How to Avoid Packet Droppers with Proactive Routing Protocols for Ad Hoc Networks

Ignacy Gawędzki and Khaldoun Al Agha

Laboratoire de Recherche en Informatique
Bât. 490, Université Paris-Sud 11, CNRS UMR 8623
91405 Orsay Cedex France
Email: i@lri.fr and alagha@lri.fr

Abstract

Routing protocols for mobile ad hoc networks (MANETs) have been an active area of research for the last decade, with many very interesting potential as well as actual applications, in the military, rescue and commercial fields. Currently emerging standard routing protocols for MANETs do not perform well in presence of nodes that intentionally drop data traffic but otherwise behave correctly with respect to control messages. In this paper, we address the problem of coping with the presence of such nodes in networks that use a proactive routing protocol. The solution, based on the principle of flow conservation, is purely protocolar and does not rely on any specific underlying OSI layer 1 or 2 technology. In addition, it is well suited for integration into existing proactive routing protocols for ad hoc networks, since it can use existing control messages and does not require any synchronization between the nodes. The method, once implemented in an actual routing protocol, proves to increase the ratio of successfully delivered data packets significantly.

Keywords: MANET, proactive routing protocols, droppers, resilience, flow conservation

1 Introduction

A wireless mobile ad hoc network (MANET) is formed by nodes each equipped with one or mode wireless network interfaces, and located in the transmission range of a subset of the others. Every node acts both as a host and as a router to enable others to use their relaying capability to possibly communicate with nodes out of their own transmission range.

Actually emerging standard routing protocols for MANETs [1, 2, 3] are distributed algorithms that rely on the cooperation of participating nodes to operate correctly. The only supported form of failure is the simple crash, in which a node suddenly stops operating and which is essentially the same as a node leaving the network.

The assumption that all participants in the network are benevolent is simply unrealistic for many practical applications. In most of them, some bad nodes are to be expected, and some may even behave in the worst possible way. Due to the specifics of MANETs, there are many possible actions by which such malevolent nodes — called *Byzantine processes* in the jargon of distributed computing [4] — could undermine the operation of the network (jamming at the physical level, denial-of-service attacks at the MAC level, control message poisoning or data packet forging/altering/dropping at network level, etc).

The proposed solution consists of requiring nodes to perform data packet accounting on each adjacent link and exchange the values of their counters with their 2-hop neighborhood. The accumulation of these counters then enables each node to perform a series of checks aimed at verifying the principle of flow conservation in the network on the nodes in its 1-hop neighborhood. Nodes that fail at some point during that verification are to be suspected of intentional dropping. This information is then used to establish a trust metric to be used by the routing protocol in order to avoid relaying data packets through these nodes as much as possible.

After presenting previous work in Sect. 2, the solution itself is exposed in Sect. 3 along with the assumptions about the network it relies on. In Sect. 4, considerations about its practical implementation are discussed, then simulation results are presented and commented in Sect. 5. Lastly, we conclude and present perspectives for future work in Sect. 6.

2 Previous Work

The issue of Byzantine robustness of network operations has received attention for as long as the network administration of the Internet has become more decentralized and thus more prone to attacks. More recently, the problem had to be addressed again for MANETs, since the direct application of solutions for wired networks appeared to be either impossible or ineffective.

Solutions have been proposed to provide Byzantine robustness to reactive routing protocols [5, 6, 7]. In this family of protocols — of which DSR [8] and AODV [2] are notorious examples — route discovery and maintenance as well as data packet forwarding is taken care of by the routing protocol itself. This allows using very clever schemes based on cryptographic methods to ensure that every data packet has been forwarded towards the destination, without being corrupted, misrouted or forged.

In the family of proactive routing protocols, the OLSR [1] protocol has received attention from Raffo *et al.* in [9] in which the authors propose to secure the content of control messages to prevent nodes from advertising false or expired information. Fourati *et al.* also proposed a way to secure its control messages [10] by requiring neighboring nodes to validate and sign TC messages using threshold cryptography. While these propositions protect the control messages and thus the correctness of the protocol in presence of malicious nodes, it still does not prevent them from behaving badly during forwarding of data packets.

Applying methods of secure packet forwarding crafted for reactive protocols to their proactive cousins would be impossible without breaking the principle of table-driven, hop-by-hop routing, which has the advantage of being very little intrusive (if ever at all) in the network stack. Additional cryptographic tools like key distribution systems, to allow sender authentication should be used, but they still do not prevent malicious nodes from simply not forwarding data packets if they decide, for example to either save power or simply harm the network.

So the idea would be to enable nodes to detect when some other nodes are dropping traffic. In [11], Marti *et al.* propose a solution based on the ability of nodes to overhear the transmissions of their neighbors and thus check whether they have retransmitted the packets as requested. Unfortunately, such an approach does not work anymore as soon as directional antennae or multiple network interfaces are used.

Active probing of nodes have also been proposed [12, 5, 13], based on the inability of nodes to either distinguish probes from plain data packets or identify their originator.

A few completely distinct approaches aimed at encouraging cooperation instead of enforcing it have also been proposed in [14] and [15]. The use of a virtual currency system enables nodes to earn money by forwarding traffic that they may then spend to send theirs. The downside of such approaches is that peripheral nodes, which have less chance to even be able to forward traffic eventually go

bankrupt, while not having behaved badly.

In the context of static wired networks, Bradley *et al.* proposed in [16] that nodes should periodically verify that each of their neighbors satisfies the principle of flow conservation. They required that nodes maintain packet counters separately on every link to every neighbor for both incoming and outgoing data traffic. In periodic rounds, nodes advertise values of their counters and collect advertisements from all their neighbors. Then a series of conditions are verified that aim to detect whether a node has failed to disclose its counters, has advertised counters that contradict the ones of at least one of its neighbors or does not satisfy the flow conservation principle. In case the first or third condition is not verified, the node is flagged as bad, whereas in case the second one fails, both contradicting nodes are. Nodes flagged as bad are then excluded from the network. Hughes et al. later argued in [17] that this particular solution was making too strong assumptions on the network model and that it was impossible to apply as such in practice. They also exposed a few possible attacks that enable malicious nodes to make the network lose traffic in another way than simple dropping.

