

Scalable Exchange of Packet Counters in OLSR

Ignacy Gawędzki and Khaldoun Al Agha

Abstract The HELLO messages of the OLSR protocol can be used as a way to exchange packet counters for each advertised link between the originating node and its neighbors. These counters are aimed to enable nodes receiving the HELLO messages to check that the property of flow conservation is verified on the originating node. The addition of this information incurs a significant overhead that could prevent the node from being able to advertise all its links in one HELLO message. Although plain OLSR has been designed to deal with partial information in its control messages, enabling the verification of the flow conservation property is not trivial. In this paper, we present a way to perform partial advertisement of packet counters and complete verification of the property of flow conservation that remains scalable. Simulation results show that in practical conditions, the use of the method incurs a limited overhead that makes it very acceptable.

1 Introduction

The technique presented by Gawędzki and Al Agha [1] (referenced as *the method* in the following) aims to detect data packet loss in mobile ad hoc networks by making each node verify the principle of flow conservation at each of its neighbors. To achieve that, every node advertises a series of differential packet counters for every link between itself and one of its neighbors. Upon reception of a counter advertisement from a neighbor, a node must be able to perform two kinds of checks: a node balance check for the originating node and a link balance check for each of the advertised links. The problem

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

here is that to be able to perform a node balance check, the counters for every link from that node are required. Moreover, all these counters have to be taking account of the same time interval, otherwise the node balance is impossible to check. As long as all the counters for all the links of the originating node are advertised at once in each periodic control message, the node balance check can be performed. Unfortunately, given the limited capacity of control messages in most practical applications, an increasing network density directly implies more and more links between a node and its neighbors, hence the impossibility, at some point, to pack all the counters in one advertisement.

The OLSR protocol [2] uses exchange of HELLO control messages to enable nodes to detect their neighbors and maintain an up-to-date set of links and neighbors. In the case where there are too many links to be advertised in one HELLO message, the protocol allows only part of the links to be advertised, provided that all the links are advertised often enough for the corresponding information not to expire at its neighbors. Depending on the different timing parameters (HELLO emission interval, default validity time, etc), a node may simply generate partial HELLO messages or additionally decrease the generation interval, to ensure no link will be advertised less often than the default validity interval.

The method does not, as such, support partial advertisements, so it needs to be adapted to be fully applied on the OLSR protocol. In the following section, a summary of the method and its requirements are provided. The adapted method is presented in section 3 and the correct way to make authentication of control messages possible in section 4. The impact of the adapted method on the performance of the network is studied in section 5 and lastly a conclusion is drawn in section 6.

2 Checking Flow Conservation

The method involves the maintaining by a node of six packet counters for every link between itself and its neighbors. The counter values are to be advertised periodically in control messages bearing a sequence number and the counters themselves are to be reset to zero right after their values have been advertised. Thus let us note $V_j^i(n)$ the set of counter values of the directed links (i, j) and (j, i) (which we will both refer to as undirected link $\{i, j\}$ in the following) transmitted by node i in its advertisement with sequence number n . Let us note $T_i(n)$ the instant in time at which i sends its advertisement number n . The sequence counter of i , noted $S_i(t)$ is the sequence number of the latest transmitted advertisement by i as of instant t . In fact, the sequence counter is incremented right before each advertisement is transmitted with the new value. Therefore, the following properties always hold:

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

Let $L_j^i(n)$ be what we will be calling the *link advertisement* and defined as follows:

$$\forall i, j, \quad \forall n \in \mathbb{N}, \quad L_j^i(n) = (n, V_j^i(n), R_j^i(n)) . \quad (2)$$

