Improving QoS Goodput in Hostile Mobile Ad-Hoc Environments

Ayman Helweh-Hannan
Network Research Group, School of Computer Science
Universiti Sains Malaysia, Penang, Malaysia. ayman@nrg.cs.usm.my

Abstract- The tremendous advances in wireless technologies have driven an immense interest in mobile ad hoc networks (MANETs). However, the performance of MANETs is highly affected by the behaviour of its constituting nodes, which must cooperate in order to provide the basic networking functionality. We present a solution that detects and avoids misbehaving nodes which agree to route packets for other nodes and subsequently drop these packets. Such misbehaviour is of direct effect on Quality of Service (QoS) solutions, namely the QoS goodput metric. The solution takes a transparent layered approach and assumes no security constraints. The solution was simulated using NS2. Experiments were done using multiple variations of mobile ad hoc environments, according to the hostility degree, mobility scenario and traffic load. The experimentation results show that the solution consistently detects and avoids misbehaving nodes, and improves the goodput by up to 25%.

I. INTRODUCTION

Mobile ad hoc networks (MANETs) are the product of the new "mobile computing" paradigm. Mobile systems use wireless technology for transparent self-organized communication. Unlike conventional wireless networks, MANETs are self-creating, self-organizing and self-administering. Mobile nodes freely join and leave the network and self-organize into arbitrary and temporary network topologies, allowing people and devices to seamlessly and instantly inter-network in areas without any pre-existing communication infrastructure. Nodes are both hosts and routers: they transiently associate with peers that are within their connectivity range and implicitly agree to provide the basic network functions. The unique benefits and flexibility offered by MANETs elicited immediate interest among military, police and rescue agencies; especially in disorganized or hostile environments. However, many challenges, including security and quality of service (QoS), must be addressed in order for MANETs to become a reliable and practical mean of communication. Since MANETs rely on its constituting nodes to provide the necessary networking functions; the presence of misbehaving nodes paralyzes most network functions. QoS performance is particularly degraded because of these misbehaving nodes. We propose and implement a solution called the Snoopy-Guard to detect and avoid misbehaving nodes under different variations of MANET environments. Experimentation results show that the Snoopy-Guard consistently detects and avoids misbehaving nodes in MANETs under various hostility degrees, traffic loads and mobility scenarios. The Snoopy-Guard improves the QoS goodput (data delivery ratio) in all hostile environments and incurs acceptable routing overhead.

Besides the lack of infrastructure and central authority, the pivotal role played by mobile nodes in creating, organizing and administering MANETs places great importance on the quality of the participating nodes. The presence of misbehaving nodes greatly affects the integrity and performance of the network. The effect is more drastic in environments where QoS is of importance. QoS performance is directly affected by the quality of the nodes participating in the QoS provisioning process. Current QoS solutions are mostly intended for operation in a safe trusted environment, they implicitly assume the participating nodes to be cooperative and well-behaved [5, 6]. Such an assumption is not valid in most realistic environments. MANETs' 'flexibility' and lack of central authority allow any node, including malicious and un-cooperative (selfish) nodes, to easily join the network. Protecting the performance of QoS solutions in such environments becomes extremely important. Incorporating wholesome security solutions within QoS frameworks rather complicates the already non-trivial process of QoS provisioning. Hence, the need arises to devise alternative approaches to protect QoS performance without radically complicating it.

II. RELATED WORK

The existing works that address the problem of misbehaving nodes in MANETS can be classified into three categories: prevention, detection and toleration. The prevention approach builds a line of protection, trying to prevent misbehaving nodes from participating in routing. Detection allows the existence of misbehaving nodes, but detects them and avoids using them in routing. Toleration on the other hand, does not detect misbehaving nodes; instead it seeks to function well in their presence [4, 7]. The Snoopy-Guard we present is a detection approach; we will next discuss two closely related detection solutions.

Marti et al. [1] proposed a solution where a node detects a misbehaving successor along a packet's path by promiscuously listening on its wireless interface waiting for the packet it forwarded to its successor to be accordingly forwarded by the successor node. They term this detection mechanism as watchdog. Upon detection of such a misbehaving successor, the detecting node sends a message to the packet's source identifying the misbehaving node. The Dynamic Source Routing (DSR) routing protocol is used and extended with the watchdog and path-rater mechanisms. The path-rater is implemented in the source node, it decrements the rating value associated with the misbehaving node and may cause the rating
of the whole route to change, resulting in the source node using another route. The watchdog identifies misbehaving nodes, while the path rater avoids routing packets through these nodes. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. The watchdog does this by listening promiscuously to the next node's transmissions. If the next node does not forward the packet, then it is misbehaving. The path rater uses this knowledge of misbehaving nodes to choose the network path that is most likely to deliver packets. Each node maintains a rating for every other node it knows about in the network. It calculates the path metric by averaging the node ratings in the path.

