This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Enhanced Dominant Pruning-based Broadcasting in Untrusted Ad-hoc Wireless Networks

Ashikur Rahman
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
Email: ashikur@cs.ualberta.ca

Pawel Gburzynski
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
Email: pawel@cs.ualberta.ca

Bozena Kaminska
School of Engineering Science
Simon Fraser University
Burnaby, BC, Canada V5A 1S6
Email: kaminska@sfu.ca

***Abstract*—Many protocols for ad-hoc wireless networks perform poorly in situations when node cooperation cannot be enforced. This may happen because of the lack of global authority over the nodes, or because of the environmental hostility. The net outcome of such scenarios is what we call a selfish behavior of nodes, which may spontaneously refuse to forward some packets. While unicast communication has been studied in this context, little has been done to improve the reliability of broadcasting in the face of limited trustworthiness of nodes. In this paper, we propose two enhancements of the popular broadcasting algorithms based on dominant pruning, which significantly improve the reachability of nodes in an unreliable ad-hoc wireless network. The improvement can be viewed as an enhancement of the network's fault tolerance, as it applies to most scenarios in which a node may fail to carry out its forwarding duties.**

keywords: **Ad-hoc Networks, Broadcast, Reliability, Reachability, Fault-tolerance, Medium Access Control.**

I. Introduction

A multi-hop ad-hoc wireless network is a collection of possibly mobile nodes devoid of authoritative centralized control. The implementation of all networking functions such as routing, broadcasting, multicasting, medium access control, assumes that all nodes cooperate in the communal task of maintaining connectivity. This assumption need not be always valid. Some nodes may be actually *selfish* and want to get a free ride without contributing to the community, or they may bona-fide fail to carry out their duties—because of blackouts or other errors. Arguably, the two cases are different. An intentionally uncooperative node may try to mislead its neighbors, e.g., by acknowledging a packet and then dropping it, while an honest failure may be easier to diagnose by the network and thus recover from. Note, however, that this need not always be the case. A node may acknowledge a received packet and then fail to forward it for objective reasons, e.g., the next-hop neighbor has become temporarily unreachable because of interference or other environmental factors. Many routing/forwarding schemes, especially those dealing with short packets (typical of many sensor networks) avoid persistent feedback (acknowledgments, explicit failure reports) for the sake of simplicity or sheer cost. Thus, there are many situations where a bona-fide failure is essentially indistinguishable from a premeditated one. Ad-hoc operation is especially promising in the areas involving sensor networks, where the nodes may be collaborative in principle, but, owing to the harshness of the environment, may exhibit failures amounting to the lack of collaboration.

Two basic data exchange modes in any network are *unicasting* and *broadcasting*. In both cases, the source relies on intermediate nodes for transporting the message to distant regions of the network. With most unicast schemes, every retransmission is addressed to a specific single neighbor, which makes it easy to ensure a high (formal) reliability of the transfer, e.g., via the four-way handshake of IEEE 802.11. With broadcasting, the notion of one hop is somewhat fuzzier because the packet legitimately can (and often should) be picked up by multiple neighbors of the forwarding node. Consequently, the question of reliable transmission is no longer a Boolean matter.

Broadcasting protocols are usually based on *node sets* rather than paths, which, combined with the somewhat confused notion of hop reliability, renders intentionally malicious node behavior difficult to target. Thus, the failure view of all node misbehavior scenarios in a broadcasting system is justified. First, honest failures are fundamentally more difficult to recognize with broadcasting than with unicasting, which means that the problem is important. Second, "smart" intentionally malicious misbehavior scenarios in a broadcasting system are not much more harmful than honest failures. Consequently, by effectively solving the failure problem in the broadcast case, we can hope to bring about a useful remedy for a wide range of problems.

The generic prerequisite of all practical broadcasting schemes is flooding. To be of value, this scheme must be contained by reducing the number of rebroadcasts to the minimum that guarantees complete (or high) coverage [19], [12], [16], [11], [21]. The main contribution of our work is an enhancement of one solution, the so-called called *dominant pruning* [11], [12], which takes into consideration the hostility of environment.

