Cross-Layer Approach to Detect Data Packet Droppers in Mobile Ad-Hoc Networks

Djamel Djenouri[1] and Nadjib Badache[2]

[1] Basic Software Laboratory, CERIST Center of research, Algiers, Algeria
ddjenouri@mail.cerist.dz
[2] LSI, USTHB University, Algiers, Algeria
badache@cerist.dz

Abstract. Mobile ad hoc networks (MANETs) are dynamic infrastructureless networks whose routing protocols are fully based on node cooperation, and where each mobile node relies on other intermediate mobile nodes to send packets towards remote ones. Being anxious about its battery shortage, a node participating in the network and using the forwarding service provided by other nodes might behave ***selfishly*** and drop packets originated from others. Such a behavior hugely threatens the QoS (Quality of Service), and particulary the packet forwarding service availability. Another motivation to drop data packets is to launch a DoS (Denial of Service) attack. To do so, a node participates in the routing protocol and includes itself in routes then simply drops data packet it receives to forward. We propose in this paper a novel ***cross-layer*** based approach to detect data packet droppers, that we optimize and decrease its overhead. Contrary to all the current detective solutions, ours is applicable regardless of the power control technique employment.

Keywords: mobile ad hoc networks, security, packet forwarding.

1 Introduction

The absence of any central infrastructure in MANET imposes new challenges, since services ensured by the central infrastructure must be ensured by mobile devices themselves in this new environment. Particularly, to ensure packets transmission between remote nodes each mobile device (node) acts as a router and devotes its resources to forward packets for others. This consumes its resources, such as its limited battery. In some MANET applications, like in battlefields or rescue operations, all nodes have a common goal and are *cooperative by nature.* However, in many civilian applications such as networks of cars and provision of communication facilities in remote areas, nodes do not belong to a single authority and do not pursue a common goal. Forwarding packets for other nodes is generally not in the direct interest of anyone in such networks, hence there is no good reason to trust nodes and assume that they always cooperate. Indeed, nodes try to preserve their batteries and might misbehave and tend to be *selfish.* A selfish node regarding the packets forwarding process is the one which asks others to forward its own data packets but drops their packets when asked to relay them. Another motivation to drop data packets is rather aggressive; one node could launch a ***DoS attack*** by participating in the routing protocol to include itself in routes and simply dropping packets it receives for forwarding. Whatever the motivation of the dropper is, this dropping threatens the QoS and the service availability in the network. In the rest of this paper we call both the previous malicious and selfish behavior misbehavior on packet forwarding.

Some solutions have been recently proposed to detect such a misbehavior, but almost all these solutions rely on the watchdog technique [1] which suffers from some problems, especially when using the power control technique employed by some new power-aware routing protocols following the watchdog's proposal.

In this paper we deal with this issue and propose a new cross-layer based approach, that consists of a protocol formed by two components. The first component is located in the network layer and is in charge of monitoring nodes' successors forwarding, whereas the second one is located in the MAC layer and is responsible for appending two-hop ACKs (acknowledgments) to the standard MAC ACKs, and for forwarding them. Note that these two-hop ACKs are special ACKs used by our network component. By exploiting the standard MAC ACKs this cross-layer architecture reduces the overhead compared with a standard one layer approach, and gives more robustness. We also propose an optimization for more reducing the overhead. To assess our solution's performance we conduct a simulation study using GloMoSim [2], where we compare our protocol vs. the watchdog.

The remainder of this paper is organized as follows: The next section shows the causes and the effects of the packet dropping misbehavior, and provides an overview the solutions proposed in literature thus far. Section 3 is devoted to the presentation and the analysis of our new approach, followed by the simulation study in the next section. Finally, section 5 concludes the paper and summarizes the future work.

2 Related Work

2.1 Misbehaving Causes and Effects

Recent studies show that most of one node's energy in MANET is likely to be devoted for packets relaying. For instance, the simulation study in [3] shows that when the average number of hops from a source to a destination is around 5, then almost 80% of a node's transmission energy will be devoted to forward packets for others. This motivates nodes to behave selfishly, and makes them unwilling to relay packets not of direct interest to them. As mentioned before, another motivation for dropping data packets is to launch a DoS attack targeting either the source or the destination of packets. The full reliance on nodes' cooperation makes ad hoc networks hugely vulnerable to this attack.