Bypassing Misbehaving nodes Routing (BMR) [2] is a detection approach able to bypass misbehaving nodes and select a "good" path to route packets. The BMR algorithm has two phases: the testing phase and the delivery phase. It identifies the shortest good path during the testing phase and transmits packets along this path in the delivery phase. The basic idea to identify a good path is motivated by the following fact: Although bad paths may exhibit various behaviours, from an end-to-end perspective, there is only one preferred packet delivery behaviour pattern of good paths, which is, delivering packets correctly with low loss rate and small delay. Paths which follow this pattern can be regarded as good paths; paths which deviate from this pattern are bad paths. From this idea, packet transmissions are used to measure the end-to-end performance of the path in use. Based on the measurement results, the testing heuristic determines whether the path is good or not. Paths are tested according to the ascending order of their lengths until a good path is found or there is no path left. In the first case, the algorithm enters the delivery phase with the good path; in the second case, the algorithm chooses the one with the highest measured packet delivery ratio.

III. TERMS AND DEFINITIONS

The Snoopy-Guard security-awareness solution employs a passive security attack. Eavesdropping is an attack against the confidentiality requirement. The attacker listens in on a communication between two or more parties in order to get the transmitted data. This is often easily done in wireless networks where there is no physical control over the links that the message is transmitted onto. The Snoopy-Guard actually uses this passive attack to improve the quality of service by enabling a node to promiscuously listen (eavesdrop) to its neighbours to ensure they are well-behaving.

Promiscuous listening is defined as stealthily listening to transmissions from other nodes within the node's transmission range. Those transmissions do not involve the eavesdropping node.

The term neighbor is used to denote a node within the wireless transmission range of the eavesdropping node. Nodes are able to eavesdrop on all their neighbors. A node's neighborhood denotes all the nodes within the node's transmission range.

After the Snoopy-Guard detects a misbehaving node, it avoids (isolates) it. Isolation is a security goal that requires the protocol to be able to identify misbehaving nodes and render them unable to interfere.

The node misbehaviour dealt with by the Snoopy-Guard is limited to a node dropping data packets originating from other nodes after agreeing to forward these packets. Two types of misbehaving nodes are covered: selfish nodes and malicious nodes. Selfish nodes use the network but do not cooperate, in order to save battery life for their own communications. They do not intend to directly damage other nodes, nonetheless they degrade QoS performance. Malicious nodes on the other hand aim at damaging other nodes' communication and interrupting normal network operation. Saving battery life is not a priority. More than one malicious node may collude with each other to intensify the damage or to escape the detection mechanism [4]. A malicious node can deploy a variety of Denial of Service (DoS) attacks, but we are mainly interested in the attacks caused by failing to forward data packets due to their effect on QoS performance.

The Snoopy-Guard deals with the blackmailing problem. A malicious node may blackmail a legitimate node by unjustifiably advertising that this node is misbehaving. This results in other nodes avoiding the legitimate node which causes the performance to drop. The Snoopy-Guard avoids blackmailing without extra authentication or trust management overhead.

Another addressed issue is the problem of false positives. A false positive is a well-behaved node falsely marked as misbehaving. Besides blackmailing, false positives can occur due to congestion or ambiguous collisions which lead the eavesdropping node to believe that this node is intentionally dropping data packets, when in fact the dropping is due to other reasons such as a software/hardware malfunction, a full queue buffer or external interfering noise.

IV. THE SNOOPY-GUARD

The Snoopy-Guard is based on the Ad hoc On-demand Distance Vector (AODV) routing protocol. AODV was chosen instead of source routing protocols (such as Dynamic Source Routing (DSR)) which were used by other solutions to the misbehaving nodes problem because it provides a local transparent mechanism to detect and avoid misbehaving nodes at the node level without informing the sender or receiver. In AODV, all routing decisions are taken locally at each forwarding node. No authentication is required since nodes do not communicate in regard to detected misbehaving nodes. This feature is important in order to build a layered transparent security-awareness solution to be integrated with QoS provisioning.

The Snoopy-Guard assumes that all links between nodes support bi-directional communication symmetry. Also wireless interfaces that support
promiscuous mode operation (eavesdropping) are assumed. Finally, we assume that the power needed to receive a packet is substantially lower than the power needed to transmit a packet.

