Resilience to Dropping Nodes in Mobile Ad Hoc Networks with Link-State Routing

Ignacy Gawędzki and Khaldoun Al Agha

Laboratoire de Recherche en Informatique
Université Paris-Sud 11, CNRS
F-91405 Orsay, France
i@lri.fr alagha@lri.fr

Abstract. Currently emerging standard routing protocols for MANETs do not perform well in presence of malicious nodes that intentionally drop data traffic but otherwise behave correctly with respect to control messages of the routing protocol. In this paper, we address the problem of coping with the presence of such nodes in networks that use a link-state routing protocol. The solution, based on the verification of the principle of flow conservation, is purely protocolar and does not rely on any specific underlying OSI layer 1 or 2 technical feature. In addition, it is well suited for integration into existing link-state routing protocols for ad hoc networks, by using existing periodic control messages and by not requiring any clock synchronization between the nodes. The method, once implemented in an actual routing protocol, proves to increase the ratio of successfully delivered data packets significantly.

Key words: ad hoc networks, link-state routing, security, resilience.

1 Introduction

Routing protocols for mobile ad hoc networks (MANETs) have received much attention in the past decade, but currently the most mature solutions still rely on the full cooperation of the participating nodes. The presence of one or more malicious nodes may have negative consequences on the proper operation of the network. In the case of proactive protocols, nodes exchange control messages allowing them to maintain their routing table and data packet forwarding is performed on a hop-by-hop basis, using a traditional protocol stack (most commonly TCP/IP). This makes these protocols easy to implement as userspace processes while the operating system's kernel remains unchanged. The downside is that securing the protocol itself is not enough to make the network resilient to malicious nodes, since they can always play nice with control packets, while not as much with data packets.

The presented solution aims to reduce the impact of malicious nodes as much as possible, allowing others to route data packets around nodes that tend to lose traffic, provided that malicious nodes are not colluding.

After related work in the field is summarized in Sect. 2, the solution is detailed in Sect. 3. Performance evaluation is presented in Sect. 4, followed finally by a conclusion and some perspectives in Sect. 5.

This is a preprint of an article published in NETWORKING 2008, 7th International IFIP-TC6 Networking Conference, Singapore, May 5-9, 2008, Proceedings, LNCS vol. 4982.
http://www.springer.com/computer/communications/book/978-3-540-79548-3

2 Related 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 exposed to attacks. The problem reemerged for MANETs, since the direct application of solutions for wired networks appeared to be either impossible or ineffective.

In the family of reactive routing protocols — of which DSR [9] and AODV [13] are notorious examples — various solutions have been proposed [2,7,1] to provide Byzantine robustness. Since route discovery and maintenance as well as data packet forwarding is taken care of by the protocol, cryptographic methods can be used to ensure that every packet has been forwarded to the destination, without being corrupted, misrouted nor forged.

As for proactive routing protocols, the OLSR [5] protocol has received attention from Raffo *et al.* [14] who proposed 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 control messages [6] by requiring neighbors to validate and sign TC messages using threshold cryptography. While these propositions aim to guarantee the correctness of the protocol in presence of malicious nodes, they do not protect against data packet dropping. Protecting data packets in the same (intrusive) way as proposed for reactive protocols would break the principle of separate table-driven hop-by-hop routing, so it is not desirable in this case. Additional cryptographic tools such as key distribution systems to allow sender authentication should be used, but they still do not enable the prevention or detection of intentional packet dropping.

In order to make packet dropping detectable, Marti *et al.* proposed [12] to exploit the ability of nodes to overhear the transmissions of their neighbors to check whether they have retransmitted the packets as requested. Unfortunately, such an approach does not work anymore when directional antennae or multiple network interfaces (or simply different communication channels) are used on nodes. Other approaches, based on active probing of nodes to detect droppers, have also been proposed [10,2,11]. They rely on the inability of nodes to either distinguish probes from plain data packets or identify their originator.

It is worth noting that many solutions to encourage cooperation [4,15] have been proposed as well. Although they may be very effective to discourage selfishness, they do not protect against malicious nodes that drop data traffic, be it at their own expense.