II. Prior work

Most of the research addressing node misbehavior has been aimed at protecting unicast routing protocols against malicious attacks. One can talk about three types of such protective schemes: cryptographic [15], incentive-based, and punishment-based (combined with detection methods).

1-4244-0353-7/07/$25.00 ©2007 IEEE

The first class includes SEAD [7], which is a secured variant of DSDV [17], whereby routes are signed with a one-way hash function, which makes it impossible for intermediate nodes to tamper with them. AODV [18] has been secured via a shared key accessed in a hierarchical manner, with the modified protocol named SAR [22]—for *Security-aware Ad-hoc Routing*. A secured modification of DSR [9], named ARIADNE [8], employs symmetric-key cryptography with a key management scheme dubbed TESLA. All those schemes suffer from three drawbacks: 1) they cannot help when a node has been compromised, 2) they do not address broadcast transmissions, and 3) they are costly to implement (being power and memory intensive), which renders them infeasible for low-cost sensor networks.

In the second class, [4] proposes two models based on the so-called *nuglets*. In one model, the sender pays for delivery by storing a certain number of nuglets in the packet header. Each intermediate node charges the packet one nuglet before forwarding it. When the packet runs out of nuglets, it is dropped. One problem with this scheme is the difficulty in estimating the correct number of nuglets needed by the packet to reach the destination. Clearly, the source would like to spend the minimum number of nuglets on each packet while still making sure that it will reach the destination. While in the unicast case, the sender may know the total number of hops needed by the packet to make it to the destination, attempts at extrapolating this kind of solution onto broadcast protocols are difficult. Another payment scheme has been proposed in [24]. As that solution depends on a central credit clearance service (and thus suffers from a single point of failure), it does not go well with ad-hoc networking.

Solutions in the thirs class postulate special tools (monitors) run by nodes, e.g., *watchdog* and *rater* proposed in [13] to mitigate routing misbehavior in DSR. The watchdog monitors the behavior of all nodes on the active paths, and the rater allows the nodes to avoid paths leading through misbehaving nodes. Other observer-based systems include COFIDANT [3], HADOF [23], CORE [14] and OCEAN [2]. None of them is directly applicable to guarding broadcast traffic against node misbehavior.

III. Dominant pruning

Let $N(u)$ be the set of all one-hop neighbors of node u. By $N^2(u) = \{v | v \in N(u) \vee \exists z[v \in N(z) \wedge z \in N(u)]\}$ we shall denote the set of all one-hop and two-hop neighbors of u. The dominant pruning algorithm [11] is a reliable broadcast protocol where a broadcast packet is guaranteed to reach every node in the network. Viewed as flooding containment scheme, dominant pruning limits the population of forwarding nodes to the so-called *connected dominating set*, which is any set of nodes $\mathcal{S}$ satisfying $\forall u\ [u \in \mathcal{S} \vee \exists v[v \in \mathcal{S} \wedge u \in N(v)]]$. By definition, every node in the network is either in $\mathcal{S}$ or is directly reachable by a node in $\mathcal{S}$; thus, when every node in $\mathcal{S}$ rebroadcasts a packet, it will reach 100% of nodes. Therefore, one solution to the optimal broadcast problem is to find a connected dominating set of the minimum size. Unfortunately, finding a minimum connected dominating set (MCDS) is NP-hard [10]. Lim and Kim [11] have proposed a heuristic algorithm called *dominant pruning*, which takes advantage of two-hop neighbor information. Each node u determines the so-called *forward list* as a subset of its one-hop neighbors whose transmissions will cover all two-hop neighbors of u. Then, when transmitting a broadcast packet, u explicitly indicates that it should be rebroadcast only by the nodes on the forward list. As it turns out, finding a minimum-size forward list is still NP-hard [6]. Thus, Lim and Kim have suggested the following greedy algorithm:

