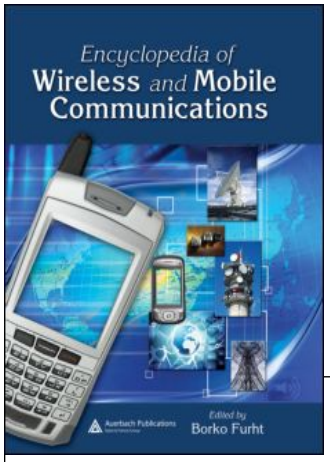

Encyclopedia of Wireless and Mobile Communications
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t789814457

MANET: Selfish Behavior on Packet Forwarding

Djamel Djenouri [a]; Nadjib Badache [b]
[a] Computer Systems Laboratory, Computer Science Department, University of Science and Technology, Algiers, Algeria
[b] Basic Software Laboratory, CERIST Center of Research, Algiers, Algeria

Online Publication Date: 15 April 2008

To cite this Article: Djenouri, Djamel and Badache, Nadjib (2008) 'MANET: Selfish Behavior on Packet Forwarding', Encyclopedia of Wireless and Mobile Communications, 1:1, 576 — 587

MANET: Selfish Behavior on Packet Forwarding

Djamel Djenouri
Basic Software Laboratory, CERIST Center of Research, Algiers, Algeria

Nadjib Badache
Computer Systems Laboratory, Computer Science Department, University of Science and Technology, Algiers, Algeria

Abstract
The limitation in energy resources along with the multi-hop nature of mobile ad hoc networks (MANETs) causes a new vulnerability, that does not exist in traditional networks. To preserve its own battery, a node may behave *selfishly* and would not forward packets originated from other nodes, while using their services and consuming their resources. This deviation from the correct behavior represents a potential threat against the quality and the availability of the forwarding service. Recently, this challenging problem received more and more attention among researchers, and some solutions have been proposed. Still, to the best of our knowledge, no paper surveying these solutions has been published yet. In this entry, we deal with the emergent problem of node selfishness in MANETs, and we provide a state-of-the-art on the current proposed solutions.

INTRODUCTION

Problematic

A mobile ad hoc network (MANET) is a collection of mobile nodes that form on-the-fly and in a self-organized way a temporary wireless multi-hop network, without relying on any established infrastructure. Due to this infrastructureless feature, all networking functions must be performed by the nodes themselves. Especially, packets sent between distant nodes are expected to be relayed by intermediate nodes, which act as routers and provide the forwarding service. The forwarding service is closely related to the routing. It consists of *correctly* relaying the received packets from node to node until reaching their final destination, following routes selected and maintained by the routing protocol. These services (routing and data forwarding) together are at the core of the network layer. The nature of MANET makes cooperation among nodes essential for the system to be operational.

In some MANET applications, such as battlefields or rescue operations, all nodes belong to a single authority (in the application layer point of view) and have a common goal, e.g., soldiers in a military unit or rescuers in a rescue team. For this reason, nodes are cooperative by nature. However, in many civilian applications, such as networks of cars and provision of communication facilities in remote areas, nodes typically do not belong to a single authority and do not pursue a common goal. In such networks, forwarding packets for other nodes is not in the direct interest of anyone, so there is no good reason to trust nodes and assume that they always cooperate. Indeed, nodes try to preserve their resources, and particularly their batteries. To address this constraint many power-aware routing protocols have been proposed,[1,2] but all these solutions do not eliminate the problem owing to the complex nature of the network. As a result, users will be permanently anxious about their limited batteries, which may lead the nodes to behave *selfishly*. A selfish node in the context of the packet forwarding process is the one that takes advantage of the distributed forwarding service and asks others to forward its own packets, but would not correctly participate in this service. This misbehavior represents a potential danger that threatens the quality of service, as well as one of the most important network security requirements, namely the availability.

In this entry, we deal with the problem of selfishness on packet forwarding in MANET, and we sketch the solutions currently proposed to mitigate this problem. We focus on the features, the advantages, and the drawbacks of each one, while avoiding useless details.

Entry Road Map

The remainder of this entry is organized as follows: in the next section, the current reactive solutions are reviewed. First, five basic monitoring solutions, end-to-end ACKs, watchdog, activity-based overhearing (ABO), two-hop ACKs, and probing are presented, respectively. Followed by four reputation-based solutions which are signed token,

Encyclopedia of Wireless and Mobile Communications DOI: 10.1081/E-EWMC-120043599

CORE, CONFIDANT, and friends and foes. Section "Preventive Techniques" is devoted to preventive solutions, where we present: two economic-based approaches (Nuglets and SPRITE), data dispersal, and the game theory based approach. Finally, section "Conclusion" concludes the entry.

REACTIVE SOLUTIONS

Here we present and discuss reactive solutions that aim at detecting selfish misbehavior on packet forwarding when it appears in the network. As we will see, the detection may be limited to the route including the selfish node, or may give deeper information and identify the selfish. Upon the detection of a selfish, routing through this node will be avoided. More stringent solutions suggest to punish these misbehaved nodes by excluding them from the service; some of them allow the redemption and the reintegration of punished nodes.

We split reactive solutions up into two main classes: monitoring- and reputation-based solutions. The monitoring class includes basic approaches that focus on the monitoring phase and suggest techniques to control the forwarding process. Reputation-based solutions are more sophisticated, they propose mechanisms to isolate the nodes detected as selfish. Still, these solutions incorporate a monitor component that use some monitoring technique or other.

Monitoring-based

We will present here four monitoring approaches, two of them are based on the promiscuous mode monitoring, while the others rely on the employment of acknowledgments (ACKs). As we will see, the advantage of the promiscuous monitoring compared with ACKs employment is that the first one requires no overhead for monitoring, and allow to monitor both directed and broadcast packets (packets sent to one neighbor and to all neighbors respectively). However, the promiscuous mode monitoring has many troubles regarding the accuracy on detections, especially when employing the power control technique as we will see later.

End-to-end ACKs

This mechanism consists of monitoring the reliability of routes by acknowledging packets in an end-to-end manner, to render the routing protocol reliable (like TCP). That is, the destination node acknowledges the successfully received packets by sending a feedback to the source. A successful reception implies that the corresponding route is operational, while a failure in the ACK reception after a timeout may be considered as an indication that the route is either broken, compromised, or includes selfish nodes. For each route the routing protocol maintains a rating reflecting the route's reliability, which is updated each time a piece of data (a set of data packets) is transmitted across the route as follows: it is increased for each successful reception (when the source receives the ACK of that piece), and decreased for each failed piece (when a timeout expires without receiving an ACK). When the path rating of a given route decreases below a defined *threshold*, assumed to be high enough to overcome the losses due to collisions, this route will not be used any more. Moreover, the routing protocol may rely on this rating as a metric and choose the most reliable routes. The ACKs must be signed to ensure no-repudiation, otherwise a selfish node may misbehave by not forwarding packets and sending back a *falsified ACK* to the source without being detected. Note that it is beneficial to a selfish to perform like this, since an ACK sending costs much less than a piece of data packet. The signature of the ACKs requires an end-to-end security association between the source and the destination.