The packet dropping misbehavior may lead to serious problems when performed by many nodes in the network, such as throughput degradation, latency rise, and network partition that threats the service availability which is one of the security requirements. All these problems affect both well-behaving and misbehaving nodes. Marti et al. [1] have shown by simulation that if 10% to

40% among the network's nodes misbehave on data forwarding, then the average throughput degrades by 16% to 32%. Another study performed by Buttyan and Hubaux [4] has been devoted to investigate the impact of the network size by simulating networks of different sizes (from 100 nodes to 400 nodes) with the same density, and comparing the effect of the same rates of misbehaving nodes on the throughput. The results show that large networks are more vulnerable to this kind of misbehavior.

2.2 Current Solutions

The emergent problem of packet dropping misbehavior in MANET has recently received attention amongst researchers, and some solutions have been proposed. These solutions could be classified into two main categories [5]: reactive solutions that aim at detecting the misbehavior when it appears in the network, and preventive solutions which try to inhibit the misbehavior either by motivating nodes to cooperate or by taking measures to prevent packets from being dropped. As far as we know, Marti et al. are the first who dealt with this problem. In [1], they have defined the *watchdog* technique that has been used by almost all the reactive solutions subsequently proposed. This technique is based on the employment of the promiscuous mode. When a node A, which may be either a source or an intermediate node, transmits a packet to B to forward to C, A monitors B's forwarding by promiscuously listening to the carrier and analyzing packets it overhears. A validates B's forwarding if and only if it overhears the packet transmission within a given timeout, otherwise it increments a counter regarding B's dropping. A will accuse B as misbehaving as soon as the counter exceeds a given threshold. In this approach, it is supposed that each transmission can be overheard by all the transmitter's neighbors if no collision takes place, and that the configured threshold is high enough to overcome possible false detections due to channel conditions. The watchdog requires no overhead, and is a relevant solution for detecting misbehaving nodes when the previous assumptions are held.

Nevertheless, subsequent works in the field of the power consumption optimization [6] [7] have proposed to use the power control technique when routing packets. That is, the transmitter does not use a fixed full-power when transmitting data packets to a given receiver, but an adaptable one according to the distance separating the two nodes. Using the watchdog with this power-efficient technique might cause false detections, as illustrated in the following example: Assume B is a well-behaving node that uses controlled powers and the required power from B to C is less than that needed to reach A from B, thereby the packets sent from B to C will not be overheard at A. Consequently, node A will not be able to validate any B's forwarding and may accuse wrongly B as misbehaving. In addition to its failure when employing the power control technique, the watchdog cannot detect the misbehavior in some other cases such as [1]:

1. Partial dropping: node B can circumvent the watchdog by dropping packets at a lower rate than the watchdog's configured minimum misbehavior threshold. Remember that A accuses B as a misbehavior when A remarks that the number of packets dropped by B exceeds a given threshold

2. Receiver collision: after a collision at node C, B could skip retransmitting the packet without being detected by A
3. False misbehavior report: a node may falsely report other innocent nodes in its neighborhood as misbehaving to avoid getting packets to forward through them
4. Insufficient transmission power: B can control its transmission power to circumvent the watchdog. If A is closer to B than C, then B could attempt to save its energy by adjusting its transmission power such that the power is strong enough to be overheard by the monitoring node (A), but less than the required transmission power for reaching the true recipient (C). Note that performing this way is power-efficient for B.
5. Cooperated misbehavior: B and C could collude to cause mischief. In this case, when B forwards a packet to C it does not report to A when C drops the packet. C does the same thing when used as a predecessor of B in some route. This kind of misbehavior is very hard to detect, and is out of the scope of our proposal.

Marti et al. also propose the path rather, which helps to route packets around misbehaving nodes detected by the watchdog. However, no punishment has been defined, and no mechanism allowing nodes to exchange their experience regarding the misbehaving nodes knowledge has been fixed by the authors. More recent proposals have dealt with these issues.

In [8], Yang et al. describe a unified network layer solution to protect both routing and data forwarding in the context of AODV. Michiardi and Molva [9] suggest a generic reputation-based mechanism, namely CORE, that can easily be integrated with any network function. CONFIDANT is another interesting reputation-based solution, proposed by Buchegger and Le-Boudec [10]. Still, all these detective solutions rely on the watchdog technique in their monitor component.

