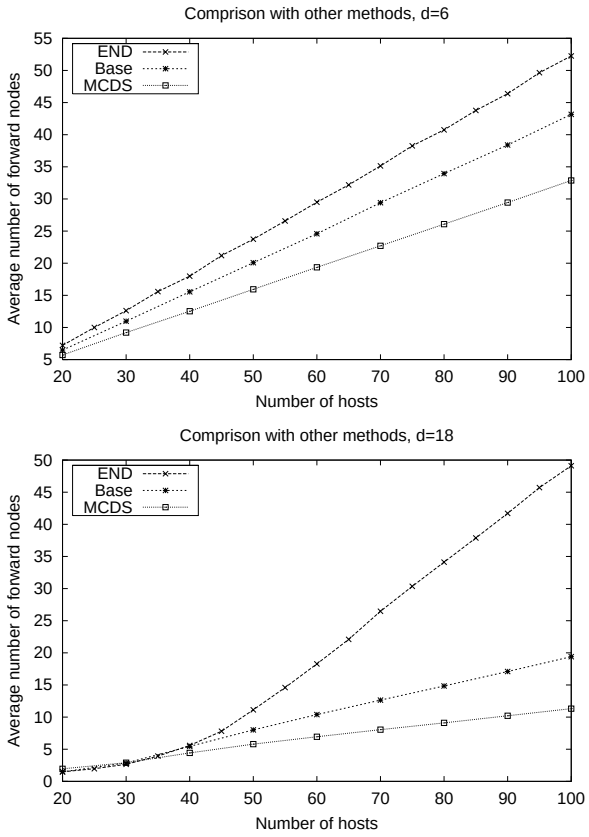


Fig. 8. Performance of coverage condition I compared with other broadcast methods.

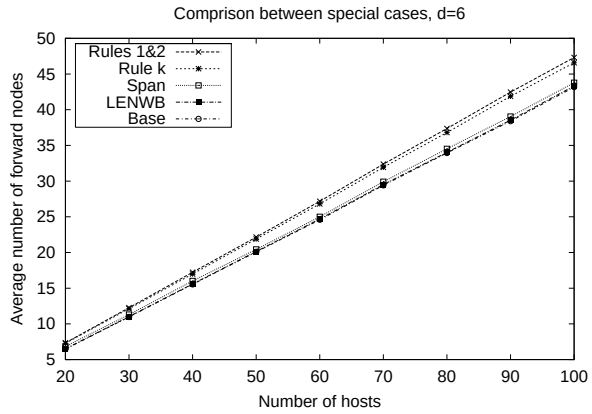
A. Algorithm efficiency

The efficiencies of various broadcast algorithms are compared in terms of the numbers of forward nodes. We say an algorithm is more efficient than another algorithm if it generates a smaller forward node set. For the sake of clarity, simulation results are organized into two groups: (1) a "base" configuration of coverage conditions versus several non-self-pruning broadcast schemes, and (2) the same base configuration versus several existing schemes that are special cases of coverage conditions. One finding in our simulation study is that the two coverage conditions have very similar properties and are hard to distinguish. Therefore, we omit lines representing coverage condition II in most figures.

Figure 8 compares efficiencies of three broadcast schemes. The base configuration (Base) is coverage condition I with 2-hop neighborhood information, 2-hop routing history, and node degrees as priority values. The enhanced neighbor-designating algorithm (END) as described in [6] is the most efficient neighbor designating algorithm. The third algorithm is based on Guha and Khuller's approximation algorithm to form a minimum connected dominating set (MCDS) [2]. This algorithm is not localized, as it requires global information to compute the forward node set. However, it can produce a near-optimal forward node set. Here we use it as a substitution of a "perfect" algorithm that produces the optimal result. The simple flooding method does not appear in this figure, because it always has n forward nodes. The probability- and area-based methods are not considered, since we only compare the algorithms that ensure the coverage.

