Adapting dynamically neighbourhood table entry lifetime in wireless sensor networks

Ahmad Ahmad Kassem, Nathalie Mitton

Adapting dynamically neighbourhood table entry lifetime in wireless sensor networks

Ahmad Ahmad-Kassem[*$] and Nathalie Mitton[*]
[$]Université de Lyon, INRIA, INSA-Lyon, CITI, [*]INRIA Lille-Nord Europe, Univ. Lille 1, CNRS
ahmad.ahmad-kassem@insa-lyon.fr, nathalie.mitton@inria.fr

***Abstract*—Neighbour discovery and maintenance of neighbourhood tables have importance in wireless sensor networks. Almost every upper layer application such as routing or self-organizing relies on neighbourhood tables. Imprecise tables may lead to failures that may be costly in terms of resources which are very limited in such networks. Neighbourhood tables are achieved thanks to the Hello protocol. Several studies propose smart schemes to dynamically adapt the frequency of Hello messages but none of them investigates the way the refreshment period of entries in table should be adapted. In this paper, we introduce the Neighbourhood Lifetime Algorithm (NLA), the very first algorithm that adapts dynamically the refreshment period of entries in neighbourhood tables, based on the speed of node and the frequency of the Hello message. Our simulation results show and demonstrate the efficiency of NLA and its high performance to keep neighbourhood tables consistent.**

I. Introduction

Wireless communications take more and more importance in the community of research. Currently, research takes into consideration the mobile networks and the problems related to sensor networks [1], [2], [3]. The applications of such networks are varied, typically involving some kind of monitoring, tracking, or controlling. Specific applications include habitat monitoring (e.g. animals), object tracking (e.g. seaport operations and goods movements), nuclear reactor control, fire detection, land slide detection and traffic monitoring (e.g. vehicles). Sensor networks are self-organized communication systems where the infrastructure is dynamically created and maintained. They are quickly deployable, nodes can move while communicating over wireless links. Two nodes are neighbours if they are in the radio range of each other. The initial state in a sensor network is a collection of nodes that are unaware of each other presence. When joining a sensor network, a node has to discover other nodes in its communication range in order to be able to self-organize and then to efficiently communicate. Topology or neighbour discovery in sensor networks is generally done by letting nodes send hello messages in order to signal their presence [4]. When a node u receives such a message from a node v, u adds v to its neighbourhood table, or updates the timestamps of the entry if v was already there. Periodically, the timestamps of these entries are regularly checked and when one of them is too old, (*i.e.* no message has been received for a long time), the corresponding entry is removed. Due to mobility, topology changes occur frequently and must be notified soon enough in order to avoid routing failures. Since the optimal HELLO frequency depends on parameters that are subject to changes, it must be dynamically adjusted to obtain the best trade-off between the network load and the freshness of neighbourhood tables. Two parameters have to be tuned: the frequency of the Hello Messages and the frequency at which an entry is refreshed. Indeed, in both cases, a badly adapted frequency leads to imprecise neighbourhood tables. While the issue relative to the adaptation of the sending frequency has been widely studied in the literature [2], [5], [6], [7], none of the suggested solutions adapts dynamically the refreshment period of entries in the neighbourhood table.

In this paper, we address the problem of refreshing dynamically the neighbourhood tables and adapting the lifetime of neighbours based on the speed of nodes and the history tables of the frequency of the hello message. In case of no reception of a hello message from a neighbour u, a node b must be able to estimate the lifetime of this neighbour before removing it from its neighbourhood table. We present the Neighbourhood Lifetime Algorithm (NLA) that adapts dynamically the lifetime of neighbours in the neighbourhood tables. In this algorithm, the faster a node, the quicker it has to be removed from a table which means a high change in the neighbourhood relation of each sensor node. We should note that this algorithm is based on the mobility of nodes and on its own history table which contains the frequency of the hello messages. NLA is coupled[1] with TAP [2], a protocol which dynamically adapts the sending frequency of Hello messages based on node turnover, *i.e.* relative mobility of node. As simulations show, NLA is very efficient at keeping neighbourhood table consistent. To date, NLA is the very first algorithm which takes into consideration the dynamic adaptation of the lifetime of entries in the neighbourhood tables.

The outline of this paper is as follows. In Section II, we present the various existing protocols and state our motivation. In Section III, we introduce and describe the protocol NLA, while in section IV, we evaluate and interpret the simulation results and the performance of NLA. Lastly, we conclude by some perspectives to improve this work in Section V.

