Proactive Resilience to Dropping Nodes in Mobile Ad Hoc Networks

Ignacy Gawędzki and Khaldoun Al Agha

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

Abstract. Proactive routing protocols for mobile ad hoc networks currently offer few mechanisms to detect and/or counter malevolent nodes. Stability and performance of most, if not all, emerging standard proactive protocols rely on cooperation between nodes. While cryptographic methods may be a solution to secure control messages, nodes not willing to cooperate may still decide not to forward data packets. In this paper, a method to enable resilience to such malevolent nodes is presented. It is non-intrusive with respect to the packet forwarding mechanisms (e.g. TCP/IP kernel stack) and particularly well suited for integration with proactive routing protocols.

Key words: proactive routing, ad hoc networks, malicious nodes, resilience.

1 Introduction

Currently emerging standardized routing protocols for mobile ad hoc networks (MANETs for short) [1–3] are distributed algorithms relying on cooperation between participating nodes to achieve efficient routing. Unfortunately, there are applications of MANETs where the assumption of benevolent collaboration is too strong and in fact nodes may be expected to behave in any conceivable way. Such potentially malevolent nodes are called *Byzantine processes* [4] in the jargon of distributed computing.

If Byzantine processes are to be expected among the participating nodes of a MANET, there are many imaginable ways by which such nodes could undermine the operation of the network as a whole (data flooding, packet dropping, . . .) or a proactive routing protocol in particular (sending fake control messages, modifying control or data packets before retransmitting them, . . .).

In Sect. 1.1 and 1.2, the problem at hand is presented and previous work resumed. In Sect. 2, the basis of the proposed idea is stated and then in Sect. 3, it is applied for the detection of cheaters. A few remarks about its practical implementation are given in Sect. 4. Evaluation results are analyzed in Sect. 5 and we finally conclude in Sect. 6.

1.1 Problem Statement

The general question of how to make a MANET resilient to Byzantine processes is a very difficult problem. Because of the complicated and compound design of MANETs, there are many weak points that could be exploited by malicious nodes to hinder network operations. At the physical level, because of the very nature of the radio medium, plain old jamming may forbid successful network communications. At the logical link level, the lack of scheduling facilities in current wireless MAC layers makes it possible to perform Denial of Service attacks. At the network level, a node may poison routing control messages to either get more resources for itself or simply, in a non-egoistic way, sap others' communications.

Some of the these issues may be addressed using cryptographic methods which enable to ensure no packet gets forged or altered and/or no data payload disclosed. Furthermore, depending on the routing protocol used, some methods enable surrounding nodes to verify that routing information diffused by a given node is correct. Consequently, in the present paper, we will focus on a more basic problem: the resilience to nodes that drop others' traffic instead of forwarding it towards their intended destination. The proposed solution relies on a simple and yet easily satisfied assumption, namely that those dropping nodes — let's call them *cheaters* — are isolated (no two of them are neighbors) and do not collude with each other.

1.2 Previous Work

Byzantine robustness of routing protocols have been actively studied for quite some time. The more and more decentralized aspect of network administration of the Internet made routing more prone to attacks. The emergence of wireless ad hoc networks called for a similar concern, but existing solutions for wired networks could not be applied directly, as was already the case with routing protocols proper.

The most trivial attack, and yet the most likely in a public network, is that of *selfish nodes*: nodes that don't want to spend their energy resources on behalf of other nodes. In [5], Marti et al. propose a solution to detect selfish nodes in a passive way, relying on the ability of neighbors to overhear a node's transmissions on the medium, it is expected to work on the condition that all nodes have only one wireless network interface, omnidirectional antennae and equal transmission power. Other approaches, based on prevention of selfish behavior by encouraging cooperation have also been proposed. The use of a virtual currency has been proposed in [6] and [7] with different rewarding rules for nodes that forward others' packets and different ways to ensure no node cheats on the currency. The downside of these solutions is that nodes on the extremities of a network have less chance that those in the center to forward others' packets and eventually go bankrupt, while having acted in a legitimate way. Solutions based on active probing of nodes have also been proposed [8–10]. They all rely on the inability of the probed node to either distinguish probes from plain data traffic or identify the originator of a probe.

A whole set of more elaborate attacks against routing protocols, such as advertisement of false routing information, tampering with others' control and data packets, misrouting, wormhole attacks, etc, have been addressed in various solutions which usually aim to provide Byzantine robustness to a reactive, on-demand routing protocol as DSR or AODV [9, 11, 12].

As for proactive, table-driven routing protocols, the problem of Byzantine robustness is twofold: how to secure the control messages and the information they contain and how to secure the correct forwarding of data packets. The OLSR protocol has received some attention lately regarding its security. In [13], Raffo et al. propose a way to secure the content of control messages and thus prevent nodes from advertising false or expired information. In [14], another such protection is proposed, based on threshold cryptography and the requirement that a node cannot sign a TC message by itself and must ask its neighbors to do it, by the same time validating its contents.

The idea proposed in the present paper has already been put forward in the case of fixed, wired networks by Bradley et al. in [15]. It has been later argued in [16] that the assumptions on the network model are too strong and that strict checking of nodes is practically impossible. Thus in this paper, a relaxed way of checking is proposed which does not rely on these assumptions and which allows nodes to drop little traffic without being noticed.

2 Theoretical Background

This section introduces the general reasoning from the flow conservation principle up to checking it in a network using packet counters. A very similar technique was proposed by Bradley et al. in [15] for static wired networks, but it relied on assumptions that do not hold in the case of MANETs.

2.1 Single and Multiple Flows Conservation

Given a flow network in the form of a directed graph $G = (V, E)$ with vertices V and edges E, a source vertex s and a sink vertex t, a function of directed flow $f : V \times V \to \mathbb{R}^+$, the principle of flow conservation states that the net flow to a vertex is zero, except for the source s and the sink t.

$$\forall i \in V \backslash \{s, t\}, \quad \sum_{j \in V} (f(j,i) - f(i,j)) = 0 \tag{1}$$

Suppose that instead of having one (s, t, f) triplet, a collection $\mathcal{F}$ of triplets is given. The principle of flow conservation for all the triplets in $\mathcal{F}$ at once can be stated as follows.

$$\forall i \in V, \quad \sum_{j \in V} \sum_{\substack{(s,t,f) \in \mathcal{F} \\ i \notin \{s,t\}}} (f(j,i) - f(i,j)) = 0 \tag{2}$$

2.2 Flow Conservation in a Network

A network is a set of nodes connected with links that can be symmetric or asymmetric. Thus it can be modelled by a directed graph which vertices are the nodes and which arcs are the links (a symmetric link between nodes i and j being represented by two arcs (i, j) and (j, i)). The flow on the arc (i, j) can be considered as the total amount of data that has flowed on the link from i to j.

For the sake of simplicity, let's define a few quantities.