In the context of static wired networks, Bradley *et al.* proposed [3] that nodes should periodically verify that each of their neighbors satisfies the principle of flow conservation based on their advertised packet counters. Nodes that fail to pass these tests are flagged as bad and excluded from the network. Their assumptions about the network have later been criticized as being too strong and a few possible attacks have been exposed by Hughes *et al.* [8]. However, this solution is also hardly applicable to MANETs, because of mobility and the difficulty to maintain tightly synchronized clocks between nodes.

3 Proposed Solution

3.1 Motivations and Assumptions

Our experience in implementation of MANET routing protocols taught us that the less a solution is intrusive in the existing framework the easier it is to implement, maintain and port. Besides, relying only on basic features of drivers allows wider hardware support. With a protocolar approach, we sought a solution easy to implement by requiring little or no change at all in the OS kernel while no cross-layer requirement eases support for a wide range of network interface cards.

The solution assumes that cryptographic mechanisms allow any node, be it the destination host or an intermediate router, to check a packet's integrity and authenticate its source. The network is assumed to be running a link-state routing protocol that exchanges periodic control packets. The bound P on the time separating two successive control packets is assumed to be finite and known.

3.2 Solution Outline

The solution is composed of three distinct phases, that are performed continuously during the operation of the network. First, perform unicast packet accounting, exchange counters periodically and verify the principle of flow conservation. Second, maintain a local degree of distrust in each other node, based on how often it fails the flow conservation tests and diffuse that value in the network. Third, combine local degrees of distrust from other nodes into a single, uniform metric (i.e. every other node comes up with the same values) and use it for routing table calculation.

Note that intentional and unintentional packet droppings are not distinguished, which is desirable since in the third phase, intentional as well as unintentional packet droppers are simply avoided. This should have a positive effect on the overall operation of the network.

3.3 Checking Flow Conservation

In this section, the method used to check whether nodes' counter values do not satisfy the flow conservation principle is presented.

Definition 1. *Let the network be a 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 sets of arcs $E(t)$ (the one-way links) existing at these instants.*

Definition 2. *Let $I_{ij}^i(t)$ and $I_{ij}^j(t)$ be respectively i's and j's counter of the amount of bytes that flowed on link (i, j) up to instant t in packets which destination was not j. Similarly, let $O_{ij}^i(t)$ and $O_{ij}^j(t)$ be respectively i's and j's counter of what flowed on (i, j) up to t in packets which source was not i.*

The flow conservation principle states that

$$\forall t, \forall i, \quad \sum_j \left(I_{ji}^i(t) - O_{ij}^i(t) \right) = 0 \ . \tag{1}$$

If i advertised its counters $I^i_{ji}(t)$ and $O^i_{ij}(t)$ periodically, its neighbors could check (1) for each instant t at which i sent an advertisement. But since that simple check is based on i's counters only, i could as well have advertised false values. Additional checks are thus needed to ensure that these are correct, namely:

$$\forall t, \forall i, j, \quad I^i_{ji}(t) = I^j_{ji}(t) \quad \text{and} \quad O^i_{ij}(t) = O^j_{ij}(t) \quad \text{and} \quad T^i_{ij}(t) = T^j_{ij}(t) \ , \quad (2)$$

where $T^i_{ij}(t)$ and $T^j_{ij}(t)$ are respectively i's and j's counter of the total amount of bytes that flowed on (i, j) up to t.

Because of mobility, the neighborhoods are changing and so absolute values of counters are inconvenient. Differential values are better advertised instead: a differential value is simply the difference between the current absolute value of a counter and its absolute value at the time of last advertisement.

Definition 3. *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 the sequence counter is incremented from $n-1$ to n and i's advertisement set with sequence number n is transmitted.*

In the following, X is used to represent either I, O or T, to avoid repeating three times each formula that holds for each type of counter.

