A Turnover based Adaptive HELLO Protocol for Mobile Ad Hoc and Sensor Networks

François Ingelrest, Nathalie Mitton, David Simplot-Ryl

A Turnover based Adaptive HELLO Protocol for Mobile Ad Hoc and Sensor Networks

François Ingelrest
LCAV, I&C School, EPFL
CH-1015 Lausanne, Switzerland
Francois.Ingelrest@epfl.ch

Nathalie Mitton and David Simplot-Ryl
IRCICA/LIFL, Univ. Lille 1
CNRS UMR 8022, INRIA Futurs, France
{Nathalie.Mitton,David.Simplot}@lifl.fr

***Abstract*—We present a turnover based adaptive HELLO protocol (TAP), which enables nodes in mobile networks to dynamically adjust their HELLO messages frequency depending on the current speed of nodes. To the best of our knowledge, all existing solutions are based on specific assumptions (*e.g.*, slotted networks) and/or require specific hardware (*e.g.*, GPS) for speed evaluation. One of the key aspects of our solution is that no additional hardware is required since it does not need this speed information. TAP may be used in any kind of mobile networks that rely on HELLO messages to maintain neighborhood tables and is thus highly relevant in the context of ad hoc and sensor networks. In our solution, each node has to monitor its neighborhood table to count new neighbors whenever a HELLO is sent. This *turnover* is then used to adjust HELLO frequency. To evaluate our solution, we propose a theoretical analysis based on some given assumptions that provides the optimal turnover when these assumptions hold. Our experimental results demonstrate that when this optimal value is used as the targeted turnover in TAP, the HELLO frequency is correctly adjusted and provides a good accuracy with regards to the neighborhood tables.**

I. Introduction and motivation

In mobile ad hoc and sensor networks, because of the path loss of radio communications, only close hosts may directly communicate to each other. Long-distance communications require packets to be forwarded by multiple intermediate nodes. While sensor networks are generally static, some applications involve mobility (*e.g.*, herd health control as targeted by the WASP European project[1]). Localized routing schemes are a resource-efficient way of achieving communication between two end hosts. In such schemes, each intermediate node is only expected to maintain knowledge about spatially nearby nodes (its *neighbors*). In most existing works, this knowledge is acquired thanks to beacon messages (the well-known *HELLO messages*): all nodes maintain a neighborhood table, and any localized protocol may make decisions based on it [3], [8].

In this paper, we are interested in studying how nodes may dynamically adjust the frequency of HELLO messages. Indeed, because of mobility, neighborhood tables have a limited lifetime and must be regularly updated. However, finding the correct frequency is not obvious: if it is too low (*i.e.*, with regards to the speed of hosts), then tables quickly become obsolete. On the contrary, if it is too high, then tables will be up to date but a high part of the available bandwidth will be wasted to the detriment of data traffic. There obviously exists a trade-off between these behaviors, but finding the optimal one is not trivial. Moreover, this trade-off actually depends on network characteristics (*e.g.*, density, speed) that may evolve over time, and a constant frequency is then not the best choice. An efficient HELLO protocol should thus be adaptive. A straightforward solution might be to let nodes know their speed and choose the correct frequency based on this information, but of course there is no easy and cheap way for a node to determine its speed.

In this paper, we propose the Turnover based Adaptive HELLO Protocol (TAP), an elegant solution that let nodes dynamically adjust their HELLO frequency based on the turnover of their neighborhood. One of the key aspects of our solution is that it highly fits mobile wireless networks since it is fully localized and does not require additional hardware. Our solution is independent of any routing protocol. Nodes only need to periodically make samples of their table to compute the current turnover and adapt their HELLO frequency. TAP may actually be seen as a generic framework rather than a frozen protocol, since it is independent of the functions used to adjust the HELLO frequency. Any such functions may be used depending on the required behavior (*e.g.*, favor bandwidth usage to the detriment of up to date tables). To evaluate our proposal, we provide a theoretical analysis in order to compute the optimal turnover for the considered environment. This allows to check the correctness of our scheme by means of simulation results using the same assumptions.

We give the needed preliminaries in the next section, while Section III proposes a literature review of related work. In Section IV, we describe TAP and provide a theoretical analysis that aims at finding the optimal turnover under known parameters. We then give experimental results about the effectiveness of TAP in Section V, and show that it is indeed able to dynamically adapt the HELLO frequency without speed information. We finally conclude in Section VI.

II. Preliminaries

A. Network Model

Wireless networks are represented by a graph $G = (V, E)$ where V is the set of nodes and $E \subseteq V^2$ the set of edges: $(u, v) \in E$ means that u and v are neighbors (*i.e.*, close enough to communicate). The neighborhood set $\mathrm{N}(u)$ of a vertex u is

[1]http://wasp-project.org

equal to $\{v : (u, v) \in E \vee (v, u) \in E\}$. The density is the average number of neighbors per node. Each node is assigned a unique identifier (*e.g.*, a MAC address). Wireless links are determined by the physical model. The most frequent one is the *unit disk graph* model [2]:

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

$|uv|$ being the Euclidean distance between nodes u and v, and R the maximum communication range.

