Advanced Detection of Selfish or Malicious Nodes in Ad hoc Networks

Frank Kargl, Andreas Klenk, Stefan Schlott, and Michael Weber

University of Ulm, Dep. of Multimedia Computing, Ulm, Germany

Abstract. The fact that security is a critical problem when implementing mobile ad hoc networks (MANETs) is widely acknowledged. One of the different kinds of misbehavior a node may exhibit is selfishness. A selfish node wants to preserve own resources while using the services of others and consuming their resources. One way of preventing selfishness in a MANET is a detection and exclusion mechanism. In this paper, we focus on the detection phase and present different kinds of sensors that can be used to find selfish nodes. First we present simulation results that show the negative effects which selfish nodes cause in MANET. In the related work section we will analyze some of the detection mechanisms proposed in literature so far. Our new detection mechanisms described next are called *activity-based overhearing*, *iterative probing*, and *unambiguous probing*. Simulation-based analysis of these mechanisms show that they are highly effective and can reliably detect a multitude of selfish behaviors.

1 Misbehaving nodes in Ad hoc Networks

Mobile ad hoc networks (MANETs) rely on the cooperation of all the participating nodes. The more nodes cooperate to transfer traffic, the more powerful a MANET gets. But supporting a MANET is a cost-intensive activity for a mobile node. Detecting routes and forwarding packets consumes local CPU time, memory, network-bandwidth, and last but not least energy. Therefore there is a strong motivation for a node to deny packet forwarding to others, while at the same time using their services to deliver own data.

In table 1 we analyze different possibilities for a selfish node to save resources in a MANET based on the DSR routing protocol [1,2]. It uses the attack-tree notation proposed by Bruce Schneier [3] that allows the categorization of attacks that all lead an attacker to reach a specific goal. Alternatives to reach this goal are denoted with OR, multiple steps that are necessary with AND. Using the numbers in the table, we can easily describe different attacks. For example, attack 3.1 stands for "Drop data packets".

Whereas most of the attacks based on manipulations of routing data can be detected by the use of a secure routing protocol like Ariadne [4], SRP [5,6,7,8,9], ARAN [10], or SAODV [11,12], there remain two attacks in the attack tree that cannot be detected this easily. When nodes simply drop packets (case 1.1 and 3.1 in the attack tree), all of the secure routing protocols fail, as they focus only

Attack tree: Save own resources
OR 1. Do not participate in routing
OR 1. Do not relay routing data (case A)
OR 1. Do not relay route requests
2. Do not relay route replies
3. Set hop limit or TTL value in route request/reply to smallest possible value
2. Modify routing data/topology
OR 1. Modify route request
OR 1. Insert additional hops
2. Modify route reply
OR 1. Replace own ID in returned route with detour leading through neighboring nodes
2. Return completely wrong route, provoking RERR and salvaging
3. Insert additional hops
4. Declare own ID in source route as external
2. Stop participation in current route
AND 1. Provoke route error
OR 1. Create arbitrary RERR messages
2. Do not send ACK messages (causing RERRs in other nodes)
2. Do not participate in following route request (A.1)
3. Do not relay data packets
OR 1. Drop data packets (case B)
2. Set hop limit/TTL to 0/1 (causing a RERR)

Table 1. Attack Tree: Save own resources

on the detection of modifications to routing data but not on the concealment of existing links.

In order to study how this behavior affects a MANET, we have done a number of simulations where we modeled a varying number of selfish nodes according to case A and B from table 1. The simulations were done using ns-2.1b8a and the DSR routing protocol. The scenario included 50 nodes moving in an area of 1500x300m according to the random waypoint model at speeds of $1\frac{m}{s}$ and $20\frac{m}{s}$ with no pause time. Twenty of the nodes were CBR sources sending 4 packets per second. Details of the simulation parameters are given in table 2. These parameters are typical for MANET simulations (see e.g. [13]) and are used for all following simulations.

Figure 1 shows the results of these simulations. We have varied the number of selfish nodes from 0 to 50 (the total number of nodes in the network). It is obvious that this number has a significant effect on the rate of packets that are successfully delivered in the network. In addition the movement rate has a clear effect. The faster nodes move, the lower the delivery ratio becomes. Finally we see that at lower speeds nodes of case B are more detrimental to the network than those of type case A whereas at higher speeds there are no big differences.

