M-044

A Method for Selfish Node Detection and Avoidance in Wireless Ad Hoc Networks

Antoine Bourlon† Oyunchimeg Shagdar† Bing Zhang†

Abstract. Unlike military applications, there is no guarantee that users fully cooperate in civilian mobile ad hoc networks. One node could stop forwarding other nodes' packets to maximize its own benefits. Such a node is called a selfish node, whose behavior would largely degrade the performance of the entire network. In this paper, we propose an algorithm to detect, punish and avoid selfish nodes to alleviate their adverse effects. Our algorithm modifies the MAC and DSDV routing protocol to detect selfish nodes and reroute around them. We evaluate the accuracy and efficiency of the mechanism by running simulation on the ns-2 network simulator.

1. INTRODUCTION

Existing routing protocols presuppose a full cooperation of participating nodes. However, in a civilian ad hoc network, a user could disable the forwarding function of his terminal in order to maximize its own benefits.

A definition of selfish nodes in ad hoc networks using reactive routing protocols can be found in [1]: selfish nodes cooperate during the route discovery phase but drop data packets routed through them. Previous works have extended on-demand protocols to mitigate [2] or discourage [3] selfishness. However, a selfish node could also be uncooperative during the route discovery phase by dropping control packets so that it will not be chosen as hop node. On the other hand, with proactive protocols, nodes must advertise their routing table information in order to send or receive data packets. This can motivate a significant number of nodes to adopt a selfish behavior, leading to the deterioration of the network performance. Therefore we choose to extend a proactive protocol, Destination-Sequenced Distance-Vector (DSDV) [4], to make it react to the presence of selfish nodes (section 2). The selfish node detection is performed at the MAC layer (section 3). We show that these modifications allow a significant increase in the network performance with a limited number of "wrong" and "missed" detections (section 4).

2. SELFISH NODE DETECTION

We extend the IEEE 802.11MAC protocol so that it can detect selfish nodes, defined as follows:

Definition. A selfish node participates in the routing process by advertising its routing table information, but drops all data packets routed through it (we call these packets "forwarding packets", in opposition to "self-generated" packets)

In [2], the "watchdog" extension of the DSR protocol allows a node to control packet forwarding by comparing overheard packets to recently sent packets. We adapt this watchdog mechanism by implementing it at the MAC layer in order to start monitoring the next hop's behavior only after getting confirmation (MAC-level ACK) of a successful transmission. This way we avoid the uncertainty related to the link layer queue delay and the multiple RTS or data packet retransmissions. Also, we take congestion in consideration by (1) accepting any packet forwarding, whatever the source is, as a proof of the next hop's willingness to cooperate, and (2) by tolerating a temporary absence of forwarding, in case the next hop is busy sending self-generated packets: its link layer queue may be full of self-generated packets and forwarding packets dropped unintentionally.

A table called Monitoring Table (MT) is used to monitor a neighbor (*hop*, identified by the *hop_MAC* and *hop_IP* fields in the MT) expected to forward data packets towards a certain end destination (field *dest*). The (*dest*, *hop*) information allows the MAC layer to be aware of route changes. MT entries also contain a counter (*sent_cnt*) of packets transmitted to *hop* for *dest*, a flag (*self_pkt*) indicating if *hop* sent self-generated, and a timer. The detailed algorithm is as follows:

Monitoring Start. When a node A successfully transmits a packet to a neighbor B, which is not the end destination of the packet, A creates an MT entry corresponding to this destination (If such entry already exists, see *Monitoring Update*). *hop_MAC* and *hop_IP* are set with B's MAC and IP addresses. The entry's *timer* and *sent_cnt* are initialized (*sent_cnt* = 1).

Monitoring Update. When A sends a packet to B, if an entry already exists for the destination and *hop* has not changed since the last transmission, *sent_cnt* is incremented. If *hop* has changed, the entry is reset, with updated *hop_ IP* and *hop_MAC*. If the timer has expired, see *Monitoring Stop.*

Overhearing. A overhears all neighbors' transmissions. When A overhears a data packet from B it checks the entries containing B's MAC address. By comparing the packet source IP address and *hop_IP*, A can determine if it is a forwarding packet or a self-generated packet. If forwarding, the corresponding entries are deleted; if self-generated, the *self_pkt* flag is set for all corresponding entries.

Link Breakage. If A detects a link breakage with B (after multiple RTS or data packet retransmissions), all entries containing B as *hop* are deleted. Also if B's MAC layer detects a link failure with the next hop, B's DSDV agent should immediately be notified (by enabling feedback from the MAC layer). In reaction, B's DSDV will stop sending packets through the broken link and inform A of the link break.

Monitoring Stop. When A sends a packet to B, if the timer corresponding to the end destination has expired: if A has sent a significant number of packets to *hop* (*sent_cnt* > 5) but *hop* has sent neither self-generated (*self_pkt* not set) nor forwarding packets (entry exists), A considers *hop* selfish: the MAC layer alerts the DSDV agent. If *hop* has not forwarded any packet but sent self-generated packets (*self_pkt* set), A gives a second chance to *hop* to forward at least one packet before the next timeout. If *hop* persists in not forwarding packets, the MAC layer alerts the DSDV agent.

3. SELFISH NODE AVOIDANCE

We extend the DSDV protocol to avoid and punish selfish nodes detected by the MAC layer.

3.1. Routing table update and advertisement

DSDV, upon receiving a selfish detection alert from the MAC layer, updates the routing table and notifies the other nodes in the network. We add a new flag (*selfish_flag*) in each routing table entry to "remember" selfish nodes. This *selfish_flag* information has to be transmitted along with usual routing information. Routing table entries are updated and advertised as follows (we call *SN* the node reported selfish by the MAC layer):

† ATR Adaptive Communications Research Lab.