Suppose that v has just received a packet from node u. The packet's header includes the forward list F_u inserted there by u. If $v \in F_u$, v has to create its own forward list F_v to be inserted into the header of the rebroadcast copy. The node starts by constructing U_v, which is the set of uncovered two-hop neighbors of v. This set includes all two-hop neighbors of v that have not been covered by the received packet, i.e., $U_v = N^2(v) - N(v) - N(u)$. Note that every node knows the population of its two-hop neighbors; thus, $N(u)$ is known to v. Then, v sets $F_v = \emptyset$ and $B(u, v) = N(v) - N(u)$. The set $B(u, v)$ represents those neighbors of v which are possible candidates for inclusion in F_v. Then, in each iteration, v selects a neighbor $w \in B(u, v)$, such that w is not in F_v and the list of neighbors of w covers the maximum number of nodes in U_v, i.e., $|N(w) \cap U_v|$ is maximized. Next v includes w in F_v and sets $U_v = U_v - N(w)$. The iterations continue for as long as U_v becomes nonempty or no more progress can be accomplished (i.e., there is no way to further improve the coverage).

IV. Improvements

Suppose that some nodes in the connected dominating set (CDS) misbehave. As CDS is aimed at eliminating the redundancy of retransmissions, a node failing to fulfill its duties may partition the network.

A. Multicover dominant pruning (MDP)

The idea behind MDP is to introduce controlled redundancy into retransmissions—to increase the reachability without detecting or specifically circumventing the misbehaving nodes. To do this methodically, we suggest to reformulate the original covering problem as a *multicovering* problem. Suppose v has received a packet from its neighbor u, and v should rebroadcast this packet. As before, v determines the two sets U_v and $B(u, v)$. This time, however, we want the set F_v to be the minimum subset of $B(u, v)$ with the property that each node in U_v is covered at least m times by the nodes in F_v. Formally, the forwarding set F_v will be a minimum subset of $B(u, v)$ such that for every node $w \in U$, $|z : w \in N(z) \wedge z \in F_v| \geq m$. Clearly, straightforward dominant pruning is a special case of MDP with $m = 1$. Consequently, similar to the original single-cover problem, MDP is NP-hard.

Our experiments indicate that $m = 2$ is usually the best choice. For $m \geq 3$, the reachability does not improve much, but the number of rebroadcasting nodes increases

quite drastically. For example, in a network with 50 nodes, where the fraction of misbehaving nodes varies between 0–40incrementing m from 2 to 3 brings about a 1–2% improvement in reachability, while increasing by 10–15% the population of retransmitting nodes. Consequently, in this paper, we limit our discussion of MDP to the case $m = 2$.

Figure 1 presents an extension of the greedy algorithm introduced in section III for finding 2-covers. It should be obvious how to further extend the algorithm—for an arbitrary value of m. The algorithm starts by painting all nodes in U_v *white*. In every iteration, it finds the node $w_k \in B(u, v)$ with the largest number of neighbors that fall into U_v. Then w_k is added to the forwarding list F_k, and the neighbors covered by it are repainted: *white* to *gray*, *gray* to *black*. Color *white* means that the node has not been covered yet, *gray* indicates a single cover, and *black* stands for double cover. The process continues until all nodes in U_v are colored *black*.

1. $F_v = \emptyset$, $Z = \emptyset$.
2. For each node $w \in U_v$ do
3. $\quad Color(w) \leftarrow white$.
5. For each node $w_i \in B(u, v)$ do
6. $\quad$ Create the set S_i such that $S_i = N(w_i) \cap U_v$.
7. Let $K = \{S_1, S_2, ..., S_n\}$.
8. Suppose, S_k is the set such that,
9. $\quad |S_k| = \max_{S_i \in K}\{|S_i|\}$
10. If $|S_k| = \emptyset$ then exit.
11. $F_v = F_v \cup \{w_k\}$
12. For each node $x \in S_k$ do
13. $\quad$ If $Color(x) = white$ then
14. $\quad\quad Color(x) \leftarrow gray$
15. $\quad$ Else if $Color(x) = gray$ then
16 $\quad\quad Color(x) \leftarrow black$
17. $\quad\quad Z = Z \cup \{x\}$
18. $\quad\quad$ For each $S_i \in K$ do
19. $\quad\quad\quad S_i = S_i - \{x\}$
20. $K = K - \{S_k\}$
21. If $Z = U_v$ then exit.
22. Otherwise go to step 8.