Definition 1 (input and output). *Let I_{ij} (resp. O_{ij}), the input (resp. output) from node i to node j, be the amount of data that flows from i to j but is not destined to j (resp. is not generated by i).*

$$I_{ij} = \sum_{\substack{(s,t,f)\in\mathcal{F}\\ j\neq t}} f(i,j) \tag{3}$$

$$O_{ij} = \sum_{\substack{(s,t,f)\in\mathcal{F}\\ i\neq s}} f(i,j) \tag{4}$$

Thus I_{ij} is the amount of data sent from i to j for retransmission and similarly O_{ij} is the amount of data retransmitted from i to j.

The multiple flow conservation is then transposed as follows.

$$\forall i \in V, \quad \sum_{j\in V} (I_{ji} - O_{ij}) = 0 \tag{5}$$

Suppose no node can forge packets (i.e. generate packets that appear as if they have been generated by another node), then the amount of outgoing forwarded data is at most equal to the amount of incoming data not destined to the current node. Therefore, any node's balance (the amount of flow non-conservation) is positive.

$$\forall i \in V, \quad \sum_{j\in V} (I_{ji} - O_{ij}) \geq 0 \tag{6}$$

Moreover, in these conditions, a node's balance is precisely the amount of data lost at that node. Hence the idea in the following is to measure how much traffic a node drops by looking at its balance.

2.3 Using Packet Counters

In a real-world scenario, the amount of data that has flowed on a link is usually implemented by using packet counters in the TCP/IP stack (OSI layer 3) or in the MAC driver (OSI layer 2). This makes it difficult to compute a node's balance directly.

Indeed, instead of having counters on links that read the absolute amount of data that has flowed so far, counters are maintained on a link's both endpoints and the transmission along a link maybe unreliable.

Definition 2 (input and output view of a node). *Let I^i_{ij} and I^j_{ij} be respectively node i's and j's view of counter I_{ij}. Similarly, let O^i_{ij} and O^j_{ij} be respectively node i's and node j's view of counter O_{ij}.*

In absence of data loss on lower layers (layers below the one that performs packet accounting), the following properties hold.

$$\forall i,\ \forall j, \quad \begin{cases} I^i_{ij} = I^j_{ij} = I_{ij} \\ O^i_{ij} = O^j_{ij} = O_{ij} \end{cases} \tag{7}$$

Unfortunately, data loss usually does occur at lower layers, so only the following properties hold.

$$\forall i,\ \forall j, \quad \begin{cases} I^j_{ij} = I_{ij} \\ O^j_{ij} = O_{ij} \end{cases} \tag{8}$$

Indeed, a packet has not surely been transmitted on a link until it has been received by the corresponding layer on the receiving side (OSI Data Link or Network layers).

Using endpoints' counters, the principle of multiple flows conservation is restated as follows.

$$\forall i \in V, \quad \sum_{j \in V} \left(I^i_{ji} - O^j_{ij} \right) = 0 \tag{9}$$

3 Detecting Cheaters

This section presents the proposed solution based on flow conservation checking based on packet counters as introduced in Sect. 2.

3.1 Network model

Cryptographic solutions are employed to be able to authenticate every originator of data and control packets. It is assumed that no node has enough resources to break the cryptographic scheme. The routing protocol employs periodic control messages for which the maximal period of time between two successive sent messages is known. The links between nodes have known finite total capacity (bandwidth).

3.2 Isolated and Non-colluding Cheaters

A cheater is a node that is not expected to collaborate with other nodes. More specifically in this case, it is expected not to perform packet accounting properly: it can change some or all of its counters with no relation to sending or receiving packets. Therefore, the content of its counters must not be relied upon.

An isolated cheater is a cheater that is not colluding with any other cheater and which has no cheater among its neighbors. If cheaters are assumed to be isolated, it is also assumed that on any link, at least one endpoint's counters are "in good faith."

Since a cheater can falsely advertise its counters, if for some counter X_{ij}, we have $X^i_{ij} \neq X^j_{ij}$, then it may be that there was some packet loss at lower layers from i to j, or either i or j is falsely advertising its version of X_{ij}. Similarly, it may happen that $X^i_{ij} = X^j_{ij}$ and there were indeed lost packets from i to j (i could falsely not count the emitted and then lost packets, or j could falsely count as received the lost packets), but this would require a good load of chance and guesswork.

It is clear that differing counters are a sign of either lost or dropped packets or false counters or both and in the following, we will not distinguish between the three situations, we will only speak about *counter discrepancy.* This is still an acceptable approach, since that information is to be eventually used to avoid routing packets through suspected or unreliable nodes (be it because they lose or drop traffic or lie about their counters).

3.3 Split Balance

Since one node's counters cannot be relied upon, a node's balance calculation has to be performed in many ways simultaneously. First, the node's internal balance (i.e. according to its own version of the counters) has to be calculated.

Definition 3 (node balance). *For a node i, its node balance $\mathcal{B}_i$ is the difference between all input and output, as viewed by itself.*

$$\mathcal{B}_i = \sum_{j \in V} \left(I^i_{ji} - O^i_{ij} \right) \tag{10}$$

A non-zero node balance would directly indicate deliberate packet dropping (or shooting itself in the foot), whereas a zero node balance does not mean anything by itself, since counters may have been manipulated incorrectly. So all link balances with that node's neighbors have to bee calculated too.

Definition 4 (input and output link balances). *An input (resp. output) link balance on link (i, j) is the difference between views of nodes i and j of counter I_{ij} (resp. O_{ij}).*

$$\mathcal{B^I}_{ij} = I^i_{ij} - I^j_{ij} \tag{11}$$

$$\mathcal{B^O}_{ij} = O^i_{ij} - O^j_{ij} \tag{12}$$

As traffic lost due to lower layer hazard is not to be distinguished from traffic lost due to deliberate packet dropping, all lost traffic should be accounted for in link balance. Unfortunately, the I_{ij} and O_{ij} counters are not enough to account for all the traffic, as for example packets generated by i, thus not counted in O_{ij}, and destined to j, thus not counted in I_{ij}, are not counted at all.

Definition 5 (total link balance). *The total link balance on link (i,j) is the difference between T^i_{ij} and T^j_{ij}, respectively i's and j's view of T_{ij}, the total amount of traffic that flows on link (i,j).*

$$\mathcal{B}^T{}_{ij} = T^i_{ij} - T^j_{ij} \tag{13}$$

The T_{ij} counter already counts packets counted by I_{ij} and O_{ij}, so for later convenience, the following counters will be used instead of T_{ij}.

Definition 6 (source and destination view of a node). *Let S^i_{ij} (resp. D^j_{ij}) be the amount of traffic generated by i (resp. destined to j) according to i (resp. j).*

$$S^i_{ij} = T^i_{ij} - O^i_{ij} \tag{14}$$
$$D^j_{ij} = T^j_{ij} - I^j_{ij} \tag{15}$$

Note that S^j_{ij} and D^i_{ij} can be defined as follows.

$$S^j_{ij} = T^j_{ij} - O^j_{ij} = D^j_{ij} + I^j_{ij} - O^j_{ij} \tag{16}$$
$$D^i_{ij} = T^i_{ij} - I^i_{ij} = D^i_{ij} + O^i_{ij} - I^i_{ij} \tag{17}$$

The total link balance can be now expressed as

$$\mathcal{B}^T{}_{ij} = S^i_{ij} + O^i_{ij} - D^j_{ij} - I^j_{ij} \tag{18}$$

3.4 Checking Split Balance

Suppose that nodes in a network have to monitor their neighbors using the split balance scheme from the previous section. During normal network operation, each node i maintains for each neighbor j the following series of counters.

