ARTICLE IN PRESS

Ad Hoc Networks xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

On eliminating packet droppers in MANET: A modular solution

Djamel Djenouri *, Nadjib Badache

CERIST Center of Research, Ben-Aknoun, BP 143, Algiers 16030, Algeria

ARTICLE INFO

Article history:
Received 2 May 2007
Received in revised form 9 October 2008
Accepted 26 November 2008
Available online xxxx

Keywords:
Mobile ad hoc networks
Security
Packet forwarding
Routing
Selfish misbehavior

ABSTRACT

In this paper we deal with misbehaving nodes in mobile ad hoc networks (MANETs) that drop packets supposed to be relayed, whose purpose may be either saving their resources or launching a DoS attack. We propose a new solution to monitor, detect, and safely isolate such misbehaving nodes, structured around five modules: (i) The monitor, responsible for controlling the forwarding of packets, (ii) the detector, which is in charge of detecting the misbehaving of monitored nodes, (iii) the isolator, basically responsible for isolating misbehaving nodes detected by the detector, (iv) the investigator, which investigates accusations before testifying when the node has not enough experience with the accused, and (v) finally the witness module that responds to witness requests of the isolator. These modules are based on new approaches, aiming at improving the efficiency in detecting and isolating misbehaving nodes with a minimum overhead. We describe these modules in details, and their interactions as well. We also mathematically analyze our solution and assess its performance by simulation, and compare it with the watchdog, which is a monitoring technique employed by almost all the current solutions.

1. Introduction

Mobile Ad hoc Networks (MANET) are dynamic and self-organized networks able to operate without depending on fixed or pre-installed infrastructure, using only wireless devices that act both as hosts and routers, and thus cooperatively provide multi-hop communications. The infrastructureless multi-hop nature of MANETs causes vulnerabilities to DoS packet dropping attack and selfish misbehavior. A node can launch a DoS attack by simply participating in the routing protocol to include itself in routes then dropping data packets it is asked to forward. Contrary to DoS attack, selfishness is an *unaggressive* motivation for dropping packets in self-organized MANETs, which merely aims at preserving resources. To save its battery a node might behave *selfishly* by not forwarding packets originated from other nodes, while using their resources to relay its own packets towards remote recipients. Regardless of whether the motivation is aggressive or not the packet dropping misbehavior harms the forwarding service in the network, and thus represents a big problem in MANETs.

In this paper we provide a full modular solution dealing with the packet dropping misbehavior and attempting to solve the complete problem, unlike the current solutions that just focus on some sub-problems. Our contribution can be summarized as follows:

- *For the monitoring*: we use the random two hops ACKs approach [1], which overcomes the watchdog's problems in detection effectiveness (presented in the following section) with reasonable overhead. Note that the watchdog [2] is a basic technique on which rely all the current sophisticated solutions that employ monitoring.
- *For the accusation*: we propose a Bayesian approach enabling redemption before judgment. The unique feature of our approach compared to the existing reputation-based ones is that each node *separately* monitors and evaluates the behavior of its successor, with no exchange (overhead) of the estimated behavior as long as the monitored node is considered to behave correctly (does not drop packets). As soon as a node considers another as misbehaving, it will proceed to its isolation

* Corresponding author. Tel.: +213 554 68 93 72; fax: +213 21 91 21 26.
E-mail addresses: ddjenouri@mail.cerist.dz (D. Djenouri), badache@mail.cerist.dz (N. Badache).

1570-8705/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2008.11.003

cooperatively with others. The current solutions based on the Bayesian approach, like [3], or generally speaking based on reputation like [4] require nodes to continuously exchange with each other their estimation of reputations. Therefore, our solution is low-cost in terms of overhead with respect to reputation evaluation.

- *For the accusation approval and isolation*: finally we suggest a social-based approach to approve detections and isolate guilty nodes. This approach's aim is to consider the vulnerability of false accusation attack (rumors), and to decrease false positives caused by channel conditions and nodes mobility. In summary, each node monitors and evaluates the behavior of its successors by itself, and as soon as it accuses a node it launches a procedure to approve this accusation and collaboratively isolate the node in the network. Compared to the current solutions, which use cooperation in the behavior estimation and perform the isolation unilaterally at every node, the collaborative isolation gives our solution many advantages in terms of reducing false positives as we will see later.

All these approaches are structured around five interacting modules, we will illustrate later. Our solution allows benign nodes running it to detect and isolate misbehaving nodes that drop data packets in many cases. It deals with both continuous and selective dropping, but the detection is inevitably slower for the second case due to the tolerance we use to prevent false detections in case of packet collision and node mobility that causes unintentional packet loss. Our isolation mechanism is global, contrary to many current solutions that perform the isolation locally in neighborhoods. In this case, a misbehaving node can simply move away from the region where it was isolated to rejoin the network. Our solution is not vulnerable to this misbehavior. As noted earlier, the rumor vulnerability that may arise from the global isolation strategy is taken into account by our social-based isolation approach. However, we do not consider reintegration of isolated nodes, which can be rational and required in some cases to make the solution fault tolerant. When adding such a reintegration mechanism the effectiveness of the solution should be reconsidered by preventing nodes from abusing the mechanism, which is problematic and presents one of our perspectives. We also do not consider collusive misbehavior where two successive nodes cooperatively misbehave, which also represents a very difficult problem to deal with. The rest of this paper is organized as follows: in the following section the related work is sketched, followed by a detailed presentation of the components of our solution in the third section. Section 4 is devoted to a mathematical analysis and discussion on our solution and its security features, and Section 5 to a simulation study where we compare our solution with the watchdog approach. Finally, the last section concludes the paper and summarizes the perspectives.

2. Related work

The first solution dealing with the problem of misbehavior on packet forwarding is the watchdog [2]. It is implemented with DSR [5], and relies on monitoring neighbors in the promiscuous mode. Each node in the source route monitors its successor after it sends it a packet to forward, by overhearing the channel and checking whether it relays or drops the packet. A monitoring node accuses a monitored node for misbehaving as soon as it detects that the latter drops more than a given number (threshold) of packets. This basic technique has been used by almost all the subsequent solutions. Nonetheless, it suffers from some problems of efficiency in detection, especially when using the power control technique employed by some new power-aware routing protocols following the watchdog's proposal [6] [7]. It may wrongly accuse innocents, or ineffectively miss the detection of misbehaving nodes. This is because the solution supposes that packets transmitted by any node can be received by all the nodes in its neighborhood, which cannot be ensured in all transmissions when using dynamic transmission powers. In addition to the problems related to detections the watchdog does not prevent nodes from misbehaving, since it does not provide any mechanism allowing nodes to exchange their experience, and does not apply any punishment against the detected nodes. More recent solutions [8] deal with this problem, and propose punishment policies together with methods to exchange information on misbehaving nodes.

Yang et al. [9] describe a unified network layer solution to protect both routing and data forwarding in the context of AODV. This solution is based on the approach of mutually according admission in neighborhood through signed tokens, issued using threshold cryptography-based signatures. The token has a period of expiration, whose value depends on how long the holder has been behaving well, and every node has to renew its token before its expatriation by collecting at least K different signatures of the token from its neighbors. Nodes in a neighborhood collaboratively monitor each other to detect any misbehavior using the watchdog, and decide about the delivery of requested token signatures according to the outcome of this monitoring. Compared to the basic solution of the watchdog, this one has the advantage of dealing with punishment policy, and preventing misbehaving nodes from accessing the network. However, it has some drawbacks. 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 from communicating, 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 breakable), whereas the choice of high values requires high connectivity, which is not always ensured in MANET.

Michiardi and Molva [4] suggest a generic reputation-based mechanism that can be easily integrated with any network function, termed CORE. In this paper the authors give rigorous definitions to the notion of reputation, by defining three types of reputations: (i) *subjective reputation* that is calculated directly from a node observations, (ii) *indirect reputation*, which is calculated basing on the information (observations) provided from other nodes, and (iii)

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

functional reputation that combines the subjective and indirect reputation. Each node maintains the three reputations for each other in a reputation table 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 are updated (relying on negative information provided from the monitor component), while 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*. For the route discovery function the two phases can easily be distinguished, as the route request propagation and the transfer of the route reply back to the source. As for the data packet forwarding, the authors suggest to add end-to-end ACKs, the transfer of which can be considered as the reply phase. Nodes with bad reputation will be punished by not relaying their packets. CONFIDANT is another interesting reputation-based solution, proposed by Buchegger and Le Boudec [3]. In this solution, the authors propose a modified Bayesian approach that allows redemption by giving less importance to past observations, contrary to CORE that gives more importance to past observations. The reputation system of CONFIDANT manages the view of the nodes' reputation. Each node reputation is represented by a rating that is updated 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 of neighbors, and the smallest one to reported experience of remote nodes. 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, it will be punished by not relaying any packet for it. The major problem of the reputation systems of both CORE and CONFIDANT is that they always (regardless of the behavior of nodes) require periodic packet exchanges, resulting in an important overhead. Further, the two solutions rely on the watchdog technique in the monitor component, and thus inherit all its shortcomings.

The second class of solutions includes preventive approaches. Buttyan and Hubaux [10] propose an efficient preventive economic-based approach stimulating nodes to cooperate. They introduce what they call *virtual currency* or *nuglets*, along with mechanisms for charging/rewarding service usage/provision. The main idea of this technique is that nodes that use a service must pay for it (virtually in nuglets) to nodes that provide the service. This solution has some drawbacks, 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 towards which it has routes). Furthermore, this technique does not prevent a node with enough nuglets from misbehaving, even when its position is critical to ensure connectivity. Another issue related to this technique is that its robustness totaly relies on the famous assumed tamper-resistent hardware, but no detail on such a hardware was provided.