Parameter	Value
Number of Nodes	50
Area X (m)	1500
Area Y (m)	300
Traffic Model	cbr
Sending rate (packets/s)	4.0
Max. number of connections	20
Packetsize (byte)	512
Simulationtime (s)	900

Table 2. Simulation parameters

Finally, when all the 50 nodes are selfish, we still get a delivery ratio of around 50%. This is due to the direct node-to-node traffic which does not need relaying. In this case the sender can reach the receiver directly.

What explanations can be found for this behavior? When the number of case A nodes rises in a network, there are fewer nodes available for building up routes. So if no alternative route can be established, there is no route to the destination which means that packets have to be discarded. That reduces the delivery rate. When movement speed rises, the delivery ratio also diminishes as the network in general gets more fragile. But the network still has a reasonable chance of routing around the selfish nodes. This changes with type case B. Here the nodes behave correctly during the route discovery phase. Thus they can be included in regular routes, but then they start to drop all packets. This isn't detected by DSR and no countermeasures are taken. So at a movement speed of $20\frac{m}{s}$ only 10% of the selfish nodes push the probability of a successful packet delivery below 50%.

Our simulations with AODV have revealed a similar behavior. This demonstrates clearly that an effective protection against selfish and malicious nodes is absolutely mandatory for ad hoc networks.

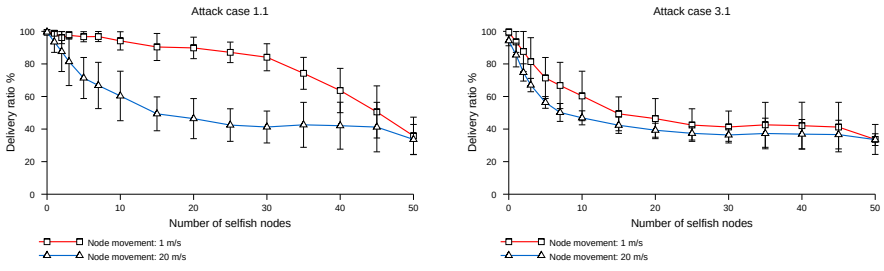

Fig. 1. Selfish attack simulation

2 Preventing selfish nodes: Motivation vs. Detection & Exclusion

There are two approaches of dealing with selfish nodes. The first approach tries to give a motivation for participating in the network function. A typical system representing this approach is Nuglets by Hubeaux et al. [14,15]. The authors suggest to introduce a virtual currency called Nuglets that is earned by relaying foreign traffic and spent by sending own traffic. The major drawback of this approach is the demand for trusted hardware to secure the currency. There are arguments that tamper-resistant devices in general might be next to impossible to be realized [16,17]. A similar approach without the need of tamper proof hardware has been suggested by Zhong et al. in [18]. There exist also other unresolved problems with virtual currencies, like e.g. nodes may starve at the edge of the network because no one needs them for forwarding etc.

Most of the existing work in this field concentrates on the second approach: detecting and excluding misbehaving nodes. The first to propose a solution to the problem of selfish (or as they call it "misbehaving") nodes in an ad hoc network were Marti, Giuli, Lai and Baker in [19]. Their system uses a watchdog that monitors the neighboring nodes to check if they actually relay the data the way they should do. Then a component called pathrater will try to prevent paths which contain such misbehaving nodes. As they indicate in their paper, their detection mechanism has a number of severe drawbacks. Relying only on overhearing transmissions in promiscuous mode may fail due to a number of reasons. In case of sensor failure, nodes may be falsely accused of misbehavior. The second drawback is that selfish nodes profit from being recognized as misbehaving. The paths in the network are then routed around them, but there is no exclusion from service. We will later present more advanced sensors that will allow a better detection of selfish nodes.

In [20,21] the authors describe a distributed intrusion detection system (IDS) for MANETs that consists of the local components "data collection", "detection" and "response" and of the global components "cooperative detection" and "global response". Whereas their architecture is very promising and similar to the one we use in our project, they neglect the aspect how their local data collection should find out on incidents like dropped packets, concealed links, etc.