S^i_{ij}	: amount of data flowed from i to j and generated by i
O^i_{ij}	: amount of data flowed from i to j but not generated by i
I^i_{ij}	: amount of data flowed from i to j but not destined to j
D^i_{ji}	: amount of data flowed from j to i and destined to i
I^i_{ji}	: amount of data flowed from j to i but not destined to i
O^i_{ji}	: amount of data flowed from j to i but not generated by j

At properly chosen times, nodes advertise their counters and calculate balances on nodes and links for which they have gathered enough of them. For a particular balance calculation, let the node performing it be the *observer* and the nodes about which it is performed the *observed*. For node balance calculations, the node that advertised the counters is the only *observed*, whereas for link balance calculations, both endpoints are. Depending on the underlying network model, an *observer* can maintain a degree of distrust in some subset of nodes (the ones it has observed so far). While a non-zero node balance affects the degree of distrust of the *observer* in the *observed* directly (we can safely assume that an *observer* has no interest in observing itself and thus the *observer* and *observed* are distinct), a non-zero link balance affects the degree of distrust of the *observer* in both endpoints, unless the *observer* is also one of the *observed* (a node is checking the balance on a link between itself and one of its neighbors).

3.5 Desynchronized Counters

In order to be able to make nodes check each other strictly, in a way that allows them to detect every single lost packet, nodes should be required to advertise their counters as soon as they send or receive one packet and to forward their neighbor's counters as soon as they receive them. Needless to say that this would incur a considerable control overhead.

A more flexible way is possible if the strict requirement is relaxed. Nodes can use periodic dedicated or existing control messages to advertise their counters, as well as all their neighbors' latest counters they have received so far.

Definition 7 (timed counters). *For each counter X^i_{ij} (resp. X^j_{ij}), let $X^i_{ij}(t)$ (resp. $X^j_{ij}(t)$) be its timed version, in other words its state at instant t, with the property that $X^i_{ij}(0) = 0$ (resp. $X^j_{ij}(0) = 0$).*

Let P be the upper bound on the interval of time elapsed between two consecutive advertisements sent by one node. Every sent advertisement message contains all that is required for a node receiving it to check its originator ($\mathcal{B}_i$) and the links between it and all its neighbors (${\mathcal{B}^I}_{ij}$, ${\mathcal{B}^I}_{ji}$, ${\mathcal{B}^O}_{ij}$, ${\mathcal{B}^O}_{ji}$, ${\mathcal{B}^T}_{ij}$ and ${\mathcal{B}^T}_{ji}$).

Each node i maintains a sequence number counter which is incremented just before sending out each advertisement.

Definition 8 (ad sequence number). *Let $\mathsf{S}_i(t)$ be the value of i's sequence number counter at instant t and let $\mathsf{T}_i(n)$ be the instant at which i sends the advertisement with sequence number n, such that*

$$\forall i, n, t \in [\mathsf{T}_i(n), \mathsf{T}_i(n+1)) , \quad \mathsf{S}_i(t) = n \tag{19}$$

Then for each instant t, the time of the latest advertisement sent by i is $\mathsf{T}_i(\mathsf{S}_i(t))$, the time of the second latest is $\mathsf{T}_i(\mathsf{S}_i(t) - 1)$ and more generally $\mathsf{T}_i(\mathsf{S}_i(t) - p)$ is the time of the pth latest advertisement.

For practical reasons explained later in Sect. 4, we will use differential timed counters.

Definition 9 (differential timed counter). *For each timed counter $X^i_{ij}(t)$ (resp. $X^j_{ij}(t)$), let $\dot{X}^i_{ij}(t)$ (resp. $\dot{X}^j_{ij}(t)$) be the differential counter defined by*

$$\dot{X}^i_{ij} = X^i_{ij}(t) - X^i_{ij}\left(\mathsf{T}_i\left(\mathsf{S}_i(t)\right)\right) \tag{20}$$

and respectively

$$\dot{X}^j_{ij} = X^j_{ij}(t) - X^j_{ij}\left(\mathsf{T}_j\left(\mathsf{S}_j(t)\right)\right) \quad . \tag{21}$$

Definition 10 (instant before advertising). *Let $\mathsf{T}^-_i(n)$ be the instant just before i sends advertisement with sequence number n, so we have $\mathsf{S}_i(\mathsf{T}_i(n)) = n$ and $\mathsf{S}_i(\mathsf{T}^-_i(n)) = n - 1$.*

The notation T^- is only used for differential timed counters, to explicitly indicate that the value of interest is the one just before sending the advertisement containing it, because usually $\dot{X}^i_{ij}(\mathsf{T}^-_i(n)) \neq \dot{X}^i_{ij}(\mathsf{T}_i(n))$, unless there has been no traffic on the link.

Definition 11 (link differential timed counter tuple). *Let $\mathcal{C}^i_j(t)$ be the tuple of i's timed counters regarding links between itself and j at instant t*

$$\mathcal{C}^i_j(t) = \left(S^i_{ij}(t), O^i_{ij}(t), I^i_{ij}(t), D^i_{ji}(t), I^i_{ji}(t), O^i_{ji}(t)\right) \tag{22}$$

and let $\dot{\mathcal{C}}^i_j(t)$ be its differential counterpart

$$\dot{\mathcal{C}}^i_j(t) = \left(\dot{S}^i_{ij}(t), \dot{O}^i_{ij}(t), \dot{I}^i_{ij}(t), \dot{D}^i_{ji}(t), \dot{I}^i_{ji}(t), \dot{O}^i_{ji}(t)\right) \quad . \tag{23}$$

Definition 12 (link ad). *Let $\mathcal{A}^i_j(n)$, i's link advertisement with sequence number n, be a tuple of i's sequence number n and differential counters regarding links between itself and j right before that advertisement is sent by i.*

$$\mathcal{A}^i_j(n) = \left(n, \dot{\mathcal{C}}^i_j\left(\mathsf{T}^-_i(n)\right)\right) \tag{24}$$

Between the transmission of two successive advertisements, for each link with one of its neighbors, node i gathers the other endpoints' advertisements regarding that link.