The nuglets concept is used and generalized to *credit* by Zhong et al. in SPRITE [11], where they attempt to propose an improved solution compared with Nuglets. However, SPRITE introduces a centralized point that is not realistic in the ad hoc context. Some other stimulating preventive approaches are based on game theory, such as [12,13]. In these approaches the forwarding process is viewed as a game where nodes have to continually decide whether to forward or not to forward packets, then the purpose consists of defining strategies to ensure fairness to all nodes, by guarantying the termed *nash equilibrium* [12]. Nash equilibrium can be defined as a strategy profile having the property that no player can benefit from *unilaterally* deviating from the strategy. 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 reach more benefits than the others. These solutions trust all the responses of the nodes regarding their decision of packet forwarding. Obviously, a misbehaving node may agree to participate in forwarding packets, in order to give impression that it executes accurately the protocol, but actually drops packets once it receives them. A monitoring solution is required here for resolving this problem.

All these preventive solutions motivate nodes to cooperate, but contrary to the previous detective ones they do not aim at detecting the misbehaving nodes and are far from being able to eliminate the problem. In [14], Papadimitratos and Haas present the SMT protocol. It is a hybrid solution that mitigates the misbehavior effects (packet loss) by dispersing packets, and detects the misbehavior by employing end-to-end feedbacks. This kind of feedbacks allows the detection of the routes containing misbehaving nodes, but fails to detect these nodes. Conti et al. [15] propose another interesting detective end-to-end feedbacks based solution, using a cross-layer interaction between the network and the transport layers to decrease the overhead. To overcome the detection problem of end-to-end feedbacks, Kargl et al. [16] propose *iterative probing* that detects links containing selfish nodes, but this method fails to detect the appropriate nodes. To find the appropriate node on a link after an iterative probing, the authors propose the termed *unambiguous probing*. This probing is based on the watchdog, thus suffers from its problems. In this work, we attempted to propose a new solution to mitigate some problems of the current solutions, and to advance a step forward towards resolving the challenging problem of packet dropping misbehavior.

3. Solution overview

3.1. Assumptions

Our solution operates with the following conditions:

- Communication links between each couple of nodes are FIFO (First-In First Out) and bidirectional, enabling communication in the two directions. If nodes use

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

heterogenous hardware then the link can be operational if the nodes are within the minimum power range of the two transceivers.
- Nodes are mobile and links are not always reliable, i.e. packet collisions and losses are possible.
- Nodes are neighborhood-aware.
- A public key infrastructure (PKI) certificate authority along with an access control scheme are implemented, in charge of authentically distributing public keys while limiting the certificates (IDs and keys) a node can authentically get to one. Although PKI and access control in ad hoc networks are problematic and open research issues some solutions have been proposed and can be used, as reported in [17,18]. Dealing with these problems is definitely out of the scope of our work. Note that the same keys can be employed for other security purposes at other layers, and thus are not specific overhead of our solution.
- The only considered misbehavior in this work is packet dropping. However, we take into account the possible vulnerabilities of our solution a misbehaving node can exploit to either circumvent the solution after dropping packets or to launch another attack on the solution.
- An attacker can spoof IP addresses, falsify packets (as long as there is no authentication on them), and launch a distributed denial of service attack by compromising other nodes. However, we suppose that it cannot get more than one authentic certificate, and therefore cannot authentically use more than one ID. This must be ensured by the assumed access control mechanism together with the PKI.
- We do not require any hierarchy among nodes, and suppose they initially have the same trustworthiness.
- Source routing is used, and promiscuous mode is enabled.
- A node can drop packets continuously (all packets) or selectively. In the latter case, however, no solution can deterministically detect the droppers in all cases due to channel conditions and mobility that cause packet loss. We will discuss the efficiency in detection in the two scenarios. We do not consider collusions where two subsequent nodes assumed to monitor one another collude and mask the misbehavior of each other. Note that currently no monitoring solution detects such a complex misbehavior.

As illustrated in Fig. 1, our solution includes five modules: The first one is the monitor, responsible for controlling the forwarding of packets. The second module is the detector, in charge of detecting the misbehavior of monitored nodes by using the information provided by the previous module. The isolator is our third module, implementing a new social-based approach we suggest for detection approval, in which a node accusing another as misbehaving is required to get witnesses before proceeding to its global isolation. The fourth module is the investigator, that investigates accusations before testifying when the node has not enough experience with the accused one. Finally, the witness is the last module that responds to testimony requests of the isolator. These modules in each node work together and cooperate in order to monitor the forwarding of each successor and make accurate judgment about it. As soon as a monitored node is considered misbehaving, the detector node proceeds to a global isolation against the detected one. Isolating a misbehaving node means: (i) do not route packets through it, to avoid losing them, and (ii) do not forward packets for it, to punish it. In this section we introduce our solution by presenting these modules, and the interactions between them as well.

3.2. Monitor

This module monitors the forwarding of both directed and broadcast packets. Due to their different natures, we provide a separate solution for each kind of packets. Therefore, this module is composed of two submodules: directed packets monitor and broadcast packets monitor. Like the

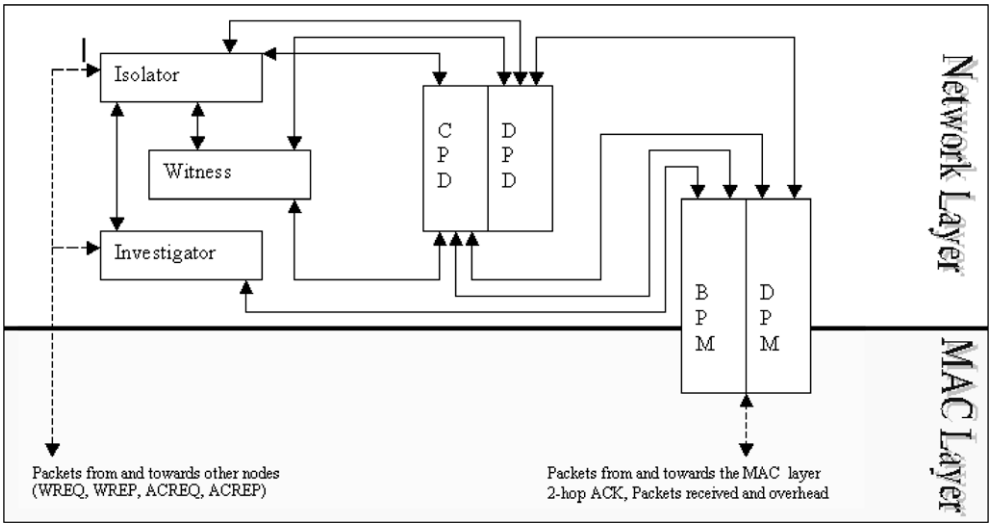

Fig. 1. Solution framework.

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

watchdog, our monitoring solution needs to be implemented with a source routing protocol (in this study we use DSR [5]). Note that contrary to all the other modules located in the network layer, this one is designed across the network and the MAC layers. The reasons will be revealed hereafter.

3.2.1. Directed packets monitor (DPM)

This module deals with packets directed to one recipient. This kind of packets includes data packets, route reply (RREP) packets, and route error (RERR) packets. Each node A monitors its successor B in the source route and checks whether the latter forwards to C each packet it provides, such that C is B's successor in the source route and A could be either the source or any intermediate node. The DPM of node A generates a random number and encrypts it with C's public key (PK), then appends it in the packet's header. When C's DPM receives the packet it retrieves the number, decrypts it using its secret key (SK), encrypts it using A's PK, and finally puts it in a two-hop ACK it sends back to A via B. In the first hop (C,B) the ACK is piggybacked to the ordinary MAC ACK instead of being transmitted in a separate packet. For this reason, a part of DPM needs to be implemented in the MAC layer. We designed this module across the MAC and the network layer (Fig. 1). When A's DPM receives the ACK it decrypts the random number and checks if it matches with the one it has generated, to validate B's forwarding regarding the appropriate packet, and informs the detector module about B's forwarding. However, if B does not forward the packet then A's DPM will not receive a valid two-hop ACK, and will detect this dropping after a timeout. In this case, A's DPM informs the detector about the detected dropping. This process is repeated on each couple of hops from the source to the final destination.

The problem with this first solution [19] is that it requires a two-hop ACK for each packet, which might result in important overhead. To decrease this cost, the DPM *randomizes* the ACK ask [1], viz. A does not ask C an ACK for each packet but upon sending a packet to forward it *randomly* decides with a coefficient p whether it asks for an ACK or not, then conceals this decision in the packet. A simple way to conceal the decision is to exploit the random number. For instance, when the node decides to ask for an ACK it selects an even number, and an odd number when it decides to not ask for the ACK. When the DPM decides to not ask for an ACK it assumes that the packet will be forwarded and notifies the detector about a positive observation (forwarding), and when it asks for an ACK it informs the detector about the result as in the ordinary ACK.

The coefficient p is continuously updated as follows: It is set to 1 (the initial value representing zero-trust) when a timeout expires without receiving the requested ACK, and to p_{trust} each time the requested ACK is received. We define p_{trust} as the trust probability, representing the minimum value of p. This way more trust is given to well-behaving nodes and the ACK requesting is enforced after a lack of one ACK, which allows to achieve all by the same performance in misbehaving detections (true positives) like the ordinary two-hop ACK as we will see later.

