Detecting Intentional Packet Loss in OLSR

Ignacy Gawędzki and Khaldoun Al Agha
Laboratoire de Recherche en Informatique
Université Paris-Sud 11, CNRS
Orsay, France
{i,alagha}@lri.fr

***Abstract*—The OLSR protocol, as specified in RFC 3626 [1], a proactive, table-driven routing protocol, is a distributed algorithm that relies on all participating nodes obeying to a set of rules. Those rules could be broken easily by some malicious nodes in order to gain some advantage in the network or simply disrupt its proper operation. In this paper, we present a method to detect packet loss on links and nodes and maintain a metric of distrust in the nodes and links that can be then used as a quality of service (QoS) metric by any QoS-capable extension of OLSR.**

I. Introduction

Wireless mobile ad hoc networks (MANETs) are formed by mobile or static nodes that communicate with each other using wireless network interfaces. The particularity of a MANET is that nodes that are located outside each other's direct communication range can still communicating by using intermediate nodes as relays. Hence each node is both a host and a router.

The OLSR protocol is one of three recently emerged experimental standards at the IETF, and its popularity is constantly growing in the scientific and industrial communities. Nevertheless, like the other two emerging protocols [2], [3], it relies on all the nodes willing to cooperate in order to function properly. Unfortunately, in many actual scenarios, this assumption is too strong and there are many easy ways by which malicious nodes could render the network unusable in part or in whole.

The problem of making a protocol resilient to the worst possibly behaving nodes — called *Byzantine processes* in the distributed computing literature [4] — is very difficult and there are many past and present efforts to address it.

Following this introduction, previous work in the field will be presented in Sect. II. Then in Sect. III, our method for the detection of dropped traffic on nodes and links will be detailed using graphical figures for a more intuitive experience. In Sect. IV, we will sketch a way to maintain a QoS metric based on the detection method. In Sect. V, we will provide a few remarks about the implications of the method on practical implementations and finally we will conclude in Sect. VI.

II. Previous Work

To date, the most advanced solutions aiming for Byzantine robustness are built on top of reactive, on-demand protocols [5], [6], [7] and remain pretty intrusive in the packet forwarding mechanisms. Solutions that remain applicable to proactive, table-driven protocols are the ones that, on the one hand, protect the protocol itself (as in [8], [9]) and on the other hand protect data traffic against dropping. The latter could be achieved either by encouraging nodes to cooperate, for example using a virtual currency system as in [10], [11], or by detecting packet dropping, for example by exploiting the ability of nodes to overhear each other's transmissions [12].

The problem with virtual currency systems is that nodes have to be able to retransmit in order to earn money for their own transmissions. Nodes located in the center of the network have thus a higher probability to become one's router than nodes on the border and the latter may go eventually bankrupt without having behaved badly. On the other hand, the problem with transmission overhearing is that it just stops working as soon as nodes can use multiple interfaces tuned to different channels in the same network, which is already the case with OLSR.

III. Detecting Problems

The goal of our solution is to use OLSR's HELLO messages to provide additional information to nodes to enable them to detect other nodes that drop data traffic. It is based on the principle of flow conservation which was already used by Bradley et al. in [13] in the context of static wired networks. Unfortunately, as stated in [14], Bradley's assumptions were too strong for any actual implementation. Therefore, we use relaxed assumptions and prove that loose checking of flow conservation is still possible and that it can be enough to detect dropping nodes.

A. Network Model

We assume that the network is composed of nodes and links that may appear and disappear at any time. The use of cryptographic schemes allows the originator of any message to be authenticated and the message's content to be protected against alteration and forgery. We assume that the OLSR protocol is used for routing and that every node cooperates in the protocol's operation. Malicious nodes are assumed to have too little processing and/or memory resources to break the cryptographic schemes and can only decide not to forward data packets.

B. Packet Counters

Each node i maintains a set of counters for every link to each of its 1-hop neighbor j (see Table I and Fig. 1).