Throughout this section, we consider the performance of each broadcast algorithm under two circumstances: relatively *sparse* networks ($d = 6$, as shown in the left graph of Figure 8), and relatively *dense* networks ($d = 18$, as shown in the right graph of Figure 8). In sparse networks, Base is about 20% worse than MCDS and 20% better than END. These ratios maintain as the number of nodes increases from 20 to 100. In dense networks, Base is about 40% worse than MCDS and about 150% better than END. That is, coverage condition I with 2-hop approximation is closer to optimal than neighbor designating algorithms, and performs much better in dense networks.

Figure 9 compares several special cases of the coverage conditions, including the same base implementation, Wu and Li's algorithm (Rules 1&2), Dai and Wu's algorithm (Rule k), enhanced Span, and LENWB. For a fair comparison, all the special cases use 2-hop neighborhood information and node degree as the priority value, except for enhanced Span, which uses neighborhood connectivity as the priority value. Under all circumstances, Base is better than all existing algorithms. It is not surprising, because Base combines the weakest coverage condition (condition I) and the longest routing history (2-hop).



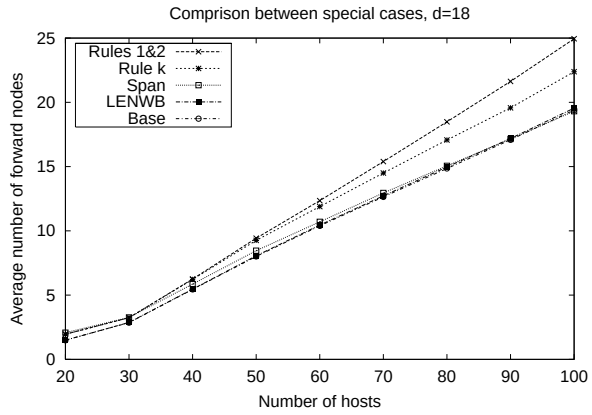
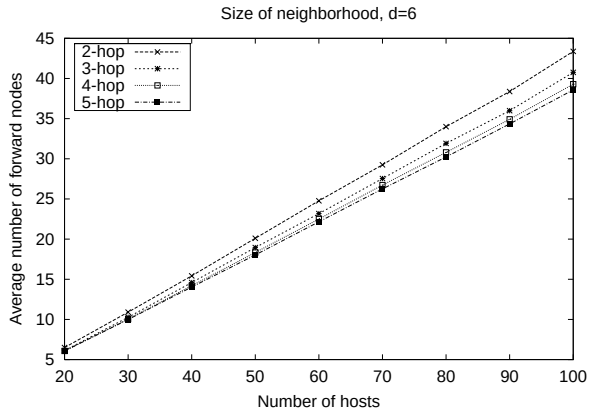


Fig. 9. Performance of severl special cases of coverage conditions.



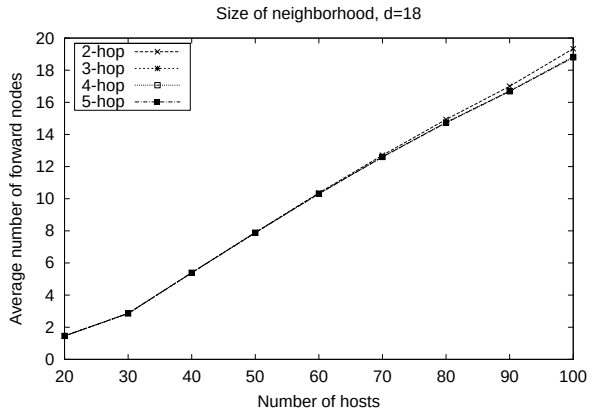


Fig. 10. Performance of k-hop approximations of coverage conditions with various k's.