Here, $V_j^i(n)$ is the set of counter values for link $\{i, j\}$ taken over the interval of time $[\mathsf{T}_i(n-1), \mathsf{T}_i(n))$ and $R_j^i(n)$ is the set of reverse link advertisements, i.e. the set defined as follows:

$$\forall i, j, \quad \forall n \in \mathbb{N}, \quad R_j^i(n) = \bigcup_{m \in \mathcal{S}_j^i(n)} \{V_i^j(m)\} , \quad (3)$$

where $\mathcal{S}_j^i(n)$ is defined as follows:

$$\forall i, j, \quad \forall n \in \mathbb{N}, \quad \mathcal{S}_j^i(n) = \{m : \mathsf{T}_i(n-1) \leq \mathsf{T}_j(m) < \mathsf{T}_i(n)\} . \quad (4)$$

So for each link, a node collects the sets of counter values regarding that link that it receives from its other endpoint. The set of collected counter values is then put in its own advertisements of links, in addition to its own counter values.

Finally, let $A^i(n)$ be the set of link advertisements with sequence number n generated by i :

$$\forall i, \quad \forall n \in \mathbb{N}, \quad A^i(n) = \bigcup_j \{L_j^i(n)\} . \quad (5)$$

When some other node receives a set of link advertisements from another node, it has that node's counters plus all its neighbors' counters for each link between that node and its neighbors. Let P be the upper bound on the time interval between two successive advertisements of any existing link between a node and a neighbor.

To perform the verification of the property of conservation of flow at the originating node, a node receiving the set of link advertisements has to calculate a *node balance* for the originating node and a *link balance* for each link between that node and its neighbors. The calculation of the link balances for a link $\{i, j\}$ based on i 's n th advertisement, noted $\tilde{\mathcal{B}}_{ij}^i(n)$ and $\tilde{\mathcal{B}}_{ji}^i(n)$, require only $L_j^i(n)$, whereas the $V_j^i(n)$ for every i 's neighbor j are needed to calculate i 's node balance, noted $\mathcal{B}_i(n)$. Fig. 1 illustrates the operation of this method in a situation where all the link advertisements do fit inside a control packet.

For instance advertisement number n contains the amount of all the traffic that flowed on the links for all the four neighbors j_0 to j_3 , since i 's $(n-1)$ th

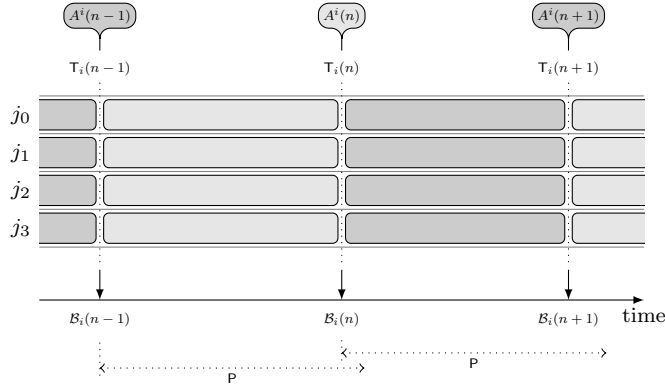


Fig. 1 Total advertisements at regular intervals. All the link advertisements fit inside each control packet.

advertisement set was sent. As soon as that advertisement is received by the neighbors of i , all the link balances $\tilde{\mathcal{B}}^{x_{ij}^i}(n)$ and $\tilde{\mathcal{B}}^{x_{ji}^i}(n)$ can be computed in addition to i 's node balance $\mathcal{B}_i(n)$.

3 Partial Advertisements

In this section, we first state the problem at hand, next study possible approaches and finally present our solution.

3.1 Partial HELLO Messages in OLSR

The HELLO messages in the OLSR protocol are exchanged periodically between direct (i.e. one hop away) neighbors in order to allow nodes to acquire their local neighborhood and detect symmetric links. As this protocol was designed to be effective in dense networks, it is allowing the advertisement of only part of the links in its HELLO messages in case all of them do not fit at once. The way to achieve that is to declare a validity period for the information contained in each HELLO message that is sufficiently long not to expire before the next time that same information is advertised. Consequently, the HELLO generation period has to be properly modulated and the links about which information is to be put in next HELLO message have to be properly chosen, preferably among the ones that are waiting to be advertised the longest.