The flow conservation principle could be stated as follows.

$$\forall i, \forall t, \quad \sum_{j} \left(I^{i}_{ji}(t) - O^{i}_{ij}(t) \right) = 0 \tag{1}$$

TABLE I
COUNTERS ACCORDING TO i FOR LINKS (i,j) AND (j,i).

$S^i_{ij}(t)$:	the amount of traffic originated from i and sent to j, up to instant t, according to i.
$O^i_{ij}(t)$:	the amount of traffic not originated from i and sent to j, up to instant t, according to i.
$I^i_{ij}(t)$:	the amount of traffic sent from i to j and not destined to j, up to instant t, according to i.
$D^i_{ji}(t)$:	the amount of traffic sent from j to i and destined to i, up to instant t, according to i.
$I^i_{ji}(t)$:	the amount of traffic sent from j to i and not destined to i, up to instant t, according to i.
$O^i_{ji}(t)$:	the amount of traffic sent from j to i and not originated from j, up to instant t, according to i.

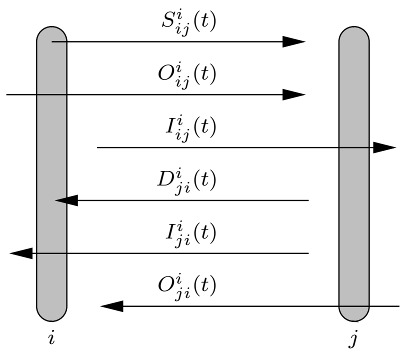

Fig. 1. Packet counters according to i.

But since i may have manipulated its counters to "hide" its dropping, the flow conservation on links has to be considered as well.

$$\forall i, \forall j, \forall t, \quad \begin{cases} I^i_{ij}(t) - I^j_{ij}(t) = 0 \\ O^i_{ij}(t) - O^j_{ij}(t) = 0 \end{cases} \tag{2}$$

Equation 2 does not take into account traffic that flows directly from i to j or from j to i. To complete the flow conservation on links, we have to state it for the overall traffic flowing on a link.

$$\forall i, \forall j, \forall t, \quad S^i_{ij}(t) + O^i_{ij} - D^j_{ij}(t) - I^j_{ij}(t) = 0 \tag{3}$$

For the sake of simplicity, let $T^i_{ij}(t)$ and $T^j_{ij}(t)$ respectively stand for $S^i_{ij}(t)+O^i_{ij}(t)$ and $D^j_{ij}(t)+I^j_{ij}(t)$, in the following.

C. Advertising Counters

Using HELLO messages of the OLSR protocol, along each link advertisement, the counters of that link are advertised too. For reasons explained in Sect. V, differential values of counters are used instead of absolute ones.

For each counter $X^k_{ij}(t)$, let $\dot{X}^k_{ij}(n)$ be its differential value, i.e. the difference between the absolute counter's value at the instant HELLO with sequence number n is sent and at the instant HELLO with sequence number $n-1$ is sent. In each HELLO message number n sent by node $k \in \{i,j\}$, there are advertisements about some links (i,j) and (j,i) with the respective counters $\dot{X}^k_{ij}(n)$ and $\dot{X}^k_{ji}(n)$. In addition, there are any differential counters received in advertisements from the other link's endpoint.

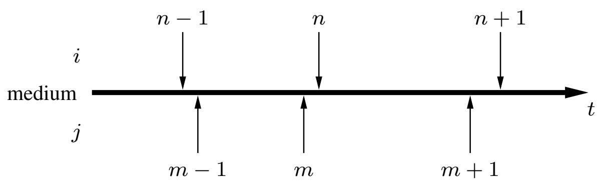

Fig. 2. Node i and j advertisements.

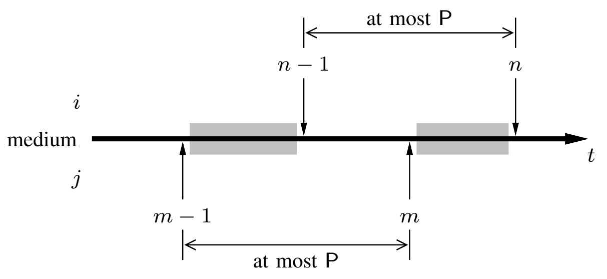

Fig. 3. Link balance bound.

