Detection and Accusation of Packet Forwarding Misbehavior in Mobile Ad-Hoc Networks

Oscar F. Gonzalez, Godwin Ansa, Michael Howarth, and George Pavlou

***Abstract*—Mobile Ad Hoc networks (MANETs) are susceptible to having their effective operation compromised by a variety of security attacks. For example, misbehaving nodes can cause general network disruption by not forwarding packets on behalf of other nodes in the network. 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 capable of detecting and accusing nodes that exhibit packet forwarding misbehavior. Our evaluation results demonstrate that our algorithm effectively detects and accuses nodes that drop a significant fraction of packets.**

***Index Terms*—Mobile ad hoc network, misbehavior detection, node accusation, packet forwarding.**

I. Introduction

A mobile ad hoc network (MANET) consists of a group of devices (or nodes) that rely solely on the wireless communication medium and themselves for data transmission. Nodes cooperate by forwarding packets on behalf of each other when destinations are out of their direct wireless transmission range. A centralized administration and/or a pre-deployed network infrastructure are not necessary for a MANET to be set up, thus making its deployment quick and inexpensive. In addition, nodes' ability to move freely ensures a flexible and versatile dynamic network topology which can be desirable in many situations. Mobile ad hoc networks are ideal in environments where installing an infrastructure is not appropriate for reasons such as cost, quality, or vulnerability, or where the network is too transient, or the infrastructure has been destroyed. Examples of MANET applications are: emergency disaster relief (destroyed infrastructure), military operations over a battlefield (vulnerable infrastructure), and wilderness expeditions (transient networks), and community networking and interaction between students during a lecture.

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. 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]. An overloaded node lacks the CPU cycles to attend its local and/or network tasks, which leads it to drop packets owing to its inability to process them. A broken node has a software or hardware fault that prevents it from performing its network duties properly. A compromised node may be victim of an attack that degrades its data forwarding capabilities. A congested node receives more packets than the bandwidth available to it allows it to send, its buffer fills and eventually it has to drop incoming packets. A selfish node is unwilling to use its resources such as battery life, bandwidth or processing power to forward packets on behalf of other nodes. A malicious node drops packets or generates additional packets solely to disrupt the network performance and prevent other nodes from accessing any network services (a denial of service attack). Both selfish and malicious nodes expect, however, other nodes to forward packets on their behalf in spite of their own misbehavior.

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 performs two tasks: a) enables packet forwarding misbehavior detection through the principle of conservation of flow [25], and b) enables the accusation of nodes that are consistently detected exhibiting packet forwarding misbehavior. A node that is accused of misbehavior is denied access to the network by its peers, which ignore any of its transmission attempts. Thus, misbehaving nodes are isolated from the rest of the network. 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 detection on 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 and accuses 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.

Manuscript received September 25, 2007.

Oscar Gonzalez, Godwin Ansa, and Michael Howarth are with the Center for Communications Systems Research, University of Surrey, Guildford, UK (e-mails: {o.gonzalez-duque, g.ansa, m.howarth}@surrey.ac.uk). George Pavlou is with Dept. of Electronic & Electrical Engineering, University College London, UK (e-mail: g.pavlou@ee.ucl.ac.uk).

In this paper we first present a framework and a relevant algorithm and protocol that deal with these attacks. 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 accusation, and Section V presents a performance evaluation. Finally, the paper is concluded in Section VI.

II. Related Work

Work has been conducted by other authors both on securing the route discovery part of routing protocols, and on packet forwarding. In this Section we initially look at ways of protecting the network against misbehaving nodes and data forwarding anomalies. We then review work that attempts to detect and penalize misbehavior in data packet forwarding.

A. Routing and Packet Forwarding Protection

Secure routing protocols have 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]. Also extending DSR to provide it with security mechanisms is CONFIDANT (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks) [18]. These approaches only secure the path discovery and establishment functionality of routing protocols, thus our approach complements them by securing the data forwarding functionality.

Some research effort has also 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] achieve routing protection assuming and making use of the existence of *a priori* relationships in a network. However, *a priori* relationships in MANETs may not exist. As with SEAD, SAODV, Ariadne and CONFIDANT these protocols can be coupled with our approach, which is not routing protocol dependent, to offer an improved security solution.

The routing protocol proposed in [16] offers resilience to Byzantine behavior (any action that results in the disruption or degradation of the data forwarding service) by an algorithm that allows the detection of an anomalous link after log *n* faults have occurred on a path, where *n* is the hop length of the path. In [19] when a node has broken the security mechanisms of a network is regarded as an intruder. 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.

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. SSP is a specific case of SMT where only one path is used at a time and the source tries a different path each time an acknowledgment is not received. 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 through the accusation phase.

Attack patterns have been the object of research in order to identify known attacks through abnormal packets. 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 that belong to trusted users and run an intrusion detection system (IDS) module to detect attacks launched by outsiders through packets with abnormal contents. Unfortunately, this framework fails to make use of well-behaved outsiders that could contribute to relevant tasks and rewards misbehaving outsiders by allowing them to use the network. In this regard, our protocol punishes misbehaving nodes by denying them access to the network and its services.

B. Misbehavior Detection

There has 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. In this approach each router executes the WATCHERS protocol at regular intervals in order to identify neighboring routers that misroute traffic and avoid them. WATCHERS requires the existence of at least one path not affected by disruptive routers between any two well behaved routers in the network. 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. Additionally, in our approach the broadcasting nature of the wireless medium allows for multiple possible paths between any two well behaved nodes.