Definition 4. *For each counter in the form $X^i_{ij}(t)$ or $X^j_{ij}(t)$, let $\dot{X}^i_{ij}(t)$ and $\dot{X}^j_{ij}(t)$ be their differential values.*

$$\forall t, \forall i, j, \forall k \in \{i, j\}, \quad \dot{X}^k_{ij}(t) = X^k_{ij}(t) - X^k_{ij}\left(\mathsf{T}_k\left(\mathsf{S}_k(t)\right)\right) \quad (3)$$

Definition 5. *Let $\mathsf{T}^-_i(n)$ be the instant right before i's nth advertisement set is sent. This notation is used only with differential values to avoid ambiguity.*

$$\forall t, \forall i, \quad \mathsf{S}_i\left(\mathsf{T}^-_i\left(\mathsf{S}_i(t)\right)\right) = \mathsf{S}_i(t) - 1 \quad (4)$$

Using periodic messages of the routing protocol, nodes advertise successive differential values of their counters.

Definition 6. *An advertised values of counters (AVC) tuple from i for neighbor j with sequence number n has the following form:*

$$V^i_j(n) = \begin{pmatrix} \dot{I}^i_{ij}\left(\mathsf{T}^-_i(n)\right), \dot{O}^i_{ij}\left(\mathsf{T}^-_i(n)\right), \dot{T}^i_{ij}\left(\mathsf{T}^-_i(n)\right), \\ \dot{I}^i_{ji}\left(\mathsf{T}^-_i(n)\right), \dot{O}^i_{ji}\left(\mathsf{T}^-_i(n)\right), \dot{T}^i_{ji}\left(\mathsf{T}^-_i(n)\right) \end{pmatrix} . \quad (5)$$

Between two successive transmissions of an AVC tuple for some neighbor j, a node i may receive zero or more AVC tuples regarding itself from j.

Definition 7. *Let $\mathbb{S}^i_j(n)$ be the (possibly empty) set of sequence numbers of j's AVC tuples received by i between its $(n-1)$th and nth advertisements.*

$$\forall i, j, \forall n, \quad \mathbb{S}^i_j(n) = \left\{m : \mathsf{T}_i(n-1) \leq \mathsf{T}_j(m) < \mathsf{T}_i(n) \wedge (j, i) \in E\left(\mathsf{T}_j(m)\right)\right\} \quad (6)$$

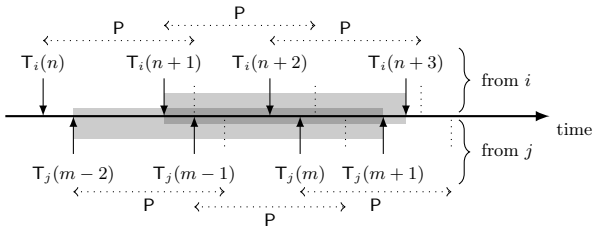

Fig. 1. Desynchronized advertisements: differential counter values from i and j are not advertised at the same time and thus do not represent flow on the same interval. Vertical arrows indicate instants at which advertisements from i (above) and j (below) are sent. Upper bound on the advertisement interval P is shown as well.

To allow neighbors of i to verify (2), i has to retransmit, along with each AVC tuple about a neighbor j, a set of reverse AVC tuples. In all, values of counters are diffused to the whole 2-hop neighborhood.

Definition 8. *Node i's nth advertisement set contains nth link advertisements about each of its neighbor j.*

$$A^i(n) = \bigcup_j \left\{ \left(n, V_j^i(n), \bigcup_{m \in \mathcal{S}_j^i(n)} \left\{ \left(m, V_i^j(m) \right) \right\} \right) \right\} \tag{7}$$

Given that advertisements are sent asynchronously, (1) can be performed directly, whereas (2) most probably cannot. The question is then how to decide that those equations are not satisfied, based on desynchronized counters, i.e. differential counter values over different intervals of time (see Fig. 1). The idea is to consider what can be called *desynchronized link balances*.

