Enhanced Dominant Pruning Applied to The Route Discovery Process of On-demand Routing Protocols

Marco Aurélio Spohn
Computer Science Department
University of California at Santa Cruz
Santa Cruz, CA 95064
maspohn@cse.ucsc.edu

J.J. Garcia-Luna-Aceves
Computer Engineering Department
University of California at Santa Cruz
Santa Cruz, CA 95064
jj@cse.ucsc.edu

***Abstract*— Dominant Pruning (DP) is a distributed connected dominating-set algorithm that can be used for reducing the impact of flooding in wireless ad hoc networks. We propose an enhanced dominant pruning (EDP) approach to be used in the route discovery process of on-demand routing protocols. To show the benefits of EDP, we integrate EDP into the *Ad-hoc On-demand Distance Vector* (AODV) protocol. We present detailed simulation results showing that our approach improves standard AODV in most aspects, and that it is simple and easy to implement. Our approach is compared against AODV and OLSR, as good representatives of on-demand and proactive routing for ad-hoc wireless networks.**

I. Introduction

On-demand route discovery is based on *route request* (RREQ) and *route reply* (RREP) messages (e.g., AODV [1] and DSR [2]). The way in which these messages are handled may differ among different protocols, but their functionality remains the same: a request is relayed until it reaches a node with a valid route to the destination or the destination itself, which triggers a reply message sent back to the originator. Several parameters (such as how long to keep requests in a cache, timeouts for requests, timeouts for hellos, and the like) are subject to tunning, and the choices made may result in improvements in the protocol performance. However, RREQs are propagated using either an unrestricted broadcast or an expanding ring search [3]. In either case, the resulting flooding operation causes considerable collisions of packets in wireless networks using contention-based channel access.

A *connected dominating set* (CDS) is a set of nodes such that every node in the network is either in the set or is the neighbor of a node in the set. The problem of determining the *minimum connected dominating set* (MCDS) is known to be NP-complete. Extensive work has been done on finding a good approximation of MCDS in terms of small approximation ratio. A protocol with a constant approximation ratio of eight has been proposed by Wan et. al. [4]. However, their approach requires that a spanning tree be constructed first in order to select the dominating nodes (forwarders), and only after that a broadcast can be performed. To improve the route discovery process we need an approach that is suitable for dynamic networks with mobile nodes, and is based on determining the CDS in real time.

This work was supported in part by CNPq (Brazil) and by the U.S. Air Force/OSR under grant No. F49620-00-1-0330.

Lim and Kim [5] show that the MCDS problem can be reduced to the problem of building a *minimum cost flooding tree* (MCFT). Given that an optimal solution for the MCFT problem is not feasible, they propose heuristics for flooding trees, resulting in two algorithms: *self-pruning* and *dominant pruning* (DP). They show that both algorithms perform better than *blind flooding*, with which each node broadcasts a packet to its neighbors whenever it receives the packet along the shortest path from the source node, and that DP outperforms *self-pruning*. Section II provides more details on DP, shows an error in the original algorithm by Lim and Kim [5] and the way to fix it.

Enhancements to dominant pruning have been reported by Lou and Wu [6], who describe the *total dominant pruning* (TDP) algorithm and the *partial dominant pruning* (PDP) algorithm. TDP requires that the two-hop neighborhood of the sender be piggybacked in the header of the packet. This information reduces the size of the two-hop neighbor set that needs to be covered by the forwarders. The header size increases proportionally to the number of nodes in the two-hop neighborhood, which may become a problem in dense or large networks. PDP enhances DP by eliminating the two-hop nodes advertised by a neighbor shared by both the sender and the receiver (forwarder). Simulation results assuming an ideal MAC layer with which no contention or collisions occur show that both TDP and PDP improve DP in a static environment. A dynamic scenario is also evaluated, and DP is shown to perform better than both TDP and PDP.

We propose modifications to DP together with some heuristics that improve its performance. These heuristics help to further reduce the number of broadcast messages at the expense of having to attach more information in the header of the control packets. We call our proposal *Enhanced Dominant Pruning* (EDP), which is described in Section III. EDP can be applied to any on-demand routing protocol that relies on broadcasting control packets when searching for a route to a given destination. To show the applicability of EDP to an existing protocol, we have implemented EDP in AODV. Nodes use hello messages to disseminate their valid one-hop neighbors for building the two-hop neighborhood, which is the minimum requirement for the connected dominating set algorithm under consideration.