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 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. In this scheme a node can distinguish between a misbehaving node and a congested node, knowing the traffic transmission rates from other nodes to the target node. 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. It uses AODV [13, 14], but argues that the same ideas are applicable to other routing protocols. SCAN assumes a network with sufficient node density that nodes can overhear packets being received by a neighbor, in addition to packets being sent by the neighbor. 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 and though nodes monitor their neighbors independently, all nodes in a local neighborhood collaborate with each other to eventually convict a suspicious node by revoking its token. The tokens' lifetime is determined by a credit strategy that helps reducing the total network overhead. Token renewal and revocation is done through threshold secret sharing and secret share updates. SCAN develops these ideas from [21] where they are used to give a valid key to a new node entering the network and from then onwards to renew its key periodically. Similar techniques have also been studied in various papers such as [22], where they are used to achieve distribution of trust throughout a network. SCAN is similar to our approach in the sense that it does not only detect the source of misbehavior, but it also punishes any misbehaving nodes. However, SCAN makes use of cryptographic techniques that may prove too resource demanding for devices with limited capabilities.

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, thus reducing the chance of finding a misbehaving node along the selected path. This work as described uses DSR but it is claimed it can easily be adapted to other source routing protocols. However, its applicability has not yet been addressed for distance-vector based routing protocols. Moreover, the watchdog might not detect a misbehaving node in the presence of ambiguous collisions, receiver collisions or nodes capable of controlling their transmission power (described in section IV.A). 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. Also, using pathrater can be considered a reward for selfish nodes since the flow is diverted towards other nodes in the network while selfish nodes preserve their resources. In contrast, our approach denies access to the network to any node that has been accused of misbehavior, thus discouraging them from dropping packets.

III. Assumptions and Terminology

A. 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) [11] 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).

We also assume that all nodes in the network are adapted with wireless interfaces that support promiscuous mode operation. This operational mode allows a node to process all transmissions from nodes within hearing range. This is required in order to determine active nodes in a node's neighborhood and schedule events to check their behavior at a later stage.

At the network layer we assume that nodes misbehave by dropping packets despite having agreed to forward them during route discovery. Other types of misbehavior are not taken into account including any type of attack by two or more colluding nodes.

This paper does not address security measures for our misbehavior detection and accusation approach since it focuses on the basic proposal of misbehavior and detection mechanisms. However, cryptographic techniques such as threshold secret sharing and secret share updates used in SCAN [3] could be used as viable ways of protecting the detection and accusation packets (Section IV.C) of our approach.

B. Terminology

We use the term *neighbor* to refer to a node that is within the direct wireless transmission range of another node. From this, it follows that both nodes are able to establish a reliable bidirectional communication. Likewise, the term *neighborhood* refers to all nodes that are neighbors of a

particular node. A node is not a neighbor of itself and, therefore, a node does not belong to its own neighborhood.

We use the term *detection* to mean that our algorithm has identified that a node appears to be misbehaving. A detection is based on a *single* check of the node's behavior. An *accusation*, on the other hand, occurs when a node reports another node as misbehaving. It is our view that an accusation should be based on more than a single positive detection to increase confidence in the assessment, as we discuss in Section IV.C below. Additionally, in Section V it is shown how the number of detections needed to raise an accusation affects the percentage of nodes correctly accused of misbehavior.

A misbehaving node is represented by a given drop characteristic, e.g. dropping packets with 30% probability. In our simulations, a uniform distribution is used. We use the parameter d to indicate the fraction of packets dropped.

IV. Detecting and Accusing Misbehaving Nodes

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.

A. 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 $V = \{v_1, v_2, v_3 \ldots v_N\}$ is the set of all nodes in the network, N is the total number of nodes in the network, and $j = 1, 2, 3 \ldots N$.
- 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:

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

Equation (1) gives 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.

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}. For instance, when node v_1 needs to transmit a packet to v_2, v_1 sends an RTS frame and v_2 replies with a CTS frame. Following the reception of the CTS, v_1 sends the data which may collide at the receiver with the transmission of some other node v_3 that heard neither the RTS nor the CTS frame for example because v_3 has just moved into range. In this case v_2 does not increase its R_{21} metric because it did not receive the data, and v_1 does not increase its T_{12} because the packet was never acknowledged. Even in the eventuality that an ACK frame gets lost the nodes would realize the error when v_1 retransmits the data. In this case, v_2 increases its R_{21} metric the first time it receives the data packet and sends back the respective ACK frame. Node v_1 does not increase its T_{12} metric since it does not receive the ACK frame and instead it retransmits the packet, sending an RTS frame and waiting for a CTS frame. The second time that v_2 receives the packet it will notice that the packet has been already received by checking the sequence control field in the MAC header, so it does not increase its R_{21} metric and it acknowledges again the packet as stipulated in the 802.11 standard. When v_1 receives the ACK it will increase its T_{12} metric and equation (1) holds again.

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 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_1 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, as in [3, 23]. 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. 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), which holds for well behaved nodes, 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, 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 and reduce the probability of an incorrect accusation.