The major problem of this technique is the lack of misbehaving node detection. This technique may detect routes containing misbehaving or malicious nodes, and those which are broken, but without any further information regarding the node causing the packet loss. However, this technique helps to avoid sending packets through unreliable routes, and it can be combined with other more sophisticated techniques. It is used in simple mail transfer protocol (SMTP)[3] along with another technique namely data dispersion, which will be presented later, as well as in Ref. 4 combined with probing, which also will be illustrated after. Note that this mechanism is also used in Ref. 5, where the authors propose a *cross-layer* mechanism that exploits TCP ACKs instead of adding explicit ACKs at the network layer, which reduces the overhead. This mechanism, however, is not combined with any detective technique in this solution, since the latter aims only at avoiding unreliable routes.

Watchdog

As far as we know, Ref. 6 is the first paper that treated the problem of nodes' misbehavior in MANET. The authors define the watchdog, which is a basic technique on which many further solutions rely. It aims to detect misbehaving nodes that do not forward packets, by monitoring neighbors in the promiscuous mode. Suppose node S sends packets to D using a route including (possibly amongst others) respectively three intermediate nodes: A, B, and C. When A transmits a packet to B to forward to C, A can check whether B forwards each packet by analyzing packets it overhears during a given timeout. If A overhears a packet it is monitoring during the fixed timeout then it validates its forwarding, otherwise it raises a rating regarding B, and will judge that B is misbehaving and notify S as soon as the rate exceeds a given threshold. This monitoring is generalized for each pair of hops in the source route. The

solution also includes the path-rater component, that selects routes based on the link reliability knowledge.

The watchdog is able to detect misbehaving nodes in many cases, and requires no overhead when no node misbehaves. It allows to monitor all packets regardless whether they are directed or broadcast. Nonetheless, the watchdog fails to detect the misbehavior in cases of collisions, partial collusion, and power control employment as illustrated below.

After a collision at C, B could circumvent re-transmitting the packet without being detected by A. B could also circumvent the watchdog by partially dropping packets, viz. at low rate than the configured accusation threshold. The watchdog fails when two successive nodes collude to conceal the misbehavior of each other, i.e., B could collude with C and do not report to A when C misbehaves. Further, the watchdog technique may cause false detections when the configured threshold fails (the number of packets lost due to mobility and channel condition exceeds the configured threshold), and especially when the monitored node uses the power control technique[1] to preserve its power. That is, when C is closer to B than A, and B transmits packets using a controlled power according to the distance separating it from C, A could not overhear B's forwarding and may accuse it wrongly. Note that the power control technique has been used by many routing protocols proposed after the watch's proposal in the field of power consumption optimization, such as in Refs.1 and 2.

Another serious problem with this solution is that it does not punish the detected misbehaving nodes. Upon the detection of a misbehavior, the detector informs the source node, thereby the rating regarding the misbehaving node is updated. Despite this rating update ensures that transmissions through the misbehaving node is avoided, no measure is taken against this node.

Activity-Based Overhearing (ABO)

In Ref. 7 the authors propose the term *ABO*, which is a generalization of the watchdog. In this technique, a node constantly monitors in the promiscuous mode the traffic activity of all its neighbors, and oversees the forwarding of each packet whose next forwarder is also in its neighborhood. This can increase the number of observations and improve the watchdog efficiency. It also mitigates the collusion problem, as illustrated in the following example:

Consider again the previous example of three aligned nodes A, B, and C such that A monitors B's forwarding toward C. As we have seen previously B could collude with C by not reporting its droppings. The watchdog just fails behind this misbehavior. Assume another node X in the neighborhood of both B and C, as in Fig. 1. In this case, when employing ABO X could overhear B's forwarding and detect afterwards C's dropping. Nonetheless, this general technique suffers from all the other problems of the watchdog, especially the one related to the power control technique as it relies on the promiscuous mode monitoring.

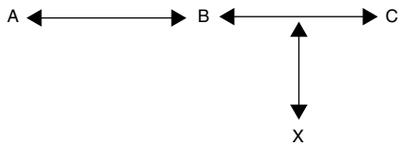

Fig. 1 Network topology.

Two-hop ACK

To mitigate the watchdog's problem related to the power control technique usage, Djenouri and Badache[8] propose a monitoring approach based on a new kind of feedbacks called *two-hop ACK*. In the context of three aligned nodes, A, B, and C such that A monitors B's forwarding to C, node C acknowledges packets sent from A by sending this latter via B a special ACK that travels two hops. Node B could, however, escape from the monitoring without being detected by simply sending A a falsified two-hop ACK. Note that performing in this way is power economic for B, since sending a short packet like an ACK consumes much less energy than sending a data packet. To avoid this vulnerability, the authors use an asymmetric cryptography-based strategy, and suggest that A generate a random number and encrypt it using C's public key, then A validate B's forwarding if and only if it receives later, the random number it generated with the two-hop ACK. Otherwise, it notice a packet dropping for B after a timeout. This random received at A is ciphered by C with A's public key, C does so (ciphers the random number with A's public key) after deciphering the number with its private key. This way B could not falsify a valid two-hop ACK, unless it gets or breaks C's private key. Like the watchdog, A accuses B as soon as the number of two-hop ACK detected drops exceeds a given threshold.

Since the validation at A is related to C's reception and not only to B's forwarding, the solution is independent of the power control usage, thus solves the watchdog's problems related to this issue. Unlike the watchdog, the two-hop ACK ensures that after a collision at C, B cannot escape from re-transmitting the packet without being detected. Further, this solution gets rid of the promiscuous mode employed by the watchdog. Nodes are therefore not required to receive all packets sent in their neighborhood and can turn-off their radios, which is power efficient. The major drawback of this solution is its important communication overhead, since a two-hop ACK is required for each data packet on each couple of hops. For a route of H hops, $O(2 \times (H-1))$ ACK packets will be engendered for each data packet. The solution was proposed for directed data packets; we remark that because of its cost it cannot be generalized for monitoring broadcast packets. If we try to

do so and require that all two hops neighbors acknowledge broadcast packets, then the complexity for broadcast packets in a network with a degree of connectivity C would be $O(2 \times C^2 \times (H-1))$, which is irrational. We think promiscuous monitoring is mandatory for this kind of packets. The problem related to the overhead has been treated by the authors in their more recent work:[9] the so-called random two-hop ACK protocol, where they suggest to randomize the ACK request. That is, instead of asking an ACK for each packet, the monitor node (node A) does this randomly with a probability continuously updated according to the behavior of the monitored node (node B), in such a way to trust more (low probability) the well behaving nodes. The analysis and the simulation results show that this randomizing of ACK requesting reduces hugely the overhead, especially, when nodes well-behave, while keeping the accuracy on detections. Another disadvantage of this solution (both the ordinary and the random two-hop ACK) is the lack of a punishment policy after detections. However, completing the solution with a punishment policy as well as a robust mechanism allowing nodes to safely exchange experiences with each other have been fixed in the author's perspectives.[9]