The Snoopy-Guard adds two tables to each mobile node to be utilized in maintaining information about the behaviour of the neighborhood. AODV's packet forwarding function is modified to enable misbehaving nodes to drop data packets other than their own. The MAC layer is also tapped before address filtering is done to allow eavesdropping on neighbouring nodes.

Each mobile node keeps track of the packets it sends in a pending packet buffer. Each buffer entry contains a unique packet identifier (e.g., a Hashed Message Authentication Code (HMAC) of the packet header and/or payload), the address of the next hop to which the packet was forwarded, the packet's destination address and an expiry time after which a still-existing packet in the buffer is considered not forwarded by the next hop. Each node also keeps ratings of neighbouring nodes it knows about in a node rating table. Each entry in this table contains the node address, a counter of data forwarding failures observed at this node and a counter of successfully forwarded data packets by this node.

A. Detection
Initially, all nodes are marked as legitimate well-behaving nodes. Each node listens to packets sent by nodes within its wireless range (i.e. neighborhood). When forwarding a data packet to a neighbor node (other than the destination node), the node adds an entry for this packet in its pending packet buffer. This is because destinations do not forward their own data packets (as the packets already arrived at its destination), so this shouldn't be considered misbehaviour. If the timer of an entry in the pending packet buffer expires without the node hearing it being forwarded, the node to which it was forwarded is considered to have committed misbehaviour.

This results in incrementing its forwarding failure counter in the node rating table. If the new rate exceeds the threshold then the node is marked as misbehaving. If -while eavesdropping- the node observes a data packet being transmitted by one of its neighbors, it checks to see if the packet exists in the pending packet buffer. If it is in the pending packet buffer it removes its entry and increments the node's forwarding success counter. If the new rating is below the threshold, then the node is rewarded with a well-behaving status.

On the other hand, if the packet it heard did not exist in the pending packet buffer it increments the node's forwarding success counter and adjusts its status according to the new rating as long as the observed data packet's source address is different from the forwarding node's address. This restriction insures that a source node does not gain forwarding credit for its own traffic. This punishes selfish nodes which only forward their own traffic. Also this insures that the rating of a congested well-behaving node that forwards the data packets after their entry in the pending buffer expires gets credit for its well-behaviour. In this manner, its rating will remain the same since both its counters will be incremented (first the forwarding failure counter at the expiry time, and later on the forwarding success counter when it actually forwards the packet).

B. Avoidance
Upon detecting a misbehaving node, the detecting node tries to do a local repair for all routes passing through the misbehaving node. This involves replacing each route that includes this misbehaving node with another one that does not contain any misbehaving nodes that this node knows about. If it fails to do so, it will not send any RERR messages upstream; as opposed to normal AODV route maintenance. Besides avoiding exhausting valuable resources such as bandwidth and battery power, this also prevents blackmailing of legitimate nodes and/or prevents a misbehaving node from dropping packets while sending upstream RERR messages claiming its innocence and accusing its downstream node.

To avoid constructing new routes which traverse misbehaving nodes, nodes drop/ignore all RREP messages coming from nodes currently marked as misbehaving. Also, all packets originating from a misbehaving node can be dropped as a form of punishment. Only dropping data packets will decrease a node's rating. Dropping or forwarding control packets, such as RREP, does not affect the rating.

C. Advantages
Unlike the watchdog/path-rater solution, the Snoopy-Guard is tolerant to black-mailing. A malicious node in [1] may blackmail a legitimate node by marking it as misbehaving and reporting it to the source. The Snoopy-Guard's avoidance actions are done locally; it does not use any kind of direct or indirect explicit messages to advertise a node's misbehaviour. Also the watchdog/path-rater lacks the punishment of misbehaving nodes, as these nodes can still send packets; hence selfish nodes are actually rewarded for their un-cooperation. The Snoopy-Guard on the other hand punishes misbehaving nodes by rejecting their routing demands.

By taking an end-to-end point of view, the BMR algorithm provides a unified solution for many node misbehaviours. However, BMR must test a path before it uses it for packet delivery, which introduces considerable overhead. Furthermore, BMR requires the misbehaving nodes to behave consistently in the testing phase and the delivery phase, which is not always the case. Problems arise if a selfish node behaves well during the testing phase and starts to drop the data packets during the delivery phase due to declining resources. The Snoopy-Guard avoids this problem by not basing a node's mark upon temporary behaviour; it continuously monitors the node and dynamically adjusts its rating. BMR also makes quite a few security assumptions. For example, a priori trust relationship between the source and destination nodes (which are both assumed to be well-behaved) must exist. The Snoopy-Guard makes no assumptions
about trust relationships or the behaviour of the source, destination or other nodes. Also BMR requires that each node must have a global unique identifier, which introduces the overhead of identifier allocation and duplicate identifiers detection. Finally, the good path model used by BMR only works well under lightly-loaded networks. In the heavily-loaded network, the behaviours of good paths and bad paths can be indistinguishable due to congestion.