B. 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 later in Section IV.C.

Every node in the network is required to keep three tables: an overhead nodes table, a detection table and an accusation table. The overheard nodes table contains the IDs of those nodes that have been overheard recently through promiscuous listening. Entries in this table are removed once they go stale (e.g. if a node in the table has not been overheard in the last *t* seconds). This process helps a node v_j to keep track of nodes that have become part of its neighborhood U_j while they were actively intervening in the network. The detection table contains the IDs of those nodes that have been detected as misbehaving and the number of times their misbehavior has been reported. Finally, the accusation table keeps the IDs of those nodes that have been accused of misbehavior. Nodes are typically accused of misbehavior because they have reached within a predefined period of time the number of misbehavior detections required to be accused.

C. Algorithm

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} (Fig. 1.a), 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.b). 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_i to check v_j's behavior (Fig. 1.c), node v_i 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. Nevertheless, two or more unsynchronized nodes performing a behavior check on the same node will generate different timestamps. In this case nodes receiving the MREQ packets will select the packet with the earliest timestamp and will reply accordingly. Thus our approach does not require accurate synchronization of the nodes' clocks. 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.d) illustrates 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.c) 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.

a. MONITORING
if node v_i successfully sends a packet to node v_j
. increase T_{ij}
endif
if node v_i receives a packet successfully forwarded by node v_j
. increase R_{ij}
endif

b. BEHAVIOR CHECK SCHEDULING
if node v_i overhears a node $v_j \in U_k$
. if node v_j is not in v_i's table of overheard nodes
. . add node v_j to v_i'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. INITIATE BEHAVIOR CHECK
if in node v_i 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

d. REQUEST HANDLING
if node v_i receives a metrics request for node v_j
. if node v_i 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_i has metrics for node v_j
. . . send a metrics reply (MREP) back to the requesting node
. . endif
. else
. . ignore request
. endif
endif

e. 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 $(1-\alpha_{threshold}) \sum_{\forall i | v_i \in U_j} R_{ij} \leq \sum_{\forall i | v_i \in U_j} T_{ij}$
. . node v_j is misbehaving (detection)
. . send a detection alert packet (DAP) with node v_j's ID
. else
. . node v_j is not misbehaving (non-detection)
. endif
endif

f. DETECTION ALERT HANDLING
if node v_i receives a detection alert for node v_j
. if node v_i has node v_j in its table of overheard nodes
. . rebroadcast detection alert packet (DAP)
. . if max_num_of_detections for node v_j has been reached
. . . broadcast na accusation packet (AP) with node v_j's ID
. . endif
. else
. . ignore detection alert
. endif
endif

g. ACCUSATION HANDLING
if node v_i receives an accusation packet for node v_j
. if node v_i has node v_j in its accusation table
. . ignore accusation packet
. else
. . add node v_j to v_i's accusation table
. . rebroadcast accusation packet (AP)
. endif
endif

h. PUNISHING ACCUSED NODES
if node v_i receives a packet from node v_j
. if node v_j is in node vi's accusation table
. . ignore packet
. else
. . handle and process the packet
. endif
endif

Figure 1. Our algorithm pseudocode.

It may also happen that node v_7 stops transmitting and receiving packets before it moves to a different network area. Then, after moving, v_7 may become active again forming a new neighborhood. In this case the old and new neighborhoods are not connected by nodes that have overheard v_7 and, therefore, a limited broadcast triggered in one neighborhood will not reach the other. In spite of this, our algorithm still works properly because two independent behavior checks will be performed on v_7: one at its old neighborhood and another at its new neighborhood. The outcome of each of these behavior checks depends on the behavior exhibited by v_7 at each neighborhood.

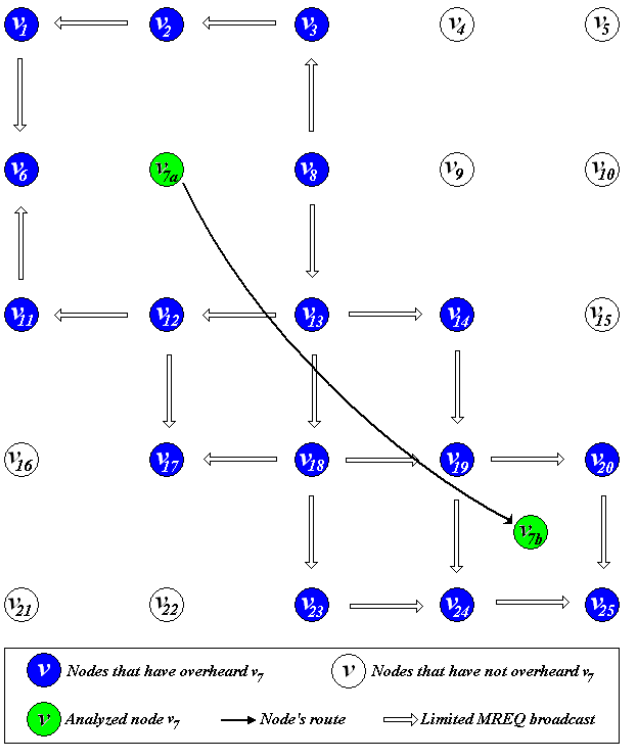


Figure 2. Example of limited broadcast to track down nodes that have overheard node v7.

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:

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

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.d) 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 requires nodes to set a backward pointer when they receive an MREQ. 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.e). If it does not, this is considered a detection and a *detection alert packet (DAP)* containing the detected node ID is broadcast with TTL = 1 in its IP header. Detection alert packets are broadcast in the same way as MREQs, i.e. they follow our limited broadcast algorithm (Fig. 1.f).