3.2.2. Broadcast packets monitor (BPM)

It is not efficient to monitor broadcast packets (RREQs) using the previous technique, because asking for an ACK for all neighbors of the monitor's neighbors (two-hop neighbors) is impractical. For RREQs packets each node monitors each RREQ it forwards (respectively, launches if it is the source) as follows: The monitoring starts from the reception of the RREQ (respectively, its launch if the node is the source) and ends after a timeout from its retransmission. For each RREQ, the transmitter's BPM monitors all its neighbors. It should receive (either as a recipient or in the promiscuous mode) either the RREQ or an appropriate RREP from them (except from which it received the RREQ if the node is not the source). If none of these packets is received from a neighbor A, then the BPM notices a packet dropping for A and informs the detector module about it. Otherwise, it informs the detector about a positive observation. The BPM also keeps track of the packets received and overheard in the promiscuous mode at the MAC level, and safeguards them during a short time for usage by the investigator module as we will see later. The packets capturing needs to be performed at the MAC layer since both data and MAC ACKs are involved, which explains the cross-layer design of BPM. Algorithms 1 and 2 illustrate the network and the MAC components of the monitor module (both DPM and BPM).

Algorithm 1

Algorithm executed by node *i*, describing the Network layer component of the Monitor

Notations:

- R_{key}: encrypting R with Key, • R^{key}: decrypting R with Key, • P_X: the public key of node X, • S_X: the secrete key of node X. $P(X)$: Probability of asking for an ACK for node X

DPM

When receive a packet D from the routing protocol to send to node X (X either the next hop or the destination and i is either the source or a forwarding node):

if (X ≠ D's destination) **then**
  Decide whether to require a two-hop ACK with probability $P(X)$
  if (Two-hop ACK required) **then**
    R=generate an even random number
    add(R,X) to the buffer Wait2HopsACK
  else
    R=generate an odd random number
  end if
  Y=X's successor in the source route
  append (R_{P_Y}, i) to D's header
end if
send D to X

When receive a packet D from the MAC protocol sent by X:

if(X ≠ D's source) **then**
  remove the random number generated by X's predecessor from the header along with the corresponding node address
end if
  send the packet to the network layer protocol

When receive a two-hop ACK packet TwoHopsACK from the MAC layer component

$R' = TwoHopsACK.Rand^{S_i}$
if $(R', TwoHopsACK.sender) \in Wait2HopsACK$ **then**
  remove $(R', TwoHopsACK.sender)$ from Wait2HopsACK
  Inform the detector about a packet forwarding regarding node X
  $P(X) = P_{trust}$
end if

When a timeout of a Wait2HopsACK entry (R,X) is expired

Inform the detector about a packet dropping regarding node X
$P(X) = 1$

BPM

(continued on next page)

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

Algorithm 1 *(continued)*

```
When receive RREQ from X
for each neighbor j except X do
  Set a timer
  Set BroadcatFlag[j] to false
end for
When a timeout of RREQ regarding X expired
if BroadcatFlag[X]=false do
  Inform the detector about a packet dropping regarding X
else
  Inform the detector about a packet forwarding regarding X
end if
```

Algorithm 2
Algorithm executed by node i, describing the MAC component of the Monitor

```
DPM
When receive a packet D sent by X
if (D.MACHeader.TwoHopsACK == true) then
  R = D.MACHeader.Rand^Si
  if (R mod 2 = 0) then
    construct an ACK packet ACKpack
    ACKpack.TwoHopsACK = true
    R' = R_{D.MACHeader.TwoHopsSrc}
    ACKpack.Rand=R'
    ACKpack.SecondDest=D.MACHeader.TwoHopsSrc
    the other fields have to be filed out by the MAC protocol
    send the ACK to X
  else
    send an ordinary ACK if the packet requires an ACK
  end if
else
  send an ordinary ACK if the packet requires an ACK
end if
pass the packet up to the network component after doing the handling required by the MAC protocol (moving the MAC header, frames defragmentation, etc.)
When receive packet D from the network layer
Do the operations required by the MAC protocol (fragmentation, making the MAC header, etc.)
if (D's source ≠ i) then
  D.MACHeader.TwoHopsACK = true
  D.MACHeader.Rand= the random number generated and encrypted by the network component
  D.MACHeader.TwoHopsSrc = i's predecessor
else
  D.TwoHopsACK=false
end if
forward the packet
When receive an ACK packet ACKpack
if (ACKpack.TwoHopsACK ==true) then
  construct a two-hop ACK packet TwoHopsACK
  TwoHopsACK.Rand= ACKpack.Rand
  TwoHopsACK.dest= ACKpack.SecondDest
  TwoHopsACK.sender=I
  send two-hop ACK to ACKpack.SecondDest
end if
do the handling required by the MAC protocol
When receive a two-hop ACK packet TwoHopsACK
pass the packet up to the network layer component
BPM
When overhear a RREQ or RREP packet from X regarding a packet being monitored
BroadcatFlag[X]=true
```

3.3. Detector

The detector is in charge of locally judging the behavior of the monitored nodes using the information provided by the monitor. This module is also composed of two parts. The first one focuses on data packets, while the second on routing control packets.

3.3.1. Data packet detector (DPD)

The monitor allows to confirm the correct forwarding of packets. Though, when the monitor (particularly DPM) notices that some packet has been dropped over a link and informs the detector, the latter should not directly accuse the monitored as misbehaving since this dropping could be caused by collisions or nodes mobility. Indeed, a threshold of tolerance should be fixed. The DPD implements a Bayesian approach enabling nodes to decide about the behavior of each other. In this approach well-behaving of nodes improves their reputation, whereas intentional or unintentional packet dropping decreases it. The Bayesian approach [20] is a mathematical estimation method, which consists in estimating a parameter the observations of which follow a Bernouli distribution by a Beta distribution. This method has already been used by Buchegger and Le-Boudec [3] for estimating node reputation regarding packet forwarding in MANET, but their solution requires periodic transmissions of huge control packets. Since misbehaving is usually an exception rather than the norm, information exchange in our solution is limited to negative impressions. Thereby, our solution is simpler and engenders no overhead when nodes behave well. Hereafter, we describe our Bayesian-based approach.

DPD of node i thinks that each other monitored node j misbehaves with probability θ_j, which is a random variable estimated by a Beta distribution $Beta(a,b)$ in the Bayesian method. For brevity we remove the indices in the following and simply denote this probability by θ. Initially with no prior information θ is assumed uniform in $[0,1]$, which is equivalent to Beta$(1,1)$ [3]. As notifications, that follow a Bernoulli distribution with parameter θ, are provided from DPM, a and b are updated as follows: $a = a + u$, $b = b + 1 - u$, where $u = 1$ if the observation consists in a dropping and 0 otherwise. Remember that a packet dropping in our monitoring solution is a lack of a required two-hop ACK. After as many observations as the decision could be made, θ can be approximated by the mathematical expectation $E(Beta(a,b))$, j will be judged. This point is denoted by the decision point, and the number of observations is expressed by $a + b$. Upon reaching this point, j will be accused as misbehaving as soon as: $E(Beta(a,b)) > E_{max}$, such that E_{max} is a fixed threshold. Note that: $E(Beta(a,b)) = a/(a+b)$.

As soon as a node is judged as misbehaving, the DPD informs the isolator. The latter tries to isolate the misbehaving node in the whole network as we will see.

3.3.2. Control packet detector (CPD)

We separate this submodule from the previous one since the Bayesian approach cannot be used for control packets. This is because there are too few of such kind of packets compared with data packets, and dropping control packets is more critical and should not be tolerated. For instance, dropping a RREQ or a RREP completely excludes a selfish node from routes, thus no data packet monitoring will be possible. Also, dropping a RERR allows a malicious

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

to launch a DoS attack by preventing the destruction of broken routes. Therefore, we should be more severe in the judgment regarding this kind of packets.

As depicted in Fig. 1 CPD receives notifications from both DPM and BPM, as it deals with both directed (RREP and RERR) and broadcast (RREQ) packets. When the CPD observes that some node drops more than a given *severe* threshold number of packets it judges it as misbehaving and informs the isolator. Optimally setting the value of this severe threshold is a hard problem, we leave it to our future work. Algorithm 3 describes the detector module.

Algorithm 3
Algorithm executed by a node i, describing the Detector

```
When receive a notification u from the monitor regarding node j
(u=1 if dropping and 0 otherwise):
if (the notification is about a data packet) {code of DPD}
  a_j = a_j + u
  b_j = b_j + 1 − u
  θ_j = a/(a + b)
  if (Decision point reached) then
    if (θ_j > E_max)
      Put j in the suspicious set
      Inform the isolator to launch WREQ against j
    end if
  end if
else {the notification is about a control packet, i.e. code of CPD}
  nbrcontroldrp_j = nbrcontroldrp_j + u
  if (nbrcontroldrp_j > threshold) then
    Put j in the suspicious set
    Inform the isolator to launch WREQ against j
  end if
end if
```

3.4. Isolator

When the isolator of some node A receives a notification about a misbehavior judgment of another node B from the detector module (DPD or CPD) it first informs the witness module (which puts B in the suspicious set), then tries to get proofs allowing it to isolate B in the whole network. Note that A should not isolate B by itself, because when it does so and punishes B unilaterally the others might consider it as misbehaving when observing it to not forward packets for B. However, it may avoid routing its own packets through this node in all cases.[1]

In social life, a person that accuses another must show proofs. One possible way to prove the accusation is to get witnesses against the accused person. Identically, we suggest a witness-based protocol (both for data and control packets) to isolate a detected node. Upon a detection the isolator informs nodes in its neighborhood about the dropper (node B), and asks for witnesses among them by locally broadcasting a WREQ (Witness REQuest) packet. The isolator of each node receiving the WREQ interrogates the witness module, if the latter testifies against B then the isolator immediately sends a *signed* WREP (Witness REPly) packet to node A's isolator (we will illustrate when the witness provides a testimony against the accused in the next subsection). Otherwise it informs the investigator, which tries to investigate the issue and to provide the isolator with an indirect testimony as we will see later. Note that the isolator informs the investigator about the nature of the last packet causing the accusation. This information is carried in the WREQ, and first deduced by the WREQ sender (A's isolator) according to the submodule from which it received the misbehavior notification, i.e. it is a data packet if the notification was provided by DPD and control packet if the notification was provided by CPD. If the investigator provides accordingly an indirect testimony, then the isolator sends a signed WREP to A's isolator. When the isolator of node A collects k validations from its neighbors with at least one provided by direct experience (without investigation), it broadcasts in the network an accusation packet (AC) containing signatures of all validating nodes. The requirement of at least one direct witness will be argued later. Each node receiving such a valid accusation isolates the guilty. If node A's isolator fails to collect k validations then it cannot isolate B, but note that its witness keeps B in the suspicious set.

