Detection of Packet Forwarding Misbehavior in Mobile Ad-Hoc Networks

Oscar F. Gonzalez[1], Michael Howarth[1], George Pavlou[1]

[1] 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. Mobile Ad Hoc networks (MANETs) are susceptible to having their effective operation compromised by a variety of security attacks. Nodes may misbehave either because they are malicious and deliberately wish to disrupt the network, or because they are selfish and wish to conserve their own limited resources such as power, or for other reasons. In this paper, we present a mechanism that enables the detection of nodes that exhibit packet forwarding misbehavior. We present evaluation results that demonstrate the operation of our algorithm in mobile ad hoc environments and show that it effectively detects nodes that drop a significant fraction of packets.

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

1. Introduction

The wireless nature and inherent features of mobile ad hoc networks makes 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. For instance, a node can behave maliciously by not forwarding packets on behalf of other peer nodes. However, when a node exhibits malicious behavior it is not always because it intends to do so. A node may also misbehave because it is overloaded, broken, compromised or congested in addition to intentionally being selfish or malicious [3,11].

Misbehavior can be divided into two categories [3]: 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. Our approach consists of an algorithm that enables packet forwarding misbehavior detection through the *principle of conservation of flow* [25]. 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.

Our scheme detects misbehaving nodes (whether selfish, malicious or otherwise) capable of launching two known attacks: the simplest of them is the black hole attack. In this attack a misbehaving node drops all the packets that it receives instead of normally forwarding them. A variation on this is a gray hole attack, in which nodes either drop packets selectively (e.g. dropping all UDP packets while forwarding TCP packets) or drop packets in a statistical manner (e.g. dropping 50% of the packets or dropping them with a probabilistic distribution). Both types of gray hole attacks seek to disrupt the network without being detected by the security measures in place.

In this paper we first present a framework and a relevant 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, as well as providing robustness against different degrees of node mobility in a network that is affected by black hole and/or gray hole attacks.

The rest of this paper is organized as follows. Section II describes related work in the area of MANET security. Section III specifies our assumptions on the network and security models and clarifies the terminology adopted. Section IV describes our algorithm for packet forwarding misbehavior detection, and Section V presents a performance evaluation. Finally, the paper is concluded in Section VI.

2. Related Work

In this Section we initially look at ways of protecting routing protocols against misbehaving nodes and then review work that attempts to detect misbehavior in data packet forwarding.

Some research effort has been focused on the development of new routing protocols whose objective is to protect the network from security threats that were not addressed by work preceding them. The Secure Routing Protocol (SRP) [7] and Authenticated Routing for Ad hoc Networks (ARAN) [8] assume the existence of *a priori* relationships in a network: in the case of SRP between the two communicating nodes, and for ARAN between each node in the network and a certificate server. Both protocols perform an end-to-end authentication and intermediate nodes are not allowed to reply to route requests even if they know a route to the destination. However, *a priori* relationships in MANETs may not exist. These approaches secure the path discovery and establishment functionality of routing protocols, and our approach complements them by securing the data forwarding functionality.

The routing protocol proposed in [16] offers resilience to diruption or degradation of the routing service by an algorithm that allows the detection of a malicious link after log n faults have occurred on a path, where n is the hop length of the path. In [19] each node is able to detect signs of intrusion locally and neighboring nodes collaborate to further investigate malicious behavior. In both these approaches a node uses its own data to identify another node as an intruder. In contrast, in our approach a node detects anomalies in packet forwarding based on data acquired by other nodes in the network as well as on its own data, thus potentially obtaining a more balanced evaluation of a node's behavior.

Secure routing protocols have also been proposed based on existing ad hoc routing protocols. These eliminate some of the optimizations introduced in the original routing protocols because they can be exploited to launch different types of attacks. Examples of such protocols are the secure efficient distance vector (SEAD) routing [6] which is based on the destination sequenced distance vector (DSDV) [12], the secure ad-hoc on-demand distance vector (SAODV) routing protocol [9, 10] based on AODV [13, 14], and the secure on-demand routing protocol for ad hoc networks (Ariadne) [2] based on the dynamic source routing (DSR) protocol [15] and the timed efficient stream loss-tolerant authentication (TESLA) protocol proposed in [17]. A second work extending DSR to provide it with security mechanisms is CONFIDANT (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks) [18]. As with SRP and ARAN these protocols can be coupled with our approach, which is not routing protocol dependent, to offer an improved security solution.