(1) Entry with *SN* as destination*:* set the metric to ∞ and set the *selfish_flag*.
(2) Entries with *SN* as next hop: set the metric to ∞ and increment the sequence number.
(3) Broadcast a triggered update for these changes.

A node that receives a DSDV update packet with the *selfish_flag* set for *SN* performs the same update process and propagates the information. Also, if a node already informed of *SN*'s selfishness receives a DSDV update packet with *selfish_flag* not set for *SN*, it broadcasts a route to *SN* with *selfish_flag* set, to inform the sender of the update packet.

3.2. Rerouting around selfish nodes

DSDV update packets from selfish nodes are dropped, so that selfish nodes will not be chosen as next hop anymore. They also become unable to receive data packets from other nodes (metric set to ∞). This penalty aims at discouraging selfishness. Data packets originated by selfish nodes are not dropped in order to avoid mutual accusations.

The sequence number of routing table entries using a selfish node as next hop is incremented so that this information is propagated in the network. A node in possession of a fresher sequence number or the destination itself will broadcast this new sequence number, allowing nodes to learn a route going around the selfish node (because update packets from selfish nodes are ignored).

4. PERFORMANCE EVALUATION

We evaluate the efficiency of our extensions by running simulations on the ns-2 network simulator. The simulation topology consists in 40 nodes moving in an 800m*800m area, according to the random waypoint mobility model [5], with a 5-second pause time and a 10m/s speed. We run 500-second simulations for 10 and 20 CBR (Constant Bit Rate) traffics with a sending rate of 10 packets (512-byte size) per second. In the simulations presented here neither senders nor receivers are selfish.

Fig. 1 represents the influence of the fraction of selfish nodes on the packet delivery ratio (PDR) when enabling (*On*) and disabling (*Off*) the MAC detection mechanism. The timeout for MT entries is fixed at 5 seconds. The extended version achieves a maximum PDR increase of 30%, with 10 traffics and 50% of selfish nodes. The improvement is still significant with a more congested network (20 sources).

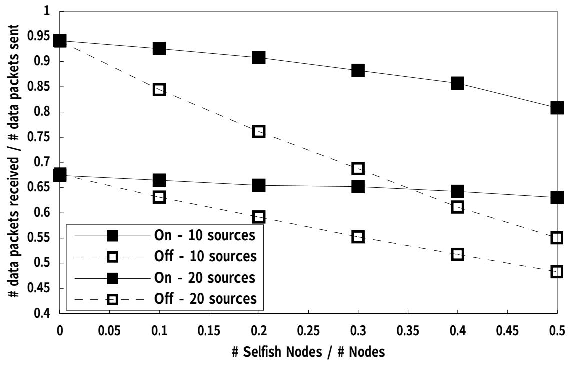

Fig. 1. Packet Delivery Ratio vs. fraction of selfish nodes

Fig. 2 represents the fraction of well-behaving nodes wrongly accused (*Wrong*) as well as the fraction of nodes that dropped packets because they were selfish but remained undetected (*Miss*). The number of nodes accused by mistake (because of link failure notifications lost in collisions, or data packets not overheard) increases with congestion but remains below 0.8%. The fraction of selfish nodes not detected (because they were only asked to forward packets for a short time) remains below 3.5%.

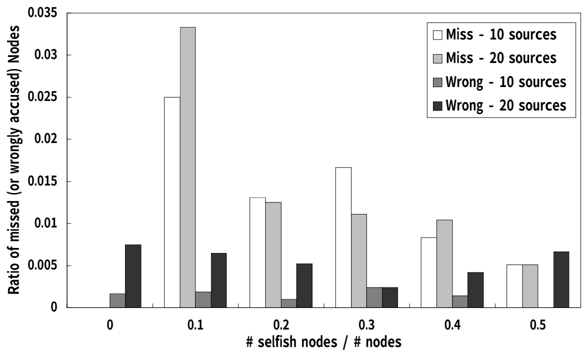

Fig. 2. Missed/wrong detections vs. fraction of selfish nodes

5. CONCLUSION

We proposed some extensions to the MAC and DSDV protocol so that nodes refusing to forward data packets can be detected and avoided. Simulation results showed that these extensions could significantly reduce the adverse effects of selfish nodes on the global network performance. However, the proposed scheme is only capable of detecting nodes dropping all forwarding packets. It enforces a forwarding rate of one packet every few seconds (5 seconds in the simulation presented). Such a low threshold is required when congestion is taken in consideration to limit the number of wrong accusations. Since nodes considered selfish are punished, the detection mechanism must differentiate between intentional drops and “forced” drops (due to congestion and link breaks). Control of packet forwarding could be more efficient if a forwarding policy was imposed to nodes.

References

[1] M. Hollick, J.Schmitt, C. Seipl and R.Steinmetz. On the effect of Node Misbehavior in Ad Hoc Networks, in *Processdings of IEEE International Conference on Communications (ICC),* June 2004.
[2] S. Marti, T. Giuli, K. Lai and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks, in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom),* Boston, August 2000.
[3] S. Sundaramurthy and E. M. Belding-Royer. The AD-MIX Protocol for Encouraging Participation in Mobile Ad hoc Networks, in *Proceedings of the International Conference on Network Protocols (ICNP),* Atlanta, November 2003.
[4] C.E. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers, in *Proceedings of the SIG-COMM '94 Conference on Communications Architectures, Protocols and Applications,* August 1994.
[5] J. Broch, D.Maltz, D. Johnson, Y.C. Hu and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols, in *Proceedings of Mobicom '98,* Dallas, October 1998.