Definition 9. *A desynchronized link balance in terms of counter X for the link (k,l) (either (i,j) or (j,i)), computed from advertisement number n from node i about node j has the following form:*

$$\forall i,j,\forall n,\forall (k,l), \quad \widetilde{\mathcal{B}}^{X\,i}_{kl}(n) = \dot{X}^i_{kl}\left(\mathsf{T}_i^-(n)\right) - \sum_{m \in \mathcal{S}_j^i(n)} \dot{X}^j_{kl}\left(\mathsf{T}_j^-(m)\right) \quad . \tag{8}$$

When a node k receives the nth advertisement set from i, it can compute $\widetilde{\mathcal{B}}^{X\,i}_{ij}(n)$, since $\dot{X}^i_{ij}(\mathsf{T}_i^-(n))$ and every $\dot{X}^j_{ij}(\mathsf{T}_j^-(m))$ with $m \in \mathcal{S}_j^i(n)$ are contained in $A^i(n)$. Obviously, $\widetilde{\mathcal{B}}^{X\,i}_{ij}(n)$ and $\widetilde{\mathcal{B}}^{X\,i}_{ji}(n)$ are seldom zero, unless the counters in $A^i(n)$ are zero themselves. Instead of looking at desynchronized link balances for individual advertisements, let us consider the sum of several desynchronized link balances for successive advertisements. It appears that these accumulated link balances have interesting properties.

Definition 10. *Let $\Delta_j^i(n)$ be the difference in time between the instant $A^i(n)$ is sent and last j's advertisement set was received by i.*

$$\forall i,j,\forall n,\quad \Delta_j^i(n) \;=\; \begin{cases} \mathsf{T}_i(n) - \max\limits_{m\in\mathcal{S}_j^i(n)} \mathsf{T}_j(m) & \text{if } \mathcal{S}_j^i(n)\neq\emptyset\ , \\ \mathsf{T}_i(n) - \mathsf{T}_i(n-1) + \Delta_j^i(n-1) & \text{otherwise.} \end{cases} \tag{9}$$

Suppose that k receives M successive advertisement sets from i, calculates the desynchronized link balances and sums them. That sum is expressed quite simply in terms of absolute counters. The intuitive argument can be seen on Fig. 1: the summed desynchronized link balance (the gray rectangles) from i's advertisements $n+2$ to $n+3$ (i's $(n+2)$th advertisement contains j's $(m-1)$th counters and i's $(n+3)$th advertisement contains j's mth and $(m+1)$th counters), is clearly equal to the difference between the amount of bytes that flowed on interval $[\mathsf{T}_j(m+1), \mathsf{T}_i(n+3))$ and on interval $[\mathsf{T}_j(m-2), \mathsf{T}_i(n+1))$.

Definition 11. *Let ${\mathcal{F}^x}_{ij}^i(n)$ (resp. ${\mathcal{F}^x}_{ji}^i(n)$) be i's* end flow *on link (i,j) (resp. (j,i)), i.e. the quantity expressed by:*

$$\forall i,j,\forall n,\forall (k,l),\quad {\mathcal{F}^x}_{kl}^i(n) = X_{kl}^i\left(\mathsf{T}_i(n)\right) - X_{kl}^i\left(\mathsf{T}_i(n) - \Delta_j^i(n)\right)\ . \tag{10}$$

Theorem 1. *If $X_{kl}^i(t) = X_{kl}^j(t)$ (for any i, j, $(k,l)=(i,j)$ or $(k,l)=(j,i)$ and any t), the summed values of desynchronized link balances over an interval of sequence numbers $[n, n+M]$ are exactly:*

$$\forall i,j,\forall n,\forall M\geq 0,\forall (k,l),\quad \sum_{p=n}^{n+M} {\widetilde{\mathcal{B}}^x}_{kl}^i(p) = {\mathcal{F}^x}_{kl}^i(n+M) - {\mathcal{F}^x}_{kl}^i(n-1)\ . \tag{11}$$

Proof. By induction on $M\geq 0$. The case of $M=0$ is trivially shown:

$$\begin{aligned} \forall i,j,\forall n,\forall (k,l),\ {\widetilde{\mathcal{B}}^x}_{kl}^i(n) = {} & X_{kl}^i\big(\mathsf{T}_i(n)\big) \quad - X_{kl}^i\big(\mathsf{T}_i(n) \quad -\Delta_j^i(n)\big) \\ & - X_{kl}^j\big(\mathsf{T}_i(n-1)\big) + X_{kl}^j\big(\mathsf{T}_i(n-1) - \Delta_j^i(n-1)\big)\ . \end{aligned} \tag{12}$$

The induction hypothesis holds since we have that $X_{kl}^i(t) = X_{kl}^j(t)$. □

From the expression of (11), bounds on the accumulated desynchronized link balances can be expressed as follows.