Buttyan and Hubaux [3] propose, model, and analyze an efficient preventive economic-based approach stimulating nodes to cooperate. The authors introduce what they call *virtual currency* or *nuglets*, along with mechanisms for charging service usage. The main idea of this technique is that nodes which use a service must pay for it (in nuglets) to the provider nodes. This concepts is used and generalized to *credit* by Zhong et al. in SPRITE [11], where they propose an improved solution compared with Nuglets. However, SPRITE introduces a centralized point that is not realistic in the ad hoc context. Other stimulating preventive approaches are based on the game theory, such as [12]. These preventive solutions motivate nodes to cooperate, but do not aim at detecting the misbehaving nodes contrary to the previous ones.

In [13], Papadimitratos and Haas present the SMTP protocol. It is a hybrid solution that mitigates the misbehavior effects (packets lost) by dispersing packets, and detects the misbehavior by employing end-to-end feedbacks. This kind of feedbacks allows the detection of routes containing misbehaving nodes, but fails to detect these nodes. Conti et al. [14] [15] propose another interesting end-to-end feedbacks based solution, that uses 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 misbehaving nodes but fails to detect the appropriate nodes. To find the appropriate node on a link after an iterative probing authors propose the so called *unambiguous probing*, which is based on the watchdog thus suffers from its problems. In this paper we propose a detective solution to mitigate some watchdog's problems, notably failures related to the problems 2 and 4 presented before, and false detections when employing the power control technique.

3 Novel Approach

3.1 Cross-Layer Two-Hop ACK

Our proposal aims at mitigating the watchdog's problems, especially those related to the power control technique use, with reasonable overhead. To reduce the cost (the overhead) of our solution we use a cross-layer based approach and exploit the MAC layer feedbacks. Our monitoring protocol is composed of two parts; the first one is located at the network layer, whereas the second one is located at the MAC layer which is generally more tamper resistant.

Like in the watchdog, each node monitors the next forwarding of each packet it transmits, and a source routing protocol is assumed to be used. To explain our monitoring process we first deal with one forwarding, and we suppose in the following (like in the previous section) that node A sends packets to B and monitors the forwarding to C. This process can be easily extended to all the hops along the route. We use a new kind of feedbacks we call *two-hop ACK*, an ACK that travels an intermediate node (two wireless links) [17]. Node C acknowledges packets it receives by sending A via B a two-hop ACK. To reduce the overhead, the two-hop ACK transmission from C to B is intergraded within the ordinary MAC ACK.

To prevent B from falsifying two-hop ACKs we suggest to use the following asymmetric cryptography based strategy: Node A generates a random number and encrypts it with C's PK (Public Key), then appends it to the packet's header. When C receives the packet, it gets the number back, decrypts it using its SK (Secret Keys), encrypts it using A's PK, and finally puts it in a two-hop ACK which is sent back to A via B. Instead of sending this ACK in a separate packet, C uses the ordinary MAC ACK it sends back to B upon the reception of the data packet and piggybacks the two-hop ACK to it. Node B, however, could not delay its acknowledgment of the data packet reception from A, hence forwards this ACK to A in a separate packet. When A receives the ACK it decrypts the random number and checks if it matches with the one it has generated, in order to validate B's forwarding regarding the appropriate packet. If B does not relay the packet, A will not receive a valid[1] two-hop ACK and will be able to detect this dropping after a timeout. This detection results in the increasing of a counter on the packets dropped at B, and like in the watchdog, A accuses B as misbehaving when this counter exceeds a configured threshold.

[1] We assume that the encryption algorithm and SKs are robust enough.

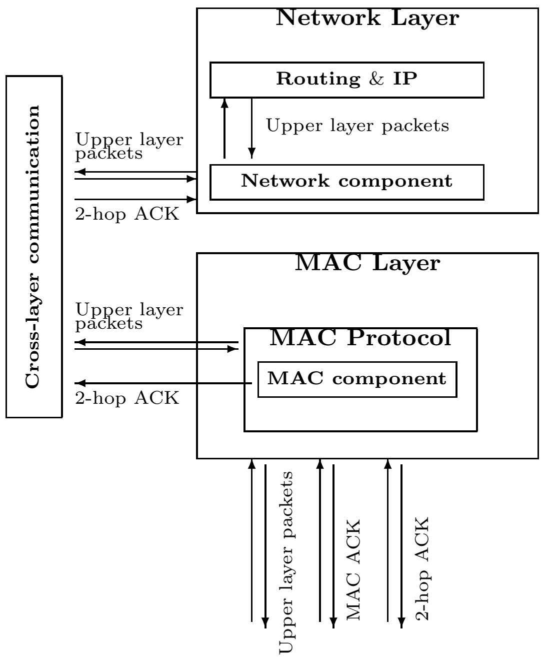

Fig. 1. Cross-layer Architecture

Our approach needs a public key distribution mechanism which is out of the scope of our purpose. However, a mechanism like the chain of trust [18] can be used. Note that the same keys could be employed for other security purposes at other layers, and are not inevitably peculiar to our solution.