Attack patterns have also been the object of research. In [4] and [5] the authors propose a framework for misuse detection which divides the nodes in a network into two categories: insiders and outsiders. Insiders are always well-behaved nodes in charge of performing network tasks. Such nodes belong to trusted users. Outsiders only communicate using the network and are always invigilated by a subset. Unfortunately, this framework assumes the existence of trust relationships prior to a MANET formation, which goes against their dynamic and spontaneous nature. In this regard, our protocol does not assume prior relationships between nodes in a network and it only evaluates their behavior after the MANET has been formed and the packet transmission has commenced.

Other work has tried to protect security mechanisms against attacks, such as [11] where the authors distinguish between two types of threats: threats to the basic mechanisms of ad hoc networks, such as routing, and threats to the security mechanisms such as key establishment and management. Their proposed algorithm uses certificates stored in repositories to find a certificate chain between the communicating nodes. Although simulations show an acceptable performance, this method requires a considerable amount of memory in every node in the network.

There has also been some work that aims to protect data packet forwarding against malicious attacks in order to provide reliable network connectivity. The final part of this section describes some approaches that detect malicious behavior in the data forwarding phase. WATCHERS (Watching for Anomalies in Transit Conservation: a Heuristic for Ensuring Router Security) [25] is a protocol designed to detect disruptive routers in fixed networks through analysis of the number of packets entering and exiting a router. Although WATCHERS is based on the principle of conservation of flow in a network in the same way as our proposed algorithm, its design focuses only on fixed networks and is not applicable to mobile ad hoc networks.

The Secure Message Transmission (SMT) and Secure Single Path (SSP) protocols are both introduced in [20]. In SMT a message that is to be sent towards a destination is first divided in N parts and then sent by N independent paths. Each part carries a limited amount of redundancy in such a way that only M parts, where $M<N$, are needed at the destination to recover the whole message. In SSP one path is used at a time and the source tries a different path each time the communication is not successful. However, SMT is very bandwidth-intensive, and these protocols do not attempt to find the source of the packet loss. Our protocol, on the contrary, identifies any source(s) that appear to be causing packet losses, allowing for their isolation at a later stage if necessary.

In [24] the authors look at traffic transmission patterns between any two communicating nodes in order to facilitate verification by a receiver. Such traffic patterns can be analyzed, and a misbehaving node can be found, if they are used in concert with suboptimal techniques at the medium access control (MAC) layer that preserve the statistical regularity from hop to hop. This work, however, has a very narrow scope of application due to its MAC layer assumptions to preserve statistical regularity, and thus it is very unlikely for it to be useful in scenarios other than military applications.

SCAN (self-organized network layer security in mobile ad hoc networks) [3] focuses on securing packet delivery. SCAN nodes monitor their neighbors by listening to packets that are forwarded to them. The SCAN node maintains a copy of the neighbor's routing table and determines the next-hop node to which the neighbor should forward the packet; if the packet is not overheard as being forwarded, it is considered to have been dropped. In contrast, in our algorithm nodes do not need to overhear transmissions to and from any neighbor in order to detect misbehavior. In SCAN each node must possess a valid token to be able to interact with the network. SCAN develops this idea from [21] where tokens are used to give a valid key to a new node entering the network. Similar techniques have also been studied in various papers such as [22], where they are used to achieve distribution of trust throughout a network.

Finally, in [23] a system that can mitigate the effects of packet dropping is proposed. This is composed of two mechanisms that are kept in all network nodes: a watchdog and a pathrater. The watchdog mechanism identifies any misbehaving nodes by promiscuously listening to the next node in the packet's path. If such a node drops more than a predefined threshold of packets the source of the communication is notified. The pathrater mechanism keeps a rate for every other node in the network it knows about. A node's rate is decreased each time a notification of its misbehavior is received. Then, nodes' rates are used to determine the most reliable path towards a destination. However, the watchdog might not detect a misbehaving node in the presence of ambiguous collisions, receiver collisions or nodes capable of controlling their transmission power. Such weaknesses are the result of using promiscuous listening to determine whether a node has forwarded a packet or not. Our approach does not have these same weaknesses since it is based on metrics obtained from nodes that are actually sending and receiving packets to and from the node whose behavior is under evaluation, as explained in section IV.A.