II. Related work and motivations

Hello protocol for neighbour discovery has been first described in OSPF [4]. This simple protocol works as follows. Every node u regularly sends a light message called *Hello message* containing its identifier to signal its presence. A node v which receives such a message from node u, adds u to its neighbours if u is unknown (*i.e.*, its identifier is not already in the neighbourhood table). Otherwise, the timestamps of the entry associated to u is updated. Deprecated entries are regularly removed thanks to a periodic timer: an entry is deprecated when its timestamps is too old (*i.e.*, that node is no longer a neighbour.) In general, this timer is equal to three periods of the frequency of the HELLO message.

Two issues are attached to this protocol. The first one is the proper sending frequency of the HELLO message and the second is the proper refreshment period of entries in neighbourhood tables. Indeed, if messages are sent too often, they will bring no new information and may saturate the network uselessly but if the sending frequency is too low, some neighbours may not be detected on time. Similarly, if refreshment frequency of tables is too high, some entries will be removed too quickly while nodes may still lay in the vicinity. On the contrary, tables need to be refreshed often enough to avoid to

[1]Can be coupled with any protocol that adapts dynamically the frequency of the hello message.

keep track of nodes that are not reachable anymore. Setting these two frequencies is thus crucial for the tables consistency on which relies every upper layer protocol [1].

In spite of the great importance of dealing with consistent tables, surprisingly, to the best of our knowledge, only few works in the literature have addressed issues relative to the Hello protocol. When they do, they mainly address the problem of the sending frequency.

Previous solutions such as the ones in [8], [9] are probabilistic. They use a timer which expires periodically in order to remove neighbours in the case of no reception of a HELLO message. These solutions, although are very simple and obviously not efficient in a mobile and dynamic environment.

Other solutions such as AHP [7] adapts the frequency of HELLO message to node mobility. Indeed, a node emits a HELLO message every S meters. A node upon receiving a HELLO beacon estimates the time at which a node can be removed from table (*i.e.* estimates the time when the node will be out of the communication range), based on a GPS. Other adaptive algorithms such as RHP [7] establish neighbourhood table purely on demand. When a node receives a data packet and needs the neighbour table, it buffers the packet and then establishes a neighbour table. Each node deletes its neighbour after every nbr-valid-time period, to cope up with mobility and to maintain up-to-date neighbour table.

As stated earlier, the HELLO frequencies should be dynamically adjusted to reflect the dynamic characteristics it depends on, such as the speed of nodes. Very few studies have considered this problem to date. In AHR[6], a simple adaptive protocol is proposed, in which nodes compute two values by monitoring their neighbourhood: the time link failure (TLF), and the time without change (TWC). Moreover, they periodically send a HELLO message at a frequency f_{low}. If a node u notices that the measured TWC becomes greater than a given threshold, it switches to the high-dynamics rate, and sends HELLO messages at a frequency f_{high}. If the estimated TLF becomes smaller than another given threshold, u switches back to the low-dynamics rate, and sends HELLO messages at a frequency f_{low}. In this solution, finding the good thresholds is not obvious since they should evolve over time along with the mobility.

At last, the protocol TAP [2] has been proposed. It presents a fully software based solution that beacons HELLO messages at an optimal frequency. This study adjusts dynamically the HELLO frequency in mobile environment, the higher the mobility the higher the sending frequency should be to ensure that every node is detected. This protocol is well-tailored to standard mobile ad hoc and sensor networks since it does not rely on any specific hardware like a GPS to aim at an optimal HELLO frequency. Nevertheless, in case of no reception of a HELLO message from a neighbour, the refreshment period is equal to three times the last period of the frequency of the HELLO message.

Yet, several studies propose to adapt the frequency of the HELLO message but none of them adapts the refreshment frequency of entries in neighbourhood table. Usually, they all use as a refreshment period equal to three times the sending period, *i.e.* if a node u receives a Hello message from node v every $\frac{1}{f}$, it will remove v from its table if it has received no new Hello message from v during $k \times \frac{1}{f}$ where k is a constant usually set to 3. Such a frequency may be seen as adaptive since f is dynamically adapted to the environment. Nevertheless, this is not enough to fit all the environment requirements. Indeed, the value of k should also be adapted to provide tables as consistent as possible to upper layer protocols.

Therefore, in this paper, we propose to address the problem of the refreshment frequency of neighbourhood tables which will have a direct effect in the neighborhood relation of each sensor node. Our solution is based on the speed of nodes and on the frequency of the Hello message. Since to date TAP [2] has shown to be the most performing protocol in changing dynamically the frequency of the HELLO message, we choose to couple our solution with TAP. Nevertheless, it can be coupled with any other protocol which adapts dynamically the frequency of the Hello message.