Span and LENWB are very close to Base. Rules 1&2 is worse than Rules k, which in turn, is worse than Span, LENWB, and Base. The difference becomes small in sparse networks (about 10%) and significant in dense networks (about 20%). This is also understandable because Rules 1&2 and Rule k are not specifically designed for broadcasting and, therefore, cannot take the advantage of routing history information. Overall, all special cases exhibit quite similar efficiencies, and the general framework is more efficient than any existing algorithms.

B. Configuration parameters

Simulation results in the last subsection show that different implementations of our generic self-pruning broadcast scheme have similar efficiencies. Since different configurations have different communication and computation overheads, fine tuning of configuration paraments may achieve lower overhead without losing much efficiency. Here we consider four parameters: (1) k, the "radius" of the neighborhood that each node considers in the coverage conditions, (2) h, the maximum length of the "trail" that can be piggybacked in each broadcast packet, which consists of the id's of recently visited nodes, (3) type of coverage condition, and (4) type of priority value. These parameters determine not only the number of forward nodes but also sizes of control and broadcast packets, amount of computation, and converging speed of neighborhood information.

Parameter (1) is related to the size of control packets and the converging speed. If k-hop information is collected at each node, the size of control packets for exchanging $(k-1)$-hop information between neighbors is $O(k^2D^2)$, and it needs k rounds of information exchange to converge after a change in network topology. Figure 10 compares four configurations: $k = h = 2$ (2-hop), $k = h = 3$ (3-hop), $k = h = 4$ (4-hop), and $k = h = 5$ (5-hop). All these configurations use node degree as the priority value. This is also the default setting in subsequent comparisons. In sparse networks, 2-hop is about 10% less efficient than 3-hop, which in turn is slightly worse than 4-hop and 5-hop. In dense networks, all configurations have almost the same efficiency. We can conclude that 2-hop information is relatively cost-effective for dense networks, and 3-hop information is relatively cost-effective for sparse networks.

Parameter (2) is related to the size of each broadcast packet. By piggybacking h-hop routing information, the size of each broadcast packet increases by $O(h)$. Figure 11 compares three configurations: no routing history (0-hop), one hop routing history (1-hop), and k hops routing history (k-hop), where k is the neighborhood radius. This simulation is conducted on networks with 100 nodes ($n = 100$), with k varying from 2 to 5. In sparse networks, 0-hop is about 5% less efficient than 1-hop and k-hop. In dense networks, 0-hop is about 10% less

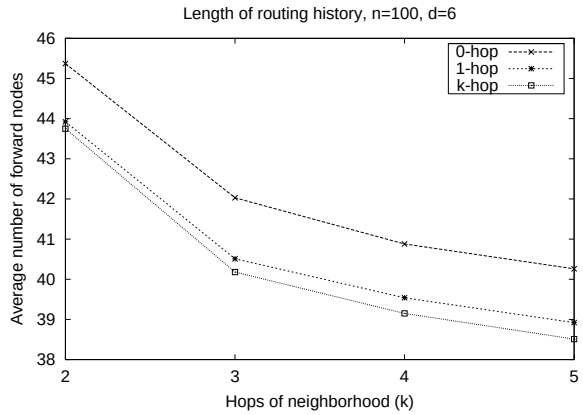
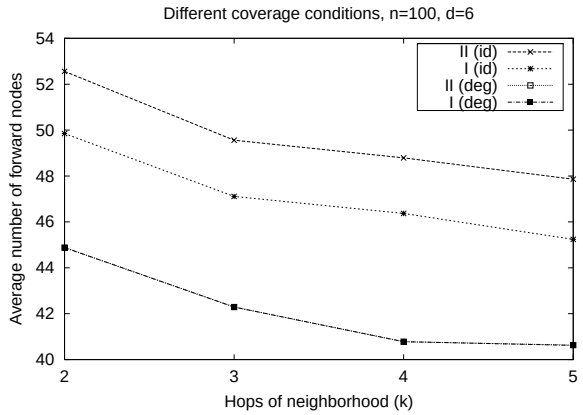


Fig. 11. Performance of coverage conditions with various lengths of routing history.