As it is the case for the acquiring of local neighborhood using partial HELLO messages, it should also still be possible to verify flow conservation in the case of partial link advertisements. Unfortunately, if sets of counters for only part of the links are provided in each control message, as illustrated on Fig. 2, where only two link advertisements fit into a single control message, the computation of node balance is impossible. Here, as soon as advertisement set number n is received by the neighbors of i , the link balances of only two out of four can be computed (for links $\{i, j_0\}$ and $\{i, j_1\}$) and i 's node balance $\mathcal{B}_i(n)$ cannot be computed, since it would require a set of $V_j^i(n)$ for all the j s accounting the traffic on the same interval of time. In fact, with this naive approach, i 's node balance is never computed at all.

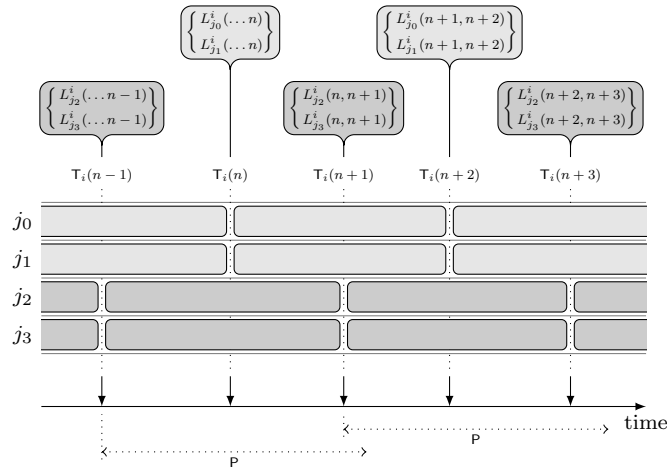


Fig. 2 Naive partial advertisements at regular intervals. Not all the link advertisements fit inside each control packet, hence the impossibility to compute the node balance. Here notations like $L_j^i(n, \dots, m)$ are a shorthand for $L_j^i(n) + \dots + L_j^i(m)$.

A straightforward approach to solve the problem would be to simply transmit at instant $T_i(n)$ a burst of as many control messages as necessary to advertise all the links and thus allow other nodes to receive all the necessary $V_j^i(n)$ they need to compute $\mathcal{B}_i(n)$. But this approach is not acceptable in practice, just for the same reason a burst of HELLO messages is not desirable to advertise all the links: the control messages should have as little impact on the medium as possible. Therefore, we have sought more clever ways to enable partial advertisements while still enabling nodes to calculate both link and node balances.

3.2 Longer Node Balance Interval Method

In section 2, it is assumed that all link advertisements have the same sequence number that is incremented each time an advertisement set is sent. This is necessary to ensure that all the local counters account for data traffic that flowed during the same interval of time. So instead of taking the values of the counters to be advertised right before the advertisement containing them is sent, they could be simply retrieved all at once at regular intervals and sent in advertisements at some later instants. Once all the link advertisements have been sent, new counter values can be retrieved. The process is illustrated on Fig. 3 where again only two link advertisements fit into a single control message. In fact, whether the interval of counter values retrieval is lengthened (and so the bound P as well) or instead the interval of transmission of control messages is shortened makes no practical difference, apart from the fact that node and link balances may be calculated more or less often.

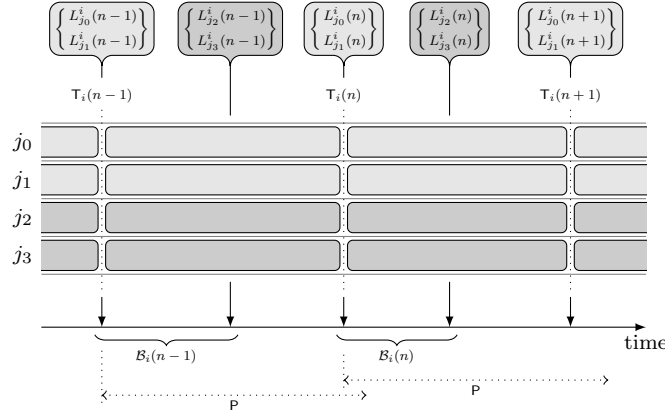


Fig. 3 Partial advertisements at regular intervals. Not all the link advertisements fit inside each control packet, but counter values are retrieved all at once, thus allowing the calculation of the node balance.