Probing

We have seen that the end-to-end ACK approach allows to monitor routes and to detect unreliable ones containing misbehaving or failed nodes, but fails to detect the actual nodes responsible for the unreliability. All the other monitoring solutions, however, directly monitor nodes. The probing approach could be viewed as a combination of route and node monitoring. This approach consists of simply incorporating into data packets commands to acknowledge their receipt. These commands are called probes and intended for selected nodes. Probes are launched when a route that contains a misbehaving node is detected (but not the ID of that node). Awerbuch et al. were the first who used this mechanism. The protocol they propose in Ref. 4, is based on the end-to-end feedbacks to monitor routes, thus requires the destination to return an ACK to the source for every successfully received data packet. The source keeps track of the number of recent losses (ACKs not received over a window of recent packets). If the number of recent losses violates the acceptable threshold, the protocol registers a fault between the source and the destination and starts a dichotomic search on the path, in order to identify the faulty link. The end-to-end ACK employed could be considered as the route monitoring phase, and the dichotomic search as the node monitoring on suspicious routes. The source controls the search by specifying a list of intermediate nodes on the future data packets. Identifiers of these nodes are onion-encrypted. Each node in the list, in addition to the destination, must send an ACK for the packet. These nodes are called probed nodes. The list of probes defines a set of non-overlapping intervals that cover the whole path, where each interval covers the sub-path between the two consecutive probes that form its endpoints. When a failure is detected on an interval, the interval is divided into two by inserting a new probe. This new probe is added to the list of probes appended to future data packets. The process of sub-division continues until a fault is detected on an interval that corresponds to a single link. In this example node I_2 is assumed a selfish node that drops all packets, including those containing probing. It is also supposed not replying to probing commands. As illustrated, I_2 will be detected after 4 probes (when the previous assumptions are held).

This solution suffers from many drawbacks. In addition to the high cost of onion-encryption and the communication overhead, there is no reliable detection of the dropper. A selfish node could analyze each packet it receives before deciding either to forward this packet or not. When it gets a probe packet it would notice that a probing is under way, and would consequently choose to cooperate and forward packets for a limited time, until the probe is over. In the context of the previous example, node I_2 could forward packets including probing and reply to the probing sent to it. This way no failure of probing will take place (no faulty interval will be detected); thus this probing will not be able to detect the selfish node in this case.

In Ref. 7 the authors propose an enhanced probing approach called *iterative probing*. It defers from the previous solution in the fact that each command is addressed to one node instead of a set of nodes. Therefore, the command contains one encrypted node ID added to a special field in data packets. If a data packet includes no probing command then the field will contain a random number, such that a recipient cannot distinguished data packets including probing from regular data packets, unless the recipient is the destination of the probing command. The solution suffers from the problem of important overhead; for an H hops route, $O(H)$ ACK transmissions is required for the first phase and $O(\log(H))$ ACK transmissions to detect a misbehaving node. Overall, $O(H+\log(H))$ is the overhead communication complexity of the solution when a misbehavior appears. This solution is also unreliable. It allows to detect the link containing the selfish node but cannot distinguish which of the two nodes forming the link is actually the misbehaving one, since there is no knowledge of the selfish node behavior upon the reception of a probing (either it sends back the ACK or not). To mitigate this problem Kargl et al.[7] proposed the *unambiguous probing*. The principle of this mechanism is simple and can be summarized as follows: Assume after an iterative probing a link (X_i,X_{i+1}) will be detected. To determine which one of the two suspicious nodes is the guilty (the selfish), the source node asks the node X_{i-1} to check if it can *overhear* the forwarding of X_i. If so then X_{i+1} is the guilty, otherwise the guilty is X_i. This mechanism (unambiguous probing) suffers from the watchdog's problems, as it relies on the promiscuous at the predecessor of the

suspicious link. Note that like two-hop *ACK*, probing was proposed and is applicable only to directed packets.

Reputation-based

The reputation of a given community member can be defined as the amount of trust granted by the other members regarding its well-behavior on a given function, according to their experience with it. Members that helpfully contribute to the community life get good reputation among community's members, while others who refuse to cooperate are badly reputed and gradually excluded from the community. In our context, the reputation of a node is the trustworthiness the other ones grant to it regarding its cooperation and participation in forwarding packets. Our definition is large such that including both solutions that assess nodes' reputation by real values or boolean values (well-behaving vs. misbehaving), provide that they punish bad reputation nodes. Reactive reputation-based solutions are more elaborate than the previous monitoring solutions, and deal with the post-detection issues. Still, to detect selfish nodes they simply incorporate approaches proposed by those basic monitoring solutions. Each node keeps track of each other's reputation according to the behavior it observes, and the reputation information may be exchanged between nodes to help each other infer the accurate values. There is a trade-off between efficiency in using available information and robustness against misinformation. If ratings made by others are naively considered, the reputation system can be vulnerable to false accusations or false praise. However, if only one's own experience is considered, the potential of learning from experiences made by others goes unused, which decreases the efficiency. In the following, we present four solutions based on this general principle of reputation.

Signed token

In Ref. 10 the authors describe a unified network layer solution, based on the approach of mutually according admission in neighborhood using signed tokens. It aims at protecting both the routing and the data forwarding. Threshold cryptography-based signature[11] and the watchdog technique[6] are at the core of this solution. The solution is structured around four closely interacting components: 1) *neighbor verification* that describes how to verify whether each node in the network is well-behaving or selfish; 2) *security enhanced routing protocol* which enhances ad hoc on-demand distance vector (AODV)[12] and extends to the termed AODV-S that explicitly incorporates the security information in routing; 3) *neighbor monitoring* that is based on the watchdog to describe how to monitor the behavior of each node in the network, and how to detect packet droppers, and finally; 4) the *intrusion reaction* which describes how to alert the network and isolate the misbehaving, and serve as a bridge between neighbor verification and neighbor monitoring. Nodes in a neighborhood mutually accord participation admissions, and nodes without *up-to-date* admissions are excluded from any network service. Each node has a token issued by its local neighbors allowing it to participate in the network operations, which implements the concept of participation admission. The token has a period of expiration, whose value depends on how long the holder node has been behaving well (its *reputation*). This latter renews (updates) the token before its expiration. Nodes in a neighborhood collaboratively monitor each other to detect any misbehavior.