Due to the nature of our algorithm nodes are not perfectly synchronized with each other. A MREQ will reach close nodes faster than nodes placed a few hops away. The last nodes to receive the MREQ have enough time to send or receive some extra packets to and from the analyzed node, thus unbalancing the values of the T_{ij} and R_{ij} metrics. This discrepancy, in which some packets may have been sent to the node being analyzed but not yet forwarded by it, is accommodated by $\alpha_{threshold}$.

A problem that has been detected in our simulations has its roots in the dynamic nature of MANETs. Nodes receiving a MREQ with non-zero metrics for the checked node send back their reply. However, such replies sometimes get lost due to collisions, noise in the wireless channel or link/path breakages due to the mobility of the nodes. If the value of the metrics contained in the lost reply is small compared to the total obtained after adding up the replies that do not get lost, $\alpha_{threshold}$ can accommodate them and equation (2) holds. Unfortunately, this is not always the case and some of those replies that get lost contain key information for the calculations and the checked node is then falsely detected as misbehaving. This is one of the reasons why an accusation should not be made based on a single detection. Using a single detection to accuse a node is not sufficient since such an approach may lead to false accusations against correctly behaving nodes. Our 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. Additionally, to circumvent the lost replies problem we propose an optional module to our algorithm. A node receiving an MREP as it is forwarded towards its destination (i.e. the node performing the behavior checking task) will also send a metrics acknowledgement packet (MACK). Thus, nodes sending/forwarding an MREP wait for an MACK from the next hop in the route. If the confirmation does not arrive then they retransmit the MREP. The process is repeated up to MAX_{Retx} retransmission retries before giving up. The results obtained in our simulations have demonstrated that this technique can significantly improve the results in MANETs with a high degree of mobility. Simulations have also shown that the most significant improvement can be seen when comparing the results for MAX_{Retx}=0 (without retransmitting any reply) and MAX_{Retx}=1. Subsequent increases to MAX_{Retx} improve the results further but not significantly.

The handling of detection alerts (Fig 1.f) is also determined by our limited broadcast algorithm. This means that the information to accuse a node of misbehavior is collected locally rather than globally. When a node receives a detection alert packet (DAP) it first checks if the reported node ID contained in the received packet is present in its table of overheard nodes; if it is not the node stops broadcasting the DAP. However, if the ID appears in its table then it rebroadcasts the DAP with TTL = 1 in the IP header. Thus, nodes can control how far the DAP broadcast is to go in the same way they do with a MREQ. For instance, assuming that v_8 in Fig. 2 detects that v_7 is misbehaving, the DAP follows the same path as that depicted in the Fig. 2 for a MREQ packet. Although this approach prevents nodes from generating excessive network overhead, it may also permit malicious nodes constantly changing their geographical position in a clever manner (without going back to previously visited areas) to avoid being accused. After a node has decided whether continue broadcasting a DAP or not, it checks if an entry for the reported node ID has been already created in its detections table. If it has not a new entry is created with its field *number of detections* equal to one. If the entry is already present its number of detections is increased and then compared against a *detections threshold*. When the number of detections reaches the detections threshold in less than a predefined period of time $T_{threshold}$ there is enough evidence to accuse the reported node of misbehavior. Therefore, an *accusation packet (AP)* is broadcast in a network wide fashion so that access is denied to the reported node all over the network.

Nodes that receive an accusation packet (Fig. 1.g) examine their accusation tables to see whether the reported node has been accused previously. When an AP with a newly accused node is received a new entry is created in the accusation table and the broadcast of the AP continues. Otherwise, the packet is ignored and dropped to prevent unnecessary network traffic. Finally, all nodes in the MANET are responsible to ensure that packets coming from an accused node (a node present in their accusation table) are immediately dropped (Fig. 1.h). This approach denies misbehaving nodes any chance to have their packets transmitted in the network as well as to participate in route discovery, thus preventing them from causing further disruption in the communication process.

V. EVALUATION

We perform our simulations using the GloMoSim simulation package. The results presented for each value are the average of 10 simulation runs. Tests with a larger number of simulations (e.g. 20) give results that vary typically no more than 1% from those presented here. Unless explicitly

stated otherwise our simulation parameters, which correspond to typical values used by other authors, take the following values: i) nodes move according to the random waypoint mobility model with a speed randomly chosen with uniform distribution between 0ms^{-1} and 10ms^{-1}, this yields a mean node speed of 5ms^{-1} and a speed standard deviation of 2.89ms^{-1}, ii) the pause time takes a value that is exponentially distributed with mean 30 seconds, iii) the wireless transmission range of every node 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 total simulation time for each scenario is 1800 seconds. Our results are presented in two parts: *part A* focuses on the detection power of our algorithm, i.e. how well our algorithm can distinguish between well behaved and misbehaving nodes, and p*art B* focuses on the capability of our algorithm to accuse misbehaving nodes and improve the average network throughput while maintaining an acceptable network overhead.

A. *Detecting Misbehaving Nodes*

An important parameter to evaluate the effectiveness of an approach that detects and accuses misbehaving nodes is its detection power. In this section we present results that demonstrate that our approach has a high probability of detecting truly misbehaving nodes while maintaining a low probability of performing false detections, i.e. wrongly detecting a well behaved node as a misbehaving one. For this set of results the network was set-up with 40% of its total nodes misbehaving by not forwarding all packets. 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 after the last time they are heard. On average an active node is checked approximately 36 times in each 1800 second simulation. The principal metric in our tests is the percentage of detections and it is 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. These values ensure that node density is preserved between both scenarios. 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 positive detections as a function of misbehavior threshold for nodes exhibiting different probabilities of misbehavior for networks with 20 and 60 nodes respectively. It can be inferred from both graphs that the criterion to select an adequate misbehavior threshold depends on the level of trust required in the network as well as on network characteristics such as network size and node density. 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.