The Optimized Link State Routing (OLSR) [7] [8] is closely related to our work, because it uses a similar mechanism for reducing duplicate control traffic retransmissions. Each node N in the network selects a set of nodes among its symmetrical one-hop neighbors, which are then responsible for retransmitting its packets. This set of nodes is called *multipoint relays* (MPR). Nodes that are not in the MPR set of N do not retransmit the packets received from N. Our approach differs from OLSR in the way the forwarders are chosen, as well as the heuristics used for prunning redundant relayers. Nevertheless, in spite of OLSR being a proactive routing protocol, it would be possible to substitute the MPR approach with ours. The impact of such modifications is the subject of another publication.

Section IV presents detailed simulations to show the benefits of EDP when it is applied to AODV and compares it against OLSR, AODV with and without hellos, and AODV with DP. The simulation results clearly show that AODV with EDP renders the best performance of all the AODV versions, and much better packet delivery ratios and end-to-end delays than OLSR, which is a direct consequence of reducing packet collisions due to RREQs by means of EDP. Section V concludes this work.

II. Dominant pruning review

We use a simple graph, $G = (V, E)$, to represent an ad hoc wireless network, where V represents a set of wireless mobile hosts (nodes) and E represents a set of edges (links). The network is seen as a *unit disk graph* [9], i.e., the nodes within the circle around node v (corresponding to its radio range) are considered its neighbors.

In *dominant pruning* (DP) [5] the sending node decides which adjacent nodes should relay the packet. The relaying nodes are selected using a distributed CDS algorithm, and the identifiers (IDs) of the selected nodes are piggybacked in the packet as the forwarder list. A receiving node that is requested to forward the packet again determines the forwarder list. The flooding ends when there is no more relaying nodes.

Nodes keep information about their two-hop neighborhood, which can be obtained by the nodes exchanging their adjacent node list with their neighbors. DP is a distributed algorithm that determines a set cover based on the partial knowledge of the two-hop neighborhood. Ideally, the number of forwarding nodes should be minimized to decrease the number of transmissions. However, the optimal solution is NP-complete and requires that nodes know the entire topology of the network. DP uses the *greedy set cover* (GSC) algorithm to determine the forwarder list of a packet (i.e., the list of nodes that should forward the packet) based just on partial knowledge of the network topology. GSC recursively chooses one-hop neighbors that cover the most two-hop neighbors, repeating the process until all two-hop neighbors are covered.

The set of nodes within two-hops from node n_i is denoted by N_2^i, and the set of one-hop neighbors of node n_i is denoted by N_1^i. If node n_i is the source of the broadcast, it determines its forwarder list so that all nodes in $U_i = N_2^i - N_1^i$ receive the packet. The set of forwarder nodes is denoted by $F_{DP}^i = \{f_1, f_2, ..., f_m\} \subseteq N_1^i$, such that $\bigcup_{f_k \in F_{DP}^i} (N_1^{f_k} \bigcap U_i) = U_i$. A forwarder node $n_j \in F_{DP}^i$ determines its own forwarder list upon receiving the broadcast. Node n_j does not need to cover the neighbors of node n_i (i.e., N_1^i), because they were already covered by the previous broadcast. In this case, $U_j = N_2^j - N_1^j - N_1^i$ is the set to be covered. The set $F_{DP'}^j \subset N_1^j$ is the temporary set cover of node n_j. Our solution includes the set of one-hop neighbors shared by nodes n_i and n_j (i.e., $N_1^j \bigcap N_1^i$), in the first part of the computation of the forwarder list. The final forwarder list is defined as $F_{DP}^j = F_{DP'}^j - F_{DP}^i$.

The solution presented by Lim and Kim [5] is incorrect, because only nodes in the subset $N_1^j - N_1^i$ are considered for the computation of the forwarder list, which can lead to incorrect results for particular topologies. The reason is simple, nodes being shared by the source and the receiver are still candidates as forwarder nodes, because node n_j may have two-hop nodes exclusively advertised by some shared node. Because a node knows the sender's forwarder list, it can get rid of those nodes that were previously chosen as forwarder nodes by the sender. It turns out that the resulting forwarder list described in [5] is in fact in the subset $N_1^j - N_1^i$.

Figure 1 shows an example where DP as proposed in [5] fails. Consider node A as the source of the broadcast. Node A selects nodes B and C as the forwarder nodes. Note that $U_B = \{F, G, H\}$, but there is no node in the set $\{D, E\}$ (i.e., $N_1^B - N_1^A$) that can cover the set $\{F, G\}$. The same can be said for $U_C = \{D, E, J\}$, where there is no node in the set $\{F, G\}$ (i.e., $N_1^C - N_1^A$) that can cover the set $\{D, E\}$.

Our solution guarantees that nodes in $N_1^A \bigcap N_1^B$ take part in the selection of the forwarder nodes of B, and that nodes in $N_1^A \bigcap N_1^C$ take part in the selection of the forwarder nodes of C. It is just a matter of consistency, even though some nodes in $F_{DP'}^B$ and in $F_{DP'}^C$ are ruled out of the resulting forwarder list when they were already in the sender's forwarder list.