Like the two-hop ACK previously presented, this solution employs asymmetric cryptography. There is a global key pair SK/PK (secret key and public key), such that each token carried by a node is signed with SK and broadcast periodically in the hello message to ask for a new validation. Note that the solution uses a hello protocol (the hello protocol consists of periodic broadcast of a special packet called *hello* or *beacon* to inform nodes in a neighborhood about the presence of the broadcaster, which allows each node to be aware about its neighbors.). PK is known by all nodes, but none has the SK. Indeed, each node has a partial key which is a part of SK and participates by providing a *partial signature of order* K, thereby K different partial signatures are sufficient to provide the right signature. In other words, SK is divided among nodes in such a way that K different signatures with K different partial keys are necessary and sufficient to make a signature equivalent to that made by SK. This technique is called polynomial secret sharing.[11] To decide whether to provide a partial signed token for the requestor or not, the requestor's historical behavior is considered, which is drawn according to information collected using the promiscuous monitoring, and detections of neighbors as well. Once a node is detected as selfish the detector informs its neighbors, and the selfish is isolated as soon as K different nodes detect it. Isolating a node in a neighborhood is achieved by not providing it with tokens.

Discussion. Although the authors do not evoke the notion of reputation, we categorize this solution in the reputation-based class since each node is accorded and denied services in its neighborhood according to its past behavior. The reputation value of each node could be simply considered boolean, i.e., well-behaving or selfish. Therefore, well-behaving nodes will be served and accorded tokens, while misbehaving ones will be isolated.

Since the detected misbehaving are isolated and excluded from any network's service, the lack of a punishment mechanism against detected misbehaving nodes problem of the previous basic solutions is resolved. However, this solution has many disadvantages. First, all the watchdog's problems described previously remain untreated, since the neighbor monitoring component completely relies on it. The second disadvantage of this solution is that it prevents a node which has less than K neighbors to communicate,

and poses a critical issue on the choice of the parameter (threshold) K for the sharing of the secret key. The choice of low K weakens the key (it will be more breakable), whereas the choice of high values requires high connectivity which is not always ensured in MANET. Finally, we point out that the solution uses the notion of token expiration, and requires a timestamping mechanism which is problematic in distributed systems. This issue was not treated by the authors.

CORE

Michiardi and Molva[13] suggest a generic reputation-based mechanism termed CORE, supposed to be easily integratable with any network function. Unlike the previous solution this one gives more rigorous definitions to the notion of reputation, and defines three types of reputations: 1) *subjective reputation* that is calculated directly from a node observations, and gives more relevance to the past observations in order to minimize the influence of sporadic misbehavior in recent observations, 2) *indirect reputation*, which is calculated basing on the information (observations) provided by other nodes, and 3) *functional reputation* that combines the subjective and indirect reputation. Each node maintains the three reputations for each other in a reputation table that is updated in two different situations; during the request phase of a given function, and during the reply phase corresponding to the result of the function execution. In the first phase, only subjective reputation related to misbehavior is updated (relying on negative information provided from the monitor component). Whereas, in the second phase only indirect reputations are updated *positively*. That is, a reply message containing a list of all the entities that *correctly* behaved is supposed to be transmitted back to the source node at the end of the function execution, so that the indirect reputations of these well-behaving nodes are *increased*. CORE is implemented with dynamic source routing (DSR), and uses the watchdog for monitoring and collecting direct observations, thus both directed and broadcasted packets could be monitored. It can be applied to packet forwarding function, both on data and route request packets. For the route discovery function, the aim is to detect misbehaving nodes that do not participate in this function and do not forward route request packets. During the request phase of the route discovery, the negative rating factor of the next provider may be observed by the requestor's watchdog, like in Ref. 6, while the identity of the nodes that participate in the function are reported to the initiator during the reply phase. The routing service will be denied to route requests issued from nodes classified as misbehaving, i.e., nodes whose functional reputation values become negative (<0). Similarly, the CORE scheme can be used to monitor the data packet forwarding function during the first step (negative rating observation). But as opposed to the route discovery function, data packet forwarding function does not include separate operations that can be qualified as request and reply phases, which harden the indirect reputation updates. However, the authors propose to add end-to-end ACKs, the transfer of which can be considered as the reply phase.

Discussion. The signed token mechanism[10] problem of preventing nodes with less than K neighbors from communicating described previously does not exist in this solution. Also, in contrast to the previous solution nodes' observations are propagated beyond neighborhoods. However, only the positive observations (of well-behaving) are so propagated but not the negative ones. The purpose is to provide robustness for the solution and prevent the vulnerability of rumors propagation which can cause DoS (denial of service) attacks. This reduces the potential of learning from observations made by others and can decrease the efficiency of misbehavior detections in the network. Contrary to the previous solution where the isolation is performed collectively by all nodes in neighborhoods, the isolation in CORE is performed *unilaterally* by each node based merely on its own view of nodes' behavior. This could represent a potential threat of possible false accusations, as when an isolator does not forward packets for another node *unilaterally* isolated, other neighboring nodes (that are not isolating the appropriate node) would consider this as illegal behavior. Further, the solution does not allow redemption after detection, as when a node is excluded by another node it will not be asked to execute the service for this detector and will never be able to redeem and increase its reputation with it. If the nodes exchange their own experiences with each other (their views of reputations and not only observations), such a *redemption* would be possible. Moreover, note that all the watchdog's drawbacks related to detections are present with this solution, since the solution relies on the watchdog mechanism for monitoring.

CONFIDANT

It is another reputation-based solution, proposed by in Ref. 14. It consists of four components present in each node. The first one is the *monitor* which is very similar to the watchdog.[6] It registers the deviations from the normal behavior and calls the reputation system as soon as a given misbehavior occurs. The *trust manager* is the second component, that deals with the incoming and the outgoing ALARM messages. ALARM messages are sent by the trust manager of a node to warn others of misbehaving nodes, i.e., the protocol is based on negative information propagation. Outgoing ALARMS are generated by the node itself according to its experience observations, or after a misbehavior report reception. The recipients of these ALARM messages are so-called *friends*, which are considered to be configured on a user-to-user basis way. Incoming alarms, originate from either outside friends or other nodes, are checked for trustworthiness before triggering a reaction. The trust manager uses a filtering of

incoming ALARM messages according to the *trust level* of the reporting node. To define trust levels, a general mechanism similar to the trust management used in PGP (pretty good privacy) for key validation and certification has been proposed. In their recent work,[15] the authors propose a modified Baysian mechanism that gives less importance to past observations than recent ones (contrary to CORE that gives *more* importance to past observations), and allows redemption. The CONFIDANT's third component is the *reputation system* that manages the node's view on reputations of the others. Each node's reputation is represented by a rating that is changed according to a rate function, assigning different weights to the type of behavior detection, i.e., the greatest weight for own experience, a smaller weight for observations in the neighborhood, and the smallest one to reported experience. The rationale for this weighting scheme is that nodes trust their own experiences and observations more than those of other nodes. Once the rating of a node exceeds a configured threshold, the *path manager* is called for action. This latter is the last component; it is responsible for punishing the misbehaving nodes by not relaying any packet to them, as well as deleting paths containing misbehaving nodes and re-ranking paths according to nodes trustworthiness.