Here, it appears that nodes need to gather i 's advertisements two by two, in order to recover the necessary information to compute i 's node balance. So for instance the calculation of $\mathcal{B}_i(n)$ is possible only at some later instant between $T_i(n)$ and $T_i(n+1)$.

There are downsides to that approach, though. One of them is that it delays the calculation of balances. For node balance, the calculation is delayed until all the required link advertisements are received, whereas for link balances, it is delayed until the very link advertisements are sent.

3.3 Double Counter Values Method

There is an alternative to the approach of section 3.2 which has the advantage of not delaying the calculation of link advertisements at the cost of an increased overhead. The idea is that of maintaining, for each counter, an auxiliary variable: at some properly chosen times, the current values of the counters are assigned to their respective auxiliary variables and the counters are reset to zero; the advertisements contain thus the values stored in the auxiliary variables and the current values of the counters. This process is illustrated in Fig. 4 where we suppose that only one link advertisement (i.e. two counter values) fits into a single control message.

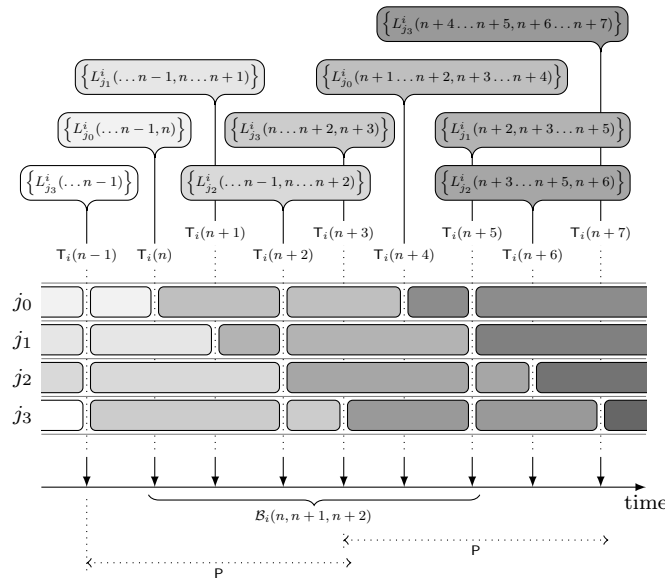


Fig. 4 Double partial advertisements. Not all the link advertisements fit inside each control packet, but double values allow the calculation of the node balance. Here, the notation $L_j^i(n \dots m, p \dots q)$ stands for $(L_j^i(n) + \dots + L_j^i(m), L_j^i(p) + \dots + L_j^i(q))$, whereas $\mathcal{B}_i(n, \dots, m)$ stands for $\mathcal{B}_i(n) + \dots + \mathcal{B}_i(m)$.

The two advertised counter values are to be simply summed for the calculation of the link balance, in order to reconstruct the required value. As for the node balance, another combination has to take place, in order to perform the node balance calculation over the interval of time separating two simultaneous assignments to auxiliary variables. For example, on Fig. 4, those assignments happen at $T_i(n-1)$, $T_i(n+2)$ and $T_i(n+5)$. Consequently, at $T_i(n+5)$, nodes that have gathered advertisements with sequence numbers from n to $n+5$ are able to compute $\mathcal{B}_i(n, n+1, n+2) = \mathcal{B}_i(n) + \mathcal{B}_i(n+1) + \mathcal{B}_i(n+2)$.