3. Model Assumptions

The physical layer of a wireless network is often vulnerable to denial of service (DoS) attacks such as frequency jamming. Spread spectrum and frequency hopping are examples of techniques that have been studied as means of preventing this type of attacks. The link layer is also subject to attacks where nodes gain unfair access to the medium or where they disrupt communications, for example by dropping packets related to typical handshake processes. We disregard attacks aimed at the physical and link layers.

We assume bidirectional communication symmetry in every direct link between a pair of nodes. This means that if a node v_2 receives a packet from node v_1, v_1 can also receive a packet from v_2. This is a sensible assumption since our approach needs MAC protocols with collision avoidance mechanisms to work properly, such as the extensively deployed IEEE 802.11, MACA (Multiple Access with Collision Avoidance) [26] and MACAW (MACA for Wireless LANs) [1], which require bidirectional communication for reliable transmission.

We assume the MAC layer protocol to be reliable (e.g. IEEE 802.11). This is required to provide confidence that a data packet has been successfully transmitted to the next-hop node, and enables us to apply the principle of flow conservation (see Section IV.A).

4. Detecting Packet Forwarding Misbehavior

Our work provides a novel methodology to secure 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.

4.1. The Principle of Flow Conservation

We now 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:

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

Equation (1) states the fundamental premise of the flow conservation principle in an ideal static network applied to packets rather than raw 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 networks 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 we propose to use a reliable MAC protocol such as IEEE 802.11, MACA or MACAW.

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 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 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 are due to the fact that overhearing is used by nodes to check for misbehavior. 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 lack the CPU cycles, buffer space or bandwidth to forward packets. 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), which holds for well behaved nodes, reflects this change.

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, as it can be inferred from our simulations (Section V). 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.

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

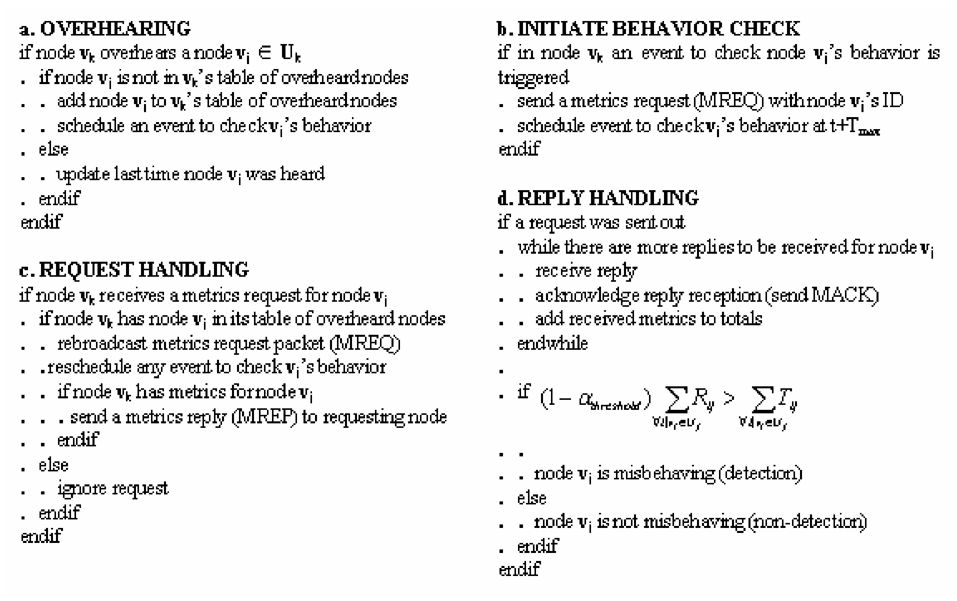
4.2. Adapting Conservation of Flow to Mobile Networks

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 as explained in this section.

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.