Theorem 2. *If $X_{kl}^i(t) = X_{kl}^j(t)$ (for any i, j, $(k,l)=(i,j)$ or $(k,l)=(j,i)$ and any t), then the accumulated desynchronized link balances are bound in the following way:*

$$\begin{aligned} \forall i,j,\forall n,\forall M\geq 0,\forall (k,l),\quad & \max_{q\in[n,n+M]} \left(\sum_{p=n}^{q} {\widetilde{\mathcal{B}}^x}_{kl}^i(p) - {\mathcal{F}^x}_{kl}^i(q) \right) \leq \sum_{p=n}^{n+M} {\widetilde{\mathcal{B}}^x}_{kl}^i(p) \\ & \leq \min_{q\in[n,n+M]} \left(\sum_{p=n}^{q} {\widetilde{\mathcal{B}}^x}_{kl}^i(p) \right) + {\mathcal{F}^x}_{kl}^i(n+M)\ . \end{aligned} \tag{13}$$

The proof of Theorem 2 is straightforward but omitted for brevity.

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

$$\forall i,j,\forall n : \mathcal{S}^i_j(n) \neq \emptyset, \quad \Gamma^i_j(n) = \mathsf{T}_i(n) - \min_{m \in \mathcal{S}^i_j(n)} \mathsf{T}_j(m) \tag{14}$$

In $A^i(n)$, the quantities $\dot{X}^i_{ij}(\mathsf{T}^-_i(n))$ and $\dot{X}^i_{ji}(\mathsf{T}^-_i(n))$ are i's claim about how much has flowed on links (i,j) and (j,i) respectively for each class X, during interval $[\mathsf{T}_i(n-1), \mathsf{T}_i(n))$. If that advertisement also contains at least one reverse AVC tuple for the links (i.e. $\mathcal{S}^i_j(n) \neq \emptyset$), 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 $\mathsf{L}^i_j(n)$ and $\mathsf{N}^i_j(n)$ be the sequence numbers defined as follows.*

$$\forall i,j,\forall n, \begin{cases} \mathsf{L}^i_j(n) = \max\left\{m : m \leq n \wedge \mathcal{S}^i_j(m) \neq \emptyset\right\} \\ \mathsf{N}^i_j(n) = \min\left\{m : m > n \wedge \mathcal{S}^i_j(m) \neq \emptyset\right\} \end{cases} \tag{15}$$

It appears that we have the following properties:

$$\begin{aligned} &\forall i,j,\forall n, \\ &\begin{cases} \left[\mathsf{T}_i(n) - \Delta^i_j(n), \mathsf{T}_i(n)\right) \subseteq \left[\mathsf{T}_i\left(\mathsf{L}^i_j(n) - 1\right), \mathsf{T}_i(n)\right) \\ \left[\mathsf{T}_i(n) - \Delta^i_j(n), \mathsf{T}_i(n)\right) \subseteq \left[\mathsf{T}_i(n) - \Delta^i_j(n), \mathsf{T}_i\left(\mathsf{N}^i_j(n)\right) - \Gamma^i_j\left(\mathsf{N}^i_j(n)\right)\right) \end{cases} . \end{aligned} \tag{16}$$

It immediately follows that bounds for the end flow can be expressed in terms of differential values of counters contained in advertisements.

$$\begin{aligned} &\forall i,j,\forall n,\forall (k,l), \\ &\mathcal{F}^x{}^i_{kl}(n) \leq \min\left(\sum_{p=\mathsf{L}^i_j(n)}^{n}\left(\dot{X}^i_{kl}\left(\mathsf{T}^-_i(p)\right)\right), \dot{X}^j_{kl}\left(\mathsf{T}^-_i\left(\mathsf{N}^i_j(n)\right) - \Gamma^i_j\left(\mathsf{N}^i_j(n)\right)\right)\right) \end{aligned} \tag{17}$$

If there is no discrepancy between i and j's counters about link (k,l) (either (i,j) or (j,i)), then the bounds on the accumulated desynchronized link balances of (13) are verified. Even more so if upper bounds of the end flows from (17) are used instead of their actual values. In the following we call this verification the *link coherence check*. When such a check fails, the absolute value of the excess of accumulated desynchronized link balances with respect to the (loose) bounds is itself a lower bound on the amount of actual discrepancy, which is used directly in the second phase of the solution (detailed in Sect. 3.4). Then the corresponding accumulated values have to be reset to zero and the bounds to be initialized anew upon reception of the next advertisement.