Nevertheless, it appears that relaxed assumptions regarding the checking of counters and another approach to use its results make the idea still applicable in practice. As shall be stated in next section, if advertisements are exchanged asynchronously but with a bound on their interval in time, there is a way of checking the second condition. Moreover, the checking makes still sense if the network topology is dynamic and nodes come and go in the neighborhood.

3 Checking Flow Conservation

3.1 Solution Outline

Each node maintains a set of counters on each link with its neighbors. In each periodic advertisement, a node diffuses a set of link advertisements (LA) composed of the differential values of counters regarding a link. In addition, it also includes any reverse link advertisements (RLA) received as LAs from the other endpoint since the previous LA.

Each node also maintains locally a degree of distrust in all the nodes that it diagnoses to have lost some traffic until that degree of distrust decreases to zero.

Upon reception of a set of LAs from some neighbor, a node performs a series of checks to verify whether the counters satisfy conditions of coherence and the principle of flow conservation. The degrees of distrust in nodes that are diagnosed to fail the various checks are increased exponentially, while being decreased linearly with passing time.

The degrees of distrust are then used as relative metrics on the distrust a node has in other nodes. In case a distributed proactive routing protocol is used, these have to be diffused in the network and combined by the nodes so as to route data packets around those nodes that happen to have the worst combined degree of distrust. While this part is obviously critical for the solution to work properly, it is still a work in progress and outside the scope of the present paper.

3.2 Network Assumptions

The network is supposed to have a dynamic topology and to be running a proactive routing protocol based on the exchange of periodic control messages. The control messages need not be strictly periodic but an finite upper bound P on the interval of time separating two successive messages has to be known.

Data packets are not supposed to be delayed too much at relaying nodes, so it is assumed the values of the counters are recorded in a way that accounts any packet scheduled for retransmission as retransmitted.

A cryptographic scheme is assumed to be deployed that allows originators of packets to be authenticated and their content to be protected against modification or forgery.

Lastly, dropping nodes are supposed not to be colluding with each other.

3.3 Network Model

Suppose the network is modeled as a dynamic graph $G = (V, E)$ with V the set of vertices (the nodes) and $E : \mathbb{R}^+ \to V \times V$ a function mapping instants in time t to set of arcs $E(t)$ (the one-way links) existing at that instant.

Let there be a pair of counters on every link, counting bytes of packets as they flow.

Definition 1. *Let $I_{ij}(t)$ (resp. $O_{ij}(t)$) be the amount of bytes that have flowed on link (i, j) up to instant t and that have not been destined to j (resp. have not been generated by i).*

We assume that for all non-existing links (i, j) during interval of time $[t, t']$, we have $I_{ij}(t) = I_{ij}(t')$ and $O_{ij}(t) = O_{ij}(t')$.

3.4 Packet Counters and Split Checking

The principle of flow conservation can be formulated as follows.

$$\forall t, \forall i, \quad \sum_j \big(I_{ji}(t) - O_{ij}(t)\big) = 0$$

If nodes were able to acquire counters $I_{ij}(t)$ and $O_{ij}(t)$ reliably for all the links adjacent to some other node, then a simple check of flow conservation would be enough to detect dropped traffic at that node. Unfortunately, this kind of information is not available in most of the practical cases. Nevertheless, packet accounting facilities are usually provided by either the TCP/IP stack (OSI Network layer) or the network interface MAC driver (OSI Data Link layer), for both incoming and outgoing traffic. Nodes can thus perform local accounting on links of which they are an endpoint and maintain what we will call their *view* of these counters, which is the amount of traffic belonging to $I_{ij}(t)$ and $O_{ij}(t)$ according to them.

Definition 2. *Let $I^i_{ij}(t)$ and $I^j_{ij}(t)$ be respectively node i's and j's view of counter $I_{ij}(t)$. Similarly, let $O^i_{ij}(t)$ and $O^j_{ij}(t)$ be respectively node i's and node j's view of counter $O_{ij}(t)$.*

Based on this notion of views, the verification of flow conservation at some node is possible on the condition that this node's view, as well as all of its neighbors' are available to the node performing the check. This is necessary since no single node is to be trusted for performing packet accounting correctly.

Before checking flow conservation as such on a node, its counters have to be checked for coherence with the counters of each other neighbor on the links adjacent to that node. Therefore, three link balances have to be calculated.

The first and second are link balances related to views of counters $I_{ij}(t)$ and $O_{ij}(t)$.

Definition 3. *Let $\mathscr{B}^I{}_{ij}(t)$ (resp. $\mathscr{B}^O{}_{ij}(t)$), the input (resp. output) link balance on link (i, j), be the difference between views of i and j of counter $I_{ij}(t)$ (resp. $O_{ij}(t)$).*

$$\forall t, \forall i, \forall j, \quad \begin{cases} \mathscr{B}^I{}_{ij}(t) = I^i_{ij}(t) - I^j_{ij}(t) \\ \mathscr{B}^O{}_{ij}(t) = O^i_{ij}(t) - O^j_{ij}(t) \end{cases} \tag{1}$$

The third is the balance related to views of counter $T_{ij}(t)$, standing for the total amount of traffic that flowed from i to j up to instant t.

Definition 4. *The total link balance on link (i, j) is the difference between $T^i_{ij}(t)$ and $T^j_{ij}(t)$, respectively i's and j's view of $T_{ij}(t)$, the total amount of bytes that flowed on link (i, j) up to instant t.*

$$\forall t, \forall i, \forall j, \quad \mathscr{B}^T{}_{ij}(t) = T^i_{ij}(t) - T^j_{ij}(t) \tag{2}$$

After that, a node's balance (i.e. amount of flow non-conservation at that node) has to be calculated according to its own view of counters.

Definition 5. *Let $\mathscr{B}_i(t)$, node i's internal balance at instant t, be the amount of flow non-conservation based on i's view of the counters.*

$$\forall t, \forall i, \quad \mathscr{B}_i(t) = \sum_j \left(I^i_{ji}(t) - O^i_{ij}(t) \right) \tag{3}$$

Non-zero balances are indicators of *counter discrepancy*. For the node balance, the discrepancy would be among that node's own counters, that directly accuses it of wrong counter manipulation. On the other hand, for link balances, the discrepancy is between the counters of the nodes at the endpoints of the link and thus does not accuse any one node in particular, so both have to be treated as possible cheaters.

3.5 Advertising Desynchronized Counters

In the rest of this paper, we will abuse of the following notation:

$$\forall X \in \{I, O, T\}, \quad \ldots$$

to express that some formula has three actual variants, with the character X standing successively for I, O and T.