Discussion. Unlike the previous reputation-based solution (CORE), with CONFIDANT reliable negative information is propagated beyond the neighborhood. To mitigate the vulnerability to DoS attacks by propagating rumors, the trust manager is proposed along with the rate function that assigns different weights to the types of behavior detections in such a way to give more importance to local observations when computing the reputation rating. Moreover, the path manager component clarifies punishments against detected misbehavior. The simulation results[14] show a significant improvement in terms of goodput compared to the standard DSR (with which CONFIDANT has been implemented). Nevertheless, like the previous solution the isolation is performed independently by the path manager of each node. Recall that this could represent a potential threat of possible false accusations, as when an isolator does not forward packets for another node unilaterally isolated, other neighboring nodes would observe that and consider it as illegal behavior when they are not isolating the appropriate requestor. Also, all the watchdog's drawbacks presented previously remain untreated in this solution, since the monitor component fully relies on this technique. Finally, note that in the recent solution employing the Bayesian approach the authors suggest that the views of nodes' reputation are periodically exchanged with each other, which causes an important overhead.

Friends and foes

Contrary to the previous solution (CONFIDANT), friends and foes[16] gives as much importance to the past observations as to the present ones. Thus, it uses a long life memory. In this solution nodes are permitted to publicly claim that they are unwilling to forward packets to some nodes, as each node maintains basically three sets: a set of *friends* to which it is willing to provide services, a set of *foes* to which it is unwilling to provide services, and finally a set of nodes known to act as if it is their foe (they do not provide service packets for it) named set of *selfish*. These three sets are *periodically* broadcasted in the neighborhood. Each node also maintains other variables for its neighbors, especially its view of their friends and foes, that are updated according to its experience and to the messages it receives periodically from its neighbors. When a node is asked to forward a packet it does so only when the asker is a friend, and count accordingly a credit for this friend. Also, every node chooses routes such that the next forwarder is its friend, then monitors the forwarding using the watchdog technique. It deletes a credit for the monitored node if this latter is perceived not to correctly forward the packet, and puts it in the selfish set as soon as the number of packets it drops exceeds a given threshold. The solution allows redemption and permits a selfish node to be reintegrated by broadcasting a special packet (SelfState) acknowledging that it has behaved selfishly with the appropriate nodes. To prevent abusing this mechanism, the selfish is first charged with penalties; it must broadcast two SelfState packets to consume additional energy, and the maximum value of its credit (the maximum number of packets it can send without providing forwarding services) is decreased by all neighbors. In addition to data packets, the authors propose the use of this solution to secure DSR control packets against selfish dropping.

Discussion. This solution defines a robust method of redemption that allows selfish nodes reintegration, while preventing these from abusing other nodes' tolerance. Nonetheless, it suffers from some problems that we illustrated below.

First, this solution has all the watchdog problems on which it relies for monitoring. The second problem is related to the overhead. The authors argue that the solution does not cause important overhead because control packets of each node are merely sent in its neighborhood. However, these packets are broadcast periodically, which could be significant in networks with high connectivity. Further, since each node only keeps information about its current neighbors and the information of nodes leaving its neighborhood are arisen, a mobile selfish can easily circumvent and would never be detected. Finally, note that the solution is integrated with DSR, and is used to secure DSR's control packets from dropping. However, a basic principle of the solution is that each forwarder chooses the next one among its friends. Therefore, routing is made hop-by-hop and the solution is not applicable to a source routing protocol as DSR. Indeed, any reactive hop-by-hop routing protocol could be integrated with this solution, such as AODV.[12]

PREVENTIVE TECHNIQUES

Thus far, we have presented reactive solutions that aim at detecting selfish misbehavior on packet forwarding when it appears in the network. Another class of solutions includes approaches that *proactively* try to mitigate the misbehavior or its effects, either by motivating nodes to cooperate or by taking measures to prevent packets from being dropped before sending them. This is helpful to reduce the problem but does not eliminate it permanently. Thus, a reactive solution which detects such a misbehavior remains essential. In this section we present three approaches we classify as preventive: 1) economic-based solutions, 2) data dispersal, and 3) game theory based solutions.

Economic-based

In the following we present two economic-based solutions, inspired by some economic principles which they project on to the forwarding service in MANET.

Nuglets

Buttyan and Hubaux[17] propose an economic-based approach stimulating nodes to cooperate for packet forwarding in MANET, which they model and analyze in Ref. 18. They introduce what they call *virtual currency* or *nuglets*, along with mechanisms for charging the service usage. The basic idea of this technique is that nodes which utilize a service must pay for it (in nuglets) to the provider ones. This makes nuglets essential for utilizing the network, and renders each node interested in increasing its stock of nuglets by providing services for other nodes. Besides stimulating for the provision of services, this mechanism can also force nodes to make a moderate usage of the network services, since they are charged. Nuglets are represented by counters at nodes: each one's value corresponds to the wealth of the holder. In order to prevent a node from illegitimately increasing its own counter, this latter is maintained by a trusted and tamper-resistant hardware module, termed *security module*. Only this module can directly perform operations on the counter. Nuglets loaded in a packet are protected from illegitimate modification and detachment from its original packet by cryptography mechanisms. The physical and data link layers (where the security module is built) are assumed to be robustly protected, such that users cannot modify them. Further, the neighborhood of a node is assumed not to change very fast, so as to make it feasible for a node to keep track of its neighbors by running a hello protocol. Besides discovering its neighbors, the security module uses the hello protocol (like the signed token described before) to establish and maintain security associations with the security modules of the neighboring nodes.

As for packet forwarding charging, the authors suggest three models: *packet pursue model* (*PPM*), *packet trade model* (*PTM*), and a *hybrid* one. In the first model the source is charged. It estimates the required nuglets on each hop and put the total number estimated of nuglets in the packet, then each forwarder acquires the required nuglets from the packet. The required nuglets charged by a forwarder may depend on many things, such as the amount of energy used for the forwarding operation, the current battery status of the forwarder, and its current nuglets number. If a packet has not enough nuglets to be forwarded then it is *discarded*. The advantage of this model is that it may deter nodes from sending useless data and overloading the network. However, the drawback is that it is difficult to estimate the total number of nuglets that are required to reach a given destination. If the source underestimates this number then the packet will be discarded and the source loses its investment in this packet, whereas an overestimation causes a wasting of the precious nuglets. On the other hand, in the PTM approach the packet does not carry nuglets, but it is traded for nuglets by intermediate nodes on each hop. Each intermediary *buys* it from the previous one for some nuglets (except the first intermediary that receives the packet for free from the source), and sells it to the next one (or to the destination) for more nuglets. This way, each intermediary that provides a service by forwarding the packet increases its number of nuglets, and the total cost of forwarding the packet is covered by the packet's destination. In contrast to the previous model, in this one the source does not need to know in advance the number of nuglets required to deliver a packet. Furthermore, letting the destination pay for the packet forwarding makes this approach applicable in the case of multicast packets. However, a serious disadvantage is that this approach does not deter nodes from overloading the network. Another disadvantage is of overhead, since a price negotiation is required on each hop for each packet. The two models can be combined in the following way: the source loads the packet with some nuglets before sending it, the packet is handled according to the PPM until it runs out of nuglets, then it is handled according to the PTM until the destination buys it. This hybrid model gets over the packet loss problem of PPM.