Formally, we have to redefine $L_j^i(n)$ as follows:

$$\forall i, j, \quad \forall n \in \mathbb{N}, \quad L_j^i(n) = (n, V_j'^i(n), V_j^i(n), R_j^i(n)) \quad , \quad (6)$$

where $V_j'^i(n)$ is the tuple of values of auxiliary variables associated with the counters of which values are the tuple $V_j^i(n)$.

The details of the algorithm used to generate and manage the auxiliary variables is given in Fig. 5. Note that when a new neighbor j is to be advertised, a pair $(j, -\infty)$ is added to \mathcal{M}_i .

Input: current node i
current sequence number n_C
sequence number of last switch n_S
set of current neighbors \mathcal{N}_i
set of marked neighbors \mathcal{M}_i (of (neighbor, last sequence number) pairs)
set of counters $\mathcal{V}^i = \cup_{j \in \mathcal{N}_i} \{V_j^i\}$
set of auxiliary variables $\mathcal{V}'^i = \cup_{j \in \mathcal{N}_i} \{V_j'^i\}$
set of reverse link advertisements $\mathcal{R}^i = \cup_{j \in \mathcal{N}_i} \{R_j^i\}$

Output: new advertisement set A^i

```

1   $A^i \leftarrow \emptyset$ 
2  while  $\exists(j, n) \in \mathcal{M}_i : n < n_C$  do
3      let  $(j, n) \in \mathcal{M}_i : \forall(j', n') \in \mathcal{M}_i, n \leq n'$ 
4       $L_j^i = (n_C, V_j'^i, V_j^i, R_j^i)$ 
5       $V_j'^i \leftarrow 0_V$ 
6       $V_j^i \leftarrow 0_V$ 
7       $R_j^i \leftarrow 0_R$ 
8      if  $\text{size}(A^i) + \text{size}(L_j^i) > \text{maximum size}$  then return
9       $\mathcal{M}_i \leftarrow (\mathcal{M}_i \setminus \{(j, n)\}) \cup \{(j, n_C)\}$ 
10      $A^i \leftarrow A^i \cup L_j^i$ 
11     if  $\forall(j, n) \in \mathcal{M}_i, n \geq n_S$  then
12         foreach  $j \in \mathcal{N}_i$  do
13              $V_j'^i \leftarrow V_j^i$ 
14              $V_j^i \leftarrow 0_V$ 
15          $n_S \leftarrow n_C$ 

```

Fig. 5 Generating advertisements. The values 0_V and 0_R are simply sets of counter values all set to zero.

The big advantage of this method is that link balances are calculated as usual, with fresh values of counters retrieved an instant before and no additional delay. The delay induced in node balance calculation is not so much an inconvenience as it would be in the case of link balances. Indeed, as pointed out about the method, a non-zero link balance tells nothing by itself and successive values of link balances have to be accumulated in order to perform the check, whereas a non-zero node balance implies lost data traffic

at that node directly. Therefore, it seems preferable to have a delay induced on node balance rather than on link balances.

In the trivial case where all advertisements fit inside the control message, the node and link balances could all be computed for each successive sequence numbers. With double counter values, for a given sequence number, only some link balances can be computed and each one of them is spanning several sequence numbers, whereas only a sum of several successive node balances can be retrieved.

4 Authentication of Messages

The method relies, among other assumptions, on the ability of nodes to authenticate the source of information contained in control messages. Therefore, the necessary infrastructure is assumed to be deployed that enables each node to verify the authenticity and integrity of any digitally signed piece of information. Since some information contained in HELLO messages is of second-hand origin, a simple signing of the whole message is not enough to verify all of its content. The details of how to protect the content of plain OLSR information in HELLO messages, though, is outside the scope of this paper.

In the present section, we are dealing with the details of the necessary elements for the verification of the additional information needed by the double counter values method of section 3.3.

Obviously, the authenticity and integrity of all first-hand information is verified by a common signature for the whole message. The second-hand information here are the sets of reverse link advertisements contained in each link advertisement. Any node receiving i 's n th advertisement set $A^i(n)$ has to be able to verify, for every link to a neighbor j , the link advertisement $L_j^i(n)$ which in turn contains, among other things, the reverse link advertisement set $R_j^i(n)$ of second-hand origin.