Fig. 1. Multicover dominant pruning

B. Trust-based dominant pruning (TDP)

Our second proposed enhancement takes the perceived trustworthiness of nodes into account. While with straightforward dominant pruning, a node is solely ranked by the number of next-hop neighbors it can cover, the proposed trust-aware scheme factors in the trust rating, as measured by the node's neighbors, into the rank.

We equip each node into a *neighbor monitor*, which resembles the idea of *watchdog* [13] or *traffic observer* [23] with due modifications for broadcasting. These modifications are mostly drastic simplifications. In contrast to the watchdog and observer, which need to monitor all the nodes lying on the active path towards the destination, our neighbor monitor only keeps track of the immediate (one hop) neighbors. Moreover, it does not suffer from the *blackmailing* problem of watchdogs [13], which results from the fact that the unicast source is usually out of direct range of many intermediate nodes on the session's path.

Consider the operation of a selected node v. Initially, for the lack of knowledge, v assigns the default trust rating of 1 to all the nodes in its neighborhood. This rating gets updated as prescribed below.

The node maintains a pair of counters labeled sc_i (success count) and fc_i (failure count) for every one-hop neighbor u_i. Having transmitted a packet p, v keeps its signature in a queue and starts a timer for δ seconds. The signature of p contains a copy of the forward set F_v included in the packet header, as well as a sufficient amount of information to let the node recognize the same packet p, if received back. Within the δ-second interval v expects to receive back all the copies of p that will be rebroadcast by its neighbors. Having received such a copy from neighbor w_i, v increments sc_i. If the timer goes off and some neighbors from F_v have not rebroadcast p, v will increment their fc_i counters. Note that v can also assess the integrity of the rebroadcast packets, e.g., by storing and comparing their hash functions. Every Δ seconds the *neighbor rater* daemon wakes up and calculates the current trust rating T_i for every neighbor w_i for which $sc_i + fc_i \neq 0$: $T_i = sc_i/(sc_i + fc_i)$. Then, the average trust rating T_i^a of w_i is updated as: $T_i^a = \alpha \times T_i^a + (1 - \alpha) \times T_i$, where α is between 0 and 1. Finally, both sc_i and fc_i are reset to zero, and the deamon goes to sleep for another interval of Δ seconds.

One more attribute associated with a neighbor w_i is its *relevance* R_i^u, which indicates how important w_i is for relaying packets received by v from u to v's two-hop neighbors. The relevance is calculated separately for every node u delivering a packet to be rebroadcast by v as:

$$R_i = \frac{|S_i|}{\sum_{w_j \in B(u,v)} |S_j|}$$

where $S_i = N(w_i) \cap U_v$. Similar to the trust rating T_i, R_i is also a fractional number between 0 and 1. Unlike trust rating, the relevance depends on the previous-hop (delivering) neighbor.

Straightforward dominant pruning can be viewed as a rank-based scheme with relevance being the sole rank: one-hop neighbors are picked up in the decreasing order of R_i until they have covered all two-hop neighbors. With two simultaneous factors, the situation becomes more complicated because tuples $(T_i, R_i) \in \mathbb{R}^2$ are not explicitly ordered. Thus we suggest the following way of transforming them into ranks: (T, R) represents a higher rank than (T', R') if any of the following two conditions holds: 1) $T > T' + \rho$, 2) $|T - T'| \leq \rho \wedge R > R'$, where $\rho > 0$ is typically small, e.g., 0.05 or less.