3.5. Witness

This module is responsible of providing testimonies against suspicious nodes. As we mentioned, when the witness receives a notification about a misbehaving node from the local isolator it puts its ID in the suspicious set. When interrogated by the isolator (after the latter receives a WREQ from another isolator as illustrated in the previous subsection), the witness testifies against the accused in the following two cases: (i) if its suspicious set includes the accused and (ii) if the accused's misbehaving expectation (E) is close to E_{max} and/or the number of control packets detected dropped by the accused is close to the configured maximum threshold. Such information (the value of E and the number of control packets dropped) can be obtained from the detector module.

3.6. Investigator

The investigator is used when the witness does not provide a direct testimony, due to lack of enough experience with the accused node. Upon demand from the isolator, the investigator investigates an accusation launched by node A against node B according to the nature of the packet for which the investigation is launched:

3.6.1. Directed packets

If the node (holding the investigator module) is a neighbor of B then it asks the successor of the latter whether it has received packets forwarded from it, by sending an ACREQ (ACcusation REQuest) packet using a route around B. But first and in order to avoid false accusations, the investigator should ensure that A has really sent a packet to B to be forwarded to the claimed successor. One possible way to do this is to check whether the BPM has recently overheard such a packet, as well as an ACK sent from B to A *just after the packet*. This ensures that B has really received the packet and that A is not impressing the investigator, i.e. attempting to isolate B, which is innocent, by

[1] Regardless of whether proofs are gotten or not.

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

claiming that the latter is dropping packets (as it will be detailed later). Note that unlike the watchdog, the information provided from the promiscuous mode are not used for the monitoring but only for indirect testimonies, aiming at improving efficiency on detections. If the investigator of B's successor finds that its monitor has not recently received any packet *forwarded* from B, then it sends a *signed* ACREP (ACcusation REPly) packet to the investigator of the ACREQ sender, which provides its isolator with an indirect testimony and allows it sending A a WREP.

3.6.2. Broadcast packets (RREQ)

In this case, the node (if it is B's neighbor) simply checks whether its BPM has recently received either RREQ *forwarded* from B, or a RREP originated from it. If none of such packets has been received, then it testifies for the accusation and provides its isolator with an indirect testimony allowing it to send A a WREP. But it must first ensure that A has really recently sent out a RREQ (using its BPM). Algorithm 4 illustrates our isolation approach (isolator, witness, and investigator modules).

Algorithm 4
Algorithm executed by a node i, describing the isolation approach (Isolator, Witness, Investigator)

When receive a WREQ sent by X against j:
if (The WREQ is about a directed packet) **then** {The Isolator and its witness}
  if ($j \in$ *the suspicious set* or $\theta_j \approx E_{\max}$ or $nbrcontroldrp_j \approx threshold$) **then**
    send a direct signed WREP to X
  else
    if (j is a neighbor of i) **then** {Investigator}
      if (a packet from X to j was overheard as well as the ACK) **then**
        send ACREQ toward j's successor using a route that does not include j
      end if
    end if
  end if
else {The WREQ is about a broadcast packet}
  if (j is a neighbor and neither RREQ nor RREP has been received from j) **then** {Investigator}
    send a direct signed WREP to X
  end if
end if
When receive a ACREQ sent by Y against j where X is the previous hop: {Investigator}
if (no packet has been recently forwarded from j including X as the previous hop) **then**
  send Y a ACREP
end if
When receive a ACREP regarding X accusation: {Investigator}
  send X a signed undirect WREP {Isolator}
When receive a WREP sent by X against j: {Isolator}
if (WREP.type = direct) **then**
  $nbrdirectwit[j] = nbrdirectwit[j] + 1$
else
  $nbrundirectwit[j] = nbrundirectwit[j] + 1$
end if
$WitnessesSet[j] = WitnessesSet[j] \bigcup (X, Singature_X)$
if $(nbrdirectwit[j] + nbrundirectwit[j] = k$ and $nbrdirectwit[j] > 0)$ **then**
  Constract AC using WitnessesSet[j]
  broadcast AC to isolate j
end if

4. Security and performance analysis

4.1. Monitoring

The watchdog's problems presented in the first section are mitigated with our monitor, since B's forwarding validation at A is not only related to B's transmission but to C's reception.[2] If B drops the packet, A will detect that after a timeout as B cannot falsify a two-hop ACK whose authentication is ensured through asymmetric cryptography. Nonetheless, false detections are possible due to channel conditions and nodes mobility.

In terms of detection effectiveness our solution is superior to all the current solutions relying on the watchdog-based monitoring when employing the power control technique, as it will be illustrated later. Although we employ asymmetric cryptography which is more computational costly than symmetric cryptography when used to encrypt/dycript data, the computational cost of our monitoring solution is minor compared to the communication cost in both delay and power, since we apply the encryption to merely short random numbers (not to the whole packets). That is, the cost (in terms of time and power) required to encrypt/decrypt a random number within a two-hop ACK is negligible compared to the cost required for gaining access to the channel and transmitting this ACK. Hence, we focus on communication overhead in our performance analysis.

The obvious and huge problem with the ordinary two-hop ACK is its important communication overhead. To get over this shortcomings our monitoring uses the random asking strategy [1]. Hereafter, we analyze the efficiency in detection as well as the overhead of this strategy vs. the ordinary two-hop ACK [19]. In this mathematical analysis we assume that channels are reliable such that there is no packet loss. Later in the simulation, we will make more investigations into real situations of mobility and collisions causing packet loss.

Notations: In the following we give definitions and notations of some parameters we use in our analysis.

pdr: the number of packets dropped among n packets monitored.

pd: the number of packets dropped and detected by the random two-hop ACK among n packets monitored.

DF: detection factor, given by: $DF = E(pd)/E(pdr)$, where E stands for the mathematical expectation.

Na_i: the number of data packets arrived at the hop i among n packets transmitted.

Np_1: the total number of two-hop ACKs requested by the ordinary monitoring solution.

Np: the total number of two-hop ACKs requested by the random monitoring solution.

RF: the communication overhead reduction factor, given by: $RF = E(Np_1)/E(Np)$.

$P[X_i = xi | X_{i-1} = x_{i-1}, X_{i-2} = x_{i-2}, \ldots, X_0 = x_0]$: The probability that $X_i = x_i$ giving that $X_{i-1} = x_{i-1}$, $X_{i-2} = x_{i-2}$,..., and $X_0 = x_0$.

[2] We reconsider here the example of three nodes A, B, and c, whre A is monitoring B's forwarding to C.

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

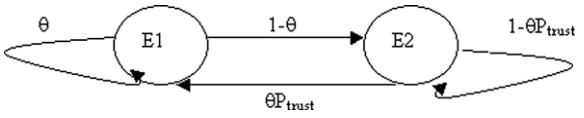

Fig. 2. Markov chain transition graph.

The model: The main difference between the random two-hop ACK and the ordinary one is that the former uses a parameter p whose value changes from a packet to another, hence is a random variable. The ordinary two-hop ACK can be considered as the random two-hop ACK with p always fixed to 1. In the following we try to model by a Markov chain the change of p at some monitor node during the monitoring of n packets, in order to compute its mathematical expectation. We assume the monitored node misbehaves for each packet with a probability θ (the behavior is independent from a packet to another, which represents the most general case). In our model we consider the state X_i as the value of p upon monitoring the packet i. X_i $i = 1, \ldots, n$ consist of n random variables. The possible values of p in the DPM's algorithm, thus of X_i in the model, are 1 and p_{trust}. We denote states representing them, respectively, by E_1 and E_2. For a given value of X_{i-1} the value of X_i depends solely on θ (the probability of misbehaving), and not on the previous values of X, i.e.:

$$(P[X_i = xi|X_{i-1} = x_{i-1}, X_{i-2} = x_{i-2}, \ldots, X_0 = x_0] = P[X_i = x_i|X_{i-1} = x_{i-1}] \quad \forall 1 \leqslant i \leqslant n, \ \forall x_i \in \{1, p_{\text{trust}}\}$$

Therefore, X_i $i = 1, \ldots, n$ form a Markov chain. The transition probabilities are:

- $P[X_i = E_1|X_{i-1} = E_1] = \theta$. If p is fixed to 1 it remains so iff the monitored packet is dropped.
- $P[X_i = E_2|X_{i-1} = E_1] = 1 - \theta$. The complement of the previous probability.
- $P[X_i = E_1|X_{i-1} = E_2] = \theta p_{\text{trust}}$. If p is fixed to p_{trust} for the packet $(i-1)$, it will be set to 1 iff that packet is dropped and detected. The probability of detection is the probability of asking,[3] which is p_{trust}, the probability of dropping is θ, and the events dropping the packet and requesting ACK for it are independent.
- $P[X_i = E_2|X_{i-1} = E_2] = 1 - \theta p_{\text{trust}}$, the complement of the previous probability.