Another system is the "Collaborative Reputation Mechanism" or CORE [22,23]. It is similar to the distributed IDS by Zhang et al. and consists of local observations that are combined and distributed to calculate a reputation value for each node. Based on this reputation, nodes are allowed to participate in the network or are excluded. In their work, the authors specify in detail how the different nodes should cooperate to combine the local reputation values to a global reputation and how they should react to negative reputations of nodes. For the actual detection of selfish nodes, they only refer to the work of Marti.

A similar approach is conducted by Buchegger et al. with the CONFIDANT system [24,25]. Again, they only marginally describe their detection mechanism and rely mostly on promiscuous overhearing.

3 The Mobile Intrusion Detection System (MobIDS)

We have developed a *Mob*ile *I*ntrusion *D*etection *S*ystem (*MobIDS*) that has a similar structure like some of the systems mentioned above. Because most of the other systems use overhearing, we tried to focus on enhancing this mechanism and develop additional sensors that can be used in parallel to have a higher detection accuracy. As you can see in figure 2, different sensors collect data from the network. As MobIDS is embedded in a complete security architecture called SAM [26], data from the routing protocol SDSR is also taken into account. SAM provides also node authentication based on public/private key pairs that are also used for authentication when distributing ratings (see below). SAM also includes a secure routing protocol called SDSR that negotiates secret session keys between the endpoints or a route and each of the inner nodes.

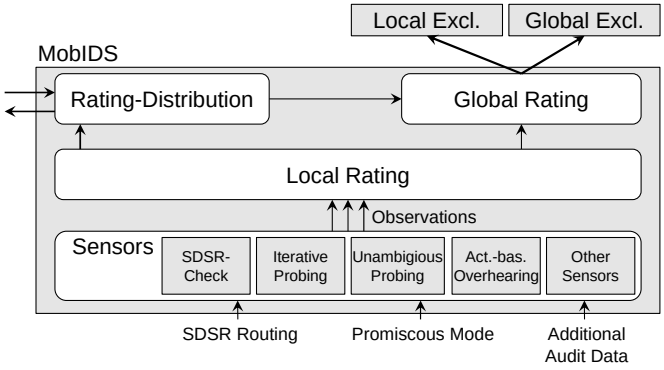

Fig. 2. Overview of MobIDS

The sensors generate *observations*. $\sigma_n^s \in [-1;1]$ represents the n^{th} observation of sensor s. Positive values represent a positive behavior whereas a negative value expresses non-cooperative behavior. All local observations of a node k_i and a sensor s regarding another node k_j at time t lead to a *sensor rating* $r_{k_i}^t(k_j|s) \in [-1;1]$:

$$r_{k_i}^t(k_j|s) = \left(\sum_{\forall n} \rho(t, t_n) \cdot \sigma_n \right) / n$$

where

$$\rho(t, t_n) = 1 - \left(\frac{t - t_n}{T} \right)^x$$

t_n is the time when a specific observation σ_n^s was made. The function ρ makes older observations less important than newer once, observations older than $t-T$

are ignored and can be discarded. x controls the degradation of older observations.

Finally all sensor ratings $r^t_{k_i}(k_j|s)$ are combined into a *local rating* $r^t_{k_i}(k_j) \in [-1;1]$ that expresses the judgment of node k_i regarding node k_j at time t:

$$r^t_{k_i}(k_j) = \sum_{\forall s} w_s \cdot r^t_{k_i}(k_j|s)$$

The local ratings are then *distributed* to neighboring nodes by flooding them periodically in a certain diameter surrounding a node. The distribution of ratings is secured by a simple acknowledge and retransmit mechanism. A node averages all received local ratings (including his own) which results in the *global rating* $gr^t_{k_i}(k_j)$.

As the initial observations are often based on statistical sensors, no node can prove that his rating is actually accurate. So when distributing ratings, these are signed by private keys of each node, but no further attempt is made to prove the credibility of a rating. Instead global ratings are only accepted when at least N nodes have contributed to the rating. This prevents alliances of less than N nodes from excluding other nodes from the network.