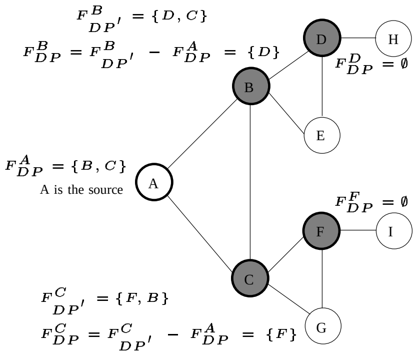


Fig. 1. Example where original DP fails (node A is the source). Modified DP determines the correct set of forwarding nodes: A selects B and C as forwarder nodes; B chooses D and C as forwarder nodes, but dismiss C because it is in the sender's forwarder list; C chooses F and B, but dismiss B because it is in the sender's forwarder list.

III. Enhanced Dominant Pruning

The objective of *Enhanced Dominant Pruning* (EDP) is to reduce the number of broadcast packets necessary to flood the network with the same guarantees provided by DP (after applying the modifications cited previously). In the following, we assume that a neighbor protocol is available to provide the two-hop neighborhood information.

The *EDP forwarder list* as determined by node n_i is denoted by $\mathcal{F}_i$. We use the term *EDP forwarder list* to emphasize that the resulting list might be different from the one obtained by simply running DP (i.e., F^i_{DP}) over U_i. The current node is denoted by n_i, and the node that sent the packet is denoted by S (if the current node is the source of the broadcast, then $\mathsf{S} = \emptyset$). The sender's forwarder list is denoted by $\mathcal{F}_\mathsf{S}$, and the second-to-previous forwarder list is denoted by $\mathcal{F}_{\mathsf{S}_\mathsf{S}}$. $\mathcal{F}_\mathsf{S} = \emptyset$ if n_i is the source of the broadcast. In a similar manner, $\mathcal{F}_{\mathsf{S}_\mathsf{S}} = \emptyset$ if n_i is the source of the broadcast or if the sender S is the source. The packet header must specify the forwarder list and the sender's forwarder list (the sender S of the packet is obtained from the packet header), but that should not be a problem given that both lists are expected to be small, because GSC is applied to the two-hop neighborhood.

Algorithm 1 shows the pseudo-code for determining the *EDP forwarder list* $\mathcal{F}_i$. Let $\mathcal{C}$ be the set of neighbors of node n_i that are also in the sender's forwarder list $\mathcal{F}_\mathsf{S}$. Let $\bigcup_{n_k \in \mathcal{C}} N^k_1$ be the set of nodes adjacent to neighbors that are also in the sender's forwarder list. These nodes do not need to be considered when running DP, because they are guaranteed to be covered by some other forwarder node. Let $\mathcal{I}$ be the set of nodes in $\mathcal{F}_\mathsf{S}$ with identifiers larger than node n_i. The set of forwarder nodes of n_i that are reachable through other forwarder node in the sender's list (i.e., there is a disjoint two-hop path to node n_k through another node in $\mathcal{F}_\mathsf{S}$ with a higher priority) is denoted by $\mathcal{P}$. The set of neighbors that are already covered by nodes in $\mathcal{F}_{\mathsf{S}_\mathsf{S}}$ (the second-to-previous forwarder node) is denoted by $\mathcal{Q}$.

As in DP, a forwarding node does not need to include in its forwarder list those neighbors that are also neighbors of the sender S (i.e., $N^i_1 \bigcap N^\mathsf{S}_1$). Because the sender S already sent the packet to all its neighbors, all the common neighbors between the sender and the receiver can be excluded from the forwarder list. Neighbors that are also in the sender's forwarder list $\mathcal{F}_\mathsf{S}$ (line 1) can have their one-hop nodes removed from U_i (lines 2 through 4). Then DP is run on this reduced set, denoted by U^i_{DP} (line 5).