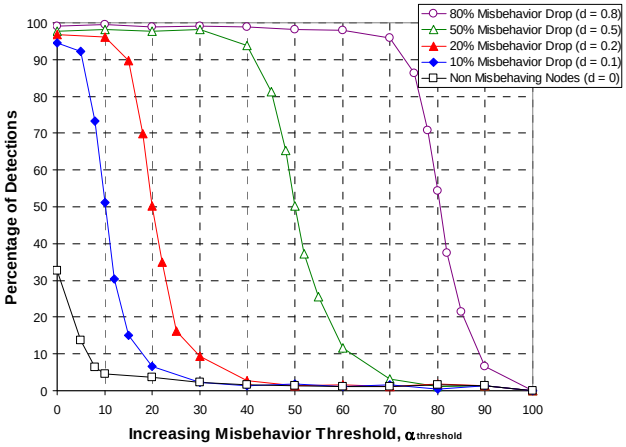


Figure 3. Percentage of positive detections as a function of the increasing misbehavior threshold (20 node network).

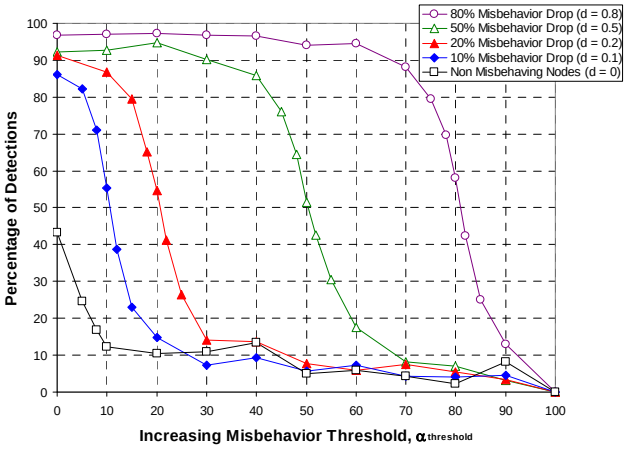


Figure 4. Percentage of positive detections as a function of the increasing misbehavior threshold (60 node network).

A similar problem occurs with misbehaving nodes that drop a small percentage of packages, e.g. less than 10% of packets. The graphs show how the less misbehavior a node exhibits the more its curve resembles that of a well behaved node, making distinguishing between them a complex task. 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.

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.

Our second set of results assesses the performance of our misbehavior detection algorithm in terms of the degree of mobility of the nodes in the network. This time the misbehaving nodes drop packets with a 50% probability while the misbehavior threshold is kept at 40%. The mean node speed varies between 0ms^{-1} (a static network) and 20ms^{-1} while the speed standard deviation for all

measurements is 0.58ms^{-1}. Whereas Fig. 5 is plotted for misbehaving and well-behaved nodes in a 20 node network, Fig. 6 is plotted for misbehaving and well-behaved nodes in a 60 node network.

It can be seen from Fig. 5 that our misbehavior detection protocol is not significantly affected by the speed of the nodes in small networks. Our approach robustly keeps a gap between misbehaving nodes and correctly behaving nodes making it easy to spot nodes that are purposefully violating the principle of flow conservation. The fluctuations seen in both curves are likely to have occurred due to the sporadic losses of metrics reply packets (MREP) rather than the node speed. However, the same is not true for large scale networks, as it can be appreciated from Fig. 6. As the mean node speed increases the gap between misbehaving and correctly behaving nodes grows smaller. Nevertheless, a good level of discrimination is maintained. These results support our hypothesis that using a single detection to accuse a node is not sufficient since such an algorithm may lead to false accusations against correctly behaving nodes (Section IV.C). An accusation should be the result of a distributed consensus mechanism such as that proposed in SCAN [3] to ensure fairness to well behaved nodes while keeping a high probability of correctly accusing misbehaving nodes.

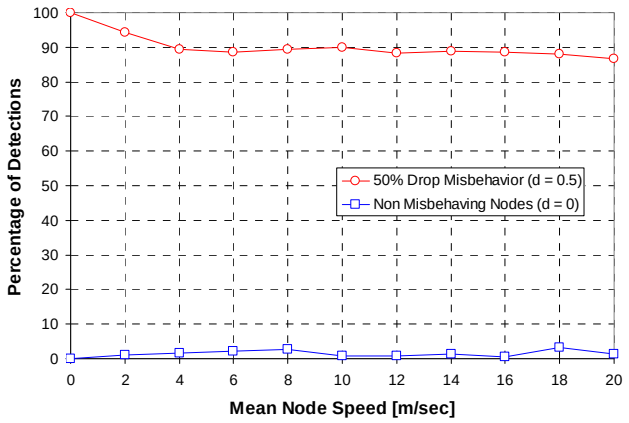


Figure 5. Percentage of detections as a function of the increasing mean node speed (20 node network).