Discussion. Nuglet is a new economic-based approach that motivates and obliges nodes to cooperate and forward packets for each other, because when a node behaves selfishly it will be unable to send its own packets. Moreover, this solution allows the nodes redemption, since a node which is unable to send its own packets because it runs out of nuglets is not excluded from being asked to participate in the data forwarding service and earning nuglets. But this approach suffers from some disadvantages. If a well-behaved node is not asked to route enough packets then it cannot send enough packets, and will be unfairly excluded. A node may be excluded from the routing process because of its position (it has few neighbors and belongs to just few routes) or because of the communication patterns of its neighbors (they have no

communications with nodes to which it has routes). Furthermore, this technique does not prevent a node with enough nuglets from misbehaving, especially if it has not enough packets to send. Another issue related to this technique is that its robustness totally relies on the famous assumed tamper-resistant hardware, but no detail on such a hardware was provided.

SPRITE

Zhong et al.[19] propose another economic-based solution termed SPRITE, in which each node has a *virtual credit* maintained and continuously updated by a central authority called credit clearance service (CCS). The principle is simple; when a node sends its own messages (as a source) it loses credits, and gains credits when it forwards messages for other nodes. To implement this each forwarder is assumed reporting to the CCS for each message it forwards a *receipt*, a small signed message derived from the original one. This reporting is assumed to be performed whenever the node switches to a fast connection with a backup power. When the CCS gets reports related to a receipt, it charges the source of the message and compensates the intermediate nodes. The credit that an intermediary receives depends on whether its forwarding has been successful, and whether the message has reached its final destination. A forwarding is considered successful if the next node on the route [note that SPRITE is implemented with a source routing protocol (DSR), and the receipt contains the source route of the appropriate message] reports a valid receipt. Signing receipts prevents nodes from forging them, so none can report a receipt without really receiving a message. However, as soon as a node receives a message, it can easily report the receipt without forwarding the message. The compensation strategy takes this problem into account, and prevents reporters that provide receipts of messages which do not reach the finale destination (messages not reported by the destination) from earning credits. The authors provide a modeling and a formal proof of the solution, which shows that the solution is cheat-proof (under a set of conditions). That is, truth telling (reporting receipt only when forwarding a message, and not denying any forwarding) is the optimal strategy for every node. The proof also illustrates that the solution is collusion-resistant. Further, the solution was extended with little modifications to broadcast control packets (like route request of the routing protocol), for which the CCS computes a tree based on receipts it receives before updating credits. This way, redundancy is avoided.

Discussion. Like nuglets, SPRITE is an economic-based strategy that motivates nodes to collaborate. However, the major advantage of SPRITE is that it does not require any tamper-resistant hardware. Also, virtual money in this solution is considered as credits and are not held in packets, contrary to nuglets. Consequently, the strategy of charging the source is efficient for SPRITE, since the problem of packet dropping due to lack of virtual money present with the PPM of nuglets (see the previous subsection) does not exist here. Remember that the source-charging strategy has the advantage of preventing nodes from sending useless data that overload the network, and makes them rational when using the network services. Further, the proposed compensation strategy overcomes collusion (on falsely reporting receipt), providing that the destination well-behaves. Nevertheless, the elimination of the tamper-resistant dependency was ensured by using a *central* authority (CCS) that manages credits, which makes the solution centralized, and thus introduces another drawback. Distributing the CCS is mandatory for this solution to be applicable in MANET, basically featuring total decentralization. Another disadvantage of this solution is that it assumes the cost of reporting a receipt to be negligible, and requires the reporting to be performed when the node switches to a fast connection and gets backup power, which is not always possible in MANET.

Data Dispersal

This scheme is based on Rabin's algorithm[20] and takes advantage of the existence of multiple routes from a source to a destination, to increase the reliability when transmitting packets. It consists of adding *redundancy* to the message to be sent, then the message and the redundancy are divided into a number of *pieces* and dispersed on the available routes, so that even a *partial* reception can lead to the successful reconstruction of the message at the receiver. Note that node-disjoint routes ensure more efficiency. This technique can overcome partial packets loss, that can occur due to misbehavior on some routes used.

This approach is based on a mathematical framework. To illustrate it let us assume that the message which has to be sent is a set of m streams, where a stream is a set of bits (e.g., character or integer) which can be considered as a data unit. We also suppose that the source of the message has N different routes to the destination. Let A be an N random M-vectors, i.e., A is a matrix of N rows and M columns, such as each line can be viewed as a vector of M elements (streams) and each vector is constructed randomly (the elements are random numbers). First, the message is divided into L sequences, each of M streams. If M does not divide m then extra redundancy is added to the message. The result can be modeled by the following matrix: $B = (S_1|S_2|,\ldots,|S_L)$, where each S_i is a column vector of size M.

Consider the following matrix $W = A \times B$, such that $\times$ denotes the matrix product. W's length is N rows and L columns, hence each line vector is a *piece* that can be sent on a different route. M pieces among the N transmitted are necessary to reconstruct B at the reception. The reconstruction of B is performed using the following formula:

$$B = A'^{-1} \times W'$$

such that W' is a matrix formed from M rows of W (M pieces among the well received ones), A' is the corresponding matrix formed from A, and A'^{-1} is the reverse matrix of A'. Note that rows used to form A and the corresponding ones in A' are not inevitably adjacent.

Discussion. The ratio *N/M* or the *redundancy factor* is a crucial parameter for this solution. Increasing this ratio ensures more reliability, since few number of pieces among the overall sent pieces would be required to reconstruct B, but high values of this ratio cause important overhead. On the other hand decreasing the redundancy factor reduces the overhead, but gives less reliability. Therefore, the choice of this parameter is a trade-off issue. It should strike a balance between reliability and overhead. Even though this mechanism does not prevent nodes from misbehaving and does not motivate nodes to cooperate, unlike the previous ones, it is helpful to reduce the selfish misbehavior effects on the communication reliability, and can be combined with a reactive solution. In Ref. 13 the authors propose SMTP, a solution that uses this mechanism. However, this solution has the end-to-end feedback technique drawbacks presented previously, since it relies on it.

Game Theory Based

In this approach the forwarding process is viewed as a game, where nodes have to continually decide whether to forward or not to forward packets. The purpose of this approach consists of defining strategies to ensure fairness to all nodes. Since users may be selfish, there is no guarantee that they will follow a particular strategy unless they are convinced that they cannot do better by following some other strategy. In the game theory terms a strategy which constitutes a *nash equilibrium*[21] needs to be identified. Nash equilibrium can be defined as a strategy profile having the property that no player can benefit from *unilaterally* deviating from the strategy.[22] In other words, it is a feature which ensures that if a cheat player tries to deviate from the strategy whereas all the others follow it, the cheat cannot reap more benefits than the others.

