

An Algorithm to Detect Packet Forwarding Misbehavior in Mobile Ad-Hoc Networks

Oscar F. Gonzalez, Michael Howarth, George Pavlou

Center for Communications Systems Research

University of Surrey

Guildford, UK

O.Gonzalez-Duque@surrey.ac.uk, M.Howarth@surrey.ac.uk, G.Pavlou@surrey.ac.uk

Abstract—Devices in a Mobile Ad Hoc network (MANET) may misbehave by dropping packets that are forwarded in the network, for example because they are malicious and deliberately intend to disrupt the network, or because they are selfish and wish to conserve their own limited resources such as power. In this paper, we present a mechanism that enables the detection of nodes that exhibit packet forwarding misbehavior. We present initial evaluation results that demonstrate the operation of our algorithm and show that it effectively detects nodes that drop a significant fraction of packets.

Keywords—mobile ad hoc network, misbehavior detection, packet forwarding.

I. INTRODUCTION

The wireless nature and inherent features of mobile ad hoc networks make them vulnerable to a wide variety of attacks by misbehaving nodes. Such attacks range from passive eavesdropping, where a node tries to obtain unauthorized access to data destined for another node, to active interference where malicious nodes hinder network performance by not obeying globally acceptable rules. Misbehavior can be divided into two categories [1]: routing misbehavior (failure to behave in accordance with a routing protocol) and packet forwarding misbehavior (failure to correctly forward data packets in accordance with a data transfer protocol). In this paper we focus on the latter type of misbehavior. Our approach consists of an algorithm that enables packet forwarding misbehavior detection through the *principle of flow conservation* [3]. Our scheme is not tightly coupled to any specific routing protocol and, therefore, it can operate regardless of the routing strategy adopted. Our criterion for judging a node is the estimated percentage of packets dropped, which is compared against a pre-established misbehavior threshold. Any node dropping packets in excess of this threshold is deemed a misbehaving node while those below the threshold are considered to be correctly behaving.

In this paper we first present a framework and an algorithm and protocol that deal with this attack. We then demonstrate through simulations that an appropriate selection of the misbehavior threshold allows for a good discrimination between misbehaved and well-behaved nodes in a network that is affected by black hole attacks, where malicious nodes drop all packets they receive, and/or gray hole attacks, where they

drop packets selectively or in a probabilistic manner in an attempt to circumvent the security measures in place.

Section II of this paper describes our algorithm for packet forwarding misbehavior detection, and Section III presents an initial performance evaluation. Finally, the paper is summarized in Section IV.

II. DETECTING PACKET FORWARDING MISBEHAVIOR

Our work provides a novel approach to securing the data forwarding functionality in mobile ad hoc networks. We propose an approach that takes advantage of the principle of flow conservation in a network. This states that all bytes/packets sent to a node, and not destined for that node, are expected to exit the node. In this Section we first present, from a theoretical point of view, how this principle works assuming it is implemented in an ideal network, and then we demonstrate that by making some reasonable assumptions and adaptations, our algorithm can cope with the practical problems that are encountered in real MANETs.

A. The Principle of Flow Conservation

We formally introduce the principle of flow conservation over an ideal static network model:

- Let v_j be a node such that $v_j \in V$, where $V = \{v_1, v_2, v_3 \dots v_N\}$ is the set of all nodes in the network, N is the total number of nodes in the network, and $j = 1, 2, 3 \dots N$.
- Let U_j be the subset of nodes in the network which are neighbors of v_j , i.e. U_j is the neighborhood of v_j . It follows that $v_j \notin U_j$ and also $U_j \subset V$.
- Let Δt be the period of time elapsed between two points in time t_0 and t_1 such that $\Delta t = t_1 - t_0$.
- Let T_{ij} be the number of packets that node v_i has successfully sent to node v_j for v_j to forward to a further node; $v_i \in U_j, v_j \in U_i, i \neq j$ and $T_{ij}(t_0) = 0$.
- Let R_{ij} be the number of packets that node v_i has successfully received from node v_j that did not originate at v_j ; $v_i \in U_j, v_j \in U_i, i \neq j$ and $R_{ij}(t_0) = 0$.