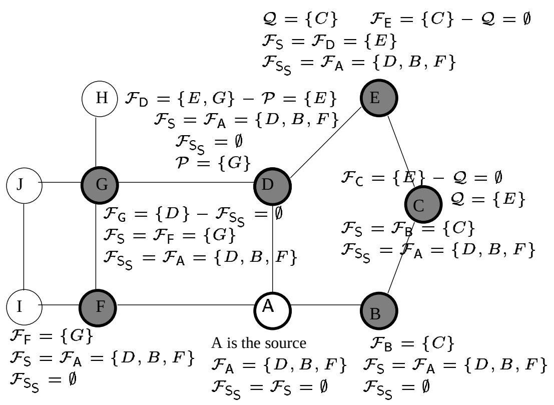
A node in F^i_{DP} that is covered by at least one more node in $\mathcal{F}_\mathsf{S}$ needs to be covered by just one of these nodes. To select one, we use node identifiers as priorities, and the node with the largest ID wins. Lines 7 through 9 present the pseudo-code that creates the set $\mathcal{I}$, which contains the nodes in $\mathcal{F}_\mathsf{S}$ with identifiers larger than the local node n_i. The set $\mathcal{P} \subset F^i_{DP}$ has the forwarder nodes that are reachable through another node in the sender's forwarder list (lines 10 through 14). That is, there is a disjoint two-hop path to node $n_k \in F^i_{DP}$ through another node in $\mathcal{F}_\mathsf{S}$ with a higher priority. Therefore, a node in the set $\mathcal{P}$ can be excluded from the forwarder list.

Algorithm 1: Enhanced Dominant Pruning

```
Data   : $n_i$, $\mathcal{F}_\mathsf{S}$, $\mathcal{F}_{\mathsf{S}_\mathsf{S}}$, $U_i$
Result : $\mathcal{F}_i$, the forwarder list
begin
1     $C \longleftarrow N^i_1 \bigcap \mathcal{F}_\mathsf{S}$
2     $U^i_{DP} \longleftarrow U_i$
3     for $n_k \in \mathcal{C}$ do
4         $U^i_{DP} \longleftarrow U^i_{DP} - N^k_1$
5     $F^i_{DP} \longleftarrow DP(U^i_{DP})$
6     $\mathcal{I} \longleftarrow \emptyset$
7     for $n_k \in \mathcal{F}_\mathsf{S}$ do
8         if $n_k > n_i$ then
9             $\mathcal{I} \longleftarrow \mathcal{I} \bigcup \{n_k\}$
10    $\mathcal{P} \longleftarrow \emptyset$
11    for $n_k \in F^i_{DP}$ do
12        for $n_l \in \mathcal{I}$ do
13            if $n_l \in N^k_1$ then
14                $\mathcal{P} \longleftarrow \mathcal{P} \bigcup \{n_k\}$
15    $\mathcal{Q} \longleftarrow \emptyset$
16    for $n_k \in N^i_1$ do
17        for $n_l \in N^k_1$ do
18            if $n_l \in \mathcal{F}_{\mathsf{S}_\mathsf{S}}$ then
19                $\mathcal{Q} \longleftarrow \mathcal{Q} \bigcup \{n_k\}$
20    $\mathcal{F}_i \longleftarrow F^i_{DP} - \mathcal{Q} - \mathcal{P} - \mathcal{F}_\mathsf{S} - \mathcal{F}_{\mathsf{S}_\mathsf{S}}$
end
```

A neighbor n_k that was previously chosen as a forwarder node by the second to previous node (i.e., $n_k \in \mathcal{F}_{\mathsf{S}_\mathsf{S}}$), and neighbors covered by a node in $\mathcal{F}_{\mathsf{S}_\mathsf{S}}$, can be removed from the forwarder list (lines 15 through 19). A neighbor n_k is covered by some node $n_l \in \mathcal{F}_{\mathsf{S}_\mathsf{S}}$ if $n_l \in N^k_1$. Finally, the EDP forwarder list $\mathcal{F}_i$ is updated on line 20.

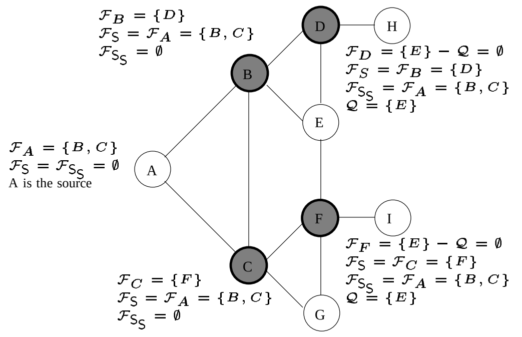
Consider the example shown in Figure 2. Node A selects nodes $\{B, D, F\}$ for its forwarder list. Node D selects nodes $\{E, G\}$ as forwarders, because E is the only neighbor covering node C, and node G, because it is the only neighbor covering nodes $\{H, J\}$. Node G can be removed from D's forwarder list, because node G is covered by another forwarder node with a higher priority (i.e., node $F \in \mathcal{F}_A$, and $ID(F) > ID(D)$). On the other hand, node F selects node G as its forwarder node, because node F wins over node D. Node G selects node D as its forwarder, because D is the only neighbor covering node E, but node D can be dismissed because node D is in the second to previous forwarder list (i.e., $D \in \mathcal{F}_{\mathsf{S}_\mathsf{S}} = \mathcal{F}_A$). Node B selects node C as its forwarder, because node C is the only neighbor to cover node E. Node C determines E as a forwarder node, because E is the only neighbor covering node D. Node C does not need to include node E in the forwarder list, because node E is covered by a node in $\mathcal{F}_A$. The same happens at node E, which selects node C as a forwarder node but it is not necessary to include node C in the forwarder list, because node C is covered by node B. It is important to note that, in order to exclude a neighbor from the forwarder list, it suffices that the node is covered by other node in $\mathcal{F}_{\mathsf{S_S}}$. It is not a requirement that the excluded node be chosen as a forwarder node by the node covering it.