Figure 1 illustrates our solution's architecture, and the type of packets exchanged between the different components and protocols. Note that the MAC component is only in charge of initiating two-hop ACK, forwarding them, and passing them up to the network module, while the other monitoring functions (requesting two-hop ACKs, validating the forwarding, counting the number o packets dropped,..etc) are the responsibility of the upper component. In spite of the cross-layer design that reduces the overhead, the major drawback of this first solution is still the overhead it engenders as a two-hop ACK is required for each data packet on each couple of hops regardless the nodes' behavior, which is costly. In the following we propose an optimization to more reduce this overhead.

3.2 Random Two-Hop ACK

To decrease this cost we suggest to *randomize* the ACK ask, i.e. node A does not ask C an ACK for each packet but upon sending a packet to forward it *randomly* decides whether it asks an ACK or not with a probability p, 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 an ACK it selects an even number, and an odd number when it decides not to ask the ACK. This *random* selection strategy prevents the monitored node from deducing which packets contain ACK requests. Note that getting such information allows a misbehaving to drop packets with no requests without being detected. When the node decides not to ask an ACK it directly validates the forwarding, whereas when it decides to ask an ACK it waits for it during a timeout like in the ordinary two-hop ACK.The probability p is continuously updated as follows: It is set to 1 (the initial value representing zero-trust) when a timeout exceeds without receiving the requested ACK, and to P_{trust} when the requested ACK is received. 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.

One possible obvious further optimization to improve the accuracy in detections of the random two-hop ACK monitoring is that node C acknowledges the number of packets transmitted from the last ACK ask, instead of acknowledging merely one packet. This way A would not directly validate packets with no ACK request, but waits for the next requesting. To do this the number of packets acknowledged needs to be carried and *encrypted* in each two-hop ACK, which increases a little bit the overhead. However, we do not investigate this optimization, since the basic random two-hop ACK converges rapidly to the ordinary two-hop ACK (as it will be illustrated in the next section).

3.3 Analysis

Unlike the current detective solutions based on the promiscuous mode monitoring (the watchdog), ours relies on the random two-hop ACK. The monitoring node (A) validates the monitored node's (B) forwarding when it receives an ACK from the successor of this latter (C). This process can be generalized along the path for each consequent two hops till the destination, and efficient encryption/decryption operations have been added in order to authenticate the two-hop ACKs and secure the solution against spoofing attacks.

Getting rid of the promiscuous mode based monitoring makes our solution independent of transmission powers, and resolves the watchdog false detection problem related to the employment of the power-control technique. Moreover, our solution resolves the problem 2 of the watchdog (section 2). When a collision appears at C, B should retransmit the packet, otherwise A would not validate its forwarding. This because B's forwarding will not be validated at A until C really receives the packet and sends back the two-hop ACK, unlike the watchdog where the validation is only related to B's first transmission. Our solution also solves the problem 4 of the watchdog. Remember that when A is closer to B than C, then B could save its energy and makes the transmission power strong enough to be overheard by A but less than the one required to reach C. This problem is eliminated in our solution, since B's forwarding validation at A is not just related to B's transmission but to C's reception. Furthermore, the two-hop ACK we use allows to detect the *appropriate* misbehaving node, unlike the end-to-end ACKs [13] and the iterative probing [16].