Based on the global rating, nodes may be excluded from the current network. MobIDS defines different thresholds t_t, t_e and t_r where t_e is the *exclusion threshold*. If the rating of a node k_i regarding a node k_j sinks below t_e, k_i will invalidate all routes containing k_j and will ignore all packets related to k_j. After some time, old negative observations will expire, so the rating of k_j will eventually increase again. As soon as the global rating exceeds the *rehabilitation threshold* t_e, k_j will be serviced again.

There is one problem: as the distribution process takes some time to deliver the local ratings to all nodes, the global ratings of different nodes regarding k_j may differ by a certain amount ϵ. If $r^t_{k_i}(k_j) < t_e < r^t_{k_l}(k_j)$ then node k_i will stop servicing k_j whereas k_l will still regard k_j as a cooperating node. So when k_i stops forwarding packets to k_j, sensors of k_l may detect this and punish k_i.

Therefor the system contains a third threshold t_t, the so called *tolerance threshold* where $t_e < t_r < t_t - \epsilon$. When $r^t_{k_l}(k_j)$ is below t_t, k_l will tolerate any node to deny service to k_j without deducing negative ratings from this.

In addition to local exclusion the security architecture contains a mechanism that allows global exclusion of nodes from MANETs by invalidating their cryptographic identity. The certified key pairs representing the identities of misbehaving nodes can be revoked, when enough incidents of selfish behavior are recorded. As this is outside the scope of this paper, please refer to [26] for details. We also will not go into any details on performance costs of the MobIDS system as we want to focus on the sensor part for the remaining part of the paper. A detailed description and analysis of the complete system can again be found in [26].

Another question is how the different thresholds should be chosen. Up to now we have adjusted them manually by running different simulations, testing the results and modifying the thresholds. In the final section we will outline future research on how to adjust them automatically.

It is obvious that without good sensors all the following steps (local and global rating, exclusion) will fail to deliver good results. So the rest of the paper focuses on this aspect of MobIDS.

4 MobIDS Sensors

4.1 Activity-Based Overhearing

We already mentioned that there are a number of problems when a node wants to determine whether another node actually relays its packet by using promiscuous mode and listening for the transmission. There are a lot of cases where a relay-node actually forwards a packet but the node overhearing the relay-node's activity will fail to realize that. If e.g. the overhearing node is currently transmitting or receiving data in an IEEE 802.11 network at a lower wire speed (e.g. 5.5 or 2 Mbps) then it won't be able to capture transmissions that happen at other speeds. Other problems include collisions, cooperating selfish nodes and many more.

We tried to improve the classical overhearing sensor, that simply tries to detect missing forwarding like described in [19], to avoid some of these drawbacks. We call the result *activity-based overhearing*. Here a node also tries to overhear forwarding of data packets by its next hop. A node constantly monitors its neighbors' traffic activity for regular data packets sent out by the neighbor nodes. The date of the last regular activity of a neighbor is stored in a table. When it sends a packet to another node and cannot detect a forwarding of the packet by the relaying node, this is esteemed a selfish behavior only when there has been a recent regular activity by this node. This way, the likelihood of a false detection in any of the cases described above is reduced. On the other hand a selfish node can evade our sensor only if it does not generate own traffic. But then it may as well leave the ad hoc network altogether.

Using this mechanism we can significantly improve the detection accuracy. Additionally our architecture introduces a detection threshold . The monitoring node will only trigger an alarm when it detects a certain number of packets being dropped within a certain timeframe. This way a small number of false detections will not lead to any actions against the assumed selfish node.

In order to verify our claim, we have performed a number of simulations which compare traditional and activity-based overhearing. The simulation setup was identical to table 2. Figure 3 shows simulation results at a movement speed of $1m/s$. It verifies the better performance of the activity-based overhearing mechanism. The left graph shows the detection rate of MobIDS in the presence of a specific number of selfish nodes that operate according to case B in the attack tree (forward routing traffic, but drop subsequent data traffic). All values are taken as the average of 10 different simulation runs. Let us assume a network with 2 selfish nodes. Then each of the two nodes is (on average) detected by 1.1 monitoring nodes using traditional overhearing. When we use activity-based overhearing there are 4.5 nodes detecting each selfish node which is 4 times