$\mathcal{Q} = \{C\}$ $\mathcal{F}_{\mathsf{E}} = \{C\} - \mathcal{Q} = \emptyset$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_{\mathsf{D}} = \{E\}$
$\mathcal{F}_{\mathsf{S_S}} = \mathcal{F}_{\mathsf{A}} = \{D, B, F\}$
H
E
$\mathcal{F}_{\mathsf{D}} = \{E, G\} - \mathcal{P} = \{E\}$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_{\mathsf{A}} = \{D, B, F\}$
$\mathcal{F}_{\mathsf{S_S}} = \emptyset$
$\mathcal{P} = \{G\}$
J
G
D
$\mathcal{F}_{\mathsf{C}} = \{E\} - \mathcal{Q} = \emptyset$
C
$\mathcal{Q} = \{E\}$
$\mathcal{F}_{\mathsf{G}} = \{D\} - \mathcal{F}_{\mathsf{S_S}} = \emptyset$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_{\mathsf{F}} = \{G\}$
$\mathcal{F}_{\mathsf{S_S}} = \mathcal{F}_{\mathsf{A}} = \{D, B, F\}$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_{\mathsf{B}} = \{C\}$
$\mathcal{F}_{\mathsf{S_S}} = \mathcal{F}_{\mathsf{A}} = \{D, B, F\}$
I
F
A
B
$\mathcal{F}_{\mathsf{F}} = \{G\}$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_{\mathsf{A}} = \{D, B, F\}$
$\mathcal{F}_{\mathsf{S_S}} = \emptyset$
A is the source
$\mathcal{F}_{\mathsf{A}} = \{D, B, F\}$
$\mathcal{F}_{\mathsf{S_S}} = \mathcal{F}_{\mathsf{S}} = \emptyset$
$\mathcal{F}_{\mathsf{B}} = \{C\}$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_{\mathsf{A}} = \{D, B, F\}$
$\mathcal{F}_{\mathsf{S_S}} = \emptyset$

Fig. 2. Node A is the source. The knowledge about the second to previous forwarder list ($\mathcal{F}_A$) allows nodes E and C to exclude each other from their forwarder list, and node G to exclude node D. Node D reduces the size of its forwarder list by using the information provided by the set $\mathcal{P}$.

Consider the example illustrated in Figure 3. Node A is the source of the broadcast. Nodes B and C are chosen as forwarders. Node B does not need to cover nodes F and G because they are adjacent to other node (i.e., node C) in the sender's forwarder list. Given that, node B determines node D as its forwarder node, which in turn determines node E as its forwarder node. Nevertheless, D does not need to forward the packet to node E because it is covered by a previous forwarder node (i.e., node B chosen as forwarder node by node A), and node E is adjacent to both node D and the sender B. In a similar manner, node C does not need to cover nodes D and E, because they are adjacent to node B that is in the sender's forwarder list. C determines node F as its forwarder node, which in turn determines node E as a forwarder node. Node F does not need to include node E because this node is covered by a previous forwarder node (i.e., node B that is in A's forwarder list).



$\mathcal{F}_B = \{D\}$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_A = \{B, C\}$
$\mathcal{F}_{\mathsf{S_S}} = \emptyset$
D
H
B
$\mathcal{F}_D = \{E\} - \mathcal{Q} = \emptyset$
$\mathcal{F}_S = \mathcal{F}_B = \{D\}$
$\mathcal{F}_{\mathsf{S_S}} = \mathcal{F}_A = \{B, C\}$
$\mathcal{Q} = \{E\}$
E
$\mathcal{F}_A = \{B, C\}$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_{\mathsf{S_S}} = \emptyset$
A is the source
A
F
I
$\mathcal{F}_F = \{E\} - \mathcal{Q} = \emptyset$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_C = \{F\}$
$\mathcal{F}_{\mathsf{S_S}} = \mathcal{F}_A = \{B, C\}$
$\mathcal{Q} = \{E\}$
C
$\mathcal{F}_C = \{F\}$
$\mathcal{F}_{\mathsf{S}} = \mathcal{F}_A = \{B, C\}$
$\mathcal{F}_{\mathsf{S_S}} = \emptyset$
G

Fig. 3. Node A is the source. Nodes B, C, D, F are the only nodes chosen as forwarders for this network.