Definition 13 (set of gathered ads' sequence numbers). *Let $\mathcal{S}^i_j(n)$ be the set of sequence numbers of j's advertisements received by i after the latter sent the advertisement with sequence number $n-1$ and before it sent the advertisement with sequence number n.*

$$\mathcal{S}^i_j(n) = \{m : \mathsf{T}_i(n-1) \leq \mathsf{T}_j(m) < \mathsf{T}_i(n)\} \tag{25}$$

Definition 14 (set of gathered ads). *Let $\mathcal{A}^j_i(n)$ be the (possibly empty) set of j's advertisements regarding links between itself and i gathered by the latter between $\mathsf{T}_i(n-1)$ (inclusive) and $\mathsf{T}_i(n)$ (exclusive).*

$$\mathcal{A}^j_i(n) = \bigcup_{m \in \mathcal{S}^i_j(n)} \left\{\mathcal{A}^j_i(m)\right\} \tag{26}$$

Definition 15 (neighbor set). *Let $\mathcal{N}_i$ be the set of i's neighbors from which it receives advertisements.*

Definition 16 (advertisement). *For each link between i and its neighbors, i's advertisement with sequence number n contains its own link advertisement regarding it and the set of all the other endpoint's advertisements regarding that same link gathered by i since its previous advertisement:*

$$\mathcal{M}^i(n) = \left\{ \left(\mathcal{A}^i_j(n), \mathcal{A}^j_i(n) \right) : j \in \mathcal{N}_i \right\} \tag{27}$$

Definition 17 (ad-based node balance). *Let $\mathcal{B}_i(n)$ be the i's node balance based on counters from its advertisement number n.*

$$\mathcal{B}_i(n) = \sum_{j \in \mathcal{N}_i} \left(\dot{I}^i_{ji} \left(\mathsf{T}^-_i(n) \right) - \dot{O}^i_{ij} \left(\mathsf{T}^-_i(n) \right) \right) \tag{28}$$

Node i's advertisement-based node balance can be checked by any node as soon as the latter receives i's $\mathcal{M}^i(n)$ containing $\mathcal{A}^i_j(n)$ for each $j \in \mathcal{N}_i$. Unfortunately, no link balance like $\mathcal{B}^I{}_{ij}$, $\mathcal{B}^I{}_{ji}$, $\mathcal{B}^O{}_{ij}$, $\mathcal{B}^O{}_{ji}$, $\mathcal{B}^T{}_{ij}$ or $\mathcal{B}^T{}_{ji}$ can be strictly checked based solely on i's advertisement number n, without the knowledge of both $\dot{\mathcal{C}}^i_j(\mathsf{T}^-_i(n))$ and the corresponding link differential counter tuple from j for the same interval of time.

Nevertheless, a loose check can still be performed, if one looks at desynchronized versions of these link balances. Depending of what endpoint's advertisement is available, those can be computed in two similar ways.

$$\forall i, j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\},$$
$$\begin{cases} \widetilde{\mathcal{B}^I}{}^i_{kl}(n) = \dot{I}^i_{kl}\left(\mathsf{T}^-_i(n)\right) - \sum\limits_{m \in \mathcal{S}^i_j(n)} \dot{I}^j_{kl}\left(\mathsf{T}^-_j(m)\right) \\ \widetilde{\mathcal{B}^O}{}^i_{kl}(n) = \dot{O}^i_{kl}\left(\mathsf{T}^-_i(n)\right) - \sum\limits_{m \in \mathcal{S}^i_j(n)} \dot{O}^j_{kl}\left(\mathsf{T}^-_j(m)\right) \\ \widetilde{\mathcal{B}^T}{}^i_{kl}(n) = \dot{S}^i_{kl}\left(\mathsf{T}^-_i(n)\right) + \dot{O}^i_{kl}\left(\mathsf{T}^-_i(n)\right) - \sum\limits_{m \in \mathcal{S}^i_j(n)} \left(\dot{D}^j_{kl}\left(\mathsf{T}^-_j(m)\right) + \dot{I}^j_{kl}\left(\mathsf{T}^-_j(m)\right) \right) \end{cases} \tag{29}$$

That loose check does not mean anything by itself. Since it is based on values of counters from different intervals, the values of these balances are most probably not zero when there is some traffic on the link. Fortunately, as should be noted in the following section, there still are ways to detect problems on links based on desynchronized balances.

3.6 Desynchronization and Tolerance

To calculate the exact values of desynchronized balances, we need to define $\delta^i_j(n)$, the minimum amount of time, as of $\mathsf{T}_i(n)$, elapsed since $\mathsf{T}_i(n-1)$ or last time j's advertisement was received. Similarly, let $\Delta^i_j(n)$ be the amount of time, as of $\mathsf{T}_i(n)$, elapsed since last time j's advertisement was received.

$$\delta^i_j(n) = \begin{cases} \mathsf{T}_i(n) - \max_{m \in \mathcal{S}^i_j(n)} \mathsf{T}_j(m) & \text{if } \mathcal{S}^i_j(n) \neq \emptyset \ , \\ \mathsf{T}_i(n) - \mathsf{T}_i(n-1) & \text{otherwise.} \end{cases} \tag{30}$$

$$\varDelta^i_j(n) = \begin{cases} \delta^i_j(n) & \text{if } \mathcal{S}^i_j(n) \neq \emptyset \ , \\ \delta^i_j(n) + \varDelta^i_j(n-1) & \text{otherwise.} \end{cases} \tag{31}$$

For the special case of $n = 0$, let's say that $\mathsf{T}_i(-1) = 0$ and that $\varDelta^i_j(-1) = 0$.

If there is no actual discrepancy between counters (due to lost packets or wrongly manipulated counters), then $X^i_{ij}(t) = X^j_{ij}(t)$ at all instants t, for all counters X_{ij} and the the value of the desynchronized link balances are

$$\forall i, j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\}\,,$$
$$\begin{cases} \widetilde{\mathcal{B}}_{I}{}^i_{kl}(n) = & I^i_{kl}\big(\mathsf{T}_i(n)\big) - I^i_{kl}\big(\mathsf{T}_i(n) - \varDelta^i_j(n)\big) \\ & - I^j_{kl}\big(\mathsf{T}_i(n-1)\big) + I^j_{kl}\big(\mathsf{T}_i(n-1) - \varDelta^i_j(n-1)\big) \\ \widetilde{\mathcal{B}}_{O}{}^i_{kl}(n) = & O^i_{kl}\big(\mathsf{T}_i(n)\big) - O^i_{kl}\big(\mathsf{T}_i(n) - \varDelta^i_j(n)\big) \\ & - O^j_{kl}\big(\mathsf{T}_i(n-1)\big) + O^j_{kl}\big(\mathsf{T}_i(n-1) - \varDelta^i_j(n-1)\big) \\ \widetilde{\mathcal{B}}_{T}{}^i_{kl}(n) = & S^i_{kl}\big(\mathsf{T}_i(n)\big) - S^i_{kl}\big(\mathsf{T}_i(n) - \varDelta^i_j(n)\big) \\ & + O^i_{kl}\big(\mathsf{T}_i(n)\big) - O^i_{kl}\big(\mathsf{T}_i(n) - \varDelta^i_j(n)\big) \\ & - D^j_{kl}\big(\mathsf{T}_i(n-1)\big) + D^j_{kl}\big(\mathsf{T}_i(n-1) - \varDelta^i_j(n-1)\big) \\ & - I^j_{kl}\big(\mathsf{T}_i(n-1)\big) + I^j_{kl}\big(\mathsf{T}_i(n-1) - \varDelta^i_j(n-1)\big) \ . \end{cases} \tag{32}$$