Let $\mathcal{S}^i_j(n)$ be set of sequence numbers of j's HELLO messages received by i between its transmission of HELLO message number $n-1$ and n. For example, on Fig. 2, we have that $\mathcal{S}^i_j(n) = \{m-1, m\}$, $\mathcal{S}^i_j(n+1) = \{m+1\}$, $\mathcal{S}^j_i(m) = \emptyset$ and $\mathcal{S}^j_i(m+1) = \{n\}$.

D. Checking Balances

Each time some node receives i's advertisement number n containing all differential counters regarding links (i,j) and (j,i), the node balance $\mathcal{B}_i(n)$ at i can be checked.

$$\forall i, \forall n, \quad \mathcal{B}_i(n) = \sum_j \left(\dot{I}^i_{ji}(n) - \dot{O}^i_{ij}(n) \right) \tag{4}$$

In addition to that, link balances for each advertised links should be checked too.

$$\forall i, \forall j, \forall n, \forall (k,l) \in \{(i,j),(j,i)\}, \\ \begin{cases} \widetilde{\mathcal{B}^I}^i_{kl}(n) = \dot{I}^i_{kl}(n) - \sum\limits_{m \in \mathcal{S}^i_j(n)} \dot{I}^j_{kl}(m) \\ \widetilde{\mathcal{B}^O}^i_{kl}(n) = \dot{O}^i_{kl}(n) - \sum\limits_{m \in \mathcal{S}^i_j(n)} \dot{O}^j_{kl}(m) \\ \widetilde{\mathcal{B}^T}^i_{kl}(n) = \dot{T}^i_{kl}(n) - \sum\limits_{m \in \mathcal{S}^i_j(n)} \dot{T}^j_{kl}(m) \end{cases} \tag{5}$$

While a non-zero node balance $\mathcal{B}_i(n)$ about i would directly accuse i of some fault (either it dropped packets, or it wrongly manipulated its counters), a non-zero link balance $\widetilde{\mathcal{B}^I}^i_{kl}(n)$, $\widetilde{\mathcal{B}^O}^i_{kl}(n)$ or $\widetilde{\mathcal{B}^T}^i_{kl}(n)$ is very likely and does not necessarily indicate a discrepancy between the nodes' counters. This is because the counters compared are not taken from the same interval of time. What we know is that if two successive HELLO messages are sent at most P seconds apart, and if B_{kl} is the maximum possible throughput on link (k,l), then the absolute value of link balances cannot exceed $\mathsf{P} \cdot \mathsf{B}_{kl}$ (see Fig. 3).

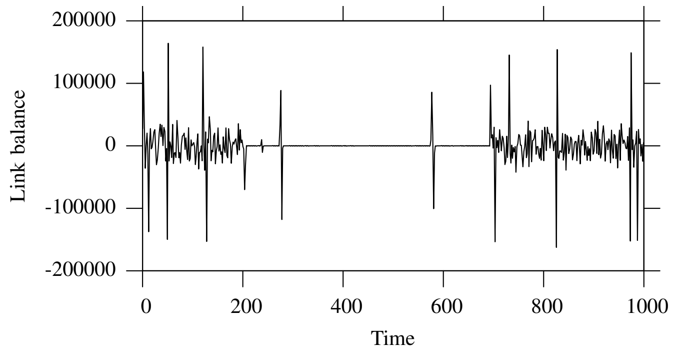

Fig. 4. Link balance value with drop probability of 0.0, $\mathsf{P} = 2$ and $\mathsf{B}_{kl} = 100000$. The absolute value of the calculated balance is never exceeding $\mathsf{P} \cdot \mathsf{B}_{kl}$.