Theorem 1 *Given a graph $G(V, E)$, let T_{DP} be a CDS of G when applying the algorithm DP, and T_{EDP} a CDS of G when applying the algorithm EDP. Then T_{EDP} is equivalent to T_{DP}.*

Proof: Nodes in the set $\mathcal{F}_{\mathsf{S_S}}$ and the set $\mathcal{F}_{\mathsf{S}}$ can be excluded without any implication besides reducing redundancy. Nodes in $\mathcal{Q}$ were already covered by some other forwarders in $\mathcal{F}_{\mathsf{S_S}}$, therefore they can be omitted. A node $n_k \in \mathcal{P}$ can be disregarded because node n_k is covered by some other node n_L in the set $\mathcal{I}$ (nodes with higher priority), i.e., all nodes in N_1^k are also in N_1^L, or $n_k \in \mathcal{F}_L$ when there is a node n_u exclusively covered by n_k (i.e. $n_u \in N_1^k$ and n_u is not a neighbor of any other node in N_1^L). Hence, all nodes covered by T_{DP} are also covered by T_{EDP}. ■

A. Applying EDP to the Route Discovery Process of AODV

This section addresses the application of EDP to the route discovery process in AODV (AODV-EDP). Our neighbor protocol uses hello packets to disseminate the one-hop neighborhood, which creates a picture of its two-hop neighborhood at any given node in the network. A hello packet advertises the node's sequence number (*mySeqNum*), the identification of its known neighbors (*neighbors[]*), and the corresponding neighbors' sequence number (*neighSeqNum[]*). We have chosen a hello interval of $1.5s$. To reduce the number of broadcast messages, RREQ also advertise the one-hop neighborhood information, working as a hello message. This event reschedules any pending hello message.

To avoid pruning too many route requests in the presence of mobility and cross-traffic, we have chosen to implement the neighbor protocol as part of AODV. We extended the hello mechanism available in AODV to include the information about the one-hop neighborhood in hello messages, and we also rely on the AODV mechanisms for evaluating the link status to neighbors.

A route request (RREQ) works in a similar way as in AODV. The main difference being that only forwarders rebroadcast a broadcast packet. The source of a RREQ calculates its forwarder list using EDP, and broadcasts the packet. Upon receiving a route request, a forwarder that cannot respond to this request calculates its own forwarder list using the information provided in the RREQ packet (i.e., forwarder list, second to previous forwarder list, and source node) and broadcast the packet after updating it with its own forwarder list. Eventually the request reaches a node with a route to the destination or the destination itself. It is expected that most of the replies will come from an intermediate node because of the two-hop neighborhood information.

Because of topology changes, nodes may not have correct two-hop neighborhood information, which may result in forwarding lists that do not cover all nodes in the neighborhood. However, this is not a major problem, because a node incorrectly excluded from the forwarder list also receives the request and can respond in the case it has a route to the destination.

IV. Simulations and Performance Results

To compare AODV with EDP (AODV-EDP) against other protocols, we use traffic and mobility models similar to those previously reported in [10]. We implemented AODV-EDP in *Qualnet* 3.5, and compare it against AODV-DP (AODV with *Dominant Pruning*), AODV with no hello messages and with $2s$ hello timers, and OLSR. We have chosen AODV and OLSR because they represent some of the most referenced reactive and proactive unicast routing protocols for wireless ad hoc networks.

A. Simulation Parameters

The network is composed of 50 nodes spread over an area of $1500m\ x\ 300m$. The radio model used is a $2Mbps$ IEEE 802.11 device with a nominal transmission range of $280m$. Initially nodes are placed uniformly over a grid. Nodes move according to the random waypoint model with velocities between 0 and $20m/s$. Seven pause times are tested: $0s$ (always moving), $50s$, $100s$, $300s$, $500s$, $700s$, and $900s$.

For traffic sources we use 30 source nodes transmitting $4\ packets/s$ of 512 bytes, making it a total of 120 data packets being injected into the network every second. Nodes begin transmitting at $50s$ plus an offset uniformly chosen over a $5s$ period to avoid synchronization in their initial transmission. Source and destination pairs are chosen uniformly among the nodes in the network. The simulation time is set to 900 seconds, and identical mobility and traffic scenarios are used across protocols.

Experiments are repeated for 10 trials with different random number seeds. Results present a 95% confidence interval. Each data point represents the mean over the 10 runs discarding the lowest and largest results (quantile of one).