Let B_{ij} be the upper bound on the amount of data that can flow on links (i,j) and (j,i) per unit of time. It is obviously a reasonable assumption that this bound exists, at least in any realistic scenario. Then since counters' values are positive by definition, we have that

$$\forall i, j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\}\,,$$
$$\max\left(\left|\widetilde{\mathcal{B}}_{I}{}^i_{kl}(n)\right|, \left|\widetilde{\mathcal{B}}_{O}{}^i_{kl}(n)\right|, \left|\widetilde{\mathcal{B}}_{T}{}^i_{kl}(n)\right|\right) \leq \mathsf{B}_{kl} \cdot \max\left(\varDelta^i_j(n-1), \varDelta^i_j(n)\right) \ . \tag{33}$$

For nodes other than i and j, values of T_j or T_i may not be known, so an upper bound to $\varDelta^i_j$ and $\varDelta^j_i$ has to be used instead. Since every node sends its advertisements at least P units of time apart, it is also the upper bound to $\varDelta^i_j$ or $\varDelta^j_i$. In the worst possible case, we have the following property:

$$\forall i, j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\}\,,$$
$$\max\left(\left|\widetilde{\mathcal{B}}_{I}{}^i_{kl}(n)\right|, \left|\widetilde{\mathcal{B}}_{O}{}^i_{kl}(n)\right|, \left|\widetilde{\mathcal{B}}_{T}{}^i_{kl}(n)\right|\right) \leq \mathsf{B}_{kl} \cdot \mathsf{P} \ . \tag{34}$$

The use of this scheme would allow only i and j to observe counter discrepancy on link (i,j) exactly (assuming that the transmission time of packets containing advertisements is negligible). Other nodes could observe packet loss on this link only in excess of $\mathsf{B}_{ij} \cdot \max(\Delta^i_j(n-1), \Delta^i_j(n))$ or $\mathsf{B}_{ij} \cdot \max(\Delta^j_i(m-1), \Delta^j_i(m))$ at best (which means they receive advertisements from both i and j) or $\mathsf{B}_{ij} \cdot \mathsf{P}$ at worst. This leaves quite a gap for a malicious node to exploit in order to drop traffic with most of the other nodes not noticing.

3.7 Accumulated Balances

There is a way to enable nodes to observe even smaller counter discrepancies on links, provided a bit of patience. Instead of looking at link balances directly, nodes could observe their accumulated value over all advertisements:

$$\begin{array}{lll} \sum\limits_n \widetilde{\mathcal{B}}{}^{I}{}^i_{ij}(n) \ , & \sum\limits_n \widetilde{\mathcal{B}}{}^{O}{}^i_{ij}(n) \quad \text{and} & \sum\limits_n \widetilde{\mathcal{B}}{}^{T}{}^i_{ij}(n) \ , \text{ or} \\ \sum\limits_m \widetilde{\mathcal{B}}{}^{I}{}^j_{ij}(m) \ , & \sum\limits_m \widetilde{\mathcal{B}}{}^{O}{}^j_{ij}(m) \quad \text{and} & \sum\limits_m \widetilde{\mathcal{B}}{}^{T}{}^j_{ij}(m) \ . \end{array}$$

It appears that in case there is no actual counter discrepancy, these accumulated values are bounded too.

Theorem 1. *For each node i and each of its neighbor j, the accumulated values of calculated link balances over i's advertisements with numbers from n to $n+M$, for any $M \geq 0$ are exactly*

$$\forall i, j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\}, n, M \geq 0,$$
$$\begin{cases} \sum\limits_{p=n}^{n+M} \widetilde{\mathcal{B}}{}^{I}{}^i_{kl}(p) = & I^i_{kl}\big(\mathsf{T}_i(n+M)\big) - I^i_{kl}\big(\mathsf{T}_i(n+M) - \Delta^i_j(n+M)\big) \\ & - I^j_{kl}\big(\mathsf{T}_i(n-1)\big) + I^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\ \sum\limits_{p=n}^{n+M} \widetilde{\mathcal{B}}{}^{O}{}^i_{kl}(p) = & O^i_{kl}\big(\mathsf{T}_i(n+M)\big) - O^i_{kl}\big(\mathsf{T}_i(n+M) - \Delta^i_j(n+M)\big) \\ & - O^j_{kl}\big(\mathsf{T}_i(n-1)\big) + O^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\ \sum\limits_{p=n}^{n+M} \widetilde{\mathcal{B}}{}^{T}{}^i_{kl}(p) = & S^i_{ij}\big(\mathsf{T}_i(n+M)\big) - S^i_{kl}\big(\mathsf{T}_i(n+M) - \Delta^i_j(n+M)\big) \\ & + O^i_{kl}\big(\mathsf{T}_i(n+M)\big) - O^i_{kl}\big(\mathsf{T}_i(n+M) - \Delta^i_j(n+M)\big) \\ & - D^j_{kl}\big(\mathsf{T}_i(n-1)\big) + D^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\ & - I^j_{kl}\big(\mathsf{T}_i(n-1)\big) + I^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \end{cases} \tag{35}$$

Proof. By induction on $M \geq 0$. Already shown for $m = 0$ in (32). Suppose that it holds for $m \geq 0$, then we have that

$\forall i,j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\}, n,m \geq 0,$

$$\begin{cases}
\displaystyle\sum_{p=n}^{n+m+1} \widetilde{\mathcal{B}^I}{}^i_{kl}(p) = & I^i_{kl}\big(\mathsf{T}_i(n+m)\big) - I^i_{kl}\big(\mathsf{T}_i(n+m) - \Delta^i_j(n+m)\big) \\
& - I^j_{kl}\big(\mathsf{T}_i(n-1)\big) + I^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\
& + I^i_{kl}\big(\mathsf{T}_i(n+m+1)\big) - I^i_{kl}\big(\mathsf{T}_i(n+m+1) - \Delta^i_j(n+m+1)\big) \\
& - I^j_{kl}\big(\mathsf{T}_i(n+m)\big) + I^j_{kl}\big(\mathsf{T}_i(n+m) - \Delta^i_j(n+m)\big) \\
\displaystyle\sum_{p=n}^{n+m+1} \widetilde{\mathcal{B}^O}{}^i_{kl}(p) = & O^i_{kl}\big(\mathsf{T}_i(n+m)\big) - O^i_{kl}\big(\mathsf{T}_i(n+m) - \Delta^i_j(n+m)\big) \\
& - O^j_{kl}\big(\mathsf{T}_i(n-1)\big) + O^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\
& + O^i_{kl}\big(\mathsf{T}_i(n+m+1)\big) - O^i_{kl}\big(\mathsf{T}_i(n+m+1) - \Delta^i_j(n+m+1)\big) \\
& - O^j_{kl}\big(\mathsf{T}_i(n+m)\big) + O^j_{kl}\big(\mathsf{T}_i(n+m) - \Delta^i_j(n+m)\big) \\
\displaystyle\sum_{p=n}^{n+m+1} \widetilde{\mathcal{B}^T}{}^i_{kl}(p) = & S^i_{kl}\big(\mathsf{T}_i(n+m)\big) - S^i_{kl}\big(\mathsf{T}_i(n+m) - \Delta^i_j(n+m)\big) \\
& + O^i_{kl}\big(\mathsf{T}_i(n+m)\big) - O^i_{kl}\big(\mathsf{T}_i(n+m) - \Delta^i_j(n+m)\big) \\
& - D^j_{kl}\big(\mathsf{T}_i(n-1)\big) + D^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\
& - I^j_{kl}\big(\mathsf{T}_i(n-1)\big) + I^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\
& + S^i_{kl}\big(\mathsf{T}_i(n+m+1)\big) - S^i_{kl}\big(\mathsf{T}_i(n+m+1) - \Delta^i_j(n+m+1)\big) \\
& + O^i_{kl}\big(\mathsf{T}_i(n+m+1)\big) - O^i_{kl}\big(\mathsf{T}_i(n+m+1) - \Delta^i_j(n+m+1)\big) \\
& - D^j_{kl}\big(\mathsf{T}_i(n+m)\big) + D^j_{kl}\big(\mathsf{T}_i(n+m) - \Delta^i_j(n+m)\big) \\
& - I^j_{kl}\big(\mathsf{T}_i(n+m)\big) + I^j_{kl}\big(\mathsf{T}_i(n+m) - \Delta^i_j(n+m)\big) ,
\end{cases} \tag{36}$$