III. Dynamic neighbour table refreshment

In this section, we describe the models and the notations used afterwards. Then, we introduce the Neighbourhood Lifetime Algorithm.

A. Preliminaries and notations

We assume that the wireless network is represented by a graph $G = (V, E)$, V being the set of nodes and $E \subseteq V^2$ the set of communication edges: $(u, v) \in E$ means that u and v are neighbours (*i.e.* they are close enough to communicate with each other):

$$E = \{(u, v) \in V^2 \mid u \neq v \wedge |uv| \leq R\},$$

$|uv|$ being the Euclidean distance between u and v. The *physical* neighbourhood set N(u) of a node u is composed of all the physical nodes laying in the communication range of node u:

$$N(u) = \{v \,|\, (u, v) \in E\}.$$

Its cardinality is called the degree of a node u, noted $\delta(u) = |N(u)|$. We note $N'(u)$ the set of neighbours known to u, *i.e.*, whose identifier is present in its neighbourhood table and $T_t u$ the sending period of node u at time t.

B. NLA

As already mentioned, sensor networks are composed of independent nodes having limited capacities in term of memory size and calculation, also strong energy constraints. The failure of sending data involves high energy consumption because source nodes will send the data several times. Each node maintains a neighbourhood table which must be up to date to avoid upper layer protocols failures. The lifetime of a nearby node in the neighbourhood table is by consequence of great importance since removing a node from the neighbourhood table too early or too lately may result in routing failures. The idea is to adapt dynamically the lifetime of entries in neighbourhood table according to the changes of neighbours. Indeed, we claim that the refreshment frequency should be based on the node speed. The higher speed of a neighbour, the higher refreshment frequency. Indeed, a node which moves away quickly is more likely to disappear from a neighbourhood than a one which moves slowly. The no reception of a Hello message may be due to medium unreliability and so a node should wait a longer time before removing this entry.

To avoid sending more information and generating more computations, since our solution is based on the speed of node and the frequency of the hello messages, NLA reuses the HELLO message frequency adaptation used by the protocol TAP [2]. NLA mainly works as follows. For each neighbour v, a node u stores a history table of sending HELLO frequency of v $f(v)$ that it retrieves from the Hello message of v (Table I, $T = \frac{1}{f}$). Let's suppose that last HELLO message from v has been received at t_0. Then, at $t = t_1 = t_0 + T_{t_0}(v)$, if no new HELLO message from v has been received, node u has to decide how long it will wait, without receiving any message from v, before removing v from its neighbourhood table. We denote this waiting time as $Wait(v)$. This waiting time is determined based on the last two periods $T_1 = T_{t_0}(v)$ and $T_2 = T_{t_{-1}}(v)$.

T_n	...	...	T_2	T_1	empty	empty

TABLE I
HISTORIC TABLE

These two values are then compared to compute the lifetime of neighbour v in the case of no reception of a hello message, *i.e.* $Wait(v)$. If at the end of this waiting period, u has not received new Hello Message from node v, it removes v from its neighbourhood table.

In order to specify the lifetime of a neighbour, three possibilities are to be considered:

1) **If period T_1 is equal to period T_2** ($T_1 = T_2$). Node v has a stabilized sending frequency and thus stabilized speed and neighbourhood. In such a case, node u sets its waiting time for node v at $Wait(v) = 3 \times T_1$.
2) **If last period T_1 is larger than period T_2** ($T_1 > T_2$). Sending Hello frequency of node v is not stabilized, node v has a decreasing speed. Since the sending frequency decreases, a period higher than the current period has to be observed since node v is currently sending Hello message less and less often. The waiting period is based on the speed of the frequency variation $T_1 - T_2$.
3) **If last period T_1 is smaller than period T_2** ($T_2 > T_1$). Sending Hello frequency of node v is not stabilized, node v has an increasing speed. Since the sending frequency increases, a period lower than the current period can be observed since node v is currently sending Hello message more and more often. The waiting period is based on the speed of the variation $T_2 - T_1$.

In the two latter cases, the waiting period time associated to node v has to be correlated with the speed of the frequency variation $|T_1 - T_2|$. Two subcases have to be considered:

1) If $|T_1 - T_2| \geq 1$, the sending frequency of v evolves quickly. $Wait(v)$ is set to $T_1 + \frac{T_1}{T_1 - T_2}$.
2) If $0 < |T_1 - T_2| < 1$, the sending frequency of v evolves slowly. $Wait(v)$ is set to $T_1 + T_1 \times (T_1 - T_2)$.