Four performance metrics are evaluated:

- *Packet delivery ratio*, the ratio of the data packets delivered to the destination to those generated by the CBR sources.
- *Average end-to-end delay* for data packets, including all possible delays caused by route discovery latency, queueing at the interface, retransmission delays at the MAC layer, and propagation and transfer times.
- *Routing load*, the number of routing packets transmitted per data packet delivered to the destination, where each hop traversed by the packet is counted as one transmission.
- *MAC collisions*, the number of collisions detected at the MAC layer.

B. Results

We show that AODV-EDP outperforms the other protocols in most of the performance metrics. OLSR performs better than AODV-EDP in terms of routing load and the number of MAC collisions (a difference of about 10% less collisions). However, we have to analyze these results together with the other metrics.

Figure 4 shows the packet delivery ratio. AODV-EDP presents an almost constant packet delivery ratio for all pause times. As the network becomes more static, the proactive approach of OLSR starts to payoff and it performs better than standard AODV, but AODV-EDP has a higher delivery ratio for all the pause times. AODV-DP shows that DP alone can improve AODV; however, it also shows that there is room for more improvement (i.e., there is some more redundancy that can be eliminated). OLSR performs better than AODV-DP for large pause times (after $500s$ pause time).

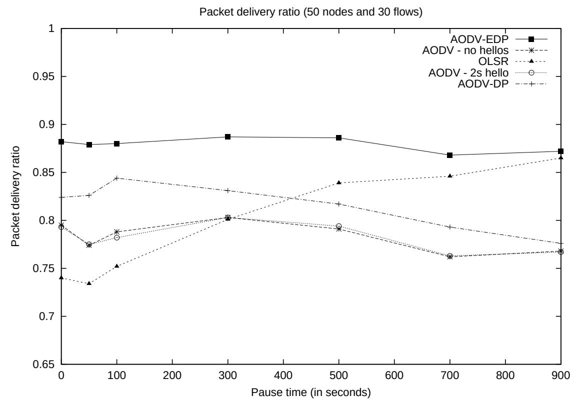


Fig. 4. Packet delivery ratio for 50 nodes and 30 flows (120 packets/s)

As pointed out in [10], the possibility of link failures is low with low mobility, but due to the node movement model (random waypoint) nodes usually get clustered. This situation is responsible for congestion in those regions in the presence of high traffic. This causes the link layer to report link failures even though the nodes are relatively static and a physical link still exists between the nodes. This is observed on Figure 4, where we notice a decreasing on the packet delivery ratio for some larger pause times.

Figure 5 shows the average end-to-end delay. AODV-EDP presents an almost constant mean latency, and is always the best for all pause times. Together with the packet delivery ratio, these results show that besides delivering more packets AODV-EDP delivers them faster than the other protocols. AODV-DP again shows that DP alone improves AODV, but OLSR is still better than AODV-DP for large pause times. Clustering of nodes has a direct impact on the latency as well. Packets spend more time waiting on the queues, and usually need to be retransmitted due to increased congestion.

Figure 6 presents the routing load. As expected, AODV-EDP has a lower routing load in comparison to standard AODV, because it reduces the number of broadcast transmissions. As expected, AODV-DP reduces the control overhead compared to AODV, but not as much as AODV-EDP. OLSR has the lowest routing load, but at the same time it gets a comparable delivery ratio only when the network is more static. As mobility increases, OLSR does not deliver as many packets as AODV-EDP, and does not improve the end-to-end delay for any pause time. In another words, less control overhead does not translate in better performance for the upper layers.

Figure 7 shows the number of collisions at the MAC layer. The number of colisions for standard AODV is noticeable

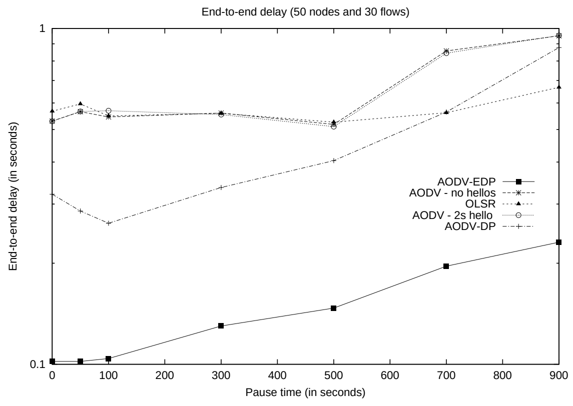


Fig. 5. End-to-end delay for 50 nodes and 30 flows (120 packets/s)