The Snoopy-Guard deals with false positives by allowing for dynamic and continuous evaluation of a node's data forwarding behaviour. Thus it gives a chance for a false positive to regain its legitimate status after forwarding a number of data packets enough so that its rating drops below the threshold.

D. Limitations

A misbehaving node can still drop packets without being detected as long as it keeps the ratio between the packets it drops to the packets it forwards below the threshold.

This is referred to as partial dropping. Though the Snoopy-Guard is not able to prevent packet dropping completely, it forces misbehaving nodes to forward a percentage of the packets passing through them, leading to a sustained performance in such a hostile environment.

Malicious nodes may collude by forwarding packets through other nodes in order to increase the number of their forwarded packets, which keeps the ratio of dropped packets to forwarded packets for this node drops below the threshold, in order to evade the Snoopy-Guard. This can be solved by giving a higher weight to forwarding packets stored in the pending packets buffer relative to packets forwarded for other nodes. This solution forces the colluding nodes to forward a lot of packets among each other in order to sustain the well-behaving rating and consume more energy.

Finally, each node promiscuously receives packets not destined to it, consuming power which is very scarce in mobile nodes. To reduce the power consumption cost of the Snoopy-Guard, only the packet headers can be read. However, this solution can not defend against changing the packet's payload.

V. SIMULATION ENVIRONMENT

The Snoopy-Guard was simulated using NS-2 [3]. The NS-2 AODV module was modified and extended in order to implement the Snoopy-Guard's detection and avoidance mechanisms. In order to analyze the performance of the Snoopy-Guard in different environments, various MANET variations were provided in the testing phase. The variations include the hostility degree of the environment, the congestion (traffic load) and the mobility levels of the environment. The simulation parameters are summarized in the following Table I.

![Table I: Summary of Simulation Parameters](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation time</td>
<td>200 seconds</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>50</td>
</tr>
<tr>
<td>Number of misbehaving nodes</td>
<td>0, 15 or 25</td>
</tr>
<tr>
<td>Number of connections</td>
<td>5 or 10</td>
</tr>
<tr>
<td>Network size</td>
<td>670 X 670 meter²</td>
</tr>
<tr>
<td>Traffic type</td>
<td>CBR</td>
</tr>
<tr>
<td>Sending rate</td>
<td>4 packets/second</td>
</tr>
<tr>
<td>Packet size</td>
<td>512 bytes</td>
</tr>
<tr>
<td>Maximum speed</td>
<td>20 meter/second</td>
</tr>
<tr>
<td>Pause time</td>
<td>0, 50, 100 or 200 seconds</td>
</tr>
</tbody>
</table>

VI. RESULTS

The overall results of experiments are presented in Table II. It can be seen that in a safe friendly environment that contains no misbehaving nodes the Snoopy-Guard performs best in the continuous.
mobility scenario. This holds true for both lightly and highly loaded traffic. The second choice would be the high mobility scenario. Low mobility introduces the least goodput gains (in highly loaded traffic it decreases the goodput by 5%) and it increases the overhead the most.

The continuous mobility scenario can be efficiently used in a hostile (30%) environment with both lightly and highly loaded traffic. It provides the best goodput/overhead tradeoff. The low mobility scenario achieves better goodput in the lightly loaded traffic but incurs more overhead. All three mobility scenarios give similar goodput results (8%) in the highly loaded traffic. High mobility is the poorest performer in the lightly loaded traffic. As for overhead, continuous mobility achieves the best results.

### TABLE II
SUMMARY OF EXPERIMENTS RESULTS

<table>
<thead>
<tr>
<th>Traffic Load</th>
<th>Hostility Degree</th>
<th>Metric</th>
<th>Continuous Mobility</th>
<th>High Mobility</th>
<th>Low Mobility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightly</td>
<td>Safe</td>
<td>Goodput</td>
<td>0.0%</td>
<td>0.23%</td>
<td>0.02%</td>
</tr>
<tr>
<td>Hostile</td>
<td></td>
<td>Overhead</td>
<td>-1.5%</td>
<td>3%</td>
<td>21%</td>
</tr>
<tr>
<td>Very</td>
<td>Hostile</td>
<td>Mobility Ratio</td>
<td>28%</td>
<td>26%</td>
<td>38%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Goodput</td>
<td>7%</td>
<td>4%</td>
<td>16%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overhead</td>
<td>24%</td>
<td>34%</td>
<td>35%</td>
</tr>
</tbody>
</table>

The continuous mobility scenario provides the best goodput improvement with the lowest overhead in the very hostile highly loaded traffic. On the other hand, the low mobility scenario provides the best goodput improvement in the very hostile lightly loaded traffic. The low mobility incurs the largest overhead increases in both lightly and highly loaded traffic very hostile environments.