Nodes use periodic control messages, which we call link advertisement sets (LA sets) to advertise their counters. For each adjacent link, the LA set contains the values of counters for that link, which we call a link advertisement (LA). Each node maintains an advertisement sequence counter that is incremented right before each control message containing an LA set is sent and which value is included in the header of the message to serve as a sequence number.

Definition 6. *Let $\mathsf{S}_i(t)$ be the value of i's sequence counter at instant t and let $\mathsf{T}_i(n)$ be the instant at which i's LA set with sequence number n is transmitted.*

From Definition 6, the following always holds:

$$\forall t, \forall i, \quad \begin{cases} \mathsf{T}_i(\mathsf{S}_i(t)) \leq t < \mathsf{T}_i(\mathsf{S}_i(t)+1) \\ \mathsf{S}_i(\mathsf{T}_i(\mathsf{S}_i(t))) = \mathsf{S}_i(t) \end{cases}.$$

For practical reasons linked to the dynamic nature of neighborhoods, as explained in Sect. 4, nodes have to advertise differential (vs. absolute) values of counters.

Definition 7. *For each maintained counter $X^i_{ij}(t)$ or $X^j_{ij}(t)$, let $\dot{X}^i_{ij}(t)$ and $\dot{X}^j_{ij}(t)$ be their differential values.*

$$\forall X \in \{I, O, T\}, \quad \forall t, \forall i, \forall j, \forall k \in \{i, j\}, \quad \dot{X}^k_{ij}(t) = X^k_{ij}(t) - X^k_{ij}(\mathsf{T}_k(\mathsf{S}_k(t)))$$

Definition 8. *Let $\mathsf{T}_i^-(n)$ be the instant right before i's nth LA set is sent. This notation is used only with differential counters to avoid ambiguity regarding their value.*

$$\forall t, \forall i, \quad \mathsf{S}_i\left(\mathsf{T}_i^-\left(\mathsf{S}_i(t)\right)\right) = \mathsf{S}_i(t) - 1$$

For nodes to be able to calculate link balances for some link (i, j), the LA for that link from both i and j must be available. So for each LA, each node must not only include its own differential counters regarding that link, but also all the other endpoint's LAs received since last advertisement was sent, which we call the reverse link advertisements (RLA).

Definition 9. *Let $\mathcal{S}_j^i(n)$ be the (possibly empty) set of sequence numbers of j's LAs included as RLAs in i's nth LA.*

$$\forall n, \forall i, \forall j, \quad \mathcal{S}_j^i(n) = \left\{m : \mathsf{T}_i(n-1) \leq \mathsf{T}_j(m) < \mathsf{T}_i(n) \wedge (j,i) \in E\left(\mathsf{T}_j(m)\right)\right\}$$

As soon as a node receives i's LA set number n, it can check i's node balance for $t = \mathsf{T}_i(n)$ using (3), but for link balances, another method has to be applied, as we shall see in the coming section.

3.6 Desynchronized Link Balances

To be able to check a link balance according to (1) and (2), LAs from both endpoints would have to be transmitted and received synchronously, which we assume is neither achievable in practice neither even desirable. Instead of checking those balances strictly, a node calculates its desynchronized value, based on the available counters in the LA.

Definition 10. *Let $\widetilde{\mathscr{B}^I}^i_{kl}(n)$ (resp. $\widetilde{\mathscr{B}^O}^i_{kl}(n)$ and $\widetilde{\mathscr{B}^T}^i_{kl}(n)$), the desynchronized $\mathscr{B}^I$ (resp. $\mathscr{B}^O$ and $\mathscr{B}^T$) link balance based on i's nth LA regarding link $(k,l) = (i,j)$ or $(k,l) = (j,i)$, be the link balance computed on the counters provided in i's LA regarding link (k,l).*

$$\forall X \in \{I,O,T\}, \quad \forall i, \forall j, \forall n, (k,l) \in \{(i,j),(j,i)\}, \quad \widetilde{\mathscr{B}^X}^i_{kl}(n) = \dot{X}^i_{kl}\left(\mathsf{T}_i^-(n)\right) - \sum_{m \in \mathcal{S}_j^i(n)} \dot{X}^j_{kl}\left(\mathsf{T}_j^-(m)\right)$$

It appears that the values of desynchronized link balances remain bounded provided there is no counter discrepancy on the link. It is easily shown with the use of the variables $\delta_j^i(n)$ and $\Delta_j^i(n)$ defined as follows.

Definition 11. *Let $\delta_j^i(n)$ be the minimum difference in time between the instant i's nth LA set is sent and last j's LA set was received by i or i's $(n-1)$th LA set was sent. Let $\Delta_j^i(n)$ be the difference in time between the instant i's nth LA set is sent and last j's LA set was received by i.*

$$\forall i, \forall j, \forall n, \quad \delta_j^i(n) = \begin{cases} \mathsf{T}_i(n) - \max_{m \in \mathcal{S}_j^i(n)} \mathsf{T}_j(m) & \text{if } \mathcal{S}_j^i(n) \neq \emptyset \ , \\ \mathsf{T}_i(n) - \mathsf{T}_i(n-1) & \text{otherwise.} \end{cases}$$

$$\Delta_j^i(n) = \begin{cases} \delta_j^i(n) & \text{if } \mathcal{S}_j^i(n) \neq \emptyset \ , \\ \delta_j^i(n) + \Delta_j^i(n-1) & \text{otherwise.} \end{cases}$$

Note that for any i, j, and n, we have that $\delta^i_j(n) > 0$ and $\Delta^i_j(n) > 0$.