Yet, if $T_1 > T_2$, the waiting lifetime has to be a value higher than T_1, it is equal to $T_1 + \frac{T_1}{T_1 - T_2}$, or $T_1 + T_1 \times (T_1 - T_2)$ according to the value of the difference $|T_1 - T_2|$. If $T_1 < T_2$, the waiting lifetime has to be a value smaller than T_1, it is equal to $T_1 - \frac{T_1}{|T_1 - T_2|}$, or $T_1 - T_1 \times |T_1 - T_2|$ again regarding the value of the difference $|T_1 - T_2|$, as needed.

Algorithm 1 formally describes the protocol NLA. It represents the estimation of the lifetime of a neighbour in the neighbourhood table, in the case of no reception of a hello message.

Algorithm 1 Neighbourhood Lifetime Algorithm computed at node u

```
for ∀v ∈ N'(u) do
  Consults history table of v and retrieves two last periods T1 and T2
  if no-message-during-T1 then
    if T1 = T2 then
      Wait(v) ← (3 * T1)
    else
      if |T2 − T1| ≥ 1 then
        {Sending frequency changes quickly.}
        Wait(v) ← (T1 + T1/(T1 − T2))
      end if
      if 0 < |T2 − T1| < 1 then
        {Sending frequency changes slowly.}
        Wait(v) ← (T1 + T1 × (T1 − T2))
      end if
    end if
  end if
  if No reception of a new Hello message from v after WAIT(u)
  then
    Removes v from neighbourhood table
  end if
end for
```

IV. EXPERIMENTAL RESULTS

A. Simulation setup

We evaluate our algorithm NLA using the WSNet simulator[2] with an IEEE 802.11 MAC layer. We uniformly deploy 50 nodes at random in a square of $500m \times 500m$. Nodes choose a random direction and random speed between 0 and $6m/s$. They all have a same transmission range $100m$, which leads to an average node degree equal to 6 nodes. Results provided here are within a 95% confident interval. These parameters are sum up in Table II.

[2]http://wsnet.gforge.inria.fr

NLA is coupled with TAP [2]. Therefore, we evaluate NLA through a comparison with the plain TAP. Both algorithms (TAP and TAP+NLA) are compared at every run over the same topology of nodes. In the following, for the sake of clarity, we use NLA for TAP+NLA.

Parameter	Value
Number of nodes	50
Time of simulation	200s
Environment	500*500m
Max Speed	6m/s
Transmission range	100m

TABLE II
PARAMETERS OF SIMULATION

B. Metrics

To estimate how good a protocol is at keeping the consistency of neighbourhood tables, we define the following metrics.

Definition 1: The accuracy $acc(u)$ is the proportion of actual neighbours of node u that have been detected by u.

$$acc(u) = \frac{|N(u) \cap N^{'}(u)|}{|N(u)|} \times 100$$

Definition 2: $err1(u)$ measures how many neighbours of node u have not been detected, *i.e.* counts the number of nodes that really lie in the neighbourhood of a node u but that u has not registered in its neighbourhood table.

$$err1(u) = \frac{|N(u) \setminus N^{'}(u)|}{|N(u)|} \times 100$$

Definition 3: $err2(u)$ measures the number of false neighbours of node u that remains in the neighbourhood table, *i.e.* counts the number of nodes that u has in its neighbourhood table but which have actually failed or gone from the communication range of u.

$$err2(u) = \frac{|N^{'}(u) \setminus N(u)|}{|N(u)|} \times 100$$

Definition 4: The error $err(u)$ measures both how many neighbours node u has not detected, and how many false neighbour remain in its neighbourhood table (*i.e.*, old neighbours that have not been removed).

$$err(u) = \frac{|N(u) \setminus N^{'}(u)| + |N^{'}(u) \setminus N(u)|}{|N(u)|} \times 100$$

C. Results

1) Frequency refreshment: Figure 1 shows the impact of NLA in the neighbourhood table of a typical node. It plots the mean value of the lifetime of a neighbour, before removing it, when no new HELLO message has been received from this neighbour.