Our solution has been designed using a cross-layer approach and has been divided into two components; one in the MAC layer and the other in the network layer. This cross-layer approach allows the integration of the two-hop ACK into the ordinary MAC ACK, which reduces the communication overhead. In the rest of this analysis we assume that there is no packet loss. Later in our simulation study we will make more investigations of more realistic scenarios with mobility and collusion. If we assume the average path length is H hops, the worst case communication complexity (when all nodes well behaves) of our first solution is: $O((H-1))$ two-hop ACK transmissions, which is identical to the end-to-end ACK employment and iterative probing [16]. If we used a standard one layer approach then two separate transmissions would be required for each hop, hence the overhead would be $O(2 \times (H-1))$. This communication complexity also represents the number of additional steps to execute the protocol (without considering the forwarding steps) for all the previous protocols, except for the iterative probing [16] in which the communication complexity is $O((H-1) + \log(H-1))$ when a misbehavior occurs; $(H-1)$ for waiting for end-to-end ACKs (used in this approach) and $\log(H-1)$ for probing, and it is $O((H-1))$ when no misbehavior takes place.

Although our first protocol is a bit faster in detection than iterative probing[2] with the same communication overhead complexity, its communication overhead remains high, since an ACK is required for each data packet on each couple of hops regardless the nodes' behavior. Our randomization of the two-hop ACK asking strategy reduces more the overhead, especially when nodes well behave as we will see later. The worst case communication complexity of this solution is $O(p_{trust} \times (H-1))$. The accurate value of the communication overhead (for both the ordinary and random two-hop ACK versions) depends on the nodes' behavior. Suppose that the monitored node misbehaves (drops the packet) with a probability (expectation) θ. Thereby, the reduction factor (RF) provided by the random approach optimization (the overhead of the ordinary two-hop ACK/ the overhead of the random version) can be approximated by:

$$RF \approx \frac{1-\theta(1-P_{trust})}{P_{trust}} \tag{1}$$

The steps for computing the overhead of both versions and thus RF are removed due to space limitation, they can be found in our technical report [19].

In the following we discuss the factor for $P_{trust} = 1/4, 1/2, 3/4$.

- when $P_{trust} = 1/4$, $RF \approx 4 - 3\theta$
- when $P_{trust} = 1/2$, $RF \approx 2 - \theta$
- and finally, when $P_{trust} = 3/4$, $RF \approx \frac{4-\theta}{3}$

Now, we discuss the efficiency (accuracy in detection) of the random two-hop ACK vs. the ordinary two-hop ACK. The behavior of the node for each packet

[2] It is faster since it involves $O(H-1)$ steps instead of $O(H-1) + \log(H-1)$.

follows a Bernoulli distribution with a parameter θ (the expectation which is the probability of dropping). Monitoring n packets could be considered as the repetition of the previous operation (monitoring one packet) n times. Therefore, the number of packets dropped (pdr) for n packets is a random variable that is the sum of n random variables which follows a Bernoulli distribution with parameter θ, thus follows a Binomial distribution with expectation: $E(pdr) = \theta \times n$.

Theoretically, the ordinary two-hop ACK detects all this number of packets (when the assumption of no packet loss is held). The purpose now is to assess the number of packets dropped and detected (pd) by the random two-hop ACK, i.e. $E(pd)$, then we will investigate the detection ratio $DR = E(pd)/E(pdr)$. This ratio shows how the random version is close to the ordinary two-hop ACK in detections. The probability of requesting an ACK *of the random two-hop ACK algorithm* is continuously updated, it differs from one operation (monitoring one packet) to another according to the result of the previous operation and the previous behavior. We denote the *algorithm's* probability of requesting an ACK for a packet i (the value of p set by the algorithm for the packet i, which is a random variable) by P_i. Consequently, the real probability (in the execution) of asking an ACK for packet i would be expressed by $E(P_i)$. P_i is fixed to 1 if in the previous operation the packet was dropped and detected, that is with the probability[3] $\theta E(P_{i-1})$, otherwise it is fixed to P_{trust}, i.e. with probability $1 - \theta E(P_{i-1})$. Therefore, the mathematical expectation of P_i could be expressed by: $E(P_i) = 1 \times \theta E(P_{i-1}) + P_{trust} \times (1 - \theta E(P_{i-1}))$. Hence:
$E(P_i) = P_{trust} + \theta(1 - P_{trust})E(P_{i-1})$.
The number of packets detected by the random strategy (pd) also follows a Binomial distribution, since it is the results of repeating a Bernoulli operation n times with parameter θP_i, but the only difference from the continuous requesting is that in this latter strategy (P_i) is not constant. We have:

$$E(pd) = \sum_{i=1}^{n} \theta E(P_i) = \theta \sum_{i=1}^{n} E(P_i) \tag{2}$$

Note that $P_1 = 1$.