All three mobility rates produce considerable goodput improvements. The low mobility scenario performs exceptionally well (10% in hostile and 25% in very hostile environments). The improvements in the safe environment are very slight. The improvements are substantially increased as the hostility degree of the environment increases.

The overhead increase actually decreases in very hostile environments compared to the results in hostile environments in both the continuous and high mobility scenarios. The high mobility overhead increase is 34% in hostile and 20% in very hostile environments. The low mobility does not show this decrement, its overhead goes slightly up from 35.5% in hostile to 37.5% in very hostile environments.

The Snoopy-Guard also performs very well in the highly loaded traffic. The continuous mobility achieves the same goodput enhancement as the other two mobility scenarios and it does not increase the overhead like them.

The continuous mobility is also the only scenario that improves the goodput in the safe highly loaded traffic environment. That improvement is accompanied by the smallest overhead increase (3.8%).

As a final result, we find that the continuous mobility scenario gives the best overall results, especially in highly loaded traffic. The low mobility can be used to greatly improve the goodput in lightly loaded traffic, with slightly more overhead than the higher mobility scenarios.

### VII. COMPARISON

By comparing the respective results of BMR and the Snoopy-Guard we find that our Snoopy-Guard outperforms BMR in all environments with high mobility. In low mobility scenarios, Snoopy-Guard outperforms BMR in the hostile environment but falls behind BMR in the very hostile environment as shown in Table III.

### TABLE III
SNOOPY-GUARD VS. BMR: GOODPUT METRIC

<table>
<thead>
<tr>
<th>Method</th>
<th>Mobility Scenario</th>
<th>Hostile Environment</th>
<th>Very Hostile Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMR</td>
<td>High</td>
<td>6%</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>5.5%</td>
<td>12%</td>
</tr>
<tr>
<td>Snoopy Guard</td>
<td>Continuous</td>
<td>8%</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>8%</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>8%</td>
<td>6.4%</td>
</tr>
</tbody>
</table>

Apart from the low mobility very hostile environment, we find that the Snoopy-Guard consistently outperforms BMR. The Snoopy-Guard underperforms only in the very hostile environment because of the problem of false positives caused by the ambiguous collision in congested environments.

While the Snoopy-Guard slightly outperformed the BMR goodput gains in most scenarios, it drastically improves over BMR in the overhead department. BMR must test each path before using it. Thus, BMR
increases the overhead ubiquitously by 200%-205% in the high mobility scenarios and by 160%-250% in the low mobility scenarios. The huge overhead in high mobility is attributed to paths getting broken more frequently in high mobility. Subsequently, new paths must be tested before being used. The Snoopy-Guard on the other hand increases the overhead by only 13%-32% in the continuous mobility, 48%-49% in the high mobility and 41%-62% in the low mobility scenarios.

Table IV displays the overhead comparison results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mobility Scenario</th>
<th>Hostile Environment</th>
<th>Very Hostile Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMR</td>
<td>High</td>
<td>205%</td>
<td>200%</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>250%</td>
<td>160%</td>
</tr>
<tr>
<td>Snoopy-Guard</td>
<td>Continuous</td>
<td>32%</td>
<td>13%</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>48%</td>
<td>49%</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>49%</td>
<td>62%</td>
</tr>
</tbody>
</table>

These considerable overhead improvements prove that the choice of AODV to implement the Snoopy-Guard is righteously justified. The cuts in the overhead (compared to BMR) can be attributed to the fact that the Snoopy-Guard does not send any messages when it discovers a misbehaving node. BMR on the other hand is similar to DSR and is thus required to make all routing decisions at the source node. Also, testing each path before rating and using it adds extra overhead. The Snoopy-Guard! AODV flexibly makes those decisions locally at each node, keeping the overhead increases to a minimum.

VIII. CONCLUSION

We have proposed and simulated a solution that detects and avoids misbehaving nodes which drop data packets. The solution operates transparently as a security-awareness layer by making all the detection and avoidance decisions locally. No security constraints are assumed, which facilitates integration with QoS solutions. Experimentation results using variations of MANET environment show that the solution improves the goodput by up to 25% with considerably less overhead trade-off compared to other proposed solutions.

REFERENCES