Using deltas, values of desynchronized link balances can be expressed in terms of the endpoints' view of the absolute counters.

$$\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,(k,l) \in \{(i,j),(j,i)\},$$
$$\widetilde{\mathscr{B}}^{X\,i}_{kl}(n) = \begin{array}{ll} X^i_{kl}\big(\mathsf{T}_i(n)\big) & -X^i_{kl}\big(\mathsf{T}_i(n) \quad -\Delta^i_j(n)\big) \\ -X^j_{kl}\big(\mathsf{T}_i(n-1)\big) & +X^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \end{array} \tag{4}$$

Since the deltas are positive and the counters increasing, we have the following bounds.

$$\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,(k,l) \in \{(i,j),(j,i)\},$$
$$\left|\widetilde{\mathscr{B}}^{X\,i}_{kl}(n)\right| \leq \max \begin{pmatrix} X_{kl}\left(\mathsf{T}_i(n-1)\right) - X_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big), \\ X_{kl}\left(\mathsf{T}_i(n)\right) \quad -X_{kl}\big(\mathsf{T}_i(n) \quad -\Delta^i_j(n)\big) \end{pmatrix}$$

But because values of $X_{kl}(t)$ are not known, the known finite bound on the link capacity, B_{kl} has to be used instead.

$$\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,(k,l) \in \{(i,j),(j,i)\}, \quad \left|\widetilde{\mathscr{B}}^{X\,i}_{kl}(n)\right| \leq \mathsf{B}_{kl} \cdot \max\left(\Delta^i_j(n-1),\Delta^i_j(n)\right) \tag{5}$$

For nodes that did not receive both i's and j's LA sets, the values of deltas are only known to be bounded by the maximum interval of time separating two successive LA sets. This implies that the bounds on the values of desynchronized link balances are finite.

$$\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,(k,l) \in \{(i,j),(j,i)\}, \quad \left|\widetilde{\mathscr{B}}^{X\,i}_{kl}(n)\right| \leq \mathsf{B}_{kl} \cdot \mathsf{P} \tag{6}$$

To summarize, a node checks a desynchronized link balance by verifying that its absolute value does not exceed the bound, at best according to (5) and at worst according to (6).

3.7 Accumulated Desynchronized Link Balances

The method described in Sect. 3.6 is still very permissive, in that nodes detect counter discrepancies only in excess of $\mathsf{B}_{ij} \cdot \mathsf{P}$ during the interval between two successive LA sets. A dropping node may still drop quite a considerable amount of traffic without anyone noticing. Instead of using only the latest values of desynchronized link balances and forgetting earlier ones, nodes can do better by summing these values over consecutive sequence numbers and considering the sum for checking.

Theorem 1. *If $X^i_{ij}(t) = X^j_{ij}(t) = X_{ij}(t)$, the summed values of desynchronized link balances over an interval of sequence numbers $[n,n+M]$ are exactly:*

$$\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,\forall M \geq 0,(k,l) \in \{(i,j),(j,i)\},$$
$$\sum_{p=n}^{n+M} \widetilde{\mathscr{B}}^{X\,i}_{kl}(p) = \begin{array}{ll} X^i_{kl}\big(\mathsf{T}_i(n+M)\big) & -X^i_{kl}\big(\mathsf{T}_i(n+M) - \Delta^i_j(n+M)\big) \\ -X^j_{kl}\big(\mathsf{T}_i(n-1)\big) & +X^j_{kl}\big(\mathsf{T}_i(n-1) \ -\Delta^i_j(n-1)\big) \end{array}$$

Proof. By induction on $M \geq 0$, with the case of $m = 0$ already shown in (4). The induction hypothesis holds since, because there is no counter discrepancy, we have that $X^i_{kl}(t) = X^j_{kl}(t)$ for each counter $X_{kl}(t)$. ■

It immediately follows that, as for single values of desynchronized link balances of (k,l), we have the following bounds.

$$\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,\forall M \geq 0,(k,l) \in \{(i,j),(j,i)\}, \\ \left|\sum_{p=n}^{n+M} \widetilde{\mathscr{B}}^{X\,i}_{kl}(p)\right| \leq \max \begin{pmatrix} X_{kl}\left(\mathsf{T}_i(n-1)\right) - X_{kl}\left(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\right), \\ X_{kl}\left(\mathsf{T}_i(n+M)\right) - X_{kl}\left(\mathsf{T}_i(n+M) - \Delta^i_j(n+M)\right) \end{pmatrix} \quad (7)$$

And here also values of $X_{kl}(t)$ are not known and the bound has to be loosened to $\mathsf{B}_{ij} \cdot \max(\Delta^i_j(n-1), \Delta^i_j(n+M))$ or even to $\mathsf{B}_{kl} \cdot \mathsf{P}$ if deltas are not known either.

So provided that a node is able to check accumulated desynchronized link balances over a long interval of time, it will eventually detect a growing counter discrepancy, even if the growing rate is low.

3.8 Bounding the Deltas Even Tighter

Let us define first a new quantity.

Definition 12. *Let $\Gamma^i_j(n)$, be the minimum difference in time between the instant i's nth LA set is sent and first j's LA set was received by i since i's $(n-1)$th LA set was sent.*

$$\forall i,\forall j,\forall n : \mathbb{S}^i_j(n) \neq \emptyset \quad \Gamma^i_j(n) = \mathsf{T}_i(n) - \min_{m \in \mathbb{S}^i_j(n)} \mathsf{T}_j(m)$$

In each nth LA from i about links (i,j) and (j,i), the quantities $\dot{X}^i_{ij}(\mathsf{T}^-_i(n))$ and $\dot{X}^i_{ji}(\mathsf{T}^-_i(n))$ are i's claim about what amount of traffic has flowed on links (i,j) and (j,i) respectively for each of classes I, O and T, during interval $[\mathsf{T}_i(n-1), \mathsf{T}_i(n))$. If that LA also contains j's RLAs for the links, then the quantities $\dot{X}^j_{ji}(\mathsf{T}^-_i(n) - \Gamma^i_j(n))$ and $\dot{X}^j_{ij}(\mathsf{T}^-_i(n) - \Gamma^i_j(n))$ are j's corresponding claim on interval $[\mathsf{T}_i(n-1) - \Delta^i_j(n-1), \mathsf{T}_i(n) - \Gamma^i_j(n))$.