3.4 Distrust System

A failed check is a proof of counter discrepancy, caused by dropped data traffic or false AVCs. Since usually some data packet loss is to be legitimately expected, a check failure should not be a direct reason for the exclusion of nodes. Nodes that drop data traffic should thus be avoided if possible, but not excluded.

In the second phase of the solution, tests from the first phase are used to maintain a distrust metric on nodes which is used for path calculation. Unlike in usual trust or reputation systems, this metric cannot be individually maintained by each node, based on first-hand observations combined with second-hand observations from neighbors. The metric has to be uniform across all the nodes, so that every node tends to know of the same value for any given node in the topology, otherwise hop-by-hop routing may not be loop-free. Therefore, the metric is a combination of all the nodes' local observations, diffused in the network.

The local observations (failures to pass checks from the first phase) are used to maintain a *local degree of distrust*. This value is initially zero and each time a check fails, it is incremented as follows:

$$D_i \leftarrow \max(1, D_i) \cdot \left(1 + L_i \cdot r^+\right) \quad , \tag{18}$$

where D_i, L_i and r^+ are respectively the degree of distrust in i, the lower bound on the amount of actual discrepancy and the ratio used as a parameter of the system set to how fast a degree has to increase. The value recovers over time in the following way:

$$D_i \leftarrow \max\left(0, D_i - \Delta t \cdot r^-\right) \quad , \tag{19}$$

where D_i, Δt and r^- are respectively the degree of distrust in i, the interval of time elapsed since last update and the ratio used as a parameter of the system set to how quickly a degree has to decrease with time.

The local degree of distrust is diffused to other nodes, to allow them to compute the distrust metric for each other node. To make the calculation resilient to slander, we propose to combine the different degrees by calculating a kind of median value, which is insensitive to pathological samples that deviate too much.

Using the local degrees of distrust directly is too simplistic and we require nodes to advertise a *confidence factor* along. That factor is used to enable each node to say to what extent its degree has to be taken into account in the calculation of the median. It is based on the amount of traffic that the observed node supposedly transmitted, according to its counters. The confidence factor increases with traffic:

$$c_i \leftarrow c_i + T_i \cdot r'^+ \tag{20}$$

(with c_i, T_i and r'^+ being respectively the confidence factor for D_i, the amount of traffic of i and the ratio used as a parameter of the system set to how quickly the confidence has to grow) and decreases with time passing:

$$c_i \leftarrow \max(0, c_i - \Delta t \cdot r'^-) \tag{21}$$

(with c_i, Δt and r'^- being respectively the confidence factor of D_i, the interval of time elapsed since last update and the ratio used as a parameter of the system set to how quickly the confidence has to decrease with time). This way, the more (alleged) traffic goes through a node, the higher the confidence factor of its neighbors in their degree of distrust in that node. The value of the confidence factor that is actually advertised is required to be in the interval $[0, 1)$,

$$C_i = 1 - \frac{1}{1 + c_i \cdot r_a} \quad , \tag{22}$$

where C_i, c_i and r_a are respectively the advertised confidence factor of D_i, the (internal) confidence factor of D_i and the ratio used as a parameter of the system set to how quickly an advertised confidence factor has to come close to 1.

The actual calculation of the metric based on individual couples (D_i^j, C_i^j) (distrust and associated confidence in i diffused by j) is a weighted median:

$$\forall i, \left\{ \left(D_i^1, C_i^1\right), \ldots, \left(D_i^n, C_i^n\right) \right\}, \\ \sum_{j:D_i^j<W_i} C_i^j \leq \frac{1}{2} \cdot \sum_j C_i^j \quad \text{and} \quad \sum_{j:D_i^j>W_i} C_i^j \leq \frac{1}{2} \cdot \sum_j C_i^j \ , \quad (23)$$

where W_i is the combined metric for node i.