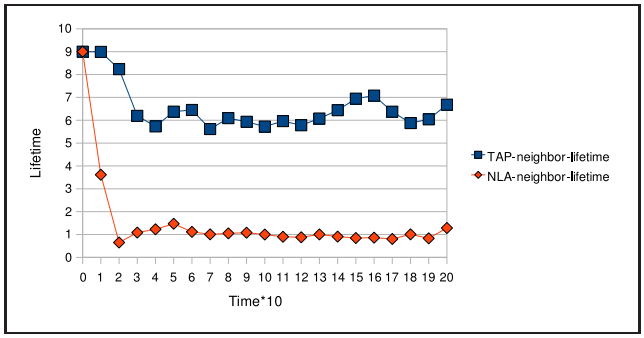


Fig. 1. Frequency refreshment of an entry

Results show that in both cases, the lifetime is dynamic and tends to stabilize. Nevertheless, as already mentioned, the lifetime is actually $a \times T_1$ where a is a multiplication factor determined by the algorithm. In TAP, as in every protocol in the literature, a is a constant usually equal to 3, so only variations of T_1 influences the dynamic of the refreshment period. In NLA, rather than T_1, a is also dynamic and determined by Algorithm 1 as a function of T_2 and T_1. At $t = 0$, we suppose that $a = 3$ and period $T = 3$. Results in Figure (1) show that NLA removes an entry after approximately $1.5s$ while TAP removes an entry after approximately $6.5s$. Even so, the refreshment period of a typical entry stabilizes.

2) Error with respect to time: Figure 2 plots the error (Def.4) *i.e.* neighbour discovery + false neighbours, of neighbourhood tables with respect to time.

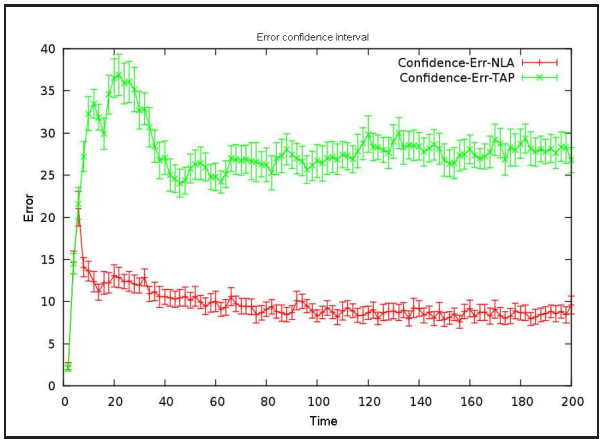


Fig. 2. Error of neighbourhood table

Results in Figure 2 show the confidence interval of error committed by both protocols. The protocol TAP provides an average error of 28%, while NLA generates only an average error of 9%. This means that NLA is approximately up-to-date, it discovers new neighbours quickly and removes false neighbours from neighbourhood table in an appropriated way. NLA outperforms TAP in terms of table consistency. In order to better understand this feature, let's have a closer look to the number of false neighbours and the number of missing neighbours in table.

3) Number of missed detections: Figure 3 represents the number of neighbours which exist in reality and do not exist in table with respect to time (Def. 2). Results show that both protocols achieve similar good results since there are between 2 and 3% of neighbours never detected with NLA against between 3 and 4% with TAP. The neighbour discovery is linked to the frequency of the HELLO message. When an old neighbour is removed from a table, this generates a turnover and thus, the sending frequency behaves accordingly. These results show that removing dynamically deprecated entries in an appropriated way, as NLA does, leads to a better discovery.

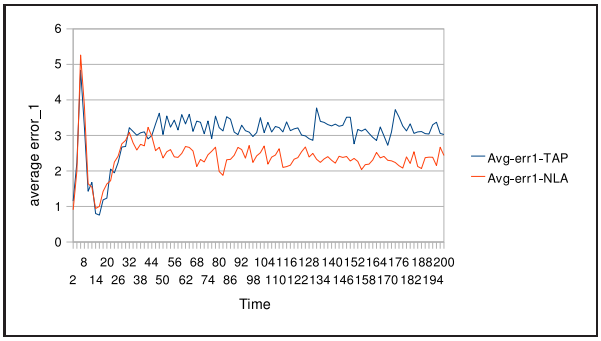


Fig. 3. Error - Number of actual neighbours not detected.

4) Number of false neighbours: Figure 4 illustrates the effectiveness of our protocol, as described in Algorithm 1. In particular, this approach reduces the percentage of false neighbours, *i.e.* the number of neighbours that are in neighbourhood table but actually do not lie anymore in the transmission area of node, by 18% with respect to TAP. Indeed, the protocol NLA provides 6% as an average error of false neighbours, while the protocol TAP has an average error of 24%. This is linked to the fact that our protocol erases deprecated entries more quickly, as shown in Figure (1), and so tables are almost up-to-date.