Definition 13. *Let $\mathsf{L}^i_j(n)$ be the sequence number, not greater than n, of last i's LA for links (i,j) and (j,i) containing at least one RLA. Similarly, let $\mathsf{N}^i_j(n)$ be the sequence number, greater than n, of next i's LA containing at least one RLA.*

$$\forall i,\forall j,\forall n, \quad \begin{cases} \mathsf{L}^i_j(n) = \max\left\{m : m \leq n \wedge \mathbb{S}^i_j(m) \neq \emptyset\right\} \\ \mathsf{N}^i_j(n) = \min\left\{m : m > n \wedge \mathbb{S}^i_j(m) \neq \emptyset\right\} \end{cases}$$

It appears that we have the following properties:

$$\forall i,\forall j,\forall n, \quad \begin{cases} \left[\mathsf{T}_i\left(\mathsf{L}^i_j(n) - 1\right), \mathsf{T}_i(n)\right) \supseteq \left[\mathsf{T}_i(n) - \Delta^i_j(n), \mathsf{T}_i(n)\right) \\ \left[\mathsf{T}_i\left(\mathsf{L}^i_j(n)\right) - \Delta^i_j\left(\mathsf{L}^i_j(n)\right), \mathsf{T}_i\left(\mathsf{N}^i_j(n)\right) - \Gamma^i_j\left(\mathsf{N}^i_j(n)\right)\right) \supseteq \left[\mathsf{T}_i(n) - \Delta^i_j(n), \mathsf{T}_i(n)\right) \end{cases}.$$

Instead of using $\mathsf{B}_{ij} \cdot \mathsf{P}$ as an upper bound to the traffic that could possibly have flowed during $[\mathsf{T}_i(n) - \Delta^i_j(n), \mathsf{T}_i(n))$ on links (i,j) and (j,i), nodes can use the maximum of both nodes' claims about it.

$$\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,\forall (k,l) \in \{(i,j),(j,i)\}, \\ X_{kl}\left(\mathsf{T}_i(n)\right) - X_{kl}\left(\mathsf{T}_i(n) - \Delta^i_j(n)\right) \leq \max \begin{pmatrix} \sum_{p=\mathsf{L}^i_j(n)}^{n} \left(\dot{X}^i_{kl}\left(\mathsf{T}^-_i(p)\right)\right), \\ \dot{X}^j_{kl}\left(\mathsf{T}^-_i\left(\mathsf{N}^i_j(n)\right) - \Gamma^i_j\left(\mathsf{N}^i_j(n)\right)\right) \end{pmatrix}$$

To be able to check these tighter bounds on accumulated desynchronized link balances from i's LAs between sequence numbers n and $n+M$, a node must get all of i's LAs between numbers $\mathsf{L}^i_j(n-1)$ and $\mathsf{N}^i_j(n+M)$ inclusive for each j. If that node also receives j's LAs separately, then only i's LAs between $\mathsf{L}^i_j(n-1)$ and $\mathsf{N}^i_j(n+M)-1$ are really necessary (for j provides itself its LAs contained in $\mathsf{N}^i_j(n+M)$.

Once the bounds for possible legitimate traffic are known, balance checks are performed according to (7).

3.9 Maximum Interval Checking

In the dynamic network model, the mobility dictates the ability of nodes to perform reliable checks. Let us call a node performing a check an *observer* and the node(s) under check the *observed*. If an observer is gathering i's LA sets on interval of time $[t_b, t_e]$, then it receives sequence numbers in $[n_b, n_e]$ with $n_b = \min\{n : \mathsf{T}_i(n) \geq t_b\}$ and $n_e = \max\{n : \mathsf{T}_i(n) \leq t_e\}$. For all j neighbor of i's, it can thus perform extraction of tighter bounds of flow on links (i,j) and (j,i) on advertisements in $[n'_b, n'_e]$ such that

$$n'_b = \min\left\{n' : \mathsf{L}^i_j(n'-1) \geq n_b\right\}$$
$$n'_e = \max\left\{n' : \mathsf{N}^i_j(n') \leq n_e\right\}$$

This implies that the observer must gather at least two LAs from i containing RLAs from j to be able to extract tighter bounds. Since at most P units of time separate two consecutive LA sets, it has to wait at worst $2\mathsf{P}$ units of time to receive one such LA (provided the topology does not change regarding it, i and j). Then the second such LA from i may not arrive later than in another $2\mathsf{P}$.

If an observer starts gathering LAs at times when big traffic flows on the links, then the tighter bounds for accumulated desynchronized link balances could be not so tight after all (since the flow during $[\mathsf{T}_i(n'_b) - \Delta^i_j(n'_b), \mathsf{T}_i(n'_b))$ could be large). Of course these bounds should always be saturated to $\mathsf{B}_{ij} \cdot \mathsf{P}$, but the extra processing used to extract them may not be paying off.

A better method would be to use a sliding window of past LAs with numbers in $[p, p+M]$, with a maximum M and an increasing p, and perform the bound extraction and accumulated balance checks over it. It would require the observer to keep a history of past LAs, but the size of the history can be a parameter of the accuracy of the method (the longer the history, the lower the minimum discrepancy growing rate detectable). Thus the bound is not minimized by the flow during $[\mathsf{T}_i(n) - \Delta^i_j(n), \mathsf{T}_i(n))$ with constant n.

3.10 Using Span to Increase Reactivity

One implication of the idea behind the method presented in Sect. 3.9 is that if an observer considers accumulated desynchronized link balances on a window starting and ending at times when no traffic flowed on the link, then if there is no counter discrepancy, their values should be exactly zero. This is because the endpoints agree on the total amount of traffic that has flowed in each class (corresponding to counters I_{ij}, O_{ij} and T_{ij}). The method of Sect. 3.9 comes a bit short of fully exploiting that implication, since as long as one end of the window is located on a moment of high traffic, it may give no chance to properly check a reduced tight bound.

If, instead of absolute values, actual values of accumulated link balances are considered, then it appears that in fact it does not really matter whether the window starts or ends on moments of high data traffic as these values tend to oscillate between a lower bound and an upper bound. We already know that the difference between the maximum and the minimum values over the window, what we