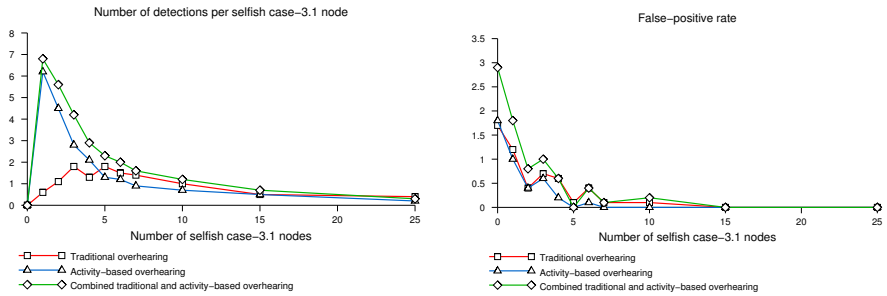

Fig. 3. Traditional vs. Activity-based Overhearing at $1m/s$

better than the classical overhearing sensor. When the number of selfish nodes gets higher (from 5 to 10), the traditional overhearing performs slightly better than activity-based overhearing. We can use both approaches when the results of both sensors are combined. The combined overhearing sensor always considers a missing forwarding activity and devalues the rating of the corresponding node. But the magnitude of this downrating is determined by the time since the last seen activity by this node. The longer a node has been inactive, the smaller the downrating. As you can see from the simulations, this delivers highly acceptable results.

When the number of selfish nodes becomes large[1], detection rates get really bad. Only one or two nodes will detect a selfish node during the average simulation run. This is partially because we assume that selfish nodes do not act as sensors anymore. So in case of 10 selfish nodes you also have to take into consideration that 20% of the sensors are gone. In order to get good results here, we need to combine the overhearing sensor with other sensors like the probing sensor described later.

The right graph in figure 3 shows the false-positives that the overhearing sensors produce. It is important to note that these values are always low compared to the correct positive identifications of selfish nodes. In MobIDS, a node is excluded from the network only if multiple nodes agree on it being selfish or malicious. So when only one node has a false-positive this has no negative effects on the detected node.

Simulations at $20m/s$ (figure 4) show that the detection rate of the activity-based and combined overhearing even increases at higher speeds. This is due to the larger number of routing protocol packets that circulate in the network. This enables the activity detector to predict more precisely whether another host is still in communication range.

[1] more than 10 nodes or 20% of all nodes!

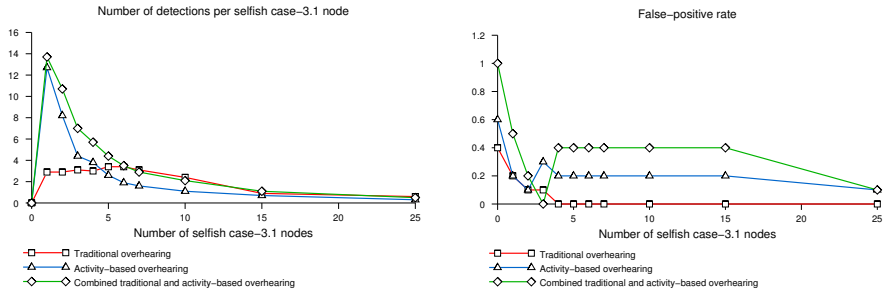

Fig. 4. Traditional vs. Activity-based Overhearing at $20m/s$

4.2 Iterative Probing

In [27] the authors describe a mechanism called *probing* to detect selfish or selfish nodes in a MANET route from source S to destination D. They use onion-encryption to embed a probe command for a specific node X into the normal data packets. When X decrypts its onion layer, it will find this command and send back an acknowledge packet to the source. As soon as an acknowledge is missing, S starts a binary search in the path to find out, where packets are being dropped. S simply sends probes to the selected nodes and waits for their replies. Figure 5 shows the binary search after which we call it *binary probing*.

This approach has a number of drawbacks. The onion-encryption is relatively expensive, as the sender has to encrypt each probe packet multiple times depending on the path length. Furthermore each node has to decrypt the packet once and each packet has to be acknowledged explicitly by the recipient D.

But there is one even more severe problem. There is no reliable detection of the node dropping packets. When a selfish node gets a probe packet it will notice that a probing is under way. Now it can choose to cooperate and forward packets for a limited time (until the probe is over) and then continue to drop packets. Even worse depending on how the probing is realized ([27] is not completely clear on this), it may even be able to selectively drop probe packets destined for another host. This host then doesn't acknowledge the probe and is marked as hostile.