This kind of ranking assigns a higher priority to the node's trustworthiness than to its relevance. Figure 2 illustrates the role of ρ, which determines half the width of the *region of uncertainty*. A node with trust rating of T^1 is assumed to

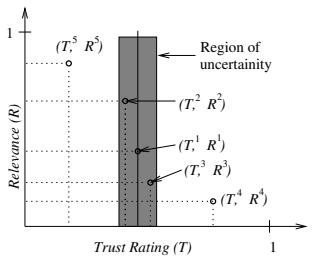


Fig. 2. Selection criteria in trust-based dominant pruning.

have the same trustworthiness as any other node with the trust rating between $T^1 - \rho$ and $T^1 + \rho$. Thus, the ordering of ranks (from high to low) is (T^4, R^4), (T^2, R^2), (T^1, R^1), (T^3, R^3), (T^5, R^5). The role of the uncertainty region is to compensate for "normal" failures and delays resulting from collisions and the inherently lossy character of the wireless medium. Note that owing to the unavailability of handshake in broadcast forwarding, this problem tends to be more pronounced here than in the unicast case.

1. $F_v = \emptyset$, $Z = \emptyset$.
2. For each node $w_i \in B(u, v)$ do
3. Create the set S_i such that $S_i = N(w_i) \cap U$.
4. Let $K = \{S_1, S_2, ..., S_n\}$.
5. For each node $w_i \in B(u, v)$ do
6. $E(w_i) == \frac{|S_i|}{\sum_{w_j \in B(u,v)} |S_j|}$.
7. Suppose, the trust ratings of nodes in $B(u, v)$ are:
8. $r(w_1), r(w_2), ...$
9. Find a node w_h having highest *trust rating* in $B(u, v)$.
10. Find the set of nodes, $N = \{w_1, w_2, ...\}$ such that,
11. $|r(w_h) - r(w_i)| < \rho$, for $w_i \in N$
12. Let w_k is the node with highest *relevance* in N.
13. $F_v = F_v \cup \{w_k\}$, $Z = Z \cup S_k$.
14. $B(u, v) = B(u, v) - \{w_k\}$.
15. For each node $S_i \in K$ do
16. $S_i = S_i - S_k$
17. $K = K - \{S_k\}$
18. If $Z = U$ or Z is unchanged then exit;
19. Otherwise go to step 5.

Fig. 3. Trust-based dominant pruning

Figure 3 lists the trust-based dominant pruning algorithm. In each iteration, v selects the most trustworthy neighbor w_h and identifies all neighbors falling within the uncertainty region of w_h (parameterized by ρ). Then, the most *relevant* node w_k is selected from that region and added to F_v; also, all the two-hop neighbors covered by it are added to Z (i.e., removed from U_v). The iterations continue until all nodes in U have been covered.

V. Experimental results

To evaluate the performance of dominant pruning and its two proposed enhancements, we built a detailed simulation model based on *ns*-2 [1] with wireless extensions. The distributed coordination function (DCF) of the IEEE standard 802.11 [5], was used as the MAC layer. The radio model characteristics were similar to Lucent's WaveLAN [20].

A. Scenario generation

We generate 10 random static scenarios of 50 nodes in a 1000m by 1000m flat two-dimensional space. The transmission range of each node is 250m (which is roughly $\frac{1}{4}$ of each dimension). A certain fraction of nodes misbehave by dropping packets without rebroadcasting them. For a particular scenario file, we first fix the number of well-behaved nodes N and the number of misbehaving nodes M, such that $N+M = 50$. The number N varies between 30 and 50, which means that the percentage of misbehaving nodes is between 0 and 40%. As we increase the number of misbehaving nodes, the larger set includes the misbehaving nodes from the previous scenario. This ensures a consistent evaluation of the impact of the enlarged population of misbehaving nodes by retaining all the "features" of the previous environment and simply making it worse. A similar approach was mentioned in [13].