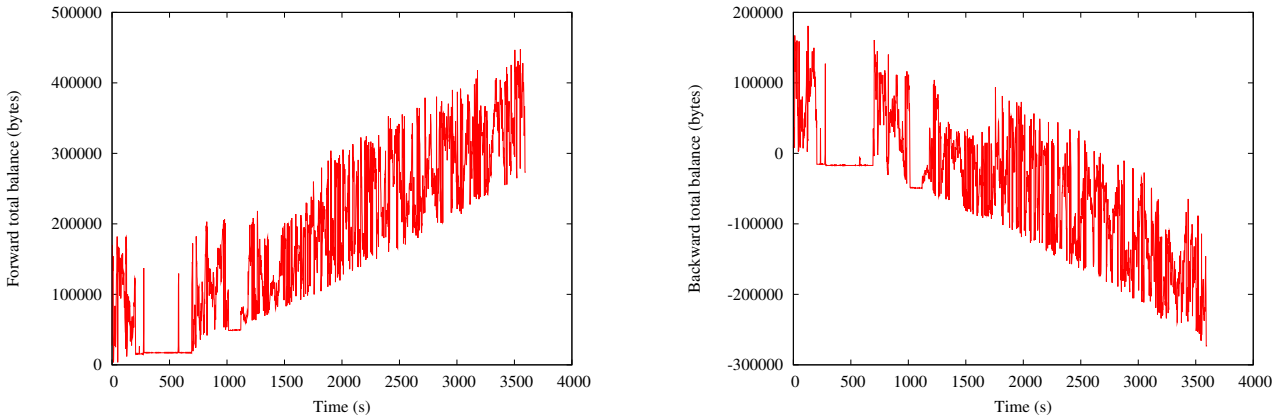

Figure 1: Forward and backward balances with packet loss. Extracted from simulations of desynchronized counter exchange on a link between two nodes with one of them dropping with a probability of 10^{-3}. The maximum throughput on the link was set to 10^5 b/s and the maximum advertisement interval to 2 s.

call the *span* is the maximum possible flow of traffic during P units of time. We also know that the minimum is attained whenever the actual traffic is zero.

For accumulated balances of the form $\sum_p \widetilde{\mathscr{B}}^{xi}_{ij}(p)$, which we call *forward* balances, loss of data packets on link (i,j) appears as increasing extrema of values over time. On the other hand, for accumulated balances of the form $\sum_p \widetilde{\mathscr{B}}^{xi}_{ji}(p)$, called *backward* balances, loss on link (j,i) appears as decreasing extrema of values over time.

Figure 1 illustrates what happens with forward and backward balances when data packet loss does occur. This variation is due to the fact that in the first case i's counters grow faster than j's thus their difference is growing positively and i's counters are not balanced with j's; and in the second case i's counters grow slower than j's, their difference is growing negatively.

What is needed here is a way to detect such "drifting" of extrema, based on the values of accumulated balances in the window. By taking a close look at the figures, we came up with the following theorem, which allows to detect such drifting pretty effectively.

Theorem 2. *If $X^i_{ij}(t) = X^j_{ij}(t) = X_{ij}(t)$, the difference between the value of an accumulated link balance and its minimum over a window is bounded by the amount of traffic that flowed at the end of that window. Moreover, the difference between the first value of an accumulated link balance in the window and its minimum over the same window is bounded by the amount of traffic that flowed at the start of the window.*

$$\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,\forall M \geq 0,\forall (k,l) \in \{(i,j),(j,i)\},$$
$$\begin{cases} 0 \leq \sum_{p=n}^{n+M} \widetilde{\mathscr{B}}^{xi}_{kl}(p) - \min_{q\in[n,n+M]} \left(\sum_{p=n}^{q} \widetilde{\mathscr{B}}^{xi}_{kl}(p) \right) \leq X_{kl}\left(\mathsf{T}_i(n+M)\right) - X_{kl}\left(\mathsf{T}_i(n+M) - \Delta^i_j(n+M)\right) \\ 0 \leq \widetilde{\mathscr{B}}^{xi}_{kl}(n) - \min_{q\in[n,n+M]} \left(\sum_{p=n}^{q} \widetilde{\mathscr{B}}^{xi}_{kl}(p) \right) \leq X_{kl}\left(\mathsf{T}_i(n)\right) - X_{kl}\left(\mathsf{T}_i(n) - \Delta^i_j(n)\right) \end{cases} \tag{8}$$

Proof. First, let us simplify things a bit.

$$\begin{aligned}
&\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,\forall M \geq 0,\forall (k,l) \in \{(i,j),(j,i)\}, \\
&\quad \min_{q\in[n,n+M]} \left(\sum_{p=n}^{q} \widetilde{\mathscr{B}}^{X}{}^{i}_{kl}(p) \right) \\
&= \min_{q\in[n,n+M]} \Big(X^i_{kl}\left(\mathsf{T}_i(q)\right) - X^i_{kl}\big(\mathsf{T}_i(q)-\Delta^i_j(q)\big) - X^j_{kl}\left(\mathsf{T}_i(n-1)\right) + X^j_{kl}\big(\mathsf{T}_i(n-1)-\Delta^i_j(n-1)\big) \Big) \\
&= \min_{q\in[n,n+M]} \big(X^i_{kl}\left(\mathsf{T}_i(q)\right) - X^i_{kl}\left(\mathsf{T}_i(q)-\Delta^i_j(q)\right) \big) - X^j_{kl}\left(\mathsf{T}_i(n-1)\right) + X^j_{kl}\left(\mathsf{T}_i(n-1)-\Delta^i_j(n-1)\right)
\end{aligned}$$

Next put that in the central parts of (8).

$$\begin{aligned}
&\forall X \in \{I,O,T\}, \quad \forall i,\forall j,\forall n,\forall M \geq 0,\forall (k,l) \in \{(i,j),(j,i)\}, \\
&\begin{cases}
\displaystyle\sum_{p=n}^{n+M} \widetilde{\mathscr{B}}^{X}{}^{i}_{kl}(p) \; - \min_{q\in[n,n+M]} \left(\sum_{p=n}^{q} \widetilde{\mathscr{B}}^{X}{}^{i}_{kl}(p) \right) = X^i_{kl}\left(\mathsf{T}_i(n+M)\right) - X^i_{kl}\left(\mathsf{T}_i(n+M)-\Delta^i_j(n+M)\right) \\
\qquad\qquad - \min_{q\in[n,n+M]} \left(X^i_{kl}\left(\mathsf{T}_i(q)\right) - X^i_{kl}\left(\mathsf{T}_i(q)-\Delta^i_j(q)\right) \right) \\
\\
\widetilde{\mathscr{B}}^{X}{}^{i}_{kl}(n) \; - \min_{q\in[n,n+M]} \left(\sum_{p=n}^{q} \widetilde{\mathscr{B}}^{X}{}^{i}_{kl}(p) \right) = X^i_{kl}\left(\mathsf{T}_i(n)\right) - X^i_{kl}\left(\mathsf{T}_i(n)-\Delta^i_j(n)\right) \\
\qquad\qquad - \min_{q\in[n,n+M]} \left(X^i_{kl}\left(\mathsf{T}_i(q)\right) - X^i_{kl}\left(\mathsf{T}_i(q)-\Delta^i_j(q)\right) \right)
\end{cases}
\end{aligned}$$

And this last step completes the proof, since $X^i_{ij}(t) = X_{ij}(t)$. ■

Checking that the span remains bounded, with the help of the method of Sect. 3.8, is much more effective, as the bound drops to zero as soon as there is no traffic on the link.