To enable a node receiving $A^i(n)$ to verify every $R_j^i(n)$ it contains, the latter have to be signed by j . In other words, instead of $R_j^i(n)$, node i should put $\bar{R}_j^i(n)$, the signed version of $R_j^i(n)$ (defined below). This means in turn that i has to be provided the signature by j itself. The trick here is to require each node i to put $\bar{L}_j^i(n)$, the signed version of $L_j^i(n)$, instead of its unsigned counterpart in its advertisement set:

$$\forall i, j, \quad \forall n \in \mathbb{N}, \quad \bar{L}_j^i(n) = (n, V_j^{ii}(n), V_j^i(n), S^i(V_j^{ii}(n)), \bar{R}_j^i(n)) \quad , \quad (7)$$

where $V_j^{ii}(n)$ is the sum, component by component, of $V_j^{ii}(n)$ and $V_j^i(n)$. Finally, the set of signed reverse link advertisements is defined as follows:

$$\forall i, j, \quad \forall n \in \mathbb{N}, \quad \bar{R}_j^i(n) = \bigcup_{m \in S_j^i(n)} \left\{ \left(V_i^{jj}(m), S^j \left(V_i^{jj}(m) \right) \right) \right\} \quad , \quad (8)$$

where S^j is the function used by j to generate a signature.

Since the separate tuples of values of auxiliary variables and counters are not needed for reverse link advertisements, they are summed component by component by the receiving node, while the necessary signature is provided directly by the originating node.

5 Performance Impact

To evaluate the performance impact of the double values method, we have run a series of simulations using the OPNET v12.0 software.

5.1 Simulation Model

The model is composed of mobile nodes moving according to the Random Ad hoc Mobility model [3], on a square area. Each node moves with a constant heading and velocity during an exponentially chosen interval of time. At the end of every interval, a new heading and a new velocity are uniformly chosen for the next period. The borders of the area in which the mobiles move are elastic, i.e. mobiles bounce off when they hit them. As the MAC and physical layers, an IEEE 802.11-style MAC with DS PHY timings [4] has been used. Nodes implement the OLSR protocol augmented with packet counters and the double counter values method from section 3.3. The relevant parameters are summarized in Table 1.

Table 1 Simulation Parameters

Area Surface	$1000 \times 1000 \text{ m}^2$	CS Power Thr.	-85 dBm ($\approx 1200 \text{ m}$)
Velocity Range	$[0, 1] \text{ m/s}$	Rx Min. SNR	3 dB
Heading Range	$[-\pi, \pi]$	Rx Power Thr.	-70 dBm ($\approx 213 \text{ m}$)
Mobility Interarrival	60 s	HELLO Interval	< 2 s
Tx Power	50 mW	TC Interval	5 s
Tx Frequency	2.4 GHz	Refresh Interval	2 s
Tx Bit Rate	11 Mbps	Maximum Jitter	0.5 s

Since a HELLO message can be lost, so can the advertisement set included in it. For the method to work with OLSR, all successive advertisements have to be received by the nodes performing the verifications. Consequently, we have implemented retransmission of lost advertisements, which details are omitted here for brevity.

5.2 Simulation Results

To evaluate the impact of the diffusion of counters on the capacity of the network, we have measured the total idle time of the medium in the MAC layer, averaged on all the mobiles. We define the idle time as the instants at which a node is considering the medium as not busy and is itself neither waiting for an inter-frame space (IFS) nor backing off. We deduce from that definition that the ratio of idle time of the medium is a good enough approximation of the ratio of available capacity of the medium at some node. The total simulated time for each run was 5 minutes.