B. Traffic generation

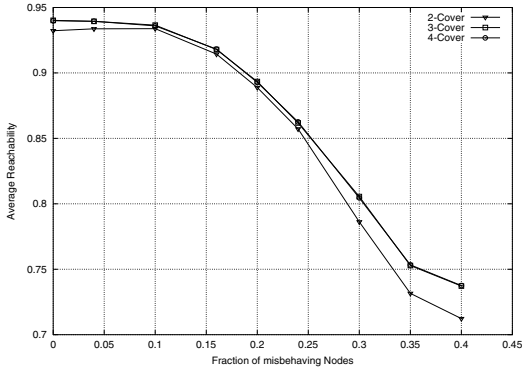
Traffic sources are CBR (continuous bit rate). Well-behaved nodes generate broadcast packets at regular 4-second intervals. Every simulation run lasts for 1000 seconds of virtual time. Traffic generators on different sources start at times uniformly distributed between 0 and 50 seconds. The packet size is fixed at 64 bytes.

C. Performance metrics

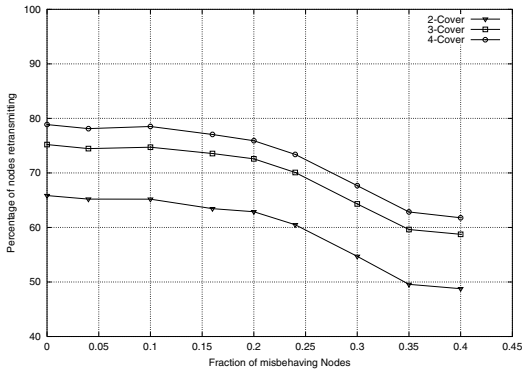
To see the effect of node failures on dominant pruning and on our proposed enhancements, we consider the following performance aspects: *Reachability*, *Redundancy* and *Delay*. Two measures of reachability are potentially interesting: (1) *coverage* defined as the average ratio of the number of well-behaved nodes receiving the broadcast packets to the total number of well-behaved nodes, and (2) *saturability* defined as the percentage of broadcast packets reaching all well-behaved nodes. *Redundancy* is formally defined as the percentage of all well-behaved nodes participating in rebroadcasting averaged over all broadcast packets. Finally, the delay metric is expressed as *broadcast latency*: the average amount of time elapsing from the moment a broadcast packet is generated at its source until the last well-behaved node in the network has received the packet.

D. Performance comparison

Let us begin by rationalizing our focus on $m = 2$ in multicover pruning. Figure 4 illustrates the impact of m on network coverage (a) and redundancy of retransmissions (b). While there is a visible (albeit small) difference between the cases $m = 2$ and $m = 3$, the difference between $m = 3$ and $m = 4$ is practically imperceptible. Moreover, note the drastic increase in redundancy from $m = 2$ to $m = 3$, which clearly nullifies the tiny gain in reachability for $m = 3$.

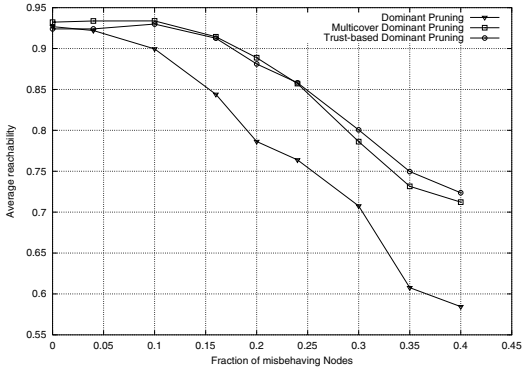


(a)

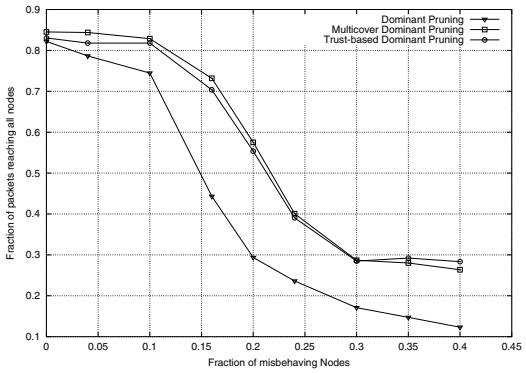


(b)

Fig. 4. Impact of m on the performance of multicover dominant pruning: (a) reachability, (b) redundancy.