If all nodes $v_j \in V$ remain static for a period of time Δt during which no collisions occur in any of the transmissions over an ideal (noiseless) wireless channel, then for a node v_j :

$$\sum_{\forall i|v_i \in U_j} R_{ij}(t_1) = \sum_{\forall i|v_i \in U_j} T_{ij}(t_1) \quad (1)$$

This equation states the fundamental premise of the flow conservation principle in an ideal static network, and is applied to packets rather than individual bytes. It states that if all neighbors of a node v_j are queried for i) the amount of packets sent to v_j to forward and ii) the amount of packets forwarded by v_j to them, the total amount of packets sent to and received from v_j must be equal.

In practice MANETs exhibit conditions that are far from ideal. First of all, the wireless channel is error prone and packets get lost while in transit. Secondly, collisions happen when the network uses protocols where nodes have to compete for the medium, such as when the link layer protocol is based on the distributed coordination function (contention period) of the IEEE 802.11 a/b standard. In order to allow equation (1) to hold our algorithm requires the use of a reliable MAC protocol such as IEEE 802.11, MACA [4] or MACAW [5], and the rest of this paper assumes that such a reliable MAC protocol is used.

A reliable MAC protocol at the link layer acknowledges each successfully transmitted packet and thus transmitter and receiver can maintain synchronized values of their metrics T_{ij} and R_{ij} . The use of a reliable MAC protocol in conjunction with the conservation of flow principle means that we are not susceptible to problems that arise when overhearing other nodes' transmissions. Thus, problems such as ambiguous collisions, receiver collisions, and the ability of a node to control its transmission power, which affect the approaches proposed in [1] and [2], do not exist in our approach. *Ambiguous collisions* occur when a node v_1 is trying to determine if another node v_2 is properly forwarding a packet. It may happen that node v_2 forwards the packet to a further node v_3 , which is out of the transmission range of v_1 , while a second transmission prevents v_1 from overhearing the forwarded packet, thus v_1 will not know if the packet was forwarded. On the other hand, in the *receiver collision* problem v_2 forwards the packet to v_3 at which point a collision occurs. Node v_1 is unaware of such a collision and assumes that the packet was forwarded even if v_2 does not attempt a retransmission. Another common problem is caused by nodes capable of controlling their transmission power. Thus, v_2 can transmit with enough power for v_1 to overhear but not enough power for v_3 to receive it, leaving v_2 unaware of the situation. All these weaknesses, which can be used by malicious nodes to disrupt the network, are due to the fact that overhearing is used by nodes to check for misbehavior in other nodes. In our algorithm the nodes that maintain statistics that are used to determine whether the forwarding was properly made are the nodes actively involved in the transmission process, i.e. the transmitter and the receiver of each transmission.

However, a node may exhibit malicious behavior even if it is not purposefully doing so. For example, an overloaded node may temporarily lack the CPU cycles, buffer space or bandwidth to forward packets [2]. In addition, some reactive routing protocols, e.g. AODV, cause buffered packets to be dropped if they go through a path that is even temporarily

unavailable. For these reasons equation (1) cannot be applied in a rigorous manner and a threshold needs to be established to account for packets dropped by a node through no fault of its own. Equation (2) reflects this change:

$$(1 - \alpha_{threshold}) \sum_{\forall i|v_i \in U_j} R_{ij}(t_1) \leq \sum_{\forall i|v_i \in U_j} T_{ij}(t_1) \quad (2)$$

The $\alpha_{threshold}$ factor can take values between 0 and 1 and as we shall see plays an important role in the detection power of our proposed algorithm, i.e. the capability of the algorithm to detect misbehaving nodes. The lower $\alpha_{threshold}$ is the more likely it is that our algorithm detects any malicious behavior. However, it also means that the probability of a false detection increases. A false detection occurs when the result of a single evaluation of a node mistakenly determines that the node appears to be misbehaving. Therefore, fine tuning is required to reach a fair point in this tradeoff.

B. Algorithm

In MANETs the neighborhood U_j of a node v_j changes dynamically over time, making it difficult to determine those nodes that have transmitted or received packets to or from a node v_j . Our scheme overcomes this problem by means of a limited broadcast that tracks down nodes that have been in contact with node v_j . The core parts of our algorithm are detailed in the pseudocode shown in figure 1. A node v_i maintains a table with two metrics T_{ij} and R_{ij} , which contains an entry for each node v_j to which v_i has respectively transmitted packets to or received packets from. Node v_i increments T_{ij} on successful transmission of a packet to v_j for v_j to forward to another node, and increments R_{ij} on successful receipt of a packet forwarded by v_j that did not originate at v_j .