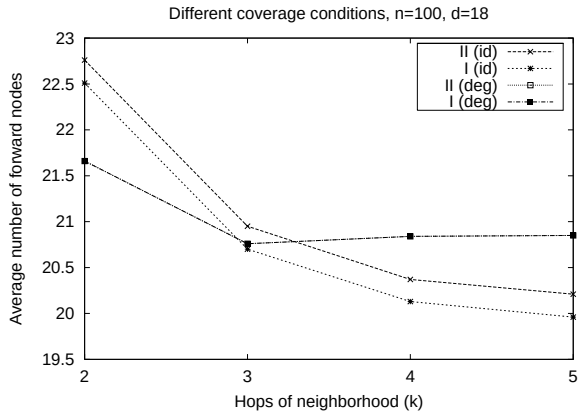


Fig. 12. Performance of different coverage conditions.

efficient than 1-hop and k-hop. Under both circumstances, 1-hop and k-hop have very similar efficiency. Therefore, using 1-hop routing history (i.e., the id of the last forward node) is more cost-effective.

Parameter (3) is related to the amount of computation. Coverage condition I consumes more CPU time than condition II. Figure 12 compares four configurations: coverage condition I with node id (I(id)) and node degree (I(deg)) as the priority value, and coverage condition II with node id (II(id)) and node degree (II(deg)) as the priority value. All these configurations use no routing history. In both sparse and dense networks, I(id) is only slightly more efficient than II(id), and no difference is observed between I(deg) and II(deg). We conclude that coverage condition II is a good approximation of coverage condition I and, considering the computation overhead, more cost-effective than coverage condition I.

Parameter (4) is related to both converging speed and computation complexity. Using node id as the priority value contributes a relatively fast converging speed and requires no extra computation. Using node degree as the priority value causes a relatively slow converging speed, because it takes an extra round of information exchange to obtain the accurate node degree value. Neighborhood connectivity, defined as the ratio of pairs of directly connected neighbors to pairs of any neighbors, is used in Span. The node with the lower neighborhood connectivity has the higher priority. Connectivity as priority is the hardest to compute and needs two extra rounds to converge. Figure 13 compares three configurations with different priority values. In sparse networks, id is the worst, and degree and connectivity are very close. In dense networks, id and degree have similar efficiencies. Degree is better with small k, and id is better with large k. Connectivity is the most efficient priority under all circumstances. There is no optimal choice of the priority type. Node id is the best for minimizing the converging time. Neighborhood connectivity is the best for relatively stationary networks. Node degree is more desirable when the computation power of each node is limited and longer converging time is tolerable.

Overall, a cost-effective configuration shall be with 2 or 3-hop information, 1-hop routing history, and coverage condition II.

V. Conclusion

This paper aims to provide a general framework for broadcasting in ad hoc networks that uses self-pruning techniques to reduce the number of forward nodes. The proposed scheme, namely neighborhood coverage conditions I and II, is the superset of several existing neighbor-knowledge-based broadcast algorithms. The general framework is more efficient in reducing the forward node set than the existing ones. Furthermore, it provides better perception on the critical mechanisms behind