These probabilities are independent of n, thus the chain is homogenous. Its transition matrix is T given hereafter, and its transition graph is illustrated in Fig. 2:

$$T = \begin{bmatrix} \theta & 1-\theta \\ \theta p_{\text{trust}} & 1 - \theta p_{\text{trust}} \end{bmatrix}$$

The chain is irreducible and aperiodic, as all T's elements are strictly positive $(0 < \theta < 1, \ 0 < p_{\text{trust}} < 1)$. Therefore, it admits a stationary distribution. In the stationary distribution, we note:

P_1: the probability to be in E_1. P_2: the probability to be in E_2.

[3] When the previous assumption of channels reliability is held an ACK.

The vector $\Pi = [P_1 P_2]$ can be obtained from the resolution of:

$$\begin{cases} \Pi \times T = \Pi \\ \sum_i \Pi_i = 1 \end{cases}$$

By resolving this system, we get

$$P_1 = \frac{\theta p_{\text{trust}}}{1 - \theta(1 - p_{\text{trust}})}, \quad P_2 = \frac{1-\theta}{1 - \theta(1 - p_{\text{trust}})}$$

This means that p of our algorithm will be fixed to 1 with probability P_1, and to P_{trust} with probability P_2. Its expectation is then:

$$E(p) = 1 \times P_1 + p_{trust} \times P_2 = \frac{p_{trust}}{1 - \theta(1 - p_{trust})} \tag{1}$$

Detection: As a monitored node drops a packet with probability θ and forwards it with probability $1 - \theta$, its behavior for each packet follows a Bernoulli distribution with a parameter θ. Monitoring n packets can be considered as simply the repetition of the previous operation (monitoring one packet) n times. Therefore pdr is a random variable representing the sum of n random variables following a Bernoulli distribution with parameter (mathematical expectation) θ, thus follows a Binomial distribution with expectation:

$$E(\text{pdr}) = \theta \times n \tag{2}$$

The latter represents the number of packets detected by the ordinary two-hop ACK. Now, our purpose is to assess pd, i.e. compute its mathematical expectation $E(\text{pd})$ in the stationary distribution, then DF, which reflects the efficiency of the random asking strategy in detection.

The variable (pd) also follows a Binomial distribution, since it is the result of repeating a Bernoulli operation n times with parameter $\theta \times p$, but the only difference from the continuous requesting (ordinary two-hop ACK) is that in the random strategy p is not constant. We have:

$$E(\text{pd}) = \sum_{i=1}^{n} \theta E(p) = \theta \times n \times E(p) = \theta \times n \times \frac{p_{trust}}{1 - \theta(1 - p_{trust})} \tag{3}$$

From (2) and (3) we get

$$\text{DF} = (E(\text{pd})/E(\text{pdr})) = \frac{p_{\text{trust}}}{1 - \theta(1 - p_{\text{trust}})} \tag{4}$$

Overhead: The number of two-hop ACK requested on the ith hop in the ordinary two-hop ACK monitoring is simply Na_i. Therefore, Np_1 (which also represents the number of two-hop ACK transmissions) can be expressed by

$$\text{Np}_1 = \sum_{i=0}^{h-2} \text{Na}_i, \quad \text{thus} : E(\text{Np}_1) = \sum_{i=0}^{h-2} E(\text{Na}_i) \tag{5}$$

As for the random solution, the number of two-hop ACKs requested (on each hop) can be expressed by: $\text{Na}_i \times P$, which is a random variable resulting from the multiplication of two random variables. The total number of two-hop ACKs requested in this solution is: $\text{Np} = \sum_{i=0}^{h-2} P\text{Na}_i$, hence

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

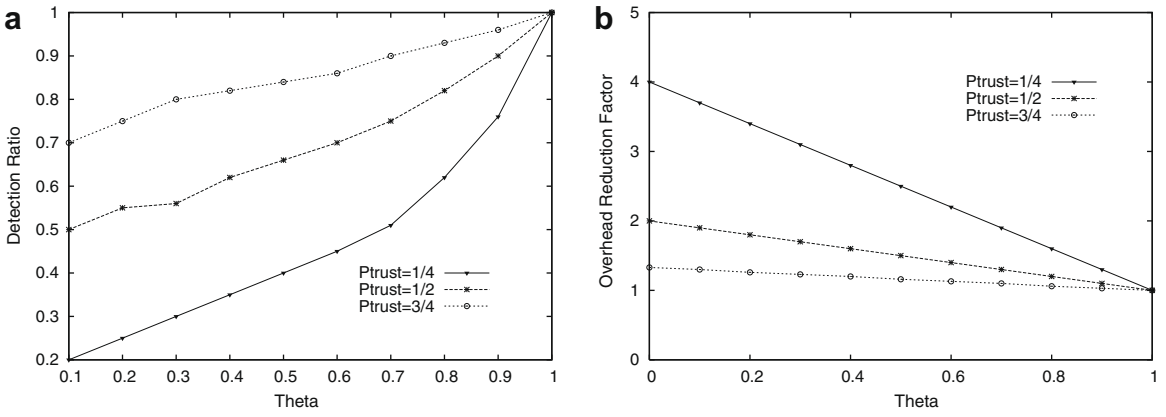

Fig. 3. Detection ratio and overhead reduction factor vs θ.

$$E(\mathrm{Np}) = \sum_{i=0}^{h-2} E(p) \times E(\mathrm{Na}_i) = E(p) \sum_{i=0}^{h-2} \times E(\mathrm{Na}_i) = E(p)E(Np_1) \quad (6)$$

From (5) and (6) we obtain

$$\mathrm{RF} = 1/E(p) = \frac{1-\theta(1-P_{\mathrm{trust}})}{P_{\mathrm{trust}}} \quad (7)$$

Fig. 3a and b, respectively, shows different values of the detection ratio (DT) and the reduction factor (RF) according to θ for some usual values of p_{trust}. We realize from the figures that $P_{\mathrm{trust}} = 0.5$ strikes a balance between efficiency (detection ratio) and cost (reduction factor). It decreases the overhead as much as half (when nodes behave well), while keeping the detection ratio good enough (always $\geqslant 0.5$). Contrary to $P_{\mathrm{trust}} = 0.25$ that has too low values of detection ratio for low and average misbehaving rates, and to $P_{\mathrm{trust}} = 0.75$ that has too low values of reduction factor. Thus, we fix $P_{\mathrm{trust}} = 0.5$ later in our simulation study.

The random two-hop ACK has been proposed for monitoring directed packets (data packets, RREP, and RRER). Nonetheless, this strategy cannot be applied to broadcast packets like RREQs, for which there is not a specific recipient. The BPM focuses on this kind of packets, and implements another simpler strategy where each node that sends a RREQ monitors its neighbors and validates one's forwarding either when it receives a RREP or a RREQ from it. Like the watchdog this strategy requires no communication overhead for monitoring, but it needs nodes to be neighborhood-aware. Neighborhood-awareness can be achieved by employing beacons either at the MAC or the routing protocol. BPM also safeguards packets the node receives, as well as those it overhears in the promiscuous mode. However, these packets are not used for the monitoring but they are only used by the investigator to provide indirect testimonies.

4.2. Misbehaving detection

Because a packet dropping might be unintentional due to nodes mobility and channel conditions, accusation should not be made upon one dropping detection. Indeed, more observations must be noted. For data packets that are taken in charge by the DPD we have proposed a Bayesian approach to make such a judgment, where each node estimates each other's misbehavior with a probability that follows a Beta(a,b) distribution whose parameters (a,b) are updated as observations are made. When enough observations with regard to a given monitored node are collected, the monitoring node will accuse the monitored one as soon as the estimated probability $E(\mathrm{Beta}(a,b))$ exceeds the configured maximum tolerance, i.e. $E(\mathrm{Beta}(a,b)) > E_{\max}$. We deduce: $E(\mathrm{Beta}(a,b)) > E_{\max} \rightarrow \frac{a}{a+b} > E_{\max} \rightarrow a > \frac{b*E_{\max}}{1-E_{\max}}$.

The latter expression $(\frac{b*E_{\max}}{1-E_{\max}})$ represents the maximum number of packets a node is tolerated to drop before its detection. This maximum tolerable threshold is proportional to b (the number of packets forwarded thus far), thus more the node forwards packets more its tolerable threshold increases. Forwarding packets after an unintentional (because of channel conditions and mobility) or intentional (selective) dropping that does not result in accusation would decrease E, which allows redemption before accusation. This redemption could not be possible when setting the tolerable threshold to a fixed number of packets. In the particular case of continuous dropping misbehavior, the value of b remains 1, its initial value, as the node does not forward any packet, and thus the maximum number a node can drop without being detected is $(\frac{E_{\max}}{1-E_{\max}})$.

$E_{\max}$ should be tuned according to the channel conditions. If the maximum successive number of packets that can be lost due to collision and mobility could be estimated by MaxLoss, then the optimal value of $E_{\max}$ to avoid false accusations is[4]: $(\frac{\mathrm{MaxLoss}}{1+\mathrm{MaxLoss}})$.