A simple route calculation using a shortest path algorithm on the global topology graph is to set the weight w_{ij} of each link (i, j) to $1 + W_i$ (instead of 1 for a simple hop-count metric).

4 Performance Evaluation

4.1 Simulation Model

We have implemented our solution on top of a model of the OLSR protocol for the OPNET 12.0 simulator. Mobile nodes move on an area of 1000 m × 1000 m, with constant velocity (chosen uniformly on $[0, 1]$ m/s) and constant direction (chosen uniformly on $[0, 2\pi)$) during an interval of time chosen exponentially with rate 1/600 per second. Data packets are generated exponentially with rate 5 per second, source and destination are chosen uniformly and data length is chosen uniformly between 40 and 1500 bytes. Cheaters, also chosen uniformly among the nodes, drop data packets with a parametrized probability. A IEEE 802.11 MAC layer is used on all nodes with 11 Mbps carrier, 50 mW transmission power, flow control (RTS/CTS) and default power thresholds (-85 dBm for carrier sense and -70 dBm for reception, resulting in ranges of respectively 1200 m and 214 m).

The method presented in Sect. 3.3 and 3.4 has been implemented, along with a simple retransmission protocol for lost advertisements (omitted here for brevity), with $r^+ = 1/50$, $r'^+ = 1$, $r^- = r'^- = 1/10$ and $r_a = 10^{-6}$. The MPR selection heuristic has been adapted so as to prefer neighbors with lower local distrust (details omitted here as well). MPRs diffuse local degrees of distrust and confidence factors of their selectors in TC messages. Cheaters do slander by always advertising a very high degree of distrust and a confidence factor of 1.

The data packet delivery rate (total received data packets over total generated data packets) has been measured in each run and compared to several reference runs: in the **No Cheater** run, plain OLSR is used and the drop probability is forced to zero to see what OLSR achieves at best without cheaters; in the **Plain** run, plain OLSR is faced with non-zero drop probability to see how bad it does alone; in the **Known** run, OLSR has an oracle telling which node is a cheater and routing table calculation is performed as suggested at the end of Sect. 3.4 with $W_i = 9$ for each cheater i. That last reference run is used to see what performance could be achieved at best.

Cheaters that intentionally drop data packets have many possible ways in which they can manipulate their counters. Among possible strategies, we have chosen four different ones: increment counters as though the packets were forwarded indeed; increment counters on all outgoing links so that the total added sum is the amount of lost data bytes; do not increment counters on outgoing links; increment counters only for every second packet.

4.2 Simulation Results

In the first batch of simulations, there were 100 nodes with 10% of cheaters. We have checked that the system stabilizes pretty quickly and after a few dozen seconds, the delivery ratio of our method is stable (though we checked for up to 10 hours) and much closer to the **Known** case than to the **Plain** one. Lowering the drop probability, apart from raising the **Plain** curve, only lengthened by a small factor the time required for the delivery ratio to be as close to the **Known** curve (approximately 3 minutes for a drop probability of 0.25). The average delivery ratios of the **Plain**, **Known** and the average of the four cheating strategies (which results turned out to be very similar) after one hour of simulation, with 100 nodes, 10% cheaters with 1.0 drop probability, are presented in Fig. 2 (left). It appears clearly that in dense networks, the performance of our method sticks to the **Known** reference.

To evaluate the performance of our method with respect to varying network density, we have run a batch of simulations with a different total number of nodes but a (almost) constant ratio of 10% of cheaters, each dropping with a probability of 1.0. The results presented in Fig. 2 (right) show that while the delivery ratio degrades with decreasing density, the performance of our method is still very close to the **Known** reference. The reason why maximum achievable performance decreases with the density is that there are fewer and fewer alternative paths for a given destination, hence fewer ways to avoid the cheaters.