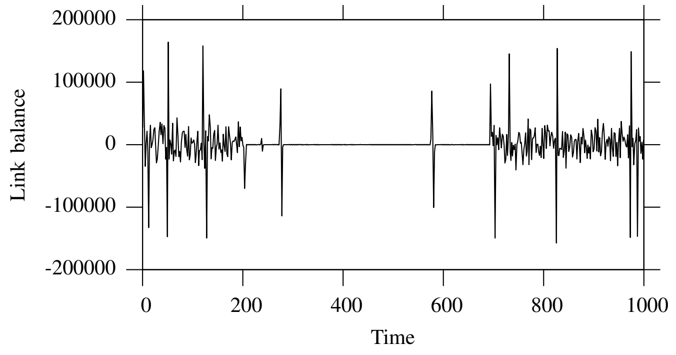

Fig. 5. Link balance value with drop probability of 0.01, $\mathsf{P} = 2$ and $\mathsf{B}_{kl} = 100000$. The absolute value of the calculated balance is rarely exceeding $\mathsf{P} \cdot \mathsf{B}_{kl}$.

As shown on Fig. 4 and 5, a low loss rate is almost not detectable when only the direct value of a balance is considered.

Instead of looking directly at the value of a balance, its accumulation over several advertisements is more interesting. As depicted on Fig. 6, the sum of successive balance values is bounded if no packet is lost. On the other hand, when there is even some low non-zero packet loss rate, the accumulated balance values grow (see Fig. 7).

It is easily provable that in absence of actual counter

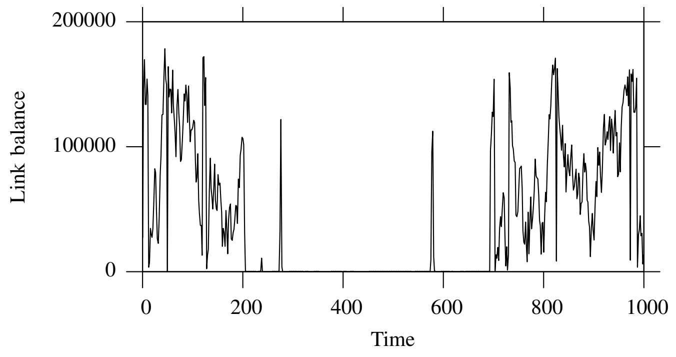

Fig. 6. Accumulated link balance value with drop probability of 0.0, $\mathsf{P} = 2$ and $\mathsf{B}_{kl} = 100000$. The accumulated value never exceeds $\mathsf{P} \cdot \mathsf{B}_{kl}$.

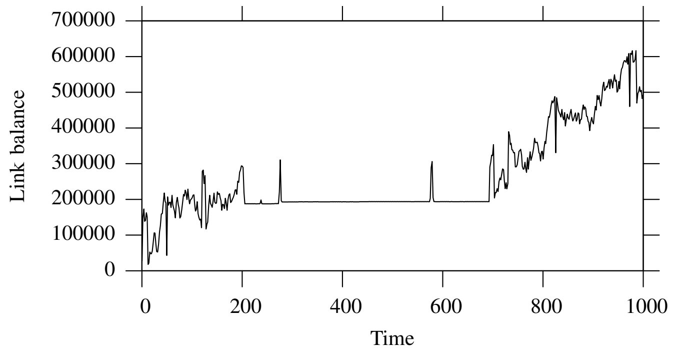

Fig. 7. Accumulated link balance value with drop probability of 0.01, $\mathsf{P} = 2$ and $\mathsf{B}_{kl} = 100000$. The accumulated value quickly exceeds $\mathsf{P} \cdot \mathsf{B}_{kl}$.