Lemma 1. $\forall i \geq 1$,
$$E(P_i) = \theta^{i-1}(1 - P_{trust})^i + P_{trust} \sum_{j=0}^{i-1} \theta^j (1 - P_{trust})^j$$

Using this lemma, formula 2 could be developed into:

$$E(pd) = \frac{\theta P_{trust}}{1-\theta(1-P_{trust})} n + \theta(1 - P_{trust}) \frac{1-\theta^n(1-P_{trust})^n}{1-\theta(1-P_{trust})} (1 - \frac{\theta P_{trust}}{1-\theta(1-P_{trust})}) \tag{3}$$

The steps of simplification, as well as the proof of lemma 1 are available in [19].

[3] The probability of detection is the probability of asking an ACK in the $(i-1)^{th}$ operation. The events dropping the $i^{th} packet$ and requesting ACK for the $(i-1)^{th}$ packet are independent.

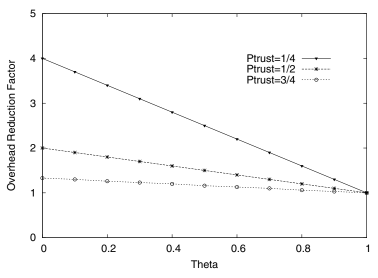

Fig. 2. Overhead Reduction Factor

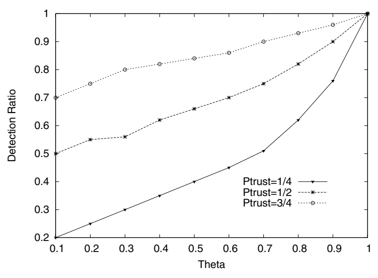

Fig. 3. Detection Ratio

This probability depends on many parameters, we will investigate it as well as the detection ratio (DR) vs. some usual values of P_{trust}.
For $P_{trust} = 1/4$, $E(pd) \approx \frac{\theta}{4-3\theta}n$, $DR \approx \frac{1}{4-3\theta}$
for $P_{trust} = 1/2$, $E(pd) \approx \frac{\theta}{2-\theta}n$, $DR \approx \frac{1}{2-\theta}$
and finally, for $P_{trust} = 3/4$: $E(pd) \approx \frac{3\theta}{4-\theta}n$, $DR \approx \frac{3}{4-\theta}$

Figures 2 and 3 illustrates respectively the approximated reduction and detection ratios according to θ. We remarque that $P_{trust} = 0.5$ strikes a balance between efficiency (detection ratio) and cost (reduction factor). It decreases the complexity overhead as much as half (when nodes well-behave), while keeping the detection ratio good enough (always ≥ 0.5). Contrary to $P_{trust} = 0.25$ that has too low values of detection ratio for low and average misbehaving, and to $P_{trust} = 0.75$ that has too low values of reduction factor. Thus, we fix $P_{trust} = 0.5$ later in our simulation study.

As illustrated, authentication of the two-hop ACK packet is ensured by employing encreption/decreption operations on a random number, generated by the monitor and piggybacked to the monitored packet. These operations have minor impact, since they are applied merely on the random number and not on the whole packet holding it. Note that we avoided the use of digital signatures in order to avoid useless packet's hash computation.

4 Simulation Results

To evaluate the performance of the proposed protocol we have driven a GloMoSim based [2] simulation study, that we present hereafter. We have simulated a network of 50 nodes located in an area of $1500 \times 1000m^2$, where they move following the random way-point model [20] with an average speed of 1m/s for 900 seconds (the simulation time). To generate traffic we have used three CBR sessions between three pairs of remote nodes, each consists of continually sending a 512 bytes data packet each second. On each hop, every data packet is transmitted using a controlled power according to the distance between the transmitter and the receiver. We compare two versions of our protocol, 2HopACK and Random 2HopACK, as well as the watchdog (WD), with regard to the true positive rate, the false positive rate, and the number of two-hop ACKs (which represents the overhead). We measured these metrics vs. the misbehaving nodes rate, which represents the rate of nodes that misbehave and drop packets they are asked to relay. Each point of the plots presented hereafter has been obtained by averaging five measurements with different seeds. Note that like the watchdog we implemented our protocol with DSR for this simulation. However, it can be implemented with any source routing protocol. Also note that WD requires no kind of ACKs, so the last metric (number of two-hop ACK) concerns merely our protocol's versions. In this study we empirically fixed the tolerance threshold[4] to 100 packets, we plane to make more investigations on optimal values and strategies in our future work.