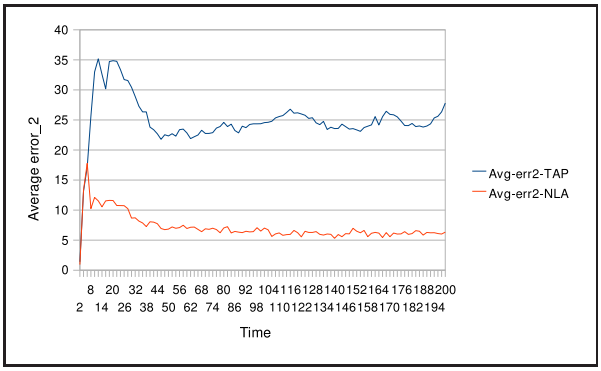


Fig. 4. Error - Number of false neighbours in table.

5) Accuracy with respect to time: So far, we have studied in what proportions our protocol provides false information. We now study how accurate the data are, *i.e.* if majority of real neighbours appear in the table. Figure 5 plots the accuracy of neighbourhood tables with respect to time, *i.e.*. Let us recall that the precision represents the number of neighbours which exist in the neighbourhood table and which actually exist. Our results clearly demonstrate the potential effectiveness of our protocol NLA, which reduces at minimum the number of false neighbours and keeps tables up-to-date with a correctness of approximately 95%. Previous results have globally

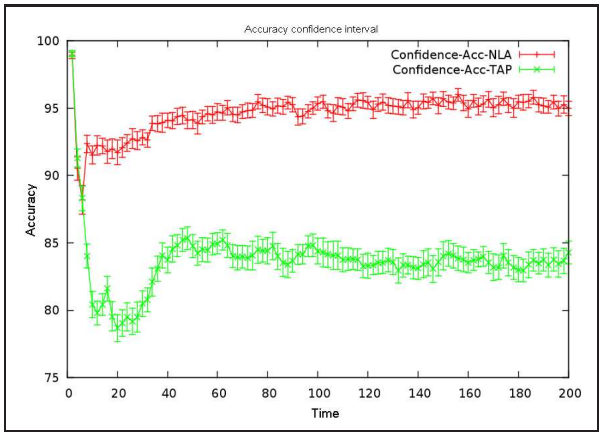


Fig. 5. Accuracy of neighbourhood table

shown that NLA outperforms TAP for different metrics and that both protocols stabilise along time. In the following, we compare both algorithms over same metrics but with respect to speed in order to check their scalability towards node speed and transmission range.

6) Error and accuracy with respect to speed: We study the impact of node speed on the consistency of neighbourhood tables. Figure 6 plots both the error and accuracy of neighbourhood tables with respect to speed. We conclude from figure 6(b) that NLA achieves a better accuracy than TAP. The accuracy obtained by using TAP tends to decrease with the node speed, while the one gotten with NLA tends to stabilise around 94%. This is due to the fact that NLA takes into consideration the tendency of the sending frequency changes of the HELLO message to set up the frequency refreshment of an entry. Thanks to it, *(i)* depreciated entries are removed quickly and thus the error is reduced and *(ii)* the turnover is refreshed and sending frequency re-adapted, which allows a quicker detection of new neighbours.

By applying the principles of the TAP protocol, when the node speed increases, to stick to a constant turnover, the node frequency should increase. In addition, when the node speed increases, a neighbour of node u remains a shorter time as a neighbour. Thus, tables have to follow these changes and be kept up to date. Figure 6(a) shows the error ($err1 + err2$) provided by both protocols. The percentage of error for TAP slightly increases, up to 30%, with the node speed while the percentage error for NLA tends to stabilize around 10%.

7) Error and Accuracy with respect to transmission range: We study the impact of the transmission range of nodes over the consistency of neighbourhood tables. The bigger transmission range, the higher number of neighbours for a node. When the transmission range increases, nodes remains neighbours a longer time. It is thus easier to maintain the link between both. In this section, nodes choose a random speed between 0 and $6m/s$.