(a)



(b)

Fig. 5. Comparison of reachability aspect: (a) coverage, (b) saturability.

Now let us compare the performance of dominant pruning with its two enhancements. The value of m for the multicover pruning algorithm is 2. For the trust-based scheme, the values of α and ρ are 0.2 and 0.05, respectively. The neighbor rater daemon is run every 5 broadcasts, which translates into $\Delta = 20$ seconds.

Figure 5 compares DP, MDP and TDP from the viewpoint of reachability. As we can see, TDP appears to slightly outperform MDP, with both enhancements offering a significantly better coverage, especially when the number of misbehaving nodes is high. A similar trend shows in saturability.

Ultimately, TDP turns up the clear winner when we look at Figures 6 and 7, which compare the redundancy and latency of the three solutions. Note that both MDP and TDP are more redundant than DP, which is expected: the algorithms intentionally incur redundancy to improve reachability. To properly interpret the graphs in Figure 6, we should realize that the spectacularly low redundancy of DP for a large number of misbehaving nodes results (at least partially) from the fact that the reachability of DP in that range is also very poor. If we scale the redundancy measure to reachability, i.e., consider *redundancy*/*coverage* as the metric of interest, then, the increase in redundancy for TDP over DP is about 17%. The same caution should be exercised when assessing the broadcast latency in Figure 7: the low latency of DP should be interpreted in the context of the poor coverage.

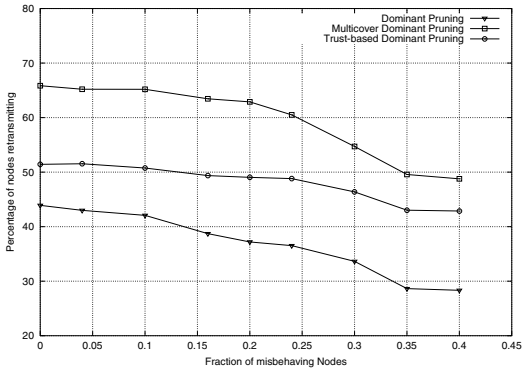


Fig. 6. Redundancy of DP, MDP and TDP.

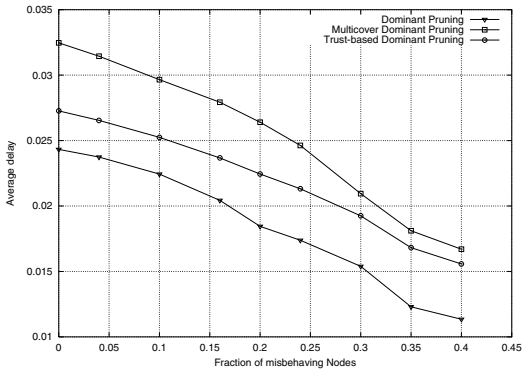


Fig. 7. Latency of DP, MDP and TDP.

VI. Conclusions and further work

We have presented two enhancements of dominant pruning (DP) aimed at improving the broadcast coverage of an ad-hoc wireless network under conditions of misbehaving nodes. Both schemes offer a significant improvement over the vanilla version of the algorithm, with TDP outperforming MDP. As TDP is the more complicated scheme (assuming certain non-trivial bureaucratic activities at the nodes), while MDP is a straightforward modification of DP, the latter can be recommended as a lightweight solution for networks built of cheap small-footprint nodes.

We plan to further investigate the proposed algorithms in more diverse networks, under realistic models of (unreliable) wireless channels and with node mobility factored in. Another interesting possibility is to consider other forms of node misbehavior.

Our present model assumes that all nodes (including the misbehaving ones) provide correct information regarding their neighbors. A malicious node might want to draw attention to itself (advertising many bogus neighbors), while a selfish node might want to do the opposite, i.e., report no neighbors—to shy away from its communal duties. An intentionally misbehaving node may try to avoid becoming a member of the connected dominating set, even though it qualifies (selfish behavior), or to be included in the connected dominating set, even though it doesn't qualify (malicious behavior).