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
```

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) to requesting node
. . endif
. else
. . ignore request
. 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 (MREQ) with node $v_j$'s ID
. schedule event to check $v_j$'s behavior at $t+T_{max}$
endif
```

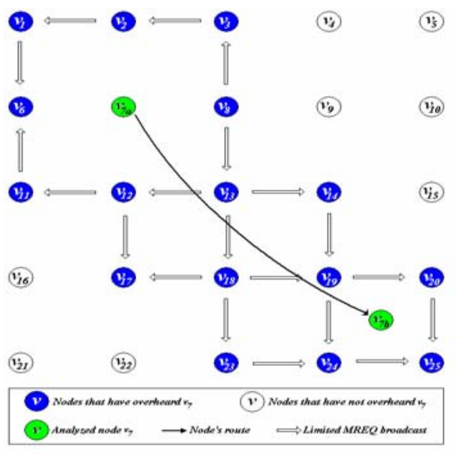
d. REPLY HANDLING

```
if a request was sent out
. while there are more replies to be received for node $v_j$
. . receive reply
. . acknowledge reply reception (send MACK)
. . add received metrics to totals
. endwhile
.
. if $(1-\alpha_{threshold}) \sum_{\forall l | v_l \in U_j} R_{lj} > \sum_{\forall l | v_l \in U_j} T_{lj}$
. .
. . node $v_j$ is misbehaving (detection)
. else
. . node $v_j$ is not misbehaving (non-detection)
. endif
endif
```

Figure 1. Our algorithm pseudocode.

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. 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. Finally, after the MREQ packet is broadcast, a task is scheduled to be triggered a period of time T_{max} (maximum elapsed period of time without checking an active node's behavior) later. This means it is highly unlikely that the same node will originate two successive checks of another node, and gives other nodes a chance to perform the behavior check.

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. 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. 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.



Nodes that have overheard v_7
Nodes that have not overheard v_7
Analyzed node v_7
Node's route
Limited MREQ broadcast

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 the average frequency with which an active node's behavior is checked is:

$$avg\_freq = \frac{1}{(T_{\min} + T_{\max})/2} = \frac{2}{(T_{\min} + T_{\max})} \tag{3}$$

The last task a node performs 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 a metrics reply packet is returned to the node that originated the MREQ following the reverse of the MREQ's path. This approach avoids the high overhead produced by reactive routing protocols when they perform route discovery for many MREP packets.

Reply handling is executed in the node that initiated the MREQ. This node, v_8 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 could use an approach similar to the distributed consensus mechanism proposed in SCAN [3].

5. Evaluation

We perform our simulations using the GloMoSim simulation package. The results presented for each value are the average of 10 simulation runs. Unless explicitly stated otherwise, our simulation parameters take the following values: i) nodes move according to the random waypoint mobility model with a mean node speed of $5ms^{-1}$ and a standard deviation of $2.89ms^{-1}$, ii) the mean pause time is 30 seconds, iii) the wireless transmission range is 100 meters, iv) the link capacity is 2 Mbps, v) the MAC layer protocol is the IEEE 802.11 DCF, vi) the underlying routing protocol is AODV, and vii) the simulation time is 1200 seconds. The network was set-up with 40% of nodes misbehaving. Nodes check the behavior of active nodes within a period chosen uniformly between 40 and 60 seconds, and keep any overheard node in their tables for 120 seconds. The principal metrics in our tests are the percentage of detections and the network traffic overhead. These metrics are assessed in terms of misbehavior threshold and node speed.

We initially consider our misbehavior detection algorithm in terms of the misbehavior threshold, which is the parameter $\alpha_{threshold}$ 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 the effect of the misbehavior threshold on nodes, simulations were carried out with networks containing 20 and 60 nodes, and areas of 40 000m^2 (200m*200m) and 120 000m^2 (346.41m*346.41m) respectively. We varied both the packet drop probability of misbehaving nodes and the misbehavior threshold between 0% and 100%.

Figures 3 and 4 show the percentage of detections as a function of the misbehavior threshold for nodes exhibiting different probabilities of misbehavior for networks with 20 and 60 nodes respectively. 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 and Fig. 4 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 figures.

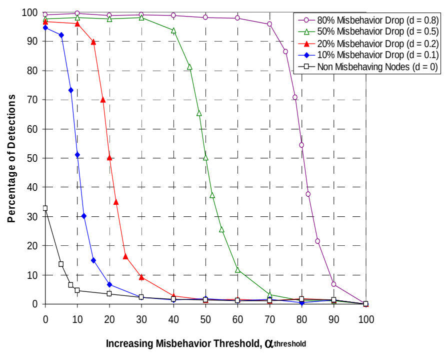


Figure 3. Percentage of positive detections as a function of the increasing misbehavior threshold (20 node network).