Figure 7 plots both the error and accuracy of neighbourhood tables with respect to the transmission range R. Results show that both protocols behave similarly but with great improvement for NLA, *i.e.* the error decreases and the accuracy increases with respect to transmission range. This is linked to the fact that a node remains a longer time in a transmission area of another node and thus in the neighbourhood table. NLA outperforms TAP and provides at most 12% errors while TAP generates more than 30% errors as seen in figure 7(a). Moreover, as we see in figure 7(b), NLA provides an accuracy of more than 96% while TAP never overpasses an accuracy of 87%. For R = 100m, NLA decreases by approximately 18% the level of error compared to that of TAP as it appears in figre 7(a). There is a large improvement of the error level between both protocols. In addition, NLA shows in figure 7(b) an improvement of 12% of the accuracy of the neighbourhood tables compared to the plain TAP protocol .

8) Number of messages: As already mentioned, NLA removes more quickly depreciated entries from neighbourhood tables compared to TAP. This modifies the turnover and thus makes nodes adapting their sending frequency accordingly. This has the benefit that new neighbours are discovered more quickly and thus improves the accuracy. Nevertheless, this also implies that NLA generates more Hello messages than the plain TAP. Figure 8 plots the mean number of hello messages sent by each node. As expected, the number of Hello messages generated by NLA is almost twice the one generated by TAP. Since more messages are sent, more energy and bandwidth are spent.

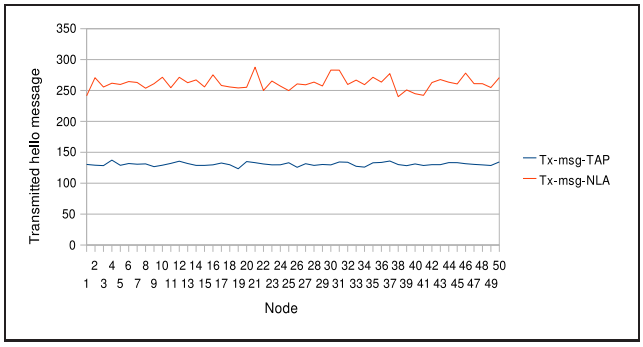
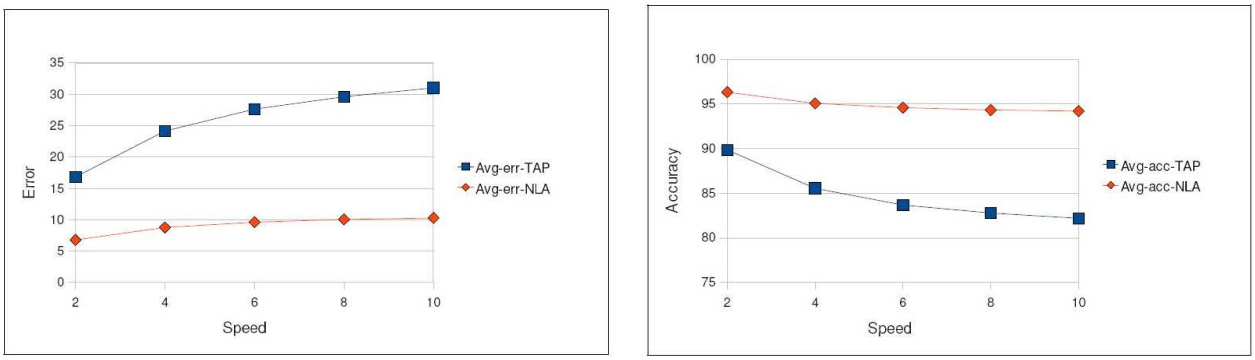


Fig. 8. Number of HELLO messages per node

There is a trade off to consider when using TAP with or without the NLA mechanism. Indeed, NLA greatly improves the quality of neighbourhood tables of nodes but at the price of more messages and thus more energy and bandwidth spending. The utilization of NLA should thus be motivated by the application. If the application is very sensitive and needs consistent tables, NLA should be preferred.