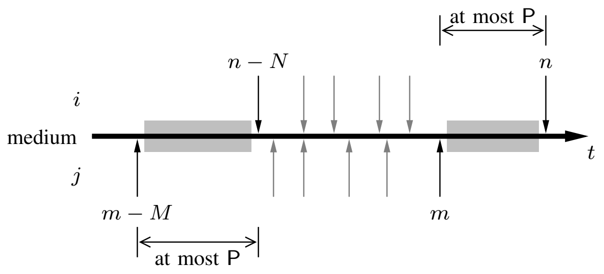

Fig. 8. Accumulated link balance bound.

discrepancy (that may otherwise indicate packet loss or faulty counter manipulation), the accumulated values of a link balance cannot grow larger than the same bound of $\mathsf{P} \cdot \mathsf{B}_{kl}$ as for absolute value of link balances. Since the proof would require the introduction of a fair load of formalism, we provide only a visual hint in Fig. 8. The idea is that because the accumulated values of a balance are never shifted more that the maximum advertisement interval, its value is bounded accordingly.

E. Window of Observation

The influence of node mobility on the previously introduced method is that some *observer* nodes (the ones that perform the balance checks) may not be able to accumulate some link's balances for the whole interval of time during which that link is active. The observer may arrive late, when there already has been traffic flowing on the link, or depart early, as the traffic continues to flow for some time.

The accumulated link balance values can then be observed either on the interval since the arrival of the observer, or on some sliding window of time. Then the size of that interval of observation is a parameter that affects the amount of detectable packet loss. The longer the observation interval, the lower the minimum amount of lost packets that can be observed.

Looking at Fig. 6 and 7, one can see that in absence of actual traffic, the accumulated balance values sort of "stabilize". If the observation interval starts at a point in time when no traffic was flowing on the link, it easy to show that the accumulated values are positive and in absence of packet loss, drop down to zero as soon as there is no actual traffic.

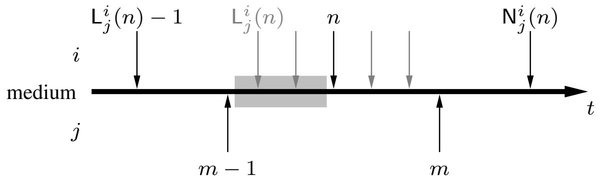

Fig. 9. Bounding the actual flow.

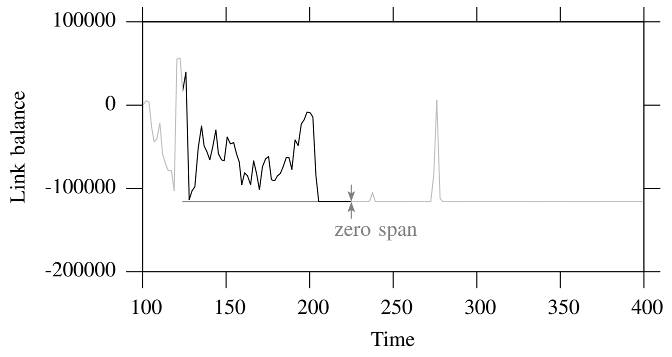

Fig. 10. Span of accumulated balance values with drop rate 0.0. The span drops to zero in absence of traffic.

More generally, looking at Fig. 8, we can suppose that the interval of observation is starting at i's advertisement number $n - N$ and ends at i's advertisement number n. We already know that the value of accumulated balance is exactly the amount of traffic that flowed in the considered class (I, O or T, depending on the type of balance) between j's last advertisement (with number m) and i's advertisement number n (the rightmost gray rectangle on Fig. 8) minus the amount of traffic that flowed in that class between j's $(m - M)$th advertisement (i.e. the last before i's $(n - N)$th) and i's $(n - N)$th (the leftmost gray rectangle on Fig. 8).

Instead of considering the maximum possible difference between these two quantities as a bound to the accumulated balance values, some advertised counters may be used. Let be the two following sequence numbers.

$$\mathsf{L}_j^i(n) = \max\left\{m : m \leq n \wedge \mathcal{S}_j^i(m) \neq \emptyset\right\} \tag{6}$$

$$\mathsf{N}_j^i(n) = \min\left\{m : m > n \wedge \mathcal{S}_j^i(m) \neq \emptyset\right\} \tag{7}$$

As shown on Fig. 9, the amount of actual traffic in the class of interest is no more than the maximum of what i claims has flowed between its advertisements $\mathsf{L}_j^i(n) - 1$ and n, and what j claims has flowed between its advertisements $m - 1$ and m. The former claim is obtained by accumulating counter values from i's advertisements between $\mathsf{L}_j^i(n)$ and n, whereas the latter is obtained by getting i's advertisement $\mathsf{N}_j^i(n)$. Since a malicious node could advertise very large differential counters, the bound on the traffic having possibly flowed using this technique must be saturated to $\mathsf{P} \cdot \mathsf{B}_{kl}$.