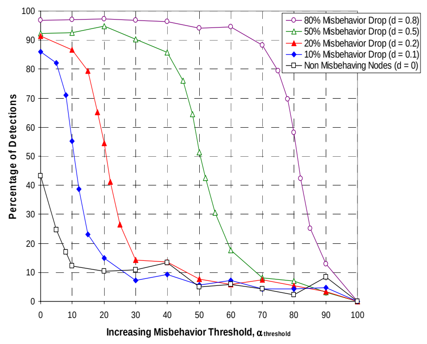


Figure 4. Percentage of positive detections as a function of the increasing misbehavior threshold (60 node network).

Selecting an acceptable or tolerable level of misbehavior *x* in a network is a policy decision. This policy then allows a value of $\alpha_{threshold}$ to be set depending on the desired detection probability. For example, for a detection probability of >90% our results suggest that $\alpha_{threshold}$ should then be set at approximately *x-0.1* for the 20 node network and *x-0.15* for the 60 node network.

The final set of results assesses the network overhead generated by our misbehavior detection algorithm. In this set of simulations misbehaving nodes drop packets with a 50% probability (*d*=0.5), the misbehavior threshold $\alpha_{threshold}$ is 40%, the node speed varies between 0ms^{-1} (a static network) and 20ms^{-1} giving a standard deviation of 0.58ms^{-1}. Fig. 5 displays the mean number of packets sent over each link per behavior check. The network resources are calculated by adding one each time a packet crosses a different link: thus a MREQ packet broadcast that traverses three hops (links) contributes three packet-links to the total. Results are displayed for networks containing 20, 40 and 60 nodes, which are distributed over areas of 40 000m^2 (200m*200m), 80 000m^2 (282.84m*282.84m), and 120 000m^2 (346.41m*346.41m) respectively.

The network overhead per behavior check shown in Fig. 5 is the sum of the overhead produced by the MREQ, MREP and MACK packets per behavior check. It is least when the nodes are stationary and increases with the mean node speed. Whereas this increment is moderate for the 20 and 40 node networks, it is very significant for the 60 node network. This can be explained as follows. First of all, the bigger the network is and the faster a node moves the more nodes it comes in contact with. This means that it is more probable that a MREQ is rebroadcast by a greater number of nodes. On top of this, in a dynamic large network it is more likely that a node that rebroadcasts a MREQ moves out of the wireless range of nodes that need to send back a MREP through it towards the node that originated the MREQ. This causes MREP retransmissions which introduces more overhead into the network.

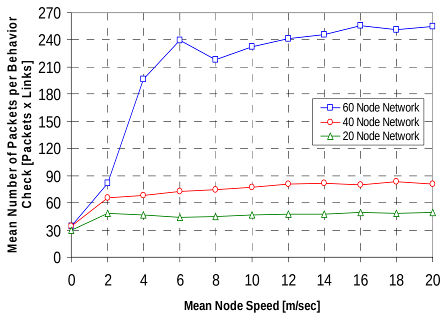


Figure 5. Network overhead as a function of the increasing mean node speed.

6. Conclusions

The self-regulating nature of MANETs requires that they be able to monitor the behavior of the network. Limited resources mean that there is an incentive for nodes to misbehave by not correctly forwarding packets (selfish nodes); nodes may also misbehave for other reasons.

In this paper we have presented an algorithm that is capable of detecting 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.

We have shown that we can detect nodes that misbehave by dropping a significant percentage of packets. Detection is successful in spite of inherent packet losses in MANETs caused by noisy links, mobility, and packet losses due to routing protocol behavior. To avoid falsely accusing correctly behaved nodes of misbehavior, a collaborative consensus mechanism such as that described in [3] can be used. This will be considered in future work.

7. References

[1] 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.

[2] Y. C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," Proceedings of the 8th ACM International Conference on Mobile Computing and Networking, pp. 12-23, September 2002.

[3] 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.