Some solutions based on this approach have been proposed, such as in Refs. 22 and 23. For instance, in Ref. 22 nodes are distributed among classes according to their energy constraints and their expectation of lifetime. The source node asks intermediate ones to relay packets before sending them, then each node has to decide whether to accept or reject forwarding packets from this source. If one node refuses to forward packets then it returns a negative ACK back to the source, and consequently the session is blocked. Otherwise, the request is forwarded until it reaches the last router (destination's predecessor) which sends a positive ACK back to the source. A node that has relayed *much* more traffic than the amount that has been relayed for it (according to a defined factor) refuses to participate in the session. A node that has relayed more traffic than a defined amount also rejects the participation. In other cases, the node agrees to forward packets.

It has been proved that the proposed algorithm leads to a nash equilibrium. That is, if all nodes accurately execute the algorithm then any *individual* deviation from a node will not allow it to reach a greater throughput than the so-called pareto optimal value, reached by all well-behaved nodes in the nash equilibrium. This solution, as well as all the ones based on game theory, trust the ACKs of intermediate nodes. Indeed, a selfish node may agree to participate in a session and to forward packets, in order to give impression that it executes accurately the protocol, but actually would not forward packets when it receives them. The approach needs to be combined with a reactive monitoring solution for resolving this problem.

CONCLUSION

In this entry, we have presented and discussed the different current proposals which deal with the selfish misbehavior on packet forwarding in MANET, focusing on their features and drawbacks. We have first classified them into two main classes: reactive and preventive solutions. Reactive (or detective) solutions aim at *actively* detecting the misbehavior when it appears, while preventive ones try to either proactively prevent any misbehavior by motivating and forcing nodes to cooperate, or take precautions to avoid packets from being lost before sending them. The history of networks security has taught us a valuable lesson: no matter how many preventive measures are inserted in a network, there are always some weaknesses in the systems that one could exploit to break in. Like intrusion detection systems (IDSs) used against attacks and intrusions, reactive solutions are essential to beat the selfishness. We have divided the reactive solutions into two subclasses, monitoring solutions and reputation-based solutions. Monitoring solutions consist of basic techniques for controlling packet forwarding. In this entry five techniques belonging to this category have been illustrated: two among them are based on the promiscuous monitoring, namely the watchdog[6] and the ABO, and three on acknowledgment; end-to-end ACK, two-hop ACK, and probing. The advantage of the promiscuous monitoring solutions is that they require no overhead as long as nodes well-behave, and they allow to easily monitor both directed and broadcast packets. Further, the second one (ABO) mitigates the collusion problem as shown before. However, these solutions fail to detect selfish nodes and may cause wrong accusations in many cases, especially when employing the power control technique. The first technique using ACK is end-to-end ACK, which consists of making the routing protocol reliable like TCP. Though this solution causes an important overhead, and does not detect selfish nodes but only routes including such nodes, it helps routing packets around unreliable routes, and may be

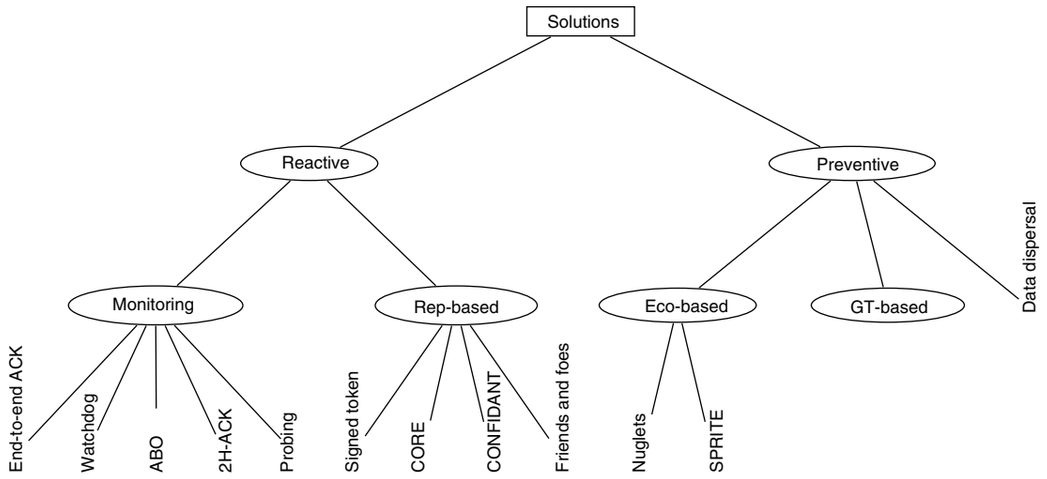

Fig. 2 Classification of the presented solutions.

combined with a more sophisticated solution like SMTP.[3–5] In this latter, the overhead issue was treated, and the TCP ACKs were exploited through a cross-layer approach to reduce it. Two-hop ACK[8] on the other hand allows to detect the selfish nodes and not only unreliable routes, and it enables the usage of the power control technique with no detection problem contrary to promiscuous monitoring solutions. But the major drawback of this solution is the important overhead it engenders. As shown, random two-hop ACK[9] decreases this cost. Probing is the last monitoring technique discussed: it uses the end-to-end ACK to monitor routes, and improves it by adding a dichotomic probing phase to detect the actual selfish nodes whenever a route becomes suspicious. As illustrated, the first solution based on this technique[4] is just unreliable. Iterative probing[7] is more effective but allows to merely detect the link including the selfish node and has high overhead. Unambiguous probing[7] deals with the node detection issue, by suggesting to utilize the promiscuous monitoring at the predecessor of a the suspicious link. This would have inevitably the watchdog's (promiscuous monitoring) problems. The major deficiency related to these monitoring solutions is the post-detection issues, i.e., punishment and selfish nodes knowledge (experience) exchanged between nodes. Reputation solutions are more elaborate and particularly deal with these issues. All the reputation-based solutions involved in this entry utilize the watchdog for the monitoring phase, thus inherit all its drawbacks. We realize that proposing a solution based on a more reliable monitoring approach may present an open research topic.