One can analyze how those goals can be accomplished. The two-hop neighbor information, requisite in all schemes discussed in this paper, can be collected in different ways. One such way involves periodic *hello* messages sent out by every node, in which the node includes its neighbor list. When a node u receives a hello message from neighbor v, it marks v as its neighbor and learns about those two-hop neighbors of itself that are reachable via v. In a stable and reliable network, this process will eventually converge, and every node will have acquired the correct lists of its one-hop and two-hop neighbors. An intentionally selfish node may specify no neighbors in its hello messages and be thus excluded from the forward lists of its neighbors. A malicious node, on the other hand, may fabricate a bogus and large neighbor list, thus posing for a highly relevant node.

Another possible way of collecting two-hop neighbor information is to send simple hello messages with the time to live (TTL) of 2. Such a message will be rebroadcast exactly once and thus propagated to all two-hop neighbors of the sender. In this case, a selfish node would never forward a hello message thus pretending that it has no neighbors, while a malicious node might want to generate spurious hello messages with TTL$= 1$.

Note that when the network is reasonably dense, the false advertisements of intentionally misbehaving nodes may be inconsistent with reports from other neighbors and thus amenable to detection or at least suspicion. A node detecting inconsistencies of this kind may use them to rank the neighbors with respect to their trustworthiness. We are currently investigating the extent of such intentional attacks and studying possible countermeasures.

References

[1] The Network Simulator: NS-2: notes and documentation. http://www.isi.edu/nsnam/ns/.

[2] S. Bansal and M. Baker. Observationbased cooperation enforcement in ad hoc networks, 2003.

[3] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the CONFIDANT protocol: Cooperation of nodes — fairness in dynamic ad-hoc networks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, CH, June 2002. IEEE.

[4] Levente Buttyan and Jean-Pierre Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proceedings of the First IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, MA, USA, 2000.

[5] IEEE Standards Department. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 1997. IEEE standard 802.11-1997.

[6] S. Even. Graph algorithms, 1979. Computer Science Press.

[7] Y. Hu, D. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks*, I:175–192, 2003.

[8] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad-hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking* (MobiCom 2002), September 2002.

[9] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[10] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[11] H. Lim and C. Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, 2000.

[12] W. Lou and J. Wu. On reducing broadcast redundancy in ad -hoc wireless networks. *IEEE Transactions on Mobile Computing*, 1(2):111–123, 2002.

[13] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[14] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks, 2001.

[15] P. Papadimitratos and Z. Haas. Secure data transmission in mobile ad hoc networks, 2003.

[16] W. Peng and X. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Mobihoc*, 2000.

[17] C.E. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance Vector routing (DSDV) for mobile computers. In *Proceedings of SIGCOMM'94*, pages 234–244, August 1993.

[18] C.E. Perkins, E.M. Royers, and S.R. Das. Ad-hoc On-demand Distance Vector Routing (AODV), February 2003. Internet Draft: draft-ietf-manet-aodv-13.txt.

[19] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbour elimination based broadcasting algorithms in wireless networksing protocol for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, January 2002.

[20] B. Tuch. Development of wavelan, an ISM band wireless LAN. *AT&T Technical Journal*, 72(4):27–33, July/Aug 1973.

[21] J. Wu and F. Dai. Broadcasting in ad hoc networks based on self-pruning. In *Proc. of INFOCOM*, March 2003.

[22] Seung Yi, Prasad Naldurg, and Robin Kravets. Security-aware ad hoc routing for wireless networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001)*, Long Beach, CA, October 2001.

[23] W. Yu, Y. Sun, and K.J.R. Liu. Hadof: Defence against routing in mobile ad hoc networks. In *Proceedings of the IEEE INFOCOM*, volume 2, pages 1252–1261, 2005.

[24] S. Zhong, Y. Yang, and J. Chen. Sprite: A simple, cheat-proof, credit-based system for mobile ad hoc networks, 2002.