Now of course, these tighter bounds on the actual flow could be used to check accumulated balance values on the observation window, but we have an even better way. As we already noticed, in absence of actual traffic, the accumulated

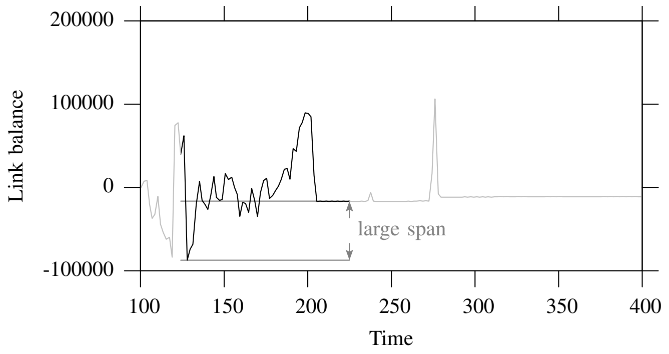

Fig. 11. Span of accumulated balance values with drop rate 0.01. The span remains large even in absence of traffic.

balance values drop down to a minimum that should be zero in case the observation window also starts at a time when there was no actual traffic on the link. The idea is then to maintain the minimums of accumulated balances on the observation window and consider the difference between the current values, as of i's advertisement number n, of accumulated balance and that minimum. We call this difference the *span* of the accumulated balance values. It is then easy to show that the span is bounded by the estimated bound on the traffic that flowed between last j's advertisement as of i's nth advertisement and that i's nth advertisement.

To check the accumulated balances in this way on i's nth advertisement, an observer has to gather up to i's $\mathsf{N}_j^i(n)$th advertisement, which should not arrive later than in $2\mathsf{P}$.

On Fig. 10, the span is zero because with a zero loss rate, the value of the accumulated counters in absence of traffic is minimizing all previous values. On the other hand, on Fig. 11, a non-zero loss rate inevitably increases the span even in absence of traffic.

Of course, a malicious node could advertise high differential counter values, in order to maintain a comfortable higher bound on the span, but then, if there is not actual traffic flowing, the other endpoint's counters are likely to be low, hence the balance value high and eventually a span exceeding the higher bound.

IV. Distrust Metric

The use of the method described in Sect. III-E allows observers to determine the presence of counter discrepancy on links. Along with the expression of node balance in (4), an observer can maintain a degree of distrust in each of the nodes it observes.

During all the interval of time in which a node can observe another node, it maintains a degree of distrust in that node. Initially the degree is 0, meaning there is no reason to distrust that node. Each time that the distrust in that node is raised, it is set to 1 if it was zero or it is doubled otherwise. The distrust then decreases linearly with time, possibly with some constant factor (the larger that factor, the longer the "recovery").

Each time an advertisement from that node is received and the node balance fails, the degree of that node is raised. If at least one of the link balances is exceeding its bound, then both of that link's endpoints' degree is raised, except when one of the endpoints is the observer itself, in which case only the other endpoint's degree is raised. When a link balance exceeds its bound, the size of the window of observation has to be reset to one.

V. Practical Considerations

In the case of OLSR, the degree of distrust is intended to be used as a QoS metric when selecting MPRs. The MPR selection heuristic could be easily adapted to make nodes prefer to select neighbors that have a lower degree of distrust.