In our mechanism, that we call iterative probing, we use a different approach. Like [27] we assume that a source S has established a secret key k_{SX_i} with each node X_i $(i = 1...n-1)$ in it's path to a destination X_n. There is a command field C included in the packet header that *may* contain a node id X_i which is encrypted by k_{SX_i}, so $C = enc_{k_{SX_i}}(X_i, P)$ $(i = 1...n)$. Otherwise the field contains a random number. P is a random padding which makes multiple probe commands to the same node still look different. So no node can tell whether C contains only garbage or a probe command to another node.

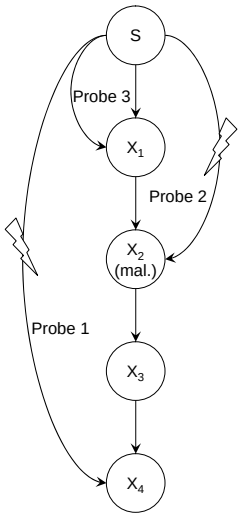

Fig. 5. Binary Probing

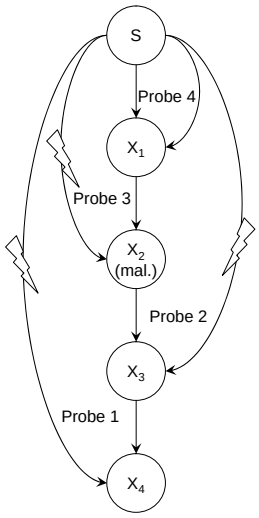

Fig. 6. Iterative Probing

Each intermediate node X_j will now try to decrypt C. If the result is its node ID, it will send an (encrypted) probe reply packet back to S, otherwise it will process the packet as usual. So S has to encrypt only a small portion of the packet and it has to do so only once (compared to the onion-encryption approach) Intermediary nodes will only have to decrypt the small command field and not the whole packet.

In normal operation (that is while S receives packets from X_n as a reply to the packets it sent to X_n) there is no need for probing. But when S hasn't received a packet from X_n for a certain amount of time t, it will sent a probe packet to X_n. If there is no reply within a certain timeout, it will send a probe to X_{n-1} and so on until it receives a reply from a node or reaches X_1. This is called *iterative probing* and shown in figure 6.

Iterative Probing has one advantage over binary probing: a selfish node only gains knowledge of an ongoing probing when it is his turn to answer a probe. So he is not able to blame any nodes on an arbitrary position later in the path by selectively filtering out or forwarding probe packets. Instead there are only two possibilities: he can reply to the probe or he can discard it. All later probe packets are sent to nodes earlier in the path and can not be manipulated any more.

But there is still one problem remaining. Let X_j be the first node from which S receives an acknowledge. There are two possibilities now. In the first case X_{j+1} is the selfish node dropping all packets. Then X_{j+1} will also dropping probe packets and X_j is working properly. In the second case X_j is the selfish node dropping packets. But before dropping a packet, X_j checks if it is a probe addressed to himself. In order to be harder to detect, X_j will then reply to the

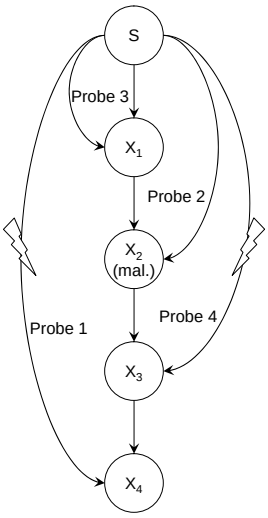

Fig. 7. X_2 answers probes: possible selfish nodes $\{X_2, X_3\}$

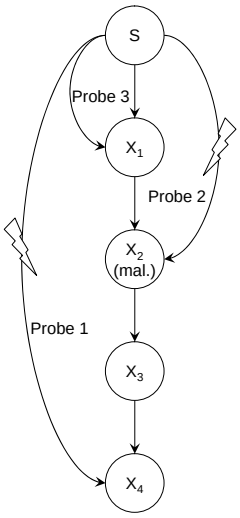

Fig. 8. X_2 answers not: possible selfish nodes $\{X_1, X_2\}$

probe. Due to space limitations we cannot show and discuss the result graphs here.