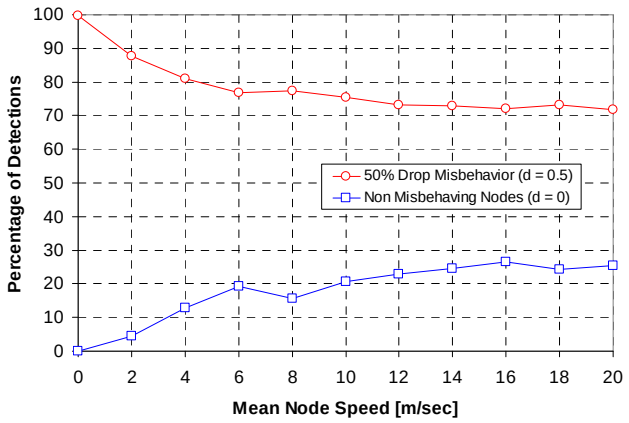


Figure 6. Percentage of detections as a function of the increasing mean node speed (60 node network).

B. *Accusing Misbehaving Nodes*

This section presents first an evaluation of the average throughput gain offered by our proposed algorithm to networks affected by nodes that drop packets in a probabilistic manner. Results are shown for networks with 20, 40, 60 and 120 nodes. Then, our final set of results analyzes the network overhead created by our approach and how it compares against the total traffic produced in the network. Networks simulated in this section were set up with 20% of its total nodes misbehaving by dropping packets with 60% percent probability (d = 0.6). The misbehavior threshold was 40% ($\alpha_{threshld} = 0.4$).

In order to see the improvement that our approach can bring to networks affected by packet forwarding misbehavior we consider the average throughput in terms of the mean node speed. Our graphs present results for networks without misbehaving nodes, networks with misbehaving nodes but without defense mechanisms in place, and networks that use our algorithm to deny access to misbehaving nodes. 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 in order to maintain a constant node density.

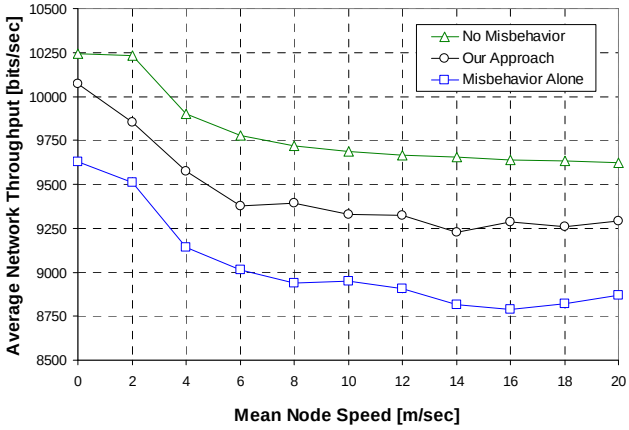


Figure 7. Average network throughput as a function of the increasing mean node speed (20 node network).

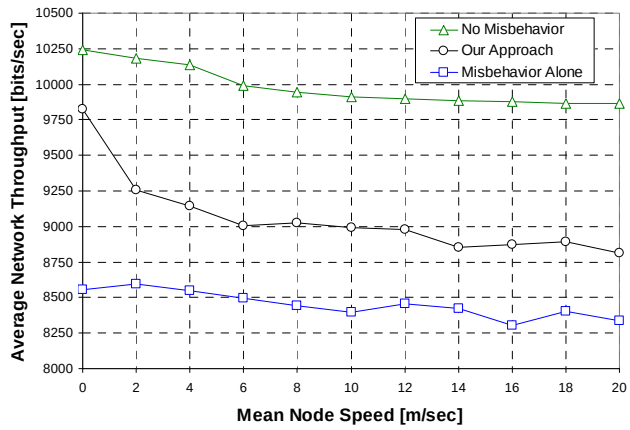


Figure 8. Average network throughput as a function of the increasing mean node speed (40 node network).

Figures 7, 8 and 9 show curves for 20, 40 and 60 node networks respectively. Each graph displays the average network throughput as a function of the increasing mean node speed for a) networks without misbehaving nodes (No Misbehavior), b) networks making use of our detection and accusation approach (Our Approach), and c) networks with misbehaving nodes but with no means of defending themselves from any type of attack (Misbehavior Alone). As it can be seen from the figures our approach improves the network throughput when it is used in networks exhibiting

packet forwarding misbehavior. However, the average throughput cannot reach that of a network where there is not any misbehavior present. This is due to the fact that our algorithm requires of certain amount of time to collect the necessary data to detect and accuse misbehaving nodes. Thus, during this initial phase (data collection) misbehaving nodes can drop packets before being accused and isolated from the network.

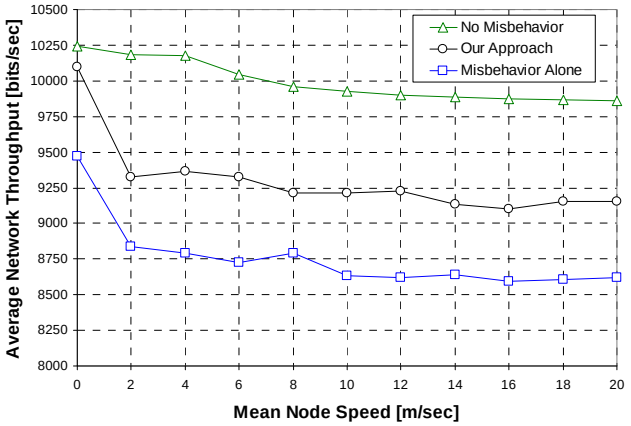


Figure 9. Average network throughput as a function of the increasing mean node speed (60 node network).