Despite the possibility of selective (partial) dropping without detection, which is rational as we consider packet unintentional dropping, the strategy of dropping up to the tolerable threshold is not efficient and sure for a misbehaving node, since it cannot know whether and how much the monitor would notice false observations because of channel conditions or mobility. Further, and owing to the random decision of the monitor it cannot know which packet requires a two-hope ACK even when reaching the

[4] By replacing a with *MaxLoss* and b with 1 in $E(\mathrm{Beta}(a,b))$.

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

trust value p_{trust}. This thwarts selective dropping and makes it risky.

The Bayesian approach was first used in [3]. In this solution every node periodically broadcasts in its neighborhood its view on nodes' behavior, which is used by recipients (as second hand information) to update their own view. To decide about the acceptance of a provided information, each node performs complicated tests on the trustworthiness of the provider. The problem with this proactive solution is its overhead, which is always important regardless of nodes' behavior. Our approach is rather reactive, where no such information are exchanged. Instead, each node performs monitoring separately and informs the others as soon as a misbehaving is approved.

As for control packets, misbehaving on their forwarding is more crucial. In addition, the number of control packets is generally minor compared with data packets. All these render the previous Bayesian approach ineffective with this kind of packets. Therefore, we have proposed another submodule for these packets (BPD), which simply uses a fixed number of packets threshold. The strategy of dropping up to the tolerable threshold is inefficient for the same reason cited before (of data packets). Providing a rigorous definition to this threshold is a big problem by itself that we leave to our future work.

4.3. Isolation

To mitigate false detection and false accusation vulnerabilities we have proposed a witness-based protocol, executed at each node by three modules: the isolator, the witness, and the investigator. Remember that upon the detection of a misbehaving the detector node launches locally in its neighborhood a call for witnesses by broadcasting a WREQ, costing only one transmission. Neighbors that have enough experience with the accused node may testify against it by sending the requestor a signed reply packet (WREP). Those which have not enough experience with the accused investigate this accusation and ask the accused node's successor whether it has recently received packets from that accused node. But first, they ensure (using the information collected by the BPM) that the accuser really sent the packet to the accused to forward to the claimed successor. To do this they must be neighbors of the accused, otherwise they do not testify. The following example illustrates and analyzes the investigation.

Assume three aligned nodes A, B and C, and another node D in A's range, as illustrated in Fig. 4. When A accuses B to not forward packets to C and sends WREQ, D investigates the issue. But before asking C, D ensures that A has really sent the packet and B has received it, by checking the data packets and ACKs overheard and collected by its BPM. If D finds that A has recently transmitted the data packet, it could not be sure that B has received it. For instance, if D is closer to A than B, A (attempting a DoS attack against B) could send the packet in a power strong enough to be overheard by D but not by B. Requiring the ACK[5] reception from B *just after* the data ensures that B has really received the data from A. This way, a node that asks the accused's successor has no doubt that the accused has received a data packet to forward towards the successor in question. Any collision at D prevents it from testifying, but does not cause any false detections.

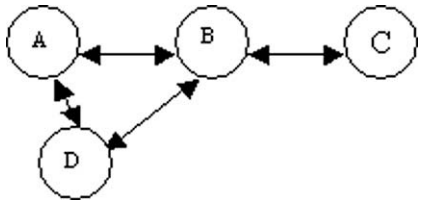

Fig. 4. Example of a nodes' connections.

Upon the reception of the ACREQ, the asked node (C) replies with a signed ACREP packet if it has not received any packet from B. Signing the ACREP prevents A from falsifying and sending it on behalf of C, as spoofing is possible in spite of the presence of an access control mechanism. A coincidental collision at C at that moment, however, would result in a false reply if A is attempting a DoS attack, thus in a false testimony. Nevertheless, the requirement of at least one direct witness (provided from a direct experience) deters wrong accusations caused by this kind of false testimonies.

The accuser has to collect k different signatures to approve its accusation. Theoretically, $k-1$ should be the maximum number of malicious or compromised nodes that can exist at any time in any vicinity. However, it is hard to determine such a number in practice, so it should be fixed to strike a balance between effectiveness and robustness. Setting k to a high value increases the robustness of the protocol against false detections and rumors, but decreases its effectiveness regarding true detections, contrary to a low value. This issue related to k will be investigated later in our simulations.

Once the accuser collects k valid signatures, it broadcasts an accusation packet including all digital signatures through the network to isolate the guilty. Again, putting digital signatures in the packet inhibits an attacker from forging it when attempting to isolate a honest node (DoS attack). This broadcast is costly, but it is not performed until a node is detected and approved as misbehaving.

In the following we summarize the capabilities of a misbehaving node, and how our solution deals with them:

- Dropping packets: First, in case of continuous dropping each node that monitors the misbehaving will detect all of the dropped packets, and thus will rapidly detect the misbehaving node as soon as it drops a given threshold and will consequently put it in its specious set, which will result soon or late in its isolation as discussed before. The second kind is the selective packet dropping. By selective when refer to both the behavior that may change from a packet to another (even of the same source) and the behavior that depends on the

[5] Note that the source of this ACK should be authenticated at the MAC level, to prevent A from forging it.

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

source, i.e. the node decides whether to forward the packet or not according to its source. The first case was discussed earlier, and we showed that a node can drop up to a given number of packets (when it gains trust and its P is set to P_{trust}) without being detected, and that this behavior is not sure for a misbehaving node as it cannot know neither the number of packets that are not validated because of channel conditions, nor which ones require ACKs. Our solution also deals with the second case. Assume some node B drops packets originated from A. When doing so an intermediate node (the predecessor of B in the source route), say C, notices that, puts B in its suspicious set, and launches a call for witnesses. Even if C's neighbors have not bad experience with B, and the latter does not drop packets except the ones of A, they can allows provide indirect testimonies via investigation and asking the successor of B if it received the appropriate packets, and then the isolation is always possible. Further, even if C fails to collect k testimonies, du to connectivity or any other problem, it will keep it in its suspicious set. This will increase the probability to isolate it later, as C will testify against B as soon as some other node accuses it. This is possible since it is likely that later some other node (other than C), say D, will monitor (will be the predecessor of) B for other session in which A is the source, i.e. A uses some link D–B in some route to some destination. Hence, D will accuse B and C is a potential node that will provide direct testimony. All this makes a misbehaving (malicious or selfish) unlikely to circumvent when applying some dropping strategy, and motivates it to always cooperate and behave correctly. Finally, note that the strategy of dropping packets then falsifying two-hop ACKs is prevented in our solution, since these packets are authenticated through the random number encryption as illustrated.

- Trying to rejoin the network after being eliminated: the obvious technique to rejoin the network after being eliminated is to get another ID. This is impossible when the access control mechanisms works well. Another possibility is to move away from the area where it is detected and rejoin the network at another neighborhood. This represents a vulnerability to many solutions that perform the isolation locally, i.e. either separately at each node or in the neighborhood. However, in our approach after getting enough evidence the isolation is performed in the whole network, which deters this possibility.
- Trying to eliminate (isolate) an innocent node (DoS attack on the solution): to do this a malicious needs to launch a WREQ against the victim and compromise k different nodes such that they provide testimonies, which is really challenging for it. Compromising k different nodes is mandatory for the attacker because each testimony is authenticated by a digital signature and thus no node can forge a testimony on behalf of another. Fixing k properly is a crucial parameter that should mitigate this DoS attack while keeping the effectiveness in detections as we will see later.

4.4. Solution limitations

Due to the high complexity of the problem we are dealing with in this work, neither our solution nor any other can eliminate it in all situations. Collusive dropping is one of the complex misbehavior with which all the current solutions (including ours) fail. To explain this behavior consider the scenario of three successive nodes A, B, and C in this order, when B and C collude and B conceals the dropping of C. As a result no current monitoring solution enables A to detect this dropping, since they all rely on hop-by-hop distributed monitoring. It is very difficult to detect such a misbehavior owing to lack of central points. In case of continues dropping, it has been illustrated that our monitoring solution detects all the dropped packets, possibly with some false positives. Components of the judgment and isolation stage deal with the latter problem (false positives) and considerably reduce it. However, in case of selective misbehavior where the behavior of the node with packets of the same origin changes over time, i.e. sometimes it forwards them and sometimes it drops them, the detection may be delayed because of the Bayesian approach for judgment that gives redemption to nodes as long as they are observed to forward packets. This is the cost of tolerating packet loss, caused by mobility and collisions. It is possible to reduce this delay by decreasing the parameter related to redemption (E_{max}), but the consequence will be high false positives. So the setting of the parameter E_{max} is a tradeoff issue, and no value can fit all scenarios and criteria. A rational adversary that aims at dropping as much packet as possible without being detected also would use selective misbehavior, but it is unfeasible for him to define a reliable strategy since it is not possible to know whether and how much the monitor would notice false observations. Thus it is likely to detect this behavior, but inevitably with some delay. Another shortcomings of our solution is that it does not consider reintegration of the isolated nodes. Although the witness-based protocol prevents innocent isolation in many situations, isolation without possible reintegration can be drastic in some cases. For instance, when a node temporary and suddenly fails, then it will be observed to not forward traffic and may be isolated. Redemption is justifiable and mandatory in this case to make the solution fault tolerant. Still, it is very difficult to define robust redemption and reinsertion mechanisms while keeping the effectiveness of the solution and preventing nodes from abusing these mechanisms. This is an open research trend that belongs to the perspectives of this work.

5. Simulation assessment

In this section we present our simulation study we made using GloMoSim [21] assessing the performance of our proposals.