4 Practical Considerations

There are some matters regarding implementation that are worth noting.

4.1 Differential Counters

The use of differential counters instead of absolute ones is essential to cope with links appearing and disappearing, as a node which has not received advertisements about some link existing in the past will not be able to check balances properly.

4.2 Contents of LAs

A node i needs to advertise a set of six of its counters for every link with a neighbor j. Apart from $\dot{I}^i_{ij}(t)$, $\dot{I}^i_{ji}(t)$, $\dot{O}^i_{ij}(t)$ and $\dot{O}^i_{ji}$, we have seen that also $\dot{T}^i_{ij}(t)$ and $\dot{T}^i_{ji}(t)$ are needed. For the sake of efficiency, the two following counters could be used instead:

$$\begin{aligned}
\dot{S}^i_{ij}(t) &= \dot{T}^i_{ij}(t) - \dot{O}^i_{ij}(t) \\
\dot{D}^i_{ji}(t) &= \dot{T}^i_{ji}(t) - \dot{I}^i_{ji}(t) \quad ,
\end{aligned}$$

as they will have smaller values and will carry as much information.

4.3 Limited Diffusion of LAs

Forwarding counters only once, as RLAs in LAs, is enough to allow any node to fully check all of its 1-hop neighbors (in terms of link and node balances) and partially check its strict 2-hop neighbors (only in terms of link balance between it and the 1-hop neighbor).

4.4 Drop Tolerance Parameter

For both node and link balance checks, a parameter of tolerance is easily added to the formulae in order to allow nodes to drop traffic below a predefined threshold rate without being suspected of cheating.

4.5 Degree of Distrust

Each node maintains locally a degree of distrust in the nodes whose LAs it has had the opportunity to verify. Since a failed check of link balances incriminates both endpoints, although only one of them is to blame, the important rule to follow for maintaining the degree of distrust is that check failures should raise the degree of distrust more if they happen more frequently. Consequently, a good start is to increase the degree exponentially at each failure and let it decrease linearly with time passing. The idea is that even if a cheater tries to conceal its droppings by "spreading" the counter discrepancy over all of its adjacent links, it will get caught more often than each of its neighbors.

Depending on the actual underlying routing protocol and its capabilities in terms of calculating routes according to some constraints, the degrees of distrust may be part of a distributed (dis)trust system. The discussion about actual fitness of existing distributed trust systems is outside the scope of this paper.

It is worth noting that in fact any loss is to be detected, be it intentional or not. This is not an issue since the suspected droppers are not really excluded from the network, they are only avoided if possible. If a dropping node is the only possible intermediate for some destinations, then it will be used anyway. This makes some of the attacks from [17] harmless.

4.6 LAs and Unreliable Control Messages

If the underlying protocol is intended initially to withstand some amount of loss in its control messages, and if LA sets are to be put in these messages for diffusion, a retransmission mechanism has to be implemented to ensure no LA is ultimately lost in the process while its originator is continuously considered a neighbor. For example, a system of negative acknowledgements (NACKs) could be used to allow nodes to request lost LAs from some of its neighbors. The stability of such a system lies in the fact that no more than some fixed amount of LA can get lost before their originator is not considered as part of the 1-hop neighborhood anymore. Obviously this would still enable link balance checks but with some additional delay.

5 Performance Evaluation

We are currently working on applying this method with the OLSR protocol. Preliminary simulation results are already showing a significant improvement of delivery rate when our method is used.

5.1 Simulation Model

We ran a batch of simulations with the OPNET simulator, with 50 mobile nodes moving according to the ad hoc waypoint model on a surface of 1000×1000 square meters. The transmission power, decoding power threshold and signal-to-noise ratio were such as to let the network have an average diameter of between 3 and 4 hops.

Data packets were generated with a parameterized exponential interarrival, uniform random source and destination nodes, uniform size in bytes on the interval $[40, 1500]$.

Among the nodes, 20% were supposed to drop traffic with a probability of 1. Cheaters always incremented their counters as though they had forwarded the packets. The width of the window of LAs on which to perform the link balance checks was of 30. The drop tolerance was set to 0 and the degrees of distrust were increased using the following formula:

$$x \mapsto \begin{cases} x \times 1.5 & \text{if } x \geq 1/1.5 \\ 1 & \text{otherwise} \end{cases} ;$$

and decreased for an interval of time dt using the following formula:

$$x \mapsto \max(0, x - dt/10) \quad .$$

Degrees of distrust were used first to select MPRs so that for each strict 2-hop neighbor, the 1-hop neighbor that reached it with the lowest degree was selected. Then MPRs included the base 1.5 logarithm of the degree for each of their MPR selector in their TC messages. Each node collecting TC messages combined for each MPR selector the logarithmic degrees coming from each of its MPRs using the median value to avoid taking exceptionally high values into account (which might come from cheaters trying to slander their neighbors). The combined value was considered as a distrust per node in the topology which in turn was used to compute a weight on all outgoing links of a node:

$$\forall i, \forall j, \quad W_{ij} = 1 + D_i$$

These weights were used by the Dijkstra algorithm to compute routes to every destination. In this preliminary application to OLSR, the neighborhood was considered as changed as soon as some neighbor had its degree increased.

5.2 Simulation Results

We evaluated the overall delivery ratio of each simulation run, that is the ratio of the number of packets successfully arrived at their intended destination over the total number of generated data packets. We observed all the different losses combined: loss at the MAC layer, missing route, expired TTL and intentional drop. In each simulation set, the data packet interarrival parameter was set successively to 0.64, 0.32, 0.16, 0.08, 0.04, 0.02 and 0.01 seconds.

First of all, we ran a set of simulations where the cheaters were not actually cheating at all, to see what ideal goal we are pursuing. Second, we ran a set of simulations where the cheaters were dropping traffic but no node did anything to prevent it: this is plain OLSR operating. Third, we ran simulations in which the cheaters were actually known to every node, with a fixed positive degree of distrust, to have an idea how well the solution could be performing with a perfect detection method. Lastly, we ran a set of simulations where the nodes performed checking of node and link balances according to our method.