The simulation runs are divided in three main batches, with successively 50, 75 and 100 nodes. In each batch, we have compared three main scenarios: plain OLSR with normal HELLO messages; OLSR with HELLO messages containing counters in the double value format; OLSR with HELLO messages containing counters in the double value format and with activated advertisement loss detection and retransmission. Finally, in each scenario, we have measured the idle time ratio with a varying maximum message payload size between 200 and 1500 bytes.

The results for the batch with 50, 75 and 100 nodes are shown in Fig. 6. The general picture is that plain OLSR has a very stable impact in that maximum message size range, while without retransmission, OLSR with counters has smaller impact than with retransmission. The reason why plain OLSR is so stable here is mostly due to the fact that in all runs, the entries for all the neighbors fit at once in each HELLO message, thus lowering the control overhead to a strict minimum. On the other hand, the visible impact on OLSR with counters is due to the fact that the added counter values are a significant overhead, thus preventing all the entries to fit inside a single HELLO message. In addition, retransmission of lost advertisements has an even larger impact, due to the additional overhead, though the difference of impact is obviously constant.

The general tendency for OLSR with counters is for impact on idle time to grow with decreasing maximum message size. This is an illustration of the fact that although approximately the same amount of information has to be diffused, the fact that it uses more independent transmissions increases the total overhead. Since there are more messages transmitted, more nodes contend for the medium more often, hence the lower average idle time ratio. Sending one big message can be considered the same as sending many smaller messages with no waiting in between.

Nevertheless, these results show that the impact of using counters on the average idle time is sufficiently close to the impact of plain OLSR, provided that HELLO message capacity is sufficiently large for the node density (i.e. average number of links to advertise). It appears that for batches with 50 and 75 nodes, message capacities of at least 250 bytes are enough, while for 100 nodes, message capacity below 450 is clearly unacceptable. The strong and irregular impact for 100 nodes below 450 bytes is due to the fact that

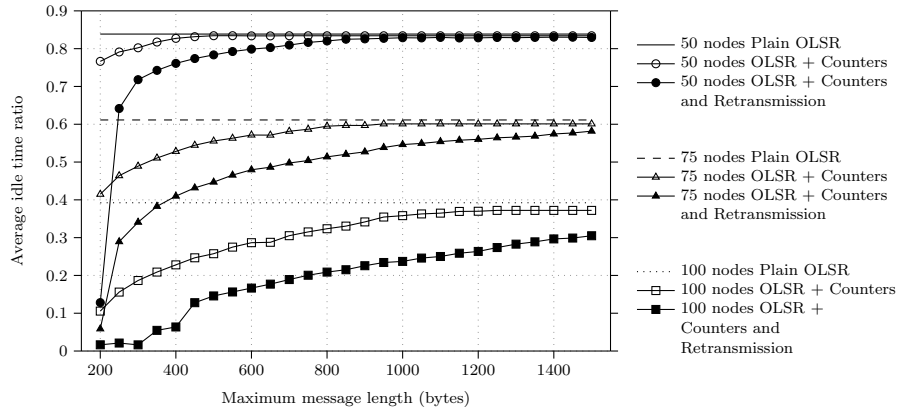


Fig. 6 Ratio of medium idle time, 50, 75 and 100 nodes.

the medium is saturated and contention for the medium is at its highest level (most of the nodes are either transmitting or waiting to transmit).

In practical cases, message capacity is more somewhere around 1400 bytes, so these results are encouraging.

6 Conclusion

We have presented a way of advertising counter values in OLSR for the application of the method of verification of flow conservation [1]. This approach was aimed to be scalable with the increasing number of counters to diffuse and the results show that it performs well in practical scenarios, compared to plain OLSR protocol without diffusion of counter values.

References

1. Ignacy Gawędzki and Khaldoun Al Agha. Proactive resilience to dropping nodes in mobile ad hoc networks. In *AINTEC*, pages 139–158, 2006.
2. T. Clausen and P. Jacquet. Optimized link state routing (OLSR) protocol. RFC 3626, IETF, October 2003.
3. A. McDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks, 1999.
4. IEEE. *IEEE Std 802.11-1999, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.