All nodes in the network continuously monitor their neighbors and update the list of those they have heard recently (Fig. 1.a). If the ID of an overheard node is not included in the table of overheard nodes a new entry is created. Otherwise, the existing entry is updated with a timestamp corresponding to the time the node was last overheard. Upon the creation of a new entry, a node schedules a task/event to check the behavior of the node whose ID has been saved in the new entry. Nodes randomly select a period of time between T_{min} and T_{max} to schedule the behavior checking task. This random selection seeks to reduce the possibility of two or more nodes starting a behavior check on the same node at the same time, wasting network bandwidth, battery energy and other network resources.

When a scheduled task is triggered in node v_k to check v_j 's behavior (Fig. 1.b), node v_k broadcasts a *metrics request packet (MREQ)* with TTL = 1 in the IP header. An MREQ includes the ID of the node emitting the request (SRC_ID), the ID of the node whose behavior is to be checked (CHK_ID), an MREQ_ID and a timestamp indicating the time at which the task was triggered. The MREQ_ID is used in the same way as in some routing protocols which base their route discovery phase on broadcasting. If a node sees an MREQ that has the same MREQ_ID and SRC_ID of a packet seen before, the MREQ is dropped. This technique prevents flooding packets from traversing a zone of the network more than once. The timestamp, on the other hand, is used to resolve conflicts when

two nodes start a behavior check on the same node at almost the same time. In such cases, nodes can see which of the packets corresponds to the earlier triggered task and disregard the other. This does not require accurate synchronization of the nodes' clocks; approximate synchronization is all that need be assumed.

a. OVERHEARING

```

if node  $v_k$  overhears a node  $v_j \in U_k$ 
. if node  $v_j$  is not in  $v_k$ 's table of overheard nodes
. . add node  $v_j$  to  $v_k$ 's table of overheard nodes
. . schedule an event to check  $v_j$ 's behavior
. else
. . update last time node  $v_j$  was heard
. endif
endif

```

b. INITIATE BEHAVIOR CHECK

```

if in node  $v_k$  an event to check node  $v_j$ 's behavior is triggered
. send a metrics request packet (MREQ) with node  $v_j$ 's ID
. schedule another event to check  $v_j$ 's behavior again at  $t+T_{max}$ 
endif

```

c. REQUEST HANDLING

```

if node  $v_k$  receives a metrics request for node  $v_j$ 
. if node  $v_k$  has node  $v_j$  in its table of overheard nodes
. . rebroadcast metrics request packet (MREQ)
. . reschedule any event to check  $v_j$ 's behavior
. . if node  $v_k$  has metrics for node  $v_j$ 
. . . send a metrics reply (MREP) back to the requesting node
. . endif
. else
. . ignore request
. endif
endif

```

d. REPLY HANDLING

```

if a request was sent out
. while there are more replies to be received for node  $v_j$ 
. . receive reply
. . add received metrics to totals
. endwhile
.
. if  $\sum_{v_i | v_i \in U_j} R_{ij} < (1 - \alpha_{threshold}) \sum_{v_i | v_i \in U_j} T_{ij}$ 
. . node  $v_j$  is misbehaving (detection)
. else
. . node  $v_j$  is not misbehaving (non-detection)
. endif
endif

```

Figure 1. Our algorithm pseudocode.

The handling of requests (Fig. 1.c) is the heart of our limited broadcast algorithm. When a node receives an MREQ it first checks if the CHK_ID is in its table of overheard nodes; if it is not the node ignores the MREQ and discards the check. However, if the CHK_ID appears in its table then it rebroadcasts the MREQ with TTL = 1 in the IP header. Setting the TTL to one allows our algorithm to control how far the