$\forall i,j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\}, n,m \geq 0,$

$$\begin{cases}
\displaystyle\sum_{p=n}^{n+m+1} \widetilde{\mathcal{B}^I}{}^i_{kl}(p) = & I^i_{kl}\big(\mathsf{T}_i(n+m+1)\big) - I^i_{kl}\big(\mathsf{T}_i(n+m+1) - \Delta^i_j(n+m+1)\big) \\
& - I^j_{kl}\big(\mathsf{T}_i(n-1)\big) + I^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\
\displaystyle\sum_{p=n}^{n+m+1} \widetilde{\mathcal{B}^O}{}^i_{kl}(p) = & O^i_{kl}\big(\mathsf{T}_i(n+m+1)\big) - O^i_{kl}\big(\mathsf{T}_i(n+m+1) - \Delta^i_j(n+m+1)\big) \\
& - O^j_{kl}\big(\mathsf{T}_i(n-1)\big) + O^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\
\displaystyle\sum_{p=n}^{n+m+1} \widetilde{\mathcal{B}^T}{}^i_{kl}(p) = & S^i_{kl}\big(\mathsf{T}_i(n+m+1)\big) - S^i_{kl}\big(\mathsf{T}_i(n+m+1) - \Delta^i_j(n+m+1)\big) \\
& + O^i_{kl}\big(\mathsf{T}_i(n+m+1)\big) - O^i_{kl}\big(\mathsf{T}_i(n+m+1) - \Delta^i_j(n+m+1)\big) \\
& - D^j_{kl}\big(\mathsf{T}_i(n-1)\big) + D^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) \\
& - I^j_{kl}\big(\mathsf{T}_i(n-1)\big) + I^j_{kl}\big(\mathsf{T}_i(n-1) - \Delta^i_j(n-1)\big) ,
\end{cases} \tag{37}$$

since there is no counter discrepancy, so $X_{ij}^i(t)$ and $X_{ij}^j(t)$ are interchangeable and additionally we also have that $S_{kl}^i(t) + O_{kl}^i(t) = D_{kl}^i(t) + I_{kl}^i(t)$. This last formula is the statement for $m+1$ and completes the proof. □

Corollary 1. *It follows from Theorem 1 that accumulated values of link balances are bounded.*

$$\forall i, j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\}, n, M \geq 0$$
$$\begin{cases} \left| \sum_{p=n}^{n+M} \widetilde{\mathcal{B}}^{I}{}_{kl}^{i}(p) \right| \leq B_{kl} \cdot \max\left(\Delta_j^i(n-1), \Delta_j^i(n+M)\right) \\ \left| \sum_{p=n}^{n+M} \widetilde{\mathcal{B}}^{O}{}_{kl}^{i}(p) \right| \leq B_{ij} \cdot \max\left(\Delta_j^i(n-1), \Delta_j^i(n+M)\right) \\ \left| \sum_{p=n}^{n+M} \widetilde{\mathcal{B}}^{T}{}_{kl}^{i}(p) \right| \leq B_{ij} \cdot \max\left(\Delta_j^i(n-1), \Delta_j^i(n+M)\right) \end{cases} \tag{38}$$

For nodes that receive advertisements from either i or j but not both, the bound has to be loosened to

$$\forall i, j \in \mathcal{N}_i, (k,l) \in \{(i,j),(j,i)\}, n, M \geq 0$$
$$\max\left(\left| \sum_{p=n}^{n+M} \widetilde{\mathcal{B}}^{I}{}_{kl}^{i}(p) \right|, \left| \sum_{p=n}^{n+M} \widetilde{\mathcal{B}}^{O}{}_{kl}^{i}(p) \right|, \left| \sum_{p=n}^{n+M} \widetilde{\mathcal{B}}^{T}{}_{kl}^{i}(p) \right| \right) \leq B_{ij} \cdot \mathsf{P} \ . \tag{39}$$

The use of accumulated values enables small counter discrepancies to accumulate and eventually make accumulated values of balances exceed their predicted bound.

3.8 What About Mobility?

The introduction of mobility has two effects. First, nodes may join a neighborhood, which means they could begin tracking other nodes' advertisements at some advanced point in time, when the latter have already been sending and receiving traffic. Second, nodes may part from a neighborhood, which means that other nodes may track their advertisements up to some point in time.

Both effects imply that some observer may receive advertisements from observed node i only on some interval of time, say $[t_b, t_e]$, which means it can gather advertisements from i on an interval of sequence numbers $[n_b, n_e]$, with

$$n_b = \min\{n : \mathsf{T}_i(n) \geq t_b\} \ , \tag{40}$$
$$n_e = \max\{n : \mathsf{T}_i(n) \leq t_e\} \ . \tag{41}$$

According to that definition, if $n_b > n_e$ then the observer has not gathered any advertisement from i. With at least one gathered advertisement ($n_b \leq n_e$), an observer can check accumulated link balances according to (39).

For all links (i, j) and (j, i) for which the observer gathered advertisements from both i and j, it may be that it knows $\Delta^i_j(n-1)$ and $\Delta^i_j(n+N)$ for some n and $N \geq 0$ or $\Delta^j_i(m-1)$ and $\Delta^j_i(m+M)$ for some m and $M \geq 0$. This enables the observer to check bounds on accumulated values of link balances according to (38) which is more restrictive than (39).

For an observer receiving advertisements from both endpoints of links (i, j) and (j, i) on some interval of time, the condition for it to know $\Delta^i_j(n)$ for some n is to receive enough advertisements from i and j so that

$$\exists m, \quad \mathsf{T}_j(m) < \mathsf{T}_i(n) \leq \mathsf{T}_j(m+1) \ , \tag{42}$$

which means that at worst, it has to wait for $2\mathsf{P}$ units of time to know both $\Delta^i_j(n)$ and $\Delta^j_i(m)$ for some n and m. Then to acquire both $\Delta^i_j(n+1)$ and $\Delta^j_i(m+1)$, it may wait at worst another P units of time.