With the purpose of seeing how our approach reacts to changes in a network's node density our next set of results has been carried out in a network that preserves the same area as our previous 60 node network (120.000 m^2 = 346.41m*346.41m), but has double its number of nodes (120 nodes) so as to double its density. Figure 10 presents results for networks without misbehaving nodes, networks with misbehaving nodes but without defense mechanisms in place, and networks that use our algorithm to deny access to misbehaving nodes.

From figure 10 it can be seen that our approach still works in networks with relative high node density. The network throughput of networks using our approach improves when compared against networks containing misbehaving nodes that are neither avoided nor penalized. However, networks that do not present node misbehavior at all still exhibit the best performance in terms of network throughput.

A network that implements our detection and accusation algorithm looks at bootstrap as a network without a protection scheme. However, as time elapses our algorithm starts detecting and accusing those nodes that drop a fraction of packets above a preset misbehavior threshold $\alpha_{threshold}$. Consequently, in such a network any misbehaving nodes will eventually be detected, accused and denied network access, allowing the network to obtain an overall performance close to a network where nodes do not misbehave.

The final set of results assesses the network overhead generated by our misbehavior detection and accusation algorithm as a function of the increasing mean node speed, and compares it against the network overhead produced by four constant bit rate (CBR) connections present in the network. Although CBR traffic is generated at the application layer, it is accounted for at the TCP/IP layer in the form of UDP network overhead. In this set of simulations misbehaving nodes drop packets with a 60% probability (d=0.6), the misbehavior threshold $\alpha_{threshold}$ is 40%, the node speed varies between 0ms^{-1} (a static network) and 20ms^{-1}, and the speed standard deviation is set at 0.58ms^{-1}. 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 a network containing 40 nodes distributed over an area of 80 000m^2 (282.84m*282.84m).

The network overhead shown in Fig. 11 is the sum of the overhead produced by the MREQ, MREP, MACK, DAP and AP packets. It is least when the nodes are stationary and increases with the mean node speed. The reader may be confused to see the UDP overhead to be the lowest in a static network (mean node speed = 0 m/sec). This can be explained as follows. In static networks link breakages are due only to collisions and the channel noise. This means that packets are more likely to arrive to their destination without need for retransmissions. This yields a high throughput and a lower network overhead since packets arrive constantly to their destination and traverse less times each link. On the other hand, in dynamic networks mobility frequently cause link breakages, retransmissions take place over almost every link increasing the network overhead. The throughput, instead, decreases due to the fact that fewer packets reach their destination. In contrast, the overhead generated by our scheme is higher in dynamic network because a node has greater probability to become in contact with more nodes. Therefore, more behavior checking tasks are scheduled which translates in a higher number of MREQ, MREP and MACK packets being transmitted.

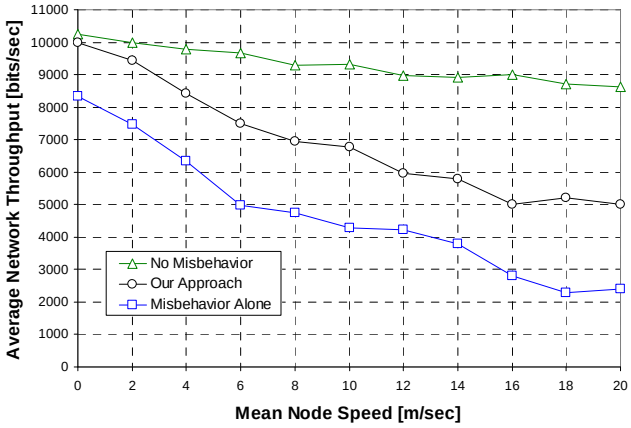


Figure 10. Average network throughput as a function of the increasing mean node speed (120 node network).

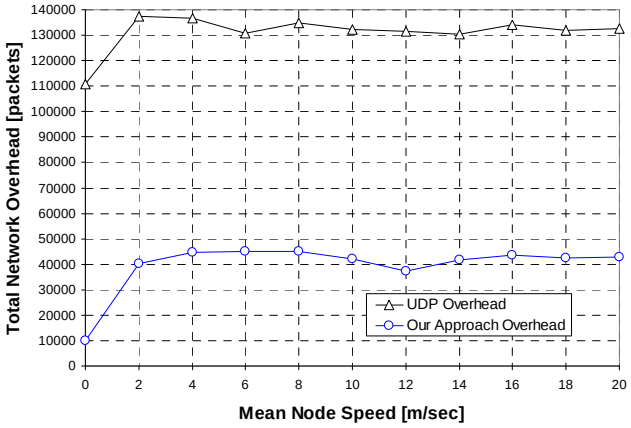


Figure 11. Network overhead as a function of the increasing mean node speed (40 nodes).

VI. 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 and accusing nodes that exhibit 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. Also, our algorithm does not interfere with the routing protocol which allows for its use regardless of the routing strategy employed.

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 an accusation in our approach is based on a predefined number of detections performed not necessarily by the same node.