broadcast of the MREQ is to go, instead of leaving this task to the IP protocol. Thus, every MREQ travels only one hop at a time, and is then analyzed and rebroadcast if the protocol so determines. By following this algorithm, our protocol is capable of tracking down nodes that have been in contact with the checked node, as illustrated in Figure 2. We assume transmissions can be overheard by vertically, horizontally and diagonally adjacent nodes. In the Figure, node v_7 is first in position a where it can be overheard by nodes $v_1, v_2, v_3, v_6, v_8, v_{11}, v_{12}$ and v_{13} . Each of these nodes makes an entry in their table of overheard nodes when v_7 first transmits and each of them schedules a task to check its behavior. At some point in time, v_7 decides to move following the path depicted in Figure 2 coming in contact with nodes $v_{14}, v_{17}, v_{18}, v_{19}, v_{20}, v_{23}, v_{24}$ and v_{25} . It finally stops in position b . In the Figure the scheduled behavior check initiation task (Fig. 1.b) in v_8 is the first to be triggered and the limited broadcast commences. All nodes that have overheard node v_7 re-broadcast the MREQ, whereas nodes such as v_4, v_9 and v_{15} also receive the MREQ but ignore it because they have not overheard node v_7 .

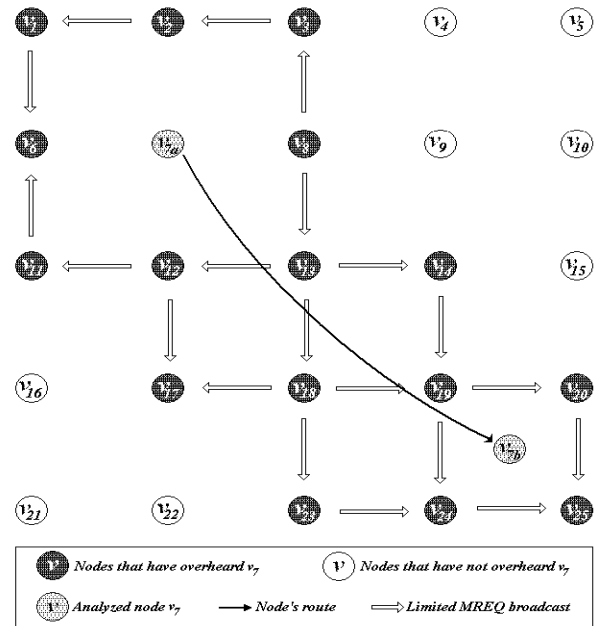


Figure 2. Example of limited broadcast to track down nodes that have overheard node v_7 .

Once a node has decided whether to continue or not broadcasting a MREQ, it reschedules any pending task to check the behavior of the checked node specified in the CHK_ID field of the MREQ. The new behavior checking task is scheduled in the same way as when a new entry is made in the table of overheard nodes, i.e. a period of time is randomly selected between T_{min} and T_{max} . In this way if the random selection is uniformly distributed and the average frequency with which an active node's behavior is checked is:

$$avg_freq = \frac{1}{(T_{min} + T_{max})/2} = \frac{2}{(T_{min} + T_{max})} \quad (3)$$

The last step a node takes when it receives a MREQ is to check if it has any metrics (R_{ij} or T_{ij}) relating to the node being

checked. If any of the metrics has a value other than zero the node returns a metrics reply packet (MREP) (Fig. 1.c) containing the metrics, but if the value of both metrics is zero then the node does not send back any response. In our scheme the task of finding a route back to the node that originated the MREQ has been left to the underlying routing protocol.

Reply handling is executed in the node that initiated the MREQ. This node, v_s in Fig. 2, waits for a period of time in order to give all nodes with metrics about the checked node the opportunity of replying. When the time expires, the node checks the behavior of the analyzed node by verifying that equation (2) holds (Fig. 1.d). If it does not, it flags the checked node as a misbehaving one; this is a detection. Using a single detection to accuse a node is not sufficient since such an algorithm may lead to false accusations against correctly behaving nodes. A scheme in which multiple detections by different nodes are necessary to accuse a node is fairer to well-behaved nodes, while keeping a high probability of correctly accusing misbehaving nodes. Thus, a system whose goal is to accuse misbehaving nodes, perhaps in order to punish them by withdrawing their right to transmit on the network, could use an approach similar to the distributed consensus mechanism proposed in SCAN [1].