We have simulated a network of 50 nodes located in an 1500×1000 m^2 area, where they move following the random way-point model with an average speed of 1 m/s. We define the misbehaving node in this simulation as the node that drops all data packets it receives to forward during all the simulation, and the misbehaving rate as the rate of

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

nodes behaving in that way (out of the 50 nodes of the network). We focus in this study on data packet dropping, and let investigations into control packets to our future work. For traffic generation we used 3 CBR sessions between three pairs of remote nodes, generating a traffic of 512 byte/s. Sources and destinations were selected such that to involve multi-hop routes during all the simulation. Note that each CBR session involves different routes, since nodes move and cause repetitive failure of the routes in use. On each hop, every data packet is transmitted using a controlled power according to the distance separating the communicating nodes. The misbehaving nodes were selected randomly, but when passing from a rate to the following one the same nodes are kept with additional random selected new ones. This enabled us to keep all the chances of getting misbehaving nodes in routes to use when passing from a rate to the next one, with possible additional chances (of the new selected ones). We measured five metrics: the true detection rate, the true isolation rate, the false detection rate, the false isolation rate, and the communication overhead, vs the misbehaving nodes rate. Each point of the curves presented hereafter is obtained after averaging five statically significant measurements, with 0.95 as the confidence interval.

In this study we compare our monitoring technique with the promiscuous monitoring of the watchdog, and we investigate the impact of the parameter k on the witness-based isolation technique. In the first comparison scenarios (scenarios where we compar our protocol's monitor with the watchdog), the tolerance threshold has been fixed empirically to 100 packets (for both protocols). Note that the Bayesian approach has not been used in this scenarios since the watchdog does not use it. However, this Bayesian approach was used when investigating the witness-based isolation. Table 1 summarizes the fixed parameters of our simulation setup.

5.1. True detection rate

The true detection rate (TDR) represents the efficiency on packet droppers *detection*. It is the average rate of true detections computed using the following formula:

$$\mathrm{TDR} = \sum_{i=0, m_i \neq 0}^{n} \frac{\mathrm{td}_i / m_i}{k} \tag{8}$$

where td_i is the true detections of node i, i.e. the number of misbehaving nodes monitored and detected by node i, m_i is the number of misbehaving nodes monitored by node i. n, the number of nodes, k, the number of nodes that have monitored misbehaving nodes (whose $m_i \neq 0$).

Table 1
Important simulation parameters.

Parameter	Value
Number of nodes	50
Terrain	$1500 * 1000\ \mathrm{m}^2$
Average speed	1 m/s
Misbehaving	Continuous
Traffic	3CBR (512 byte/s)

Plots of Fig. 5a allow us to compare two versions of our protocol, two-hop ACK and random two-hop ACK, as well as the watchdog (WD), with respect to TDR. Note that P_{trust} has been fixed to 0.5 in the random version. As mentioned, in all protocols a node is considered as misbehaving as soon as it is detected to drop more than 100 packets by a DPD of some node. We remark that the two-hop ACK has usually[6] the best TDR, and the watchdog the worst one. As for the random two-hop ACK, it has values between the two protocols, but very close to the ordinary two-hop ACK.

5.2. True isolation rate

This metric is identical to the previous one except that it deals with isolated nodes instead of detected ones, viz nodes detected as misbehaving by detector modules and further isolated by isolator modules. Unlike the previous metric, the results depicted in Fig. 5b have been obtained using the Bayesian approach in judgment instead of a fixed number of packets, with $E_{\max} = 0.5$. In the monitor module, the random approach with $P_{\mathrm{trust}} = 0.5$ has been used. In this figure two versions of our protocol are depicted. The first one with one witness ($k = 1$), and the second with two witnesses ($k = 2$). We can see that the first version has always exactly 100% of true isolation, but the second protocol's isolation declines a little bit, especially when misbehaving rate exceeds 10%. However, it remains beyond 70%. The high rates observed in this figure (regardless of the difference between the two versions) compared to the previous one illustrate the advantage of the Bayesian approach over the fixed threshold.

5.3. False detection rate

This metric (FDR) is the average rate of false detections, given by the following formula:

$$\mathrm{FDR} = \sum_{i=0, m'_i \neq 0}^{n} \frac{\mathrm{fd}_i / m'_i}{k'} \tag{9}$$

where fd_i is the false detections of node i, viz the number of well-behaving nodes monitored and wrongly detected by node i, m'_i is the number of well-behaving nodes monitored by node i, and k' is the number of nodes that have monitored well-behaving nodes (whose $m'_i \neq 0$).

Fig. 6a shows how our protocol enormously decreases FDR compared with the watchdog, and that the two versions of our protocol have almost the same performance regarding this metric. The false isolations of our protocol is mainly caused by collisions and nodes mobility, whereas the big difference between our protocol and *WD* is due to false detections engendered by the power control technique use.

5.4. False isolation rate

This metric is identical to the previous one except that it deals with isolations and not only detections. In Fig. 6b we remark huge difference between one-witness and two-witness

[6] Apart from 10% misbehaving rate.

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

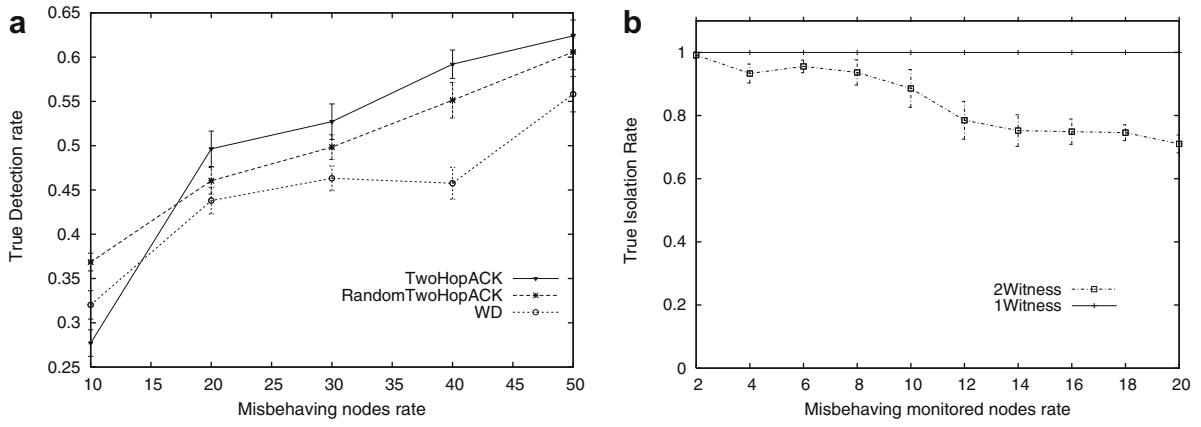

Fig. 5. True Detections and Isolations vs Misbehaving Nodes Rate.

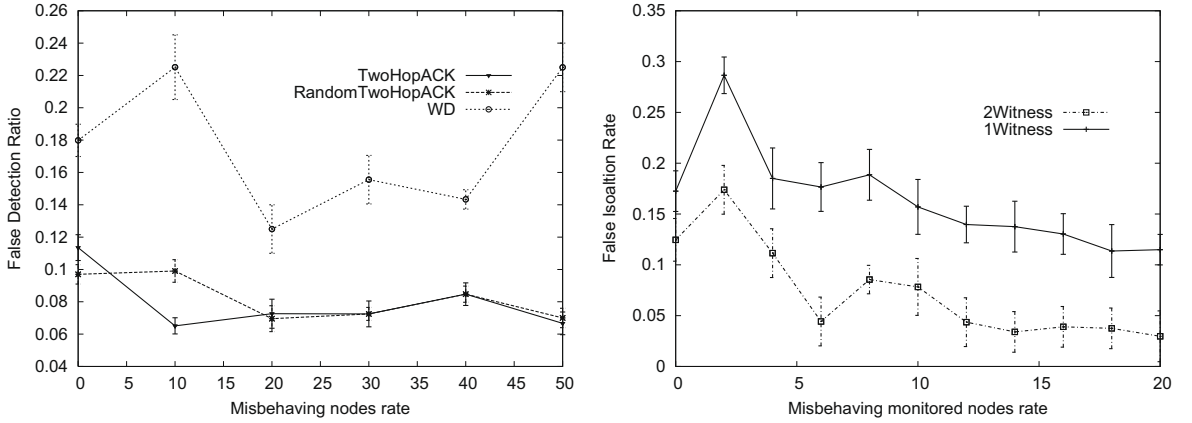

Fig. 6. False detections and isolations vs. misbehaving nodes rate.

versions, and we can see how the last version reduces false isolations. The results illustrated in this figure show how the rising of the number of required witnesses overcomes false positives.

5.5. Communication overhead

It is simply the number of two-hop ACK exchanged between monitor modules. The watchdog is excluded here since it requires no communication overhead for monitoring. As illustrated in Fig. 7, the random two-hop ACK decreases dramatically the overhead, especially when the misbehaving rate is low (most nodes behave well). This improvement is basically due to the technique of ACK asking randomization that gives more trust to well-behaving nodes. Note that the other control packets of our protocol we did not measure are of minor effect, as they are not launched until a misbehaving is detected contrary to the two-hop ACKs that are continually exchanged.