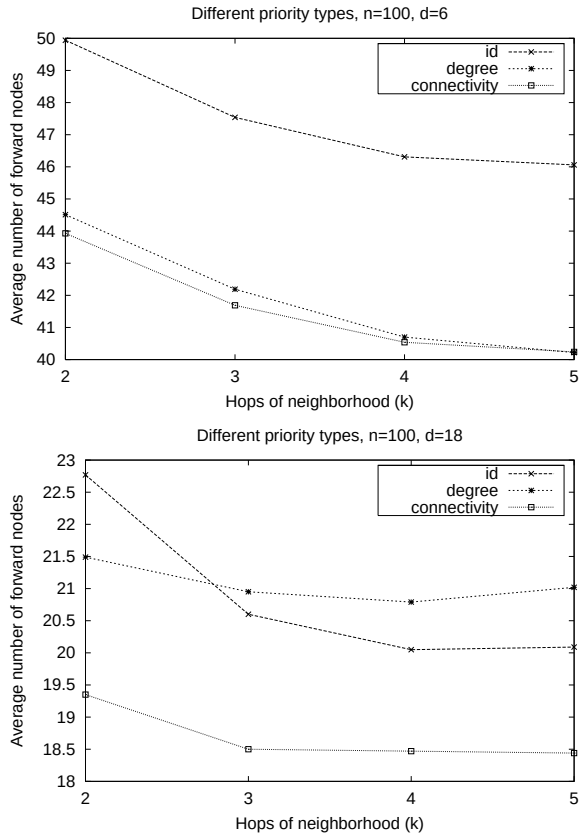


Fig. 13. Performance of coverage conditions with various types of priority values.

the self-pruning algorithms; e.g., neighborhood information, routing history, coverage conditions, and priority functions.

A comprehensive simulation study reveals that: (1) The self-pruning scheme in general is more efficient in reducing the forward node set than several existing schemes that ensure the broadcast coverage, and new algorithms can be derived from the proposed framework that outperform several existing self-pruning schemes. (2) To achieve a good balance between efficiency and overhead, 2- or 3-hop neighborhood information, 1-hop routing history, and coverage condition II are appropriate as configuration parameters. There are no obvious answer on the type of priority function that should be adopted. Node id, node degree and neighborhood connectivity shall be selected in a sensitive way to achieve a better tradeoff between pruning efficiency and converging speed. Our future work includes enhancement of the general framework to interpret other existing neighbor-knowledge-based broadcast schemes, including neighbor-designating methods.

REFERENCES

[1] S. Giordano and W. W. Lu, "Challenges in mobile ad hoc networking," *IEEE Communications Magazine*, vol. 39, no. 6, pp. 129–181, June 2001.

[2] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, Apr. 1998.

[3] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, 2002, pp. 194–205. [Online]. Available: citeseer.nj.nec.com/williams02comparison.html

[4] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, no. 2/3, pp. 153–167, Mar.-May 2002.

[5] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc netowrks," in *Proceedings of the Third ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Aug. 2000.

[6] W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *IEEE Transactions on Mobile Computing*, vol. 1, no. 2, pp. 111–123, Apr.-June 2002.

[7] W. Peng and X. Lu, "AHBP: An efficient broadcast protocol for mobile ad hoc netowrks," *Journal of Science and Technology, Beijing, China*, 2002.

[8] A. Qayyum, L. Viennot, and L. Laouiti, "Multipoing relaying: An efficient technique for flooding in mobile wireless netowrks," INRIA-Rapport de recherche, Tech. Rep. 3898, 2000.

[9] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Proceedings of the Seventh ACM International Conference on Mobile Computing and Networking (MobiCom)*, July 2001, pp. 85–96.

[10] F. Dai and J. Wu, "Distributed dominant pruning in ad hoc wireless networks," Florida Atlantic University, Technical Report TR-CSE-FAU-02-02, Feb. 2002.

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

[12] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14–25, Jan. 2002.

[13] J. Sucec and I. Marsic, "An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks," Rutgers University, CAIP Technical Report 248, Sep. 2000.

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

[15] J. Wu and W. Lou, "Forward-node-set-based broadcast in clustered mobile ad hoc networks," *accepted to appear in Wireless Communication and Mobile Computing*, 2003.

[16] P. Sinha, R. Sivakumar, and V. Bharghavan, "Enhancing ad hoc routing with dynamic virtual infrastructures," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, vol. 3, Apr. 2001, pp. 1763–1772.

[17] F. Dai, "Wireless routing simulation suit," http://sourceforge.net/projects/wrss/, 2001.