Although the iterative probing sensor is harder to fail than the binary probing, it can not distinguish which of the two nodes is actually the malicious one. We call this problem the *probing dilemma*. In the next section we will present an approach to prevent this. But first we give an analysis of the iterative probing.

Figure 9 (left side) shows the simulation results for the iterative probing sensor facing the standard adversary – a selfish case B node. Even for 10 selfish nodes we still have an average of 4.9 nodes detecting each selfish node. The

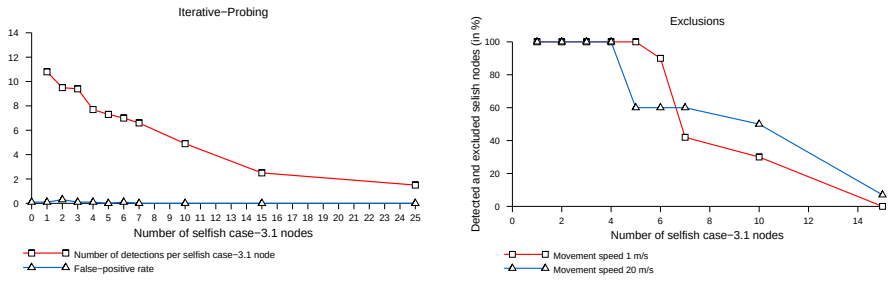

Fig. 9. Iterative Probing and exclusion of selfish nodes

false-positives are negligibly low. So probing is an efficient way of detecting selfish nodes.

4.3 Unambiguous-Probing

As indicated above, the probing techniques described so far face a serious problem: probing can not unambiguously detect a selfish node. Even worse, the standard probing described in [27] allows a malicious node to make another arbitrary node look selfish. Our iterative-probing can narrow the potential adversary nodes down to two nodes. In order to clearly identify one of these nodes as being responsible for the dropped data packets, we can combine the iterative probing with overhearing. Let X_j and X_{j+1} be the nodes that are suspicious of dropping packets like described above. Now we can verify if X_j is dropping the packet by asking X_{j-1} to check if he can overhear the forwarding of a following probe packet by node X_j. If this probe fails and X_{j-1} can't hear X_j forwarding the packet, then it is very likely that X_j is dropping the packets, otherwise X_{j+1} is the node responsible for the packet drop.

4.4 Overall Detection Rate

MobIDS combines all the presented sensors in order to make a decision on excluding nodes from the network. Our simulation results show that the detection of misbehaving nodes is very accurate and we have practically no false accusations. Figure 9 (right side) shows the percentage of discovered and excluded selfish nodes at different movement speeds. In this scenario, three different nodes were needed to detect another node as selfish in order to exclude it from the network. In the simulations, we used combined-overhearing, unambiguous-probing and route-request scanning sensors in parallel. The route-request scanning sensor is a specialized overhearing sensor that specifically checks whether route requests of the routing protocol are rebroadcasted correctly by neighboring nodes. So it can detect misbehaving nodes that do not forward route requests properly. Due to space limitations, it was not presented here.

5 Conclusion and Future Work

As we have seen the construction of sensors to detect selfish or malicious nodes in ad hoc networks is a complex task. In this paper we have presented a number of different sensors that can detect different kinds of selfish nodes with a good confidence as shown by our simulation results. If multiple sensors are active in parallel and a selfish node is detected by a number of these sensors, then this is a good indication for excluding the node from the network.

One remaining problem with our current simulations is that all the thresholds need to be set manually in order to get good detection results. So in the future we will try to find ways how these values can be set and adjusted automatically during operation. Possible candidates might be some kind of an adjustment

algorithm or a self-learning system using neural networks. Furthermore we plan to develop and test additional sensors that will e.g. use topology information from the routing protocol in order to detect selfish nodes.