Overall, we realize from our simulation results that the random two-hop ACK is almost as efficient as the ordinary two-hop ACK in high true positives and low false positives,

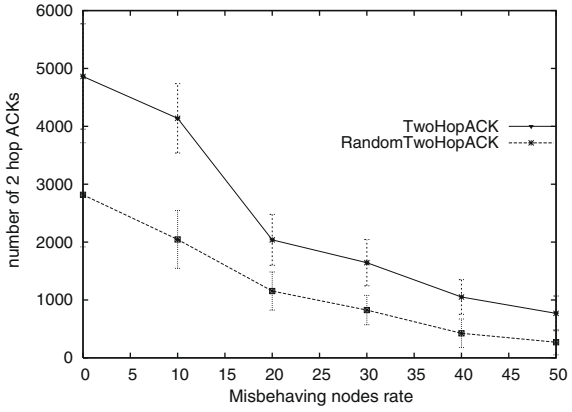

Fig. 7. Number of two-hop ACK vs. misbehaving rate.

both clearly outperform the watchdog, and that the random two-hop ACK educes the cost (overhead) of our solution. We also realize that fixing the required number of witnesses to two reduces considerably the false positives (false isolation rate), while keeping the true positives (true

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

isolation rate) satisfactory. But we point out that this decreases a little bit the true positives, particularly for high misbehaving rates.

6. Conclusion and perspectives

In this manuscript we have suggested a modular solution dealing with the packet dropping misbehavior in mobile ad hoc networks, which monitors, detects, and isolates misbehaving nodes that do not forward packets. The solution is composed of five modules: the monitor, the detector, the isolator, the witness, and the investigator. For the monitoring, we proposed the efficient technique of two-hop ACK, and we reduced its cost by suggesting the random requesting approach. Analysis and simulation results show that the random two-hop ACK is all but as efficient as the ordinary one in high true and low false detections, while hugely reducing the overhead. For local judgment, we have proposed the detector module that uses a Bayesian approach allowing redemption before making decisions, reducing the false accusations that are due to channel conditions and node mobility. However, we have been less tolerant with control packets whose dropping is crucial. Compared with the Bayesian-based solution of [3], our detection and isolation approach does not use any periodic packets exchange, thus requires no overhead as long as nodes behave well. Once a node is judged locally as misbehaving by the detector of some other node, the latter must approve its detection to ensure the misbehaving isolation by all the nodes. For this we proposed a witness-based protocol to be executed cooperatively by the isolator, the witness, and the investigator modules, each time a detector module informs the local isolator about a misbehaving detection. The protocol requires the detector to collect at least k witnesses before isolating the detected node. Fixing k is a trade-off problem, since high values mitigates rumors aiming DoS attacks as well as false detections (especially for control packets with which we have been more severe) but reduces the effectiveness on detections, contrary to low values. In our simulation, the protocol with two witnesses showed considerable improvement regarding false accusations while keeping the true detection good enough. This parameter can be risen to ensure more robustness, but should depend on the connectivity to keep efficiency.

This work is a step towards countering the packet dropping misbehavior, and brings many improvements over the solutions currently proposed in the literature. Nonetheless, the problem is too complex and far from being completely resolved, owing to the varying and unpredictable conditions and features of the ad hoc network environment. Therefore, we did not treat some important issues; such as collusive misbehavior where two nodes collude and conceal the dropping of each other, and node reinsertion after that is justifiable in case of temporary node failure leading to wrong isolation of benign node. As a perspective we plan to make more investigations with misbehaving on control packets. Dealing with the untreated issues represents challenging perspectives, viz. proposing a solution to collusive misbehaving and defining a robust mechanism to allow the reintegration of isolated nodes. This mechanism should keep the solution efficiency in preventing misbehavior, which is really challenging.

Acknowledgement

This work is supported by the Algerian Ministry of Higher Education and Scientific Research.

References

[1] D. Djenouri, N. Ouali, A. Mahmoudi, N. Badache, Random feedbacks for selfish nodes detection in mobile ad hoc networks, in: The 5th IEEE International Workshop on IP Operations and Management, IPOM'05, LNCS, vol. 3751, Springer Verlag GmbH, Barcelona, Spain, 2005, pp. 68–75.
[2] S. Marti, T. Giuli, K. Lai, M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, in: ACM Mobile Computing and Networking, MOBICOM, Boston, MA, USA, 2000, pp. 255–265.
[3] S. Buchegger, J.-Y. Le-Boudec, A robust reputation system for p2p and mobile ad-hoc networks, in: Second Workshop on the Economics of Peer-to-Peer Systems, Harvard University, MA, USA, 2004.
[4] P. Michiardi, R. Molva, Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks, in: Communication and Multimedia Security Conference, Portoroz, Slovenia, 2002.
[5] B. David, A. David, Dynamic source routing in ad hoc wireless networks, Mobile Computing, t. imielinskiand and h. korth edition, vol. 353, Kluwer Academic, 1996, pp. 153–181.
[6] S. Doshi, T. Brown, Minimum energy routing schemes for a wireless ad hoc network, in: The 21st IEEE Annual Joint Conference on Computer Communications and Networking(INFOCOM'02), New York, USA, 2002.
[7] D. Djenouri, N. Badache, New power-aware routing for mobile ad hoc networks, The International Journal of Ad Hoc and Ubiquitous Computing 1 (3) (2006) 126–136.
[8] D. Djenouri, N. Badache, Manet: selfish behavior on packet forwarding, in: Encyclopedia of Wireless and Mobile Communications, CRC Press,Taylor & Francis Group, 2008, pp. 576–587.
[9] H. Yang, X. Meng, S. Lu, Self-organized network layer security in mobile ad hoc networks, in: ACM MOBICOM Wireless Security Workshop (WiSe'02), Georgia, Atlanta, USA, 2002.
[10] L. Buttyán, J.-P. Hubaux, Stimulating cooperation in self-organizing mobile ad hoc networks, Mob. Network Appl. 8 (5) (2003) 579–592.
[11] S. Zhong, J. Chen, Y.R. Yang, Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks, in: The 22st IEEE Annual Joint Conference on Computer Communications and Networking (INFOCOM'03), San Francisco, CA, USA, 2003.
[12] V. Srinivasan, P. Nuggehalli, C.F.Chiasserini, R.R.Rao, Cooperation in wireless ad hoc networks, in: The 22st IEEE Annual Joint Conference on Computer Communications and Networking (INFOCOM'03), San Francisco, CA, USA, 2003.
[13] W. Wang, X.-Y. Li, Low-cost routing in selfish and rational wireless ad hoc networks, IEEE Trans. Mobile Comput. 5 (5) (2006) 596–607.
[14] P. Papadimitratos, Z.J. Haas, Secure data transmission in mobile ad hoc networks, in: ACM MOBICOM Wireless Security Workshop (WiSe'03), San Diego, California, USA, 2003.
[15] M. Conti, E. Gregori, G. Maselli, Improving the performability of data transfer in mobile ad hoc networks, in: the Second IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'05), Santa Clara, CA, USA, 2005.
[16] F. Kargl, A. Klenk, M. Weber, S. Schlott, Advanced detection of selfish or malicious nodes in ad hoc networks, in: 1st European Workshop on Security in Ad-Hoc and Sensor Networks, ESAS'04, Heidelberg, Germany, 2004.
[17] H. Huang, S.F. Wu, An approach to certificate path discovery in mobile ad hoc networks, in: SASN'03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, ACM, 2003, pp. 41–52.
[18] S. Zhu, S. Xu, S. Setia, S. Jajodia, LHAP: a lightweight network access control protocol for ad hoc networks, Ad Hoc Networks 4 (5) (2006) 567–585.
[19] D. Djenouri, N. Badache, A novel approach for selfish nodes detection in manets: proposal and petri nets based modeling, in: 8th IEEE

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003

International Conference on Telecommunications (ConTel'05), Zagreb, Croatia, 2005, pp. 569–574.
[20] A. Davison, Bayesian Models, Springer, 2000 (Chapter 11).
[21] X. Zeng, R. Bagrodia, M. Gerla, Glomosim: A library for the parallel simulation of large-scale wireless networks, in: The 12th Workshop on Parallel and Distributed Simulation, PADS'98, Banff, Alberta, Canada, 1998, pp. 154–161.

Djamel Djenouri obtained the engineer degree in computer science, the master degree in computer science, and finally the PhD in computer science from the University of Science and Technology USTHB (Algiers) respectively in 2001, 2003 and 2007. During his PhD, he visited the technical university of Compienge, France, and John Moors university of Liverpool, UK, as well. He also participated in many international conferences and published some journal papers. Djamel Djenouri works mainly on ad hoc networking,especially on the following topics: security, power management, routing protocols, MAC protocols, and vehicular networks. He is a permanent research assistant at the CERIST center of research in Algiers, and is currently a research fellow at the Norwegian university of Science and Technology (NTNU), in Trondheim Norway, granted by the European Research Consortium on Informatics and Mathematics (ERCIM).

Nadjib Badache was with CISTTT (Centre d'Information Scientifique et Technique et de Transfert Technologique) now CERIST (Centre de Recherche sur l'Information Scientifique et Technique) at Algiers, from 1978 to 1983, working on applied research projects in information retrieval, operating systems and library management. In 1983, he joined USTHB University of Algiers, as assistant professor and then professor, where he taught operating systems design, distributed systems and networking with research mainly in distributed algorithms and mobile systems. From 2000 to 2008, he was head of LSI (Laboratoire des SystFmes Informatiques of USTHB University) where he conducted many projects on routing protocols, energy efficiency and security in Mobile Ad hoc Networks and Wireless Sensor Networks. Since March 2008 he is director of CERIST and professor at USTHB University.

Please cite this article in press as: D. Djenouri, N. Badache, On eliminating packet droppers in MANET: A modular solution, Ad Hoc Netw. (2009), doi:10.1016/j.adhoc.2008.11.003