[4] D. Subhadrabandhu, S. Sarkar, and F. Anjum, "A framework for misuse detection in ad hoc networks – part I," IEEE Journal on Selected Areas in Communications, vol. 24, pp. 274-289, February 2006.

[5] D. Subhadrabandhu, S. Sarkar, and F. Anjum, "A framework for misuse detection in ad hoc networks – part II," IEEE Journal on Selected Areas in Communications, vol. 24, , pp. 290-304, February 2006.

[6] Y. C. Hu, D. B. Johnson, and A. Perrig, “SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks,” Proceedings of the 4th IEEE workshop on Mobile Computing Systems & Applications, pp. 3-13, June 2002.

[7] P. Papadimitratos, and Z. J. Haas, “Secure routing for mobile ad hoc networks,” Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference, pp. 193-204, January 2002.

[8] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, “A secure routing protocol for ad hoc networks,” Proceedings of the 10th IEEE International Conference on Network Protocols, pp. 78-87, November 2002.

[9] M. Guerrero-Zapata, and N. Asokan, “Securing ad hoc routing protocols,” Proceedings of the 3rd ACM Workshop on Wireless Security, pp. 1-10, 2002.

[10] M. Guerrero-Zapata, “Secure ad hoc on-demand distance vector (SAODV) routing,” Internet Draft, IETF Mobile Ad Hoc Networking Working Group, February 2005, *draft-guerrero-manet-saodv-05.txt*

[11] J. P. Hubaux, L. Buttyán, and S. Čapkun, “The quest for security in mobile ad hoc networks,” Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing, pp. 146-155, 2001.

[12] C. E. Perkins, and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers,” Proceedings of the ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications, vol. 24, issue 4, pp. 234-244, 1994.

[13] C. E. Perkins, and E. M. Royer, “Ad-hoc on-demand distance vector routing,” Proceedings of the 2nd IEEE Workshop on Mobile Computer Systems & Applications, pp. 90-100, 1999.

[14] C. E. Perkins, “Ad hoc on-demand distance vector (AODV) routing,” Request For Comments (RFC) 3561, July 2003, available at: *http://www.ietf.org/rfc/rfc3561.txt*

[15] D. B. Johnson, D. A Maltz, and Y. C. Hu, “The dynamic source routing protocol for mobile ad hoc networks,” Internet Draft, IETF MANET Working Group, July 2004, *draft-ietf-manet-dsr-10.txt*

[16] B. Awerbuch, D. Holmes, C. Nita-Rotaru, and H. Rubens, “An on-demand secure routing protocol resilient to Byzantine failures,” Proceedings of the 3rd ACM Workshop on Wireless Security, pp. 21-30, 2002.

[17] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “Efficient authentication and signing of multicast streams over lossy channels,” Proceedings of the IEEE Symposium on Security and Privacy, pp. 56-73, May 2000.

[18] S. Buchegger, and J. Le Boudec, “Performance analysis of the CONFIDANT protocol,” Proceedings of the 3rd ACM Symposium on Mobile Ad Hoc Networking & Computing, pp. 226-236, 2002.

[19] Y. Zhang, and W. Lee, “Intrusion detection in wireless ad-hoc networks,” Proceedings of the 6th ACM International Conference on Mobile Computing and Networking, pp. 275-283, August 2000.

[20] P. Papadimitratos, and Z. Haas, “Secure data communication in mobile ad hoc networks,” IEEE Journal on Selected Areas in Communications, vol. 24, issue 2, pp. 343-356, February 2006.

[21] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, “Providing robust and ubiquitous security support for mobile ad-hoc networks,” Proceedings of the 9th IEEE International Conference on Network Protocols, pp. 251-260, November 2001.

[22] L. Zhou, and Z. Haas, “Securing ad hoc networks,” IEEE Network Magazine, vol. 13, issue 6, November/December 1999.

[23] 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.

[24] R. Rao, and G. Kesidis, “Detecting malicious packet dropping using statistically regular traffic patterns in multihop wireless networks that are not bandwidth limited,” Proceedings of the 2003 IEEE Global Telecommunications Conference, vol.5, pp. 2957-2961, 2003.

[25] 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.

[26] P. Karn, “MACA – a new channel access method for packet radio,” ARRL/CRRL Amateur Radio 9th Computer Networking Conference, pp 134-240, September 1990.