References

1. David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). `http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt`, April 2003.
2. Charles E. Perkins, editor. *Ad Hoc Networking*. Addison-Wesley, 2001.
3. Bruce Schneier. Modeling security threats. *Dr Dobb's Journal*, December 1999. also available as `http://www.ddj.com/documents/s=896/ddj9912a/9912a.htm`.
4. Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks. In *Proceedings of MobiCom 2002*, Atlanta, Georgia, USA, September 2002.
5. Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Routing for Mobile Ad hoc Networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 2002. also available as `http://wnl.ece.cornell.edu/Publications/cnds02.pdf`.
6. Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Routing for Mobile Ad Hoc Networks. Working Session on Security in Wireless Ad Hoc Networks, EPFL, (published in Mobile Computing and Communications Review, vol.6, no.4), June 2002.
7. Panagiotis Papadimitratos and Zygmunt J. Haas. Securing Mobile Ad Hoc Networks. In M. Ilyas, editor, *Handbook of Ad Hoc Wireless Networks*. CRC Press, 2002.
8. Panagiotis Papadimitratos, Zygmunt J. Haas, and P. Samar. The Secure Routing Protocol (SRP) for Ad Hoc Networks. draft-papadimitratos-secure-routing-protocol-00.txt, December 2002.
9. Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Link State Routing for Mobile Ad Hoc Networks. In *IEEE Workshop on Security and Assurance in Ad hoc Networks, in conjunction with the 2003 International Symposium on Applications and the Internet*, Orlando, FL, January 2003.
10. Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. In *Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP)*, November 2002. also available as `http://signl.cs.umass.edu/pubs/aran.icnp02.ps`.
11. Manel Guerrero Zapata. Secure Ad hoc On-Demand Distance Vector Routing. *ACM Mobile Computing and Communications Review (MC2R)*, 6(3):106–107, July 2002. also available as `http://doi.acm.org/10.1145/581291.581312`.
12. Manel Guerrero Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe 2002)*, pages 1–10, September 2002. also available as `http://doi.acm.org/10.1145/570681.570682`.
13. Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Mobile Computing and Networking*, pages 85–97, 1998. also available as `http://citeseer.nj.nec.com/broch98performance.html`.

14. Levente Buttyán and Jean-Pierre Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Technical Report DSC/2001/001, EPFL-DI-ICA, January 2001.
15. Levente Buttyán and Jean-Pierre Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.
16. R. Anderson and M. Kuhn. Tamper Resistance - a Cautionary Note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996. also available as `http://citeseer.nj.nec.com/article/anderson96tamper.html`.
17. Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In *IWSP: International Workshop on Security Protocols, LNCS*, 1997. also available as `http://citeseer.nj.nec.com/anderson97low.html`.
18. Sheng Zhong, Jiang Chen, and Yang Richard Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of IEEE Infocom '03*, San Francisco, CA, April 2003.
19. Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000. also available as `http://citeseer.nj.nec.com/marti00mitigating.html`.
20. Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *Mobile Computing and Networking*, pages 275–283, 2000. also available as `http://citeseer.nj.nec.com/zhang00intrusion.html`.
21. Yongguang Zhang, Wenke Lee, and Yi-An Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *to appear in ACM Wireless Networks (WINET)*, 9, 2003. also available as `http://www.wins.hrl.com/people/ygz/papers/winet03.pdf`.
22. Pietro Michiardi and Refik Molva. Prevention of Denial of Service attacks and Selfishness in Mobile Ad Hoc Networks. http://www.eurecom.fr/ michiard/pub/michiardi_adhoc_dos.ps.
23. Pietro Michiardi and Refik Molva. A Collaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks. In *Proceedings of the 6th IFIP Communication and Multimedia Security Conference*, Portorosz, Slovenia, September 2002.
24. Sonja Buchegger and Jean-Yves Le Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403–410, Canary Islands, Spain, January 2002. IEEE Computer Society. http://citeseer.nj.nec.com/article/buchegger02nodes.html.
25. Sonja Buchegger and Jean-Yves Le Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes - Fairness in Distributed Ad-hoc Networks. In *Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, CH, June 2002.
26. Frank Kargl. *Sicherheit in Mobilen Ad hoc Netzwerken*. PhD thesis, University of Ulm, Ulm, Germany, 2003. also available as `http://medien.informatik.uni-ulm.de/~frank/research/dissertation.pdf`.
27. Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *ACM Workshop on Wireless Security (WiSe)*, Atlanta, Georgia, September 2002. also available as `http://citeseer.nj.nec.com/article/awerbuch02demand.html`.