In Signed token,[10] the first reputation-based solution presented, as well as in the last one (friends and foes) a node's reputation might be viewed as a boolean (selfish vs. well-behaving), contrary to the two others, that provide more rigorous definitions. The signed token uses threshold cryptography involving a parameter K, and nodes in each neighborhood cooperate to provide participation admission to each other. The major drawback of this solution is preventing nodes with less than K neighbors from communicating. Consequently, the parameter K poses a critical and a trade-off issue between operability and robustness. This problem does not exist with CORE.[13] In this latter, nodes' observations are propagated beyond the neighborhood, but only the positive ones. Not propagating negative observations would prevent the vulnerability of propagating rumors aiming DoS attacks, but this way the experience of others gets unused. In contrast, CONFIDANT[14] propagates negative observations beyond the neighborhood, while considering the rumors problem and taking measures to mitigate it at the trust manager component. Further, CONFIDANT with its modified Baysian approach for reputation gives less and less importance to past observations, which allows redemption contrary to CORE that gives more importance to past observations. Nonetheless, in both CORE and CONFIDANT the isolation is performed unilaterally by each node, which might result in false accusation. As when a node isolates another unilaterally and denies forwarding packets for it (punishes it), other neighbors would consider its behavior illegal. The problem is more serious with CORE, than in CONFIDANT where nodes' experiences are exchanged, and it allows redemption. Note that this problem does not exist with signed token. As for friends and foes,[16] it gives as much importance to the past observations as to the present ones, but defines a robust method for redemption. However, this solution suffers from the important overhead it might cause, particularly in highly connected networks, since it relies on periodic broadcast of control messages whose sizes are proportional to the number of neighbors. It also suffers from the mobile selfish problem, as each node only keeps information about its current neighbors.

Regarding preventive techniques, four solutions have been presented: two amongst them are based on economic approaches: Nuglets[17] and SPRITE.[19] The former

assumes a tamper-resistent hardware that manage virtual money or the so-called nuglets, whereas the latter eliminates this assumption, but introduces a central authority (CCS) which is inappropriate for MANET. Distributing this CCS could represent a research trend. Data dispersal is another technique that mitigates packets loss. Even though it does neither prevent selfishness nor motivate cooperation, it could be helpful when combined with a reactive solution like in SMTP. The last preventive solution we presented is relying on the game theory approach. We noted that the major drawback of this solution (and of this subclass in general) is that it totally trusts nodes' ACKs; hence it needs to be combined with a reactive monitoring technique. Finally, Fig. 2 illustrates our classification of all the solutions involved in this manuscript.

REFERENCES

1. Doshi, S.; Brown, T. Minimum energy routing schemes for a wireless ad hoc network. The 22th IEEE Annual Joint Conference on Computer Communications and Networking INFOCOM 2003, San Francisco, California, Mar 30–Apr 3, 2003.
2. Djenouri, D.; Badache, N. New power-aware routing for mobile ad hoc networks. Intl. J. Ad Hoc Ubiquitous Comput. **2006**, *1* (3), 126–136.
3. Papadimitratos, P.; Haas, Z.J. Secure data transmission in mobile ad hoc networks. ACM MOBICOM Wireless Security Workshop (WiSe 2003), San Diego, California, Sept 19, 2003.
4. Awerbuch, B.; Holmer, D.; Nita-Rotaru, C.; Rubens, H. An on-demand secure routing protocol resilient to byzantine failures. ACM Workshop on Wireless Security (WiSe), Atlanta, Georgia, Sept 28, 2002.
5. Conti, M.; Gregori, E.; Maselli, G. Improving the performability of data transfer in mobile ad hoc networks. The 2nd IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005), Santa Clara, CA, Sept 26–29, 2005.
6. Marti, S.; Giuli, T.; Lai, K.; Baker, M. Mitigating routing misbehavior in mobile ad hoc networks. ACM Conference on Mobile Computing and Networking, MobiCom 2000, Boston, MA, Aug 6–11, 2000; 255–65.
7. Kargl, F.; Klenk, A.; Weber, M.; Schlott, S. Advanced detection of selfish or malicious nodes in ad hoc networks. 1st European Workshop on Security in Ad-hoc and Sensor Networks, ESAS 2004, Heidelberg, Germany, Aug 5–6, 2004.
8. Djenouri, D.; Badache, N. A novel approach for selfish nodes detection in MANETs: proposal and petri nets based modeling. The 8th IEEE International Conference on Telecommunications ConTel 2005, Zagreb, Croatia, June 15–17, 2005; 569–574.
9. Djenouri, D.; Ouali, N.; Mahmoudi, A.; Badache, N. Random feedbacks for selfish nodes detection in mobile ad hoc networks. The 5th IEEE International Workshop on IP Operations and Management, IPOM 2005, Barcelona, Spain, Oct 26–28, 2005; Springer-Verlag GmbH: Berlin, Great Britain, 2005; 68–75.
10. Yang, H.; Meng, X.; Lu, S. Self-organized network layer security in mobile ad hoc networks. ACM MOBICOM Wireless Security Workshop (WiSe 2002), Atlanta, GA, Sept 28, 2002.
11. Shamir, A. How to share a secret. Communications of the ACM. **1979**, *22* (11), 612–613.
12. Perkins, C.; Royer, E. Ad hoc on demand distance vector (AODV) algorithm. The 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1999), New Orleans, LA, Feb 25–26, 1999; IEEE Computer Society: 1999; 90–100.
13. Michiardi, P.; Molva, R. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. The 6th IFIP Communication and Multimedia Security Conference CMS 2002, Portoroz, Slovenia, Sept 26–27, 2002.
14. Buchegger, S.; Le-Boudec, J. Performance analysis of the CONFIDANT, protocol cooperation of nodes fairness in dynamic ad hoc networks. 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002), Lausanne, Switzerland, June 9–11, 2002; 80–91.
15. Buchegger, S.; Le-Boudec, J.-Y. A robust reputation system for p2p and mobile ad-hoc net- works. 2nd Workshop on the Economics of Peer-to-Peer Systems, Barkeley, CA, June 4–5, 2004.
16. Miranda, H.; Rodrigues, L. Friends and foes: preventing selfishness in open mobile ad hoc networks. The 23rd IEEE International Conference on Distributed Computing Systems (ICDCS 2003), Providence, RI, May 19–22, 2003; 440–445.
17. Buttyan, L.; Hubaux, J. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Swiss Federal Institute of Technology, Lausanne, Switzerland, Tech. Rep. DSC/2001/001. Jan 2001.
18. Buttyan, L.; Hubaux, J.-P. Stimulating cooperation in self-organizingmobile ad hoc networks. ACM/Kluwer Mobile Netw. Appl. **2003**, *8* (5).
19. Zhong, S.; Chen, J.; Yang, Y.R. SPRITE: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. The 22nd IEEE Annual Joint Conference on Computer Communications and Networking INFOCOM 2003, San Franciso, CA, Mar 30–Apr 3, 2003; 25.
20. Rabin, M. Efficient dispersal of information for security, load balancing, and fault tolerance. J. ACM. 1989, *36* (2), 335–348.
21. Myerson, R.B. *Game Theory: Analysis of Conflict*; Harvard University Press: Cambridge, Mass, 1991.
22. Srinivasan, V.; Nuggehalli, P.; Chiasserini, C.F.; Rao, R.R. Cooperation in wireless ad hoc networks. The 22nd IEEE Annual Joint Conference on Computer Communications and Networking INFOCOM 2003, San Francisco, CA, Mar 30–Apr 3, 2003.
23. Wang, W.; Li, X.-Y. Low-cost routing in selfish and rational wireless ad hoc networks. IEEE Trans. Mob. Comput. **2006**, *5* (5), 596–607.