III. EVALUATION

We perform our simulations using the GloMoSim simulation package. The results presented for each value are the average of 5 simulation runs. Our simulation parameters take the following values: i) The number of nodes in the network is 20, ii) Nodes are placed over an area of 200 x 200 squared meters, iii) Nodes move according to the random waypoint mobility model with a constant speed of 10ms^{-1} , iv) The pause time takes a value that is exponentially distributed with mean 30 seconds, v) the wireless transmission range of every node is 100 meters, vi) the link capacity is 2 Mbps, vii) the MAC layer protocol is the IEEE 802.11 DCF, viii) the underlying routing protocol is AODV, and ix) the total simulation time for each scenario is 300 seconds. The network was set-up with 25% of nodes misbehaving by not forwarding all packets. Nodes check the behavior of active nodes within a period chosen uniformly between 15 and 30 seconds, and keep any overheard node in their tables for 60 seconds after the last time they are heard. The principal metric in our tests is the percentage of detections, which is assessed in terms of misbehavior threshold.

We consider the precision of our misbehavior detection algorithm in terms of the misbehavior threshold, which is the parameter $\alpha_{\text{threshold}}$ in equation (2), i.e. the maximum percentage of packets that a node is allowed to drop without being detected as a misbehaving node. In order to see properly the effect of the misbehavior threshold on nodes, simulations were carried out with nodes dropping packets with probabilities of 0%, 10%, 20%, 50% and 70% while the misbehavior threshold was varied from 0% to 100%.

Figure 3 depicts the percentage of detections as a function of the misbehavior threshold for nodes exhibiting different probabilities of misbehavior. It can be inferred from the graph that the criterion to select an adequate misbehavior threshold

depends on the level of trust required in the network. The lower the threshold is the more packets nodes need to forward to be considered well-behaved. However, since characteristics inherent to MANETs such as mobility and the noisy wireless medium can cause some packets to be lost (including packets of our own protocol), it also means that an increasing number of correctly behaving nodes can be falsely detected as misbehaving ones. Finally, it can also be seen from Fig. 3 that selecting a misbehavior threshold equal to a node's misbehaving probability prevents our approach from identifying misbehaving nodes with certainty, i.e. the probability of detection is approximately 50%. These occurrences are all contained in the zone between 40% and 60% probability of detection in the figure.

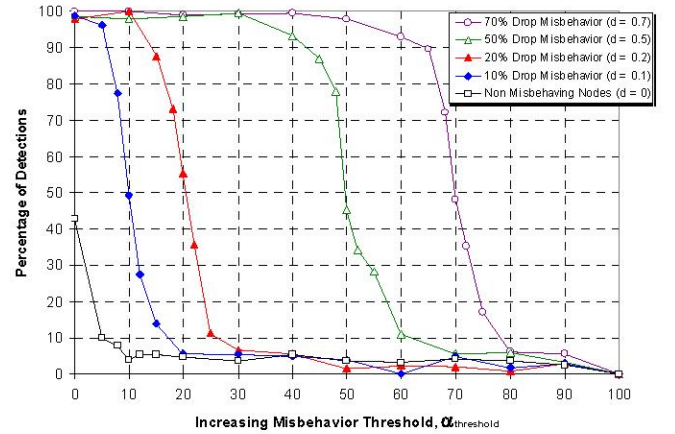


Figure 3. Percentage of positive detections as a function of the increasing misbehavior threshold (mean node speed = 10ms^{-1}).

IV. SUMMARY

In this paper we have presented an algorithm that is capable of detecting packet forwarding misbehavior. The algorithm does not require high density networks in which many nodes can overhear each others' received and transmitted packets, but instead uses statistics accumulated by each node as it transmits to and receives data from its neighbors.

V. REFERENCES

- [1] H. Yang, J. Shu, X. Meng, and S. Lu, "SCAN: Self-organized network-layer security in mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, issue 2, pp. 261-273, February 2006.
- [2] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile ad hoc networks," *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking*, pp. 255-265, August 2000.
- [3] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson, "Detecting disruptive routers: a distributed network monitoring approach," *Proceedings of the 1998 Symposium on Security and Privacy*, pp. 115-124, May 1998.
- [4] P. Karn, "MACA - a new channel access method for packet radio," *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pp. 134-240, September 1990.
- [5] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs," *Proceedings of the ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, vol. 24, issue 4, pp. 212-225, 1994.