4.1 True Positive Rate

The true positive rate (TPR) represents the efficiency on packet droppers detection. It is the average rate of true detections computed as follows:
$TPR = \sum_{i=0, m_i \neq 0}^{n} \frac{td_i/m_i}{k}$ td_i: is the true detections of node i, i.e. the number of misbehaving nodes monitored by node i that are detected.
m_i: the number of misbehaving nodes monitored by node i.
n: the number of nodes.
k: number of nodes that have monitored misbehaving nodes (whose $m_i \neq 0$).

We remark in figure 4 that except for low misbehaving rate (10%) TwoHopACK has the best detection rate above 0.5, and that RandomTwoHopAck has relatively lower TPR values but very close to those of TwoHopACK. We can also see that both protocols outperform WD. The increase of the TPR rate with misbehaving rate can be argued by: In low misbehaving rates the rate of nodes monitored but not judged (for which numbers of packets monitored did not exceed the threshold) is important thus they are not detected because of the experience lack, but as the misbehaving rate increases this rate decreases and monitors get enough samples to make judgments. Defining optimally the tolerance threshold could improve the TPR, especially for low misbehaving rate. This issue will be investigated in our future work.

4.2 False Positive Rates

This metric, we denote by FPR, will show how our protocol mitigates false detections of packet dropping due to the power control use.

It is the average rate of false detections giving by the following formula.
$FPR = \sum_{i=0, m'_i \neq 0}^{n} \frac{fd_i/m'_i}{k'}$

[4] The number of packets detected dropped upon which the node is accused.

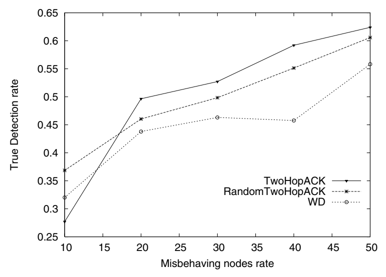

Fig. 4. True positives

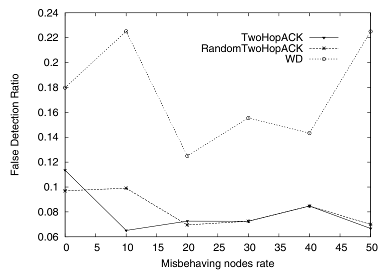

Fig. 5. False positives

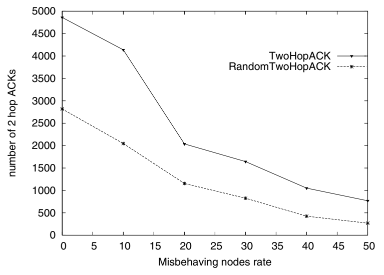

Fig. 6. Number of two-hop ACK packets

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

As illustrated in figure 5 both versions of our protocol have low FPR, contrary to WD which causes too high FPR. The FPR of our protocol is basically 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. Still, optimal definition of the tolerance threshold could minimize the FPR due to channel and mobility conditions.

4.3 Overhead

The overhead of our protocol is the number of two-hop ACK packets, depicted in figure 6. Note that WD is excluded here since it requires no communication overhead for monitoring. RandomTwoHopACK reduces considerably the overhead particularly for low and average misbehaving rates. This improvement is due to the efficient technique of randomizing the ACK requests.

5 Conclusion and Future Work

In this paper we have proposed a novel cross-layer based solution, aiming at detecting the misbehaving nodes that do not correctly cooperate for forwarding data packets. Instead of using the end-to-end ACK [13] or iterative probing [16] our solution is based on the two-hop ACK, which allows to detect the misbehaving node and not just the route containing such a node or just the appropriate link. Unlike the watchdog, our solution is applicable regardless the power control technique employment. Moreover, it allows the misbehaving nodes detection in some cases where the watchdog fails (in cases 2 and 4 presented in section 2). Our solution is composed of two components, one located at the MAC layer and the other at the network layer. This cross-layer design decreases the communication overhead as much as half compared with a standard network layer implementation of the proposed technique. In spite of this improvement, the first solution requires an ACK for every data packet on each hop, which is still costly. To reduce this cost we have suggested to randomize the two-hop ACK requesting, with probabilities continuously update in such way to give more trust to well behaving node and zero-trust to any node observed dropping a packet. The analysis and simulation results show that the random two-hop ACK is all but as efficient as the ordinary two-hop ACK in high true and low false detections, while hugely reducing the overhead, and that both versions clearly outperform the watchdog especially on false detections. Simulation results also show that there are always possibility of false detections due to collisions and nodes mobility, thus the tolerance threshold is of high importance.