3.9 Grading System

The various checks performed by observers on the advertisements coming from observed nodes can be used to compute a *degree of distrust* the former maintains for the latter. For each link (i, j), when the accumulated values of some link balance exceed the bound according to (39) or (38), the level of distrust is increased on both i and j and a bias is introduced in the accumulated link balance that has triggered the increase to reset it to some "reasonable" state.

The lesser the counter discrepancies, possibly coming from a node dropping little traffic, the longer the time necessary for accumulated values of link balances to exceed their bound, at worst.

Using the grading system on each node for each other node in its two-hop neighborhood causes the malicious nodes that either drop traffic or tamper with their counters to accumulate a higher level of distrust by other nodes. The level of distrust may be used as a quality of service metric in routing protocols to enable the preference of more trusted nodes when computing routes.

4 Practical Considerations

A few conclusions drawn from initial simulations led us to consider some important aspects of the solution.

There are actual settings that make it difficult to make counters visible globally: diffusion of counters may be cumbersome and requiring each node to track balances for every other node and link in the network is seldom scalable. In such situations, it is advisable to limit the diffusion of counter values to only some hops away and make only those node that are in the vicinity of a node and link measure the balances. Note however that there is a lower bound on the hop limit to make the scheme useful: if the counters are not allowed to diffuse farther than one hop away, only the observed node's balance can be checked in addition to the balance of the link between it and the observer. If the counters diffuse to at least two hops away, then the observer can check the observed node's node balance and link balances on all the links between the latter and its one-hop neighbors.

If actual (vs. differential) counters are used, this limited scheme works well as long as the topology is not allowed to change. As soon as node mobility is introduced, two problems appear. First, when a node "enters" a neighborhood, when it begins receiving counters from a node it has not heard of before (or in a near past), its balance calculations regarding that node or its adjacent links are biased: it has no way to know about the values of counters advertised by past neighbors that have since gone away. Second, when some neighbor goes away, all other nodes in the neighborhood have to keep the values of its counters in memory, which would, with time, bring back the very problem that limited scope was aimed to solve: nodes would have to keep in memory a potentially growing number of values.

The solution lies in using a differential convention for counters: instead of advertising the actual value of a counter, a node advertises only the difference between the actual counter and its value the previous time it was advertised. Thus the bias of past neighbors does not last more that a period of counter advertisement and values of that bias do not have to be kept in memory at all.

The solution presented in Sect. 3 relies on the fact that advertisements are never lost. Although this assumption is unrealistic, there are ways to make lost advertisements sent again, so that eventually, every observer is able to perform all the checks, be it retroactively. To ensure that a node's advertisements may not be lost repeatedly for too much time (and to be able to detect nodes that leave a neighborhood), it is reasonable to put a limit on the number of consecutively lost advertisements before considering the node as departed.

Advertisements from some node i may contain a series of link advertisements from some of its neighbor j. As a matter of fact, it would be enough for the checks to be possible if i summed j's counters to be sent in one advertisement, but observers would not be able to authenticate that information anymore, so all j's link advertisements have to be included separately. To ensure that the number of j's link advertisements included in one i's advertisement is limited, a lower bound p on the time separating two consecutive advertisements has to be put. Setting p to be equal to P is not desirable in practice (as nodes would not be able to desynchronize their transmissions), but setting $\mathsf{P}/2 \leq \mathsf{p} < \mathsf{P}$ assures that no more than two j's advertisements are sent between two consecutive i's advertisements.

5 Evaluation Results

To check the accuracy and performance of the solution based on accumulated values of link balances, a batch of simulation runs has been launched using the OPNET network simulator.

5.1 Simulation Model

We created a simplified model of an ad hoc network, to avoid all the artefacts of routing and to best evaluate the solution (Fig. 1). One dropping node has four symmetric links to four 1-hop neighbors which can in turn reach 15 other nodes. During the simulation, flows are generated (exponential interarrival with $\lambda^{-1} = 30$ s) from one of the total 20 nodes to another of the remaining 19 (chosen uniformly), necessarily passing through the dropping node. Within each flow, packets have a constant size (chosen uniformly between 64 and 1500 bytes) and a rate (of constant packet interarrival) such that the total available capacity of links is never exceeded. Mobility is modeled simply by switching one of the four 1-hop neighbors on or off (exponential interarrival): when a neighbor is switched off, its counters are reset to zero and it does not participate in any flow until it is switched on again. Advertisements of counters are simulated with uniform interarrival in $[\mathsf{P} - \mathsf{J}, \mathsf{P}]$, P being the maximum period of advertisement and J being the amount of jitter in the advertisement.

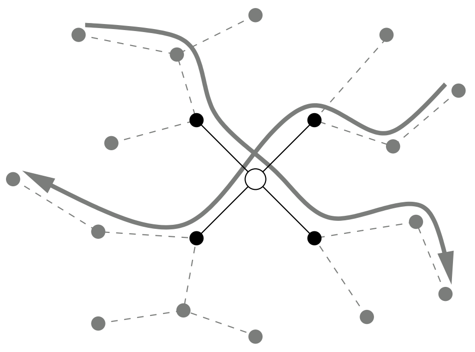

Fig. 1. Simulation model

5.2 Counter Maintenance and Advertisement

Every time a data packet gets through the links either way between the dropping node and some of its neighbors, counter maintenance is performed. Let (i, j) be the directed link on which the packet is going. First i's counters are updated as explained in Sect. 3.4 and then j's counters are updated accordingly.

Each one of the dropping node and its neighbors performs periodic counter advertisement of its own, using its own timer. On the one hand, each time a neighbor is advertising its counters, their values are added into separate secondary variables and the neighbor's counters are reset to 0. On the other hand, when the dropping node is advertising its counters, its node timed balance and all desynchronized timed balances of its links to its neighbors are calculated and logged. Balance calculations use the dropping node's counters directly, but use secondary variables into which the neighbors' counters have been accumulated (to model counter desynchronization). After balances have been calculated and logged, the dropping node's counters as well as secondary variables are reset to 0.

5.3 Influence of Mobility

To evaluate the influence of mobility on the estimation of packet loss using accumulated values of link balances, we have run two batches of simulations, varying the drop probability on the dropping node and the mean period of mobility. In the first batch, the dropping node always pretended to have retransmitted the packets it dropped, whereas in the second batch, it always pretended not to have ever received them. The simulation time of all simulations was set to one hour.

On Fig. 2, plots of the estimated total of lost packets (in fact the summed values of $\widetilde{\mathcal{B}}{\mathcal{T}}^{i}_{ij}(t)$ over the total packets to be transmitted (averaged on the four links) are shown, for drop rates of 1.0, 0.8, 0.6, 0.4, 0.2 and 0.0 from the first batch. Obviously, with larger mobility periods (less mobility), the estimations are more accurate, whereas with shorter periods, estimation tends to 1, since the counters are often reset and thus it may appear as though all the traffic has been dropped.