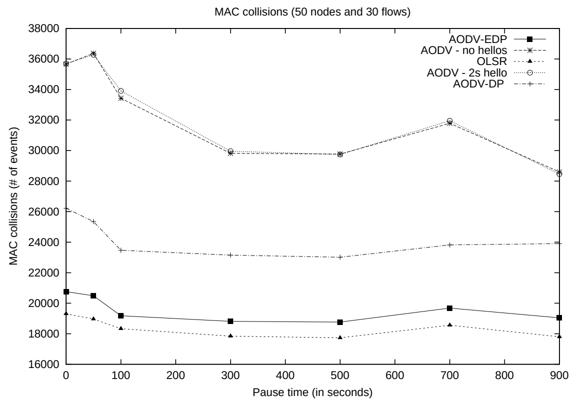


Fig. 7. MAC collisions for 50 nodes and 30 flows (120 packets/s)

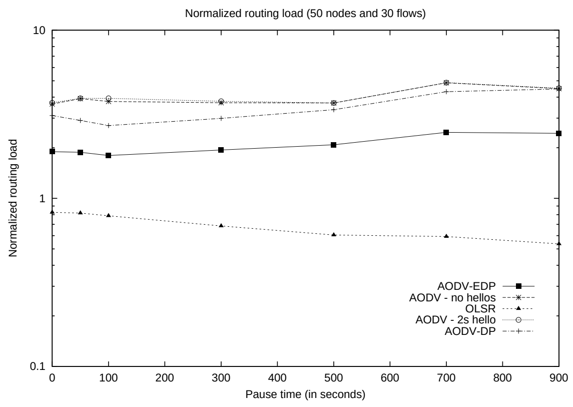


Fig. 6. Routing load for 50 nodes and 30 flows (120 packets/s)

larger than the other protocols, because a node always responds to the first received RREQ (if the TTL is valid, i.e., greater than zero). Because both AODV-EDP, AODV-DP, and OLSR reduce the number of necessary broadcasts, it translates in less collisions. OLSR produces slightly fewer collisions than AODV-EDP. However, these results when interpreted together with the packet delivery ratio and the end-to-end latency of both protocols indicate that AODV-EDP incurs a few more collisions because it delivers more packets.

Because the scenarios we have used to evaluate our approach differ from those presented in [6], and because we implemented our solution together with a neighbor and routing protocol, we do not know how our solution compares to TDP and PDP. The relation between the savings of pruning (too much, or too little) and the degree of broadcast redundancy achieved, can be different, depending on the physical environment under consideration. If we take into account that more packets being broadcasted translate into more contention and collisions, we could have a different picture, depending on the number of broadcasts that are avoided.

V. Conclusions

We presented an enhanced dominant prunning approach that allows prunning redundant broadcasts even more than the conventional dominant prunning heuristic. Because EDP requires the two-hop neighborhood to determine the forwarder list, we built a neighbor protocol as part of AODV. By making the neighbor protocol part of AODV, the result is a more accurate view of the local topology, and therefore more accurate is the determination of the forwarder list.

AODV-EDP improves the packet delivery ratio for all the pause times tested in the 50 nodes and 30 flows scenario. The other protocols (standard AODV and OLSR) deliver fewer packets than AODV-EDP (the only exception is at $900s$ when OLSR has the same delivery ratio as AODV-EDP). The end-to-end delay is much better in AODV-EDP, and is less than half of the delays incurred by the other two protocols. The better delivery ratio and lower latency do not come for free, and AODV-EDP incurs more normalized routing load than OLSR, but less than standard AODV. The reduction of broadcast replicas by AODV-EDP and OLSR translates into a lower number of collisions at the MAC layer.

References

[1] C. Perkins, “Ad-hoc on-demand distance vector routing,” in *Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[2] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353. [Online]. Available: citeseer.nj.nec.com/johnson96dynamic.html

[3] A. Segall, “Distributed network protocols,” *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 23–35, Jan 1983.

[4] P.-J. Wan, K. M. Alzoubi, and O. Frieder, “Distributed construction of connected dominating set in wireless ad hoc networks,” in *Proceedings of INFOCOM 2002*, June 2002.

[5] H.Lim and C. Kim, “Flooding in wireless ad hoc networks,” *Computer Communications*, vol. 24, february 2001.

[6] W. Lou and J. Wu, “On reducing broadcast redundancy in ad hoc wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 1, no. 2, April-June 2002.

[7] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot, “Optimized link state routing protocol for ad hoc networks,” in *Proceedings of the IEEE International Multi Topic Conference*, December 2001.

[8] ——, “Optimized link state routing protocol,” IETF Internet draft, draft-ietf-manet-olsr-06.txt, Sep 2001.

[9] B. Clark, C. Colbourn, and D. Johnson, “Unit disk graphs,” *Discrete Math*, vol. 86, pp. 165–177, 1990.

[10] C. Perkins, E. Royer, S. R. Das, and M. K. Marina, “Performance comparison of two on-demand routing protocols for ad hoc networks,” *IEEE Personal Communications*, vol. 8, no. 1, pp. 16–28, Feb 2001.