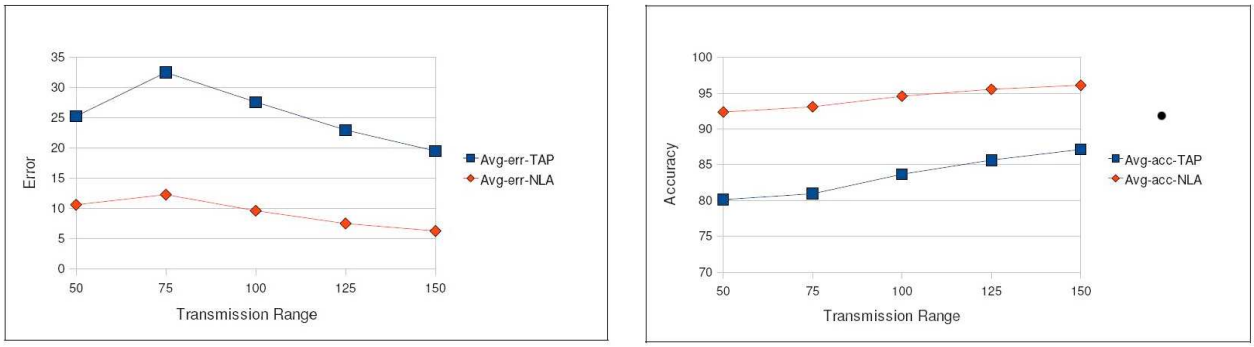
V. Conclusion and perspectives

In this paper, we have introduced the very first algorithm which adapts dynamically the refreshment period of entries in neighbourhood tables without the need of a GPS or another means of localization. NLA presents an approach that allows to estimate dynamically the lifetime of neighbours while basing on the speed of nodes and the history table of the hello message frequency. If a node does not receive a hello message from a neighbour, it consults the history table



(a) Error (b) Accuracy

Fig. 6. Error and accuracy of neighbourhood tables wrt speed



(a) Error (b) Accuracy

Fig. 7. Error and accuracy of neighbourhood tables wrt range

of this neighbour, it compares the last two frequencies and estimates the time to wait for an eventual hello message before removing this neighbour.

The implemented protocol provides notable improvements compared to when it is not used. Nevertheless, we show that this improvement of the table consistency implies more message overhead. Therefore, there is a trade off to consider, driven by the application, between the message overhead and table consistency.

As future works, we will study the protocol in the presence of different traffic models. In addition, we will study the power and the overhead introduced by additional hello message with additional routing overhead which is caused by wrong information of the neighbourhood table in order to compare the trade off. Moreover, we will investigate means to optimize the energy consumption and minimize the message overhead generated by NLA. The objective is to be able to adapt to a compromise between the supported flow of information and the energy consumption.

REFERENCES

[1] E. B. Hamida, G. Chelius, and E. Fleury, "Revisiting neighbor discovery with interferences consideration." Spain: ACM, October 2006.

[2] F.Ingelrest, N.Mitton, and D.Simplot-Ryl, "A turnover based adaptive hello protocol for mobile ad hoc sensor networks," in *Proc. IEEE MASCOTS*, 2007.

[3] T.Razafindralambo and N.Mitton, "Analysis of the impact of hello protocol parameters over a wireless network self-organization," in *4th ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN07)*, Greece, October 2007.

[4] J.Moy, "OSPF - Open Shortest Path First," March 1994, RFC 1583.

[5] I.Chakeres and E.Belding-royer, "The utility of hello messages for determining hello connectivity," in *IEEE International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2002.

[6] C.Gomez, A.Cuevas, and J.Paradells, "a two-state adaptive mechanism for link connectivity maintenance in aodv," in *Proc. REALMAN*, 2006.

[7] V.Giruka and M.Singhal, "Hello protocols for ad hoc networks : Overhead and accuracy trade-offs," in *Proc. IEEE WoWMoM*, 2005.

[8] G.Sawchuk, G.Alonso, E.Kranakis, and P. Widmayer, "Randomized protocols for node discovery in ad hoc multichannel networks," in *Proc. of Ad Hoc Now*, 2003.

[9] G.Alonso, E.Kranakis, G.Wattenhofer, and P. Widmayer, "Probabilistic protocols for node discovery in ad hoc, single broadcast channel networks (extended abstract)," in *Proc. IEEE IPDPS*, 2003.

[10] L. Ding, W. Zhang, H. Yu, X. Wang, and Y. Xu, "Incorporating tcp acknowledgements in mac layer in ieee 802.11 multihop ad hoc networks." IEEE, 2009.

[11] G.Wattenhofer, G.Alonso, E.Kranakis, and P. Widmayer, "Randomized protocols for node discovery in ad hoc, single broadcast channel networks," in *Proceedings of the IEEE International Parallel and Distributed Processing symposium (IPDPS)*, 2003.

[12] M. Rosenschon, T. Manz, J. Habermann, and V. Rakocevic, "Gateway Discovery Algorithm for Ad-Hoc Networks Using HELLO Messages. Fachhochschule Giessen-Friedberg University of Applied Science, Germany."

[13] M. Saxena and R. R. Kompella, "On the inadequacy of link connectivity monitoring," February 2008.

[14] A. Buhrig and M. Renaudin, "Gestion de la consommation des noeuds de reseau de capteurs sans fil. Laboratoire TIMA-INPG, Grenoble."