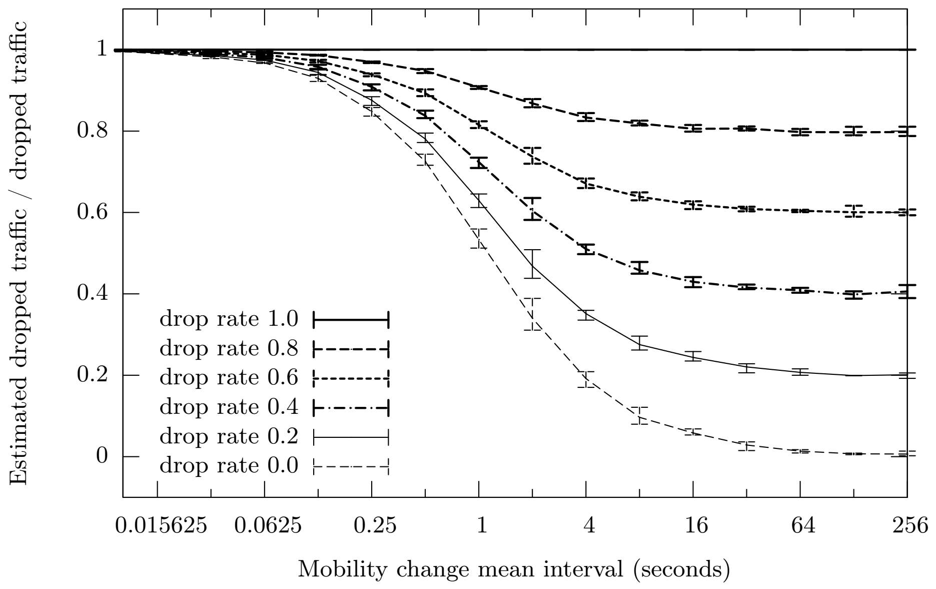

Fig. 2. Drop estimation error vs. mobility in scenario 1

On Fig. 3, the same plots are shown, but from the second batch. Here less mobility still yields better estimations, but more mobility gives negative values which modules tend to the ratio of indeed retransmitted traffic. This was to be expected, since again with high mobility, counters from the neighbors tend to zero and do not counterbalance counters of the dropping node.

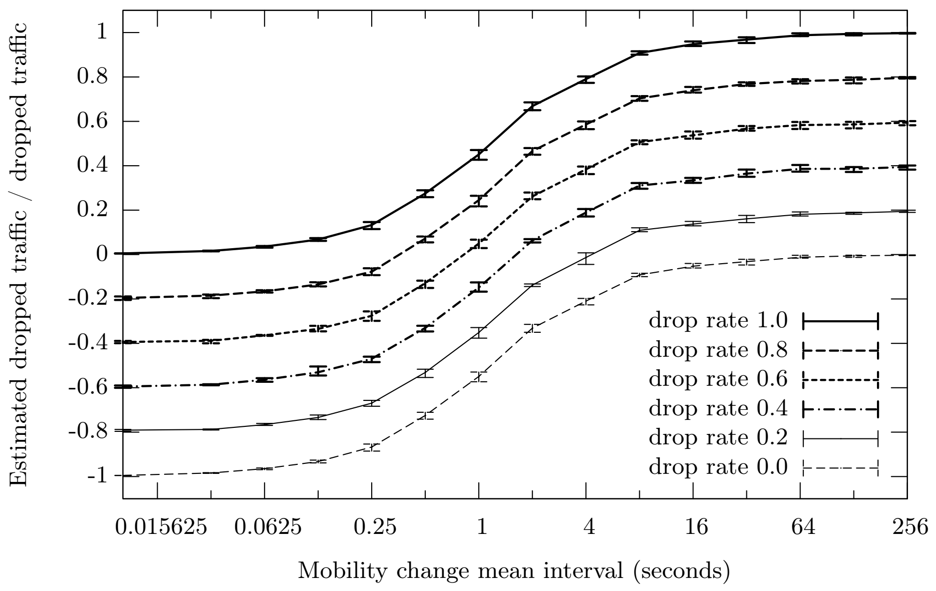

Fig. 3. Drop estimation error vs. mobility in scenario 2

6 Conclusion

The proposed solution to detect dropped traffic on nodes and links and/or counter cheating is very promising. The method integrates well into proactive protocols since it only needs packet accounting and periodic advertisement of counters.

Our current perspective is to provide even better methods of detecting counter discrepancy as well as analytical evaluations of the accuracy of the methods with respect to several parameters such as the maximum capacity of a link, node mobility, advertisement interval, etc.

References

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

2. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing. RFC 3561, IETF (2003)
3. Ogier, R., Templin, F., Lewis, M.: Topology dissemination based on reverse-path forwarding (TBRPF). RFC 3684, IETF (2004)
4. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3) (1982) 382–401
5. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking, New York, NY, USA, ACM Press (2000) 255–265
6. Buttyán, L., Hubaux, J.P.: Stimulating cooperation in self-organizing mobile ad hoc networks. MONET **8**(5) (2003) 579–592
7. Zhong, S., Chen, J., Yang, Y.R.: Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In: INFOCOM. (2003)
8. Just, M., Kranakis, E., Wan, T.: Resisting malicious packet dropping in wireless ad hoc networks. In Pierre, S., Barbeau, M., Kranakis, E., eds.: 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., Springer (2003) 151–163
9. Awerbuch, B., Holmer, D., Nita-Rotaru, C., Rubens, H.: An on-demand secure routing protocol resilient to byzantine failures. In: WiSE '02: Proceedings of the 3rd ACM workshop on Wireless security, New York, NY, USA, ACM Press (2002) 21–30
10. Mahajan, R., Rodrig, M., Wetherall, D., Zahorjan, J.: Sustaining cooperation in multi-hop wireless networks. 2nd Symposium on Networked System Design and Implementation, Boston, MA, USA, May (2005)
11. Hu, Y.C., Perrig, A., Johnson, D.B.: 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, New York, NY, USA, ACM Press (2002) 12–23
12. Avramopoulos, I., Kobayashi, H., Wang, R., Krishnamurthy, A.: Highly secure and efficient routing. In IEEE, ed.: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. Volume 1., IEEE, IEEE (2004) 208–220
13. Raffo, D., Adjih, C., Clausen, T., Mühlethaler, P.: An advanced signature system for olsr. In: SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, New York, NY, USA, ACM Press (2004) 10–16
14. Fourati, A., Agha, K.A.: A shared secret-based algorithm for securing the OLSR routing protocol. Telecommunication Systems **31**(2–3) (2006) 213–226
15. Bradley, K.A., Cheung, S., Puketza, N., Mukherjee, B., Olsson, R.A.: Detecting disruptive routers: a distributed network monitoring approach. Network, IEEE **12**(5) (1998) 50–60
16. Hughes, J.R., Aura, T., Bishop, M.: Using conservation of flow as a security mechanism in network protocols. In: IEEE Symposium on Security and Privacy. (2000) 132–131