In the basic formulation provided here, the counters are not diffused farther than two hops away, using the HELLO mechanism. This forbids nodes to maintain a degree of distrust outside of this 2-hop neighborhood. This is both a good point, since it limits somewhat the amount of processing and memory resources consumed for that task, and a bad one, since the degree of distrust cannot be used as a QoS metric for route calculation. Although the authors acknowledge the existence of various schemes to diffuse observers' degree of distrust in observed nodes to the whole network, the discussion of whether the use of such a scheme is advisable is outside the scope of this paper.

The use of differential counters appeared very important in early study of this approach. When nodes depart from a neighborhood, the observers would have to keep that nodes' absolute counters in memory to avoid sudden jumps in the node balances. On the other hand, when a node arrives in a neighborhood, it does not know of all the nodes and their counters that have been there in the past, which is another source of jumps. The use of differential counters avoids these problems, as each observer builds its own view of the absolute counters using accumulated balances values.

As for the case of lost HELLO messages, observers can keep track of their sequence numbers and use a signaling system akin to negative acknowledgements to request nodes to retransmit lost advertisements with their next transmission. During the interval of time in which an observer waits for the next advertisement batch to come, it is itself given a delay before its own balances are checked by other observers. Since nodes are required to send their HELLO messages with a maximal period, a malicious node cannot request too much retransmissions successively, as it should consider the observed node as lost if more than a few consecutive HELLOs are lost.

Lastly, since nodes are required to include all their neighbors' advertisements received since last advertisement in the current one, nodes should be forbidden to send advertisements too often, by enforcing a minimal period between HELLO transmissions. Advertisements not conforming to that requirement could be dropped or their originator's degree raised.

VI. Conclusion and Perspectives

The solution has been presented in a way more visual than formal, hopefully making it easier to understand, but at the expense of the proofs of the claims. A formal description of the method along with proofs is in the works and available upon request. We are also working on using simulations to quantify the impact of the method in terms of processing time, memory consumption and control message overhead.

Our next goal is to show how the use of our method could increase the network's performance in the presence of one and then more malicious nodes that use some well-defined strategies.

References

[1] T. Clausen and P. Jacquet, "Optimized link state routing (OLSR) protocol," http://www.ietf.org/rfc/rfc3626.txt, IETF, RFC 3626, October 2003.

[2] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," http://www.ietf.org/rfc/rfc3561.txt, IETF, RFC 3561, July 2003.

[3] R. Ogier, F. Templin, and M. Lewis, "Topology dissemination based on reverse-path forwarding (TBRPF)," http://www.ietf.org/rfc/rfc3684.txt, IETF, RFC 3684, February 2004.

[4] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[5] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "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, pp. 21–30.

[6] Y.-C. Hu, A. Perrig, and D. 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*. New York, NY, USA: ACM Press, 2002, pp. 12–23.

[7] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy, "Highly secure and efficient routing," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, IEEE, Ed., vol. 1, IEEE. IEEE, March 2004, pp. 208–220.

[8] D. Raffo, C. Adjih, T. Clausen, and P. Mühlethaler, "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, pp. 10–16.

[9] A. Fourati and K. A. Agha, "A shared secret-based algorithm for securing the OLSR routing protocol," *Telecommunication Systems*, vol. 31, no. 2–3, pp. 213–226, March 2006.

[10] L. Buttyán and J.-P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *MONET*, vol. 8, no. 5, pp. 579–592, 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1025146013151

[11] S. Zhong, J. Chen, and Y. R. Yang, "Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks," in *INFOCOM*, 2003. [Online]. Available: http://www.ieee-infocom.org/2003/papers/48_04.PDF

[12] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "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, pp. 255–265.

[13] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson, "Detecting disruptive routers: a distributed network monitoring approach," *Network, IEEE*, vol. 12, no. 5, pp. 50–60, Sep/Oct 1998.

[14] J. R. Hughes, T. Aura, and M. Bishop, "Using conservation of flow as a security mechanism in network protocols," in *IEEE Symposium on Security and Privacy*, 2000, pp. 132–131. [Online]. Available: http://computer.org/proceedings/s%26p/0665/06650132abs.htm