As perspective we plane to provide a rigorous definition to this tolerance threshold that should be well configured to overcome dropping caused by channels and mobility conditions. We also plan to complete the proposal by defining actions to take when a node is accused as a misbehaving, and by proposing a mechanism allowing nodes to exchange their knowledge regarding nodes that misbehave. Monitoring routing control packets, especially those broadcasted with which two-hop ACK is impractical, is also in our agenda.

References

1. Marti, S., Giuli, T., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: The 6th ACM Conference on Mobile Computing and Networking, MOBICOM 2000, Boston, MA, USA (2000) 255–65
2. Zeng, X., Bagrodia, R., Gerla, M.: 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) 154–161
3. Buttyan, L., Hubaux, J.: Stimulating cooperation in self-organizing mobile ad hoc networks. ACM/Kluwer Mobile Networks and Applications **8** (2003)
4. L.Buttyan, Hubaux, J.: Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology, Lausanne, Switzerland (2001)
5. Djenouri, D., Khalladi, L., Badache, N.: Security issues in mobile ad hoc and sensor networks. IEEE Communications Surveys and Tutorials **7** (2005) 2–29

6. Doshi, S., Brown, T.: 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. Djenouri, D., Badache, N.: New power-aware routing for mobile ad hoc networks. The International Journal of Ad Hoc and Ubiquitous Computing (Inderscience) **1** (2006) 126–136
8. Yang, H., Meng, X., Lu, S.: Self-organized network layer security in mobile ad hoc networks. In: ACM MOBICOM Wireless Security Workshop (WiSe'02), Georgia, Atlanta, USA (2002)
9. Michiardi, P., Molva, R.: Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In: Communication and Multimedia Security 2002 Conference, Portoroz, Slovenia (2002)
10. Buchegger, S., Le-Boudec, J.Y.: A robust reputation system for p2p and mobile ad-hoc networks. In: Second Workshop on the Economics of Peer-to-Peer Systems, Harvard university, Cambridge, MA, USA (2004)
11. Zhong, S., Chen, J., Yang, Y.R.: 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. Srinivasan, V., Nuggehalli, P., F.Chiasserini, C., R.Rao, R.: Cooperation in wireless ad hoc networks. In: The 22st IEEE Annual Joint Conference on Computer Communications and Networking(INFOCOM'03), San Francisco, California, USA (2003)
13. Papadimitratos, P., Haas, Z.J.: Secure data transmission in mobile ad hoc networks. In: ACM MOBICOM Wireless Security Workshop (WiSe'03), San Diego, California, USA. (2003)
14. Conti, M., Gregori, E., Maselli, G.: Towards reliable forwarding for ad hoc networks. In: Personal Wireless Communications (PWC 03). Number 2775 in LNCS, Venice, Italy, Springer-Verlag GmbH (2003) 169–174
15. Conti, M., Gregori, E., Maselli, G.: 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. Kargl, F., Klenk, A., Weber, M., Schlott, S.: 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. Djenouri, D., Badache, N.: New approach for selfish nodes detection in mobile ad hoc networks. In: The first IEEE/Creat-net Workshop on Integration of Security and Quality of Service (SecQoS'05), Athens, Greece (2005)
18. Capkun, S., Buttyan, L., Hubaux, J.P.: Self-organized public-key management for mobile ad hoc networks. IEEE Transactions on Mobile Computing **2** (2003) 52–64
19. Djenouri, D., Badache, N.: Cross-layer approach to detect data packet droppers in mobile ad-hoc networks: extended version. Technical Report LSI-TR-0606, USTHB University, Algiers, Algeria (2006)
20. Maltz, J.B.D., Hu, Y.C., Jetcheva, J.: A performance comparison of multi-hop wireless ad hoc network routing protocols. In: The fourth Annual ACM/IEEE International Conference On Mobile Computing And Networking (MobiCom'98), Dallas, TX, USA (1998) 85–97