Selecting an acceptable or tolerable level of misbehavior in a network is a policy decision that enables us to choose an adequate misbehavior threshold $\alpha_{threshold}$. However, this threshold also depends on dynamic network parameters such as node density and network area. Therefore finding a way to calculate an optimal misbehavior threshold in a dynamic manner is of great importance, especially in autonomic environments where the network should automatically adjust its parameters so as to adapt itself to changes in its surroundings. In this respect our future research will focus on the study of methods to collect and synthesize network context information, and techniques to calculate an adaptive misbehavior threshold using such information.

VII. References

[1] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs", in *Proc. ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, London, UK, September 1994.

[2] Y. C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks", in *Proc. 8th ACM International Conference on Mobile Computing and Networking*, Atlanta, USA, 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, no. 2, February 2006, pp. 261-273.

[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, no. 2, February 2006, pp. 274-289.

[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, no. 2, February 2006, pp. 290-304.

[6] Y. C. Hu, D. B. Johnson, and A. Perrig, "SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks," in *Proc. 4th IEEE workshop on Mobile Computing Systems & Applications*, New York, USA, June 2002.

[7] P. Papadimitratos, and Z. J. Haas, "Secure routing for mobile ad hoc networks," in *Proc. SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Antonio, USA, 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", in *Proc. 10th IEEE International Conference on Network Protocols*, Paris, France, November 2002.

[9] M. Guerrero-Zapata and N. Asokan, "Securing ad hoc routing protocols", in *Proc. 3rd ACM Workshop on Wireless Security*, New York, USA, 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.

[11] P. Karn, "MACA – a new channel access method for packet radio," in *Proc. ARRL/CRRL Amateur Radio Computer Networking Conference*, September 1990.

[12] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers", in *Proc. ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, London, UK, September 1994.

[13] C. E. Perkins, and E. M. Royer, "Ad-hoc on-demand distance vector routing", in *Proc. 2nd IEEE Workshop on Mobile Computer Systems & Applications*, New Orleans, USA, February 1999.

[14] C. E. Perkins, "Ad hoc on-demand distance vector (AODV) routing", *Request For Comments (RFC) 3561*, July 2003, Available online 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 (DSR)", Internet Draft, IETF MANET Working Group, July 2004.

[16] B. Awerbuch, D. Holmes, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to Byzantine failures", in *Proc. 3rd ACM Workshop on Wireless Security*, New York, USA, 2002.

[17] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels", in *Proc. IEEE Symposium on Security and Privacy*, Berleley, USA, May 2000.

[18] S. Buchegger, and J. Le Boudec, "Performance analysis of the CONFIDANT protocol," in *Proc. 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, Lausanne, Switzerland, June 2002.

[19] Y. Zhang, and W. Lee, "Intrusion detection in wireless ad-hoc networks", in *Proc. 6th ACM International Conference on Mobile Computing and Networking*, Boston, USA, 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, no. 2, February 2006, pp. 343-356.

[21] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks", in *Proc. 9th IEEE International Conference on Network Protocols*, Riverside, USA, November 2001.

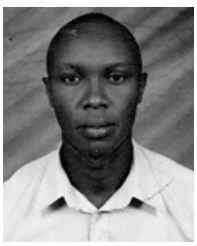
[22] L. Zhou, and Z. Haas, "Securing ad hoc networks", *IEEE Network Magazine*, vol. 13, no. 6, November/December 1999, pp. 24-30.

[23] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad hoc Networks", in *Proc. 6th ACM International Conference on Mobile Computing and Networking*, pp. 255-265, Boston, USA, 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", in *Proc. IEEE Global Telecommunications Conference*, San Fransisco, USA, December 2003.

[25] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson, "Detecting disruptive routers: a distributed network monitoring approach", in *Proc. Symposium on Security and Privacy*, May 1998.

Oscar Gonzalez received the BEng degree from the National University of Colombia, Manizales, in 2002, and the MSc degree from the University of Surrey, Guildford, in 2005. He is currently working towards the PhD degree in the Faculty of Engineering and Physical Sciences, University of Surrey, Guildford. His research interests include wireless networking, network security and trust, and autonomic communications.



Godwin Ansa received the BSc degree in Computer Science from the University of Uyo, Nigeria, in 2002, and the MSc degree in Communications Networks and Software from the University of Surrey, UK, in 2008. He is currently working towards his PhD degree in the Department of Electronic Engineering at the University of Surrey. His research interests include network security, wireless networking and satellite network architectures.

Michael Howarth received the bachelor's degree in engineering science and the DPhil degree in electrical engineering, both from Oxford University and the MSc degree in telecommunications from the University of Surrey, United Kingdom. Prior to joining the University of Surrey, he worked for several networking and IT consultancies. He is a lecturer in networking at the Centre for Communication Systems Research (CCSR), University of Surrey. His research interests include traffic engineering, quality of service, security systems, protocol design, and optimization of satellite communications. He is a chartered electrical engineer and a member of the United Kingdom IET.

George Pavlou is Professor of Communication Networks in the Dept. of Electronic & Electrical Engineering, University College London, United Kingdom, where he coordinates the activities of the Networks and Services Research Lab. He holds a BEng in Electrical and Mechanical Engineering from the National Technical University of Athens, Greece, and MSc and Ph.D. degrees in Computer Science from University College London, UK. He is responsible for a number of European and UK research projects and industrial collaborations. His research interests focus on network management, networking and service engineering, including aspects such as network dimensioning, traffic engineering, quality of service management, ad hoc/mesh/sensor networks, policy-based systems, autonomic networking and communications middleware. He has also contributed to standardization activities in ISO, ITU-T, TMF, OMG and IETF.