The results are presented in Figure 2 and show that with fixed drop probability, the method performs better when there is generally more traffic. This is due to the fact that with less traffic, there are

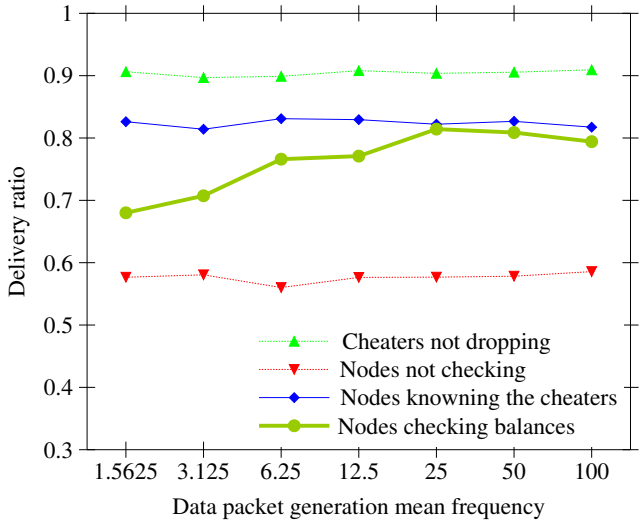

Figure 2: Our method applied to OLSR

less occasions for the link balances to fail the checks and thus cheaters to get "caught". Since there always is some amount of dropped traffic by benevolent nodes, these get caught as well and the less the cheaters get caught, the harder they are to distinguish from benevolent nodes and finally the less effective the method is.

6 Conclusion and Perspectives

The solution presented in this paper is based on theoretical derivations from the principle of flow conservation. The methods of checking balances have been inspired by observation of various simulations and have later been proved formally. The overall method proved to be detecting dropped traffic reliably, but has to be coupled with a trust system and interfaced with the routing protocol in order to be useful. The big advantage of our solution is that it provides an essential building block that is entirely protocolar, so it does not rely on any capability from lower layers.

The next step which we are currently exploring is the implementation of this solution in a particular proactive routing protocol: OLSR. The work focuses on a better distrust-aware MPR selection heuristic and on a trust system that would make the avoidance of droppers effective even with a low traffic or low drop probability.

While colluding cheaters are not supported at this time, we believe a solution similar to the one in [16] could be adapted easily, but requiring more control message, memory and processing overhead.

References

1. T. Clausen and P. Jacquet. *Optimized link state routing (OLSR) protocol*. RFC 3626, IETF, October 2003.

2. C. Perkins, E. Belding-Royer, and S. Das. *Ad hoc on-demand distance vector (AODV) routing*. RFC 3561, IETF, July 2003.

3. R. Ogier, F. Templin, and M. Lewis. *Topology dissemination based on reverse-path forwarding (TBRPF)*. RFC 3684, IETF, February 2004.

4. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

5. Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *WiSE '02: Proceedings of the 3rd ACM workshop on Wireless security*, pages 21–30, New York, NY, USA, 2002. ACM Press.

6. Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne:: a secure on-demand routing protocol for ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 12–23, New York, NY, USA, 2002. ACM Press.

7. Ioannis Avramopoulos, Hisashi Kobayashi, Randolph Wang, and Arvind Krishnamurthy. Highly secure and efficient routing. In IEEE, editor, *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 208–220. IEEE, IEEE, March 2004.

8. David B. Johnson, David A. Maltz, and Yih-Chun Hu. *The dynamic source routing protocol for mobile ad hoc networks (DSR)*. Internet-Draft 10, IETF, July 2004.

9. Daniele Raffo, Cédric Adjih, Thomas Clausen, and Paul Mühlethaler. An advanced signature system for olsr. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 10–16, New York, NY, USA, 2004. ACM Press.

10. Alia Fourati and Khaldoun Al Agha. A shared secret-based algorithm for securing the OLSR routing protocol. *Telecommunication Systems*, 31(2–3):213–226, March 2006.

11. Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 255–265, New York, NY, USA, 2000. ACM Press.

12. Mike Just, Evangelos Kranakis, and Tao Wan. Resisting malicious packet dropping in wireless ad hoc networks. In Samuel Pierre, Michel Barbeau, and Evangelos Kranakis, editors, *Ad-Hoc, Mobile, and Wireless Networks, Second International Conference, ADHOC-NOW 2003 Montreal, Canada, October 8-10, 2003, Proceedings*, volume 2865 of *Lecture Notes in Computer Science*, pages 151–163. Springer, 2003.

13. Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Sustaining cooperation in multi-hop wireless networks. *2nd Symposium on Networked System Design and Implementation, Boston, MA, USA, May*, 2005.

14. Levente Buttyán and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *MONET*, 8(5):579–592, 2003.

15. Sheng Zhong, Jiang Chen, and Yang Richard Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *INFOCOM*, 2003.

16. Kirk A. Bradley, Steven Cheung, Nick Puketza, Biswanath Mukherjee, and Ronald A. Olsson. Detecting disruptive routers: a distributed network monitoring approach. *Network, IEEE*, 12(5):50–60, Sep/Oct 1998.

17. John R. Hughes, Tuomas Aura, and Matt Bishop. Using conservation of flow as a security mechanism in network protocols. In *IEEE Symposium on Security and Privacy*, pages 132–131, 2000.

About the authors

Ignacy Gawędzki graduated from EPITA in 2003 and received his Master's Degree in 2004 from the University of Paris Sud 11. Since 2004, he is preparing a PhD thesis under advisory of Khaldoun Al Agha and on the matter of distributed algorithms applied to quality of service and security in mobile ad hoc networks.

Khaldoun Al Agha received the Engineering Degree from Supelec (Paris) in 1993. He obtained his Master's Degree (1995) and his PhD (1993) from the University of Versailles. In 2002, he received the HDR from University of Paris Sud 11. He was an Assistant Professor in University of Versailles in 1998. In 1999, he joined INRIA for one year. He is a Full Professor at University of Paris Sud 11. He created and conducts the Networking group at the LRI laboratory. He participates in different projects (BRAIN, SAMU, Arcade, SAFARI, OCARI, SARAH, etc). His research interests are on resource allocation, security and QoS for cellular and ad hoc networks.