A last batch of simulations has been run with a duration of 15 minutes, a total number of 100 nodes but a varying number of cheaters with 1.0 drop probability. Results show (see Fig. 3) that our method improves the delivery

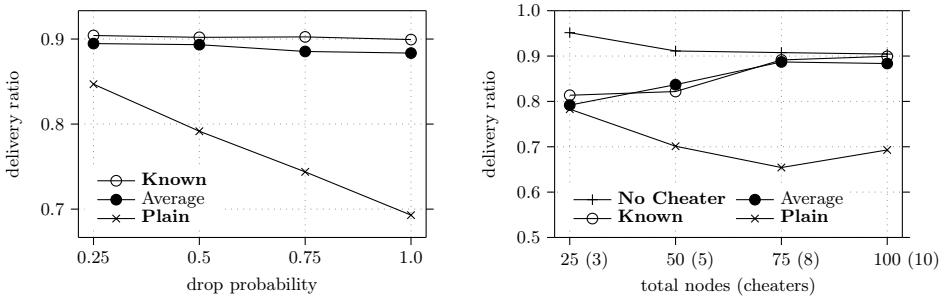

Fig. 2. Delivery ratio: with fixed density of 100 nodes and variable drop probability (left); with variable density and fixed drop probability of 1.0 (right).

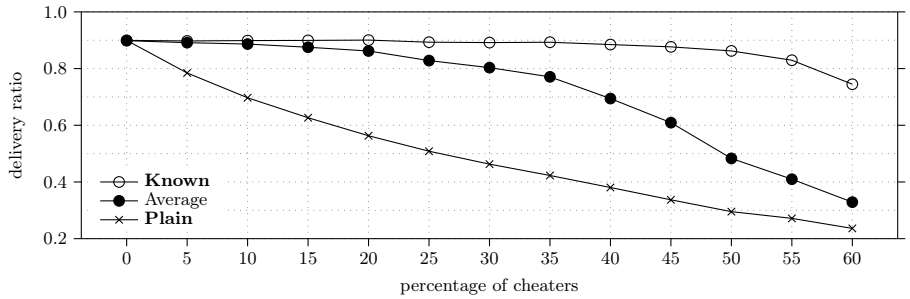

Fig. 3. Delivery ratio vs. cheaters: 1.0 drop probability and a total of 100 nodes.

ratio significantly, but less and less than the **Known** reference as the number of cheaters increases.

4.3 Discussion

Hughes *et al.* strongly criticized [8] the use of the principle of flow conservation as a way to detect malicious nodes in a network. They presented a number of attack scenarios which make simplistic approaches fail anyway.

In our solution, we require all packets to be checked for integrity and their source address to be authenticated by all intermediate nodes, to exclude the possibility of packet modification or forgery going unnoticed. Some attacks, like "ghost routers" or the ones based on the reactivity of link-state routing protocols, are targeted at the routing protocol itself, which we assume here to be already protected. Note that the "kamikaze" attack, in which a malicious node intentionally advertises false counter values in order to be excluded along with some of its neighbors is simply handled by our solution, in which nodes are never excluded.

5 Conclusion

We have presented a way to augment a link-state routing protocol to make it resilient to the presence of nodes intentionally dropping data traffic. The method consists in three parallel operations: detect flow non-conservation on neighboring nodes, maintain a degree of distrust in all checked nodes, combine all degrees of distrust into a metric for routing table calculation.

The method is very little intrusive into the system, provided that necessary cryptographic tools are available and that the routing protocol itself is protected. It is independent of OSI Data Link and Physical layers which makes it easier to implement and workable with different hardware setups (unidirectional antennae, multiple interfaces, etc). The method showed satisfactory performance, very close to an ideal case in which all cheating nodes are known to an oracle.

Among the possible future works, we intend to explore the possibility of making the method work even in the presence of colluding cheaters. The interaction with some intrusion detection system can also be studied, in order to counter the "premature aging" attack.

References

1. 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.
2. 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.
3. 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.
4. Levente Buttyán and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *MONET*, 8(5):579–592, 2003.
5. T. Clausen and P. Jacquet. Optimized link state routing (OLSR) protocol. RFC 3626, IETF, October 2003.
6. 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.
7. 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.
8. 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 132–141, 2000.
9. David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4. RFC 4728, IETF, February 2007.
10. 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*, *LNCS* vol. 2865, pages 151–163. Springer, 2003.
11. 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.
12. 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.
13. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, IETF, July 2003.
14. 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.
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.