B. Plain Periodic HELLO Protocol

The basic HELLO protocol, first described in OSPF [7], works as follows. Nodes regularly send HELLO messages to signal their presence to close nodes, and maintain a neighborhood table. The frequency of these messages is noted f_{HELLO} and the delay between them d_{HELLO} (*i.e.*, $d_{\text{HELLO}} = 1/f_{\text{HELLO}}$). When a node u receives such a message from a node v, u adds v to its table, or updates the timestamp of the entry if v was already there. We do not make assumptions about the content of HELLO messages, but they must contain the identifier of the sender.

III. Related Work

In [1], authors claim that the usefulness of a HELLO protocol depends on the size of the beacon messages, their transmission rate and the lifetime of deprecated entries of the neighborhood table. While these aspects are important, only a few studies have been performed about them. Among the proposed enhancements of the basic HELLO protocol, most of them require a slotted network. For instance, [6] aims at reducing the overall energy consumption, assuming a static network. Nodes can be in three different states: *sending*, *listening* or *sleeping*. At each slot, nodes choose a state with a probability p_{state}: the difficulty is then to determine optimal values of $p_{smboxstate}$. In [10], [12], three different protocols are proposed for both single channel and multichannel networks. In the first one (RP), nodes send a HELLO at each slot with probability p and listens with probability $1 - p$. In the second one (AP), nodes immediately answer upon reception of a message. In the last one (LP), the following condition is added to RP: nodes listen the carrier if they have sent a HELLO on the slot before. Upon reception of a HELLO, nodes trigger a backoff time and send a new HELLO when the countdown expires. As a result, LP presents the best trade-off when the number of nodes is known. Of course, knowing the number of nodes in a decentralized and dynamic network is a rather unrealistic assumption. Moreover, all the proposals assume a slotted network, which means that synchronization among nodes is needed, which is by itself not a trivial problem.

As stated earlier, a low HELLO frequency leads to obsolete tables while a high one may saturate bandwidth to the detriment of data traffic. The trade-off depends on characteristics (*e.g.*, density, speed) that may evolve over time, leading to the need for an adaptive protocol. Nevertheless, only a few studies have tried to tackle this problem. In [5], a simple adaptive protocol is proposed, in which nodes evaluate two values by monitoring their neighborhood: the time link failure (TLF) and the time without change (TWC). Moreover, they periodically send HELLO at a frequency f_{low} again. If a node 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}. On the contrary, if the estimated TLF becomes smaller than a threshold, it goes back to the "low dynamics rate" and sends HELLO at a frequency f_{low}. In this solution, finding the correct thresholds is not obvious since the thresholds themselves may need to evolve over time.

In [4], authors propose three protocols in order to approach the best trade-off. The first one is called *Adaptive HELLO protocol*: each node simply sends a HELLO each time it has gone through X meters. Though straightforward, this scheme assumes that nodes are aware of their speed and their location. The second protocol is called *Reactive HELLO protocol*. It is based on the idea that nodes should build their neighborhood table only when needed. Thus, when a node sends a data packet, it first sends a HELLO message and waits during a time t for an answer. If no answer is received, then it repeats the same behavior up to X times. Upon reception of a HELLO message, nodes trigger a timeout before answering, to avoid collisions. While this scheme minimizes the quantity of HELLO messages, it introduces a high latency before sending data packets, and should not be used in networks with high mobility. The third protocol proposed in [4] is called *event-based HELLO protocol*. Nodes perform the classic periodic HELLO protocol, but if they do not receive any message and do not need to send data packets during a given time period, then they stop sending beacon messages until reception of a HELLO message. The main drawback of this protocol is that some nodes may never be detected by mobile nodes.

In [11], an optimal HELLO frequency which depends on the relative speed between objects is described. The idea is that a node which strides more than a given distance in the communication area of another node has to be detected by the latter. If the two nodes move with a speed S, the optimal frequency f_{opt} is equal to:

$$f_{\text{opt}} = \frac{2S}{aR}, \tag{1}$$

where aR is the threshold distance in communication area to be detected ($a < 1$).

IV. The TAP Protocol

A. Description

We suppose that each node sends HELLO messages at the frequency f_{HELLO}. Whenever a node receives a HELLO message, it updates its neighborhood table, thus generating some *turnover*. We assume in this paper that given a period of time Δt, the turnover $r_{\Delta t}$ is equal to the ratio between the number of new neighbors (*i.e.*, nodes that were not yet neighbors Δt units of time earlier) and the current total number of neighbors. Obviously, the turnover depends on both Δt and f_{HELLO}, but if we assume that $\Delta t = 1/f_{\text{HELLO}}$ (that is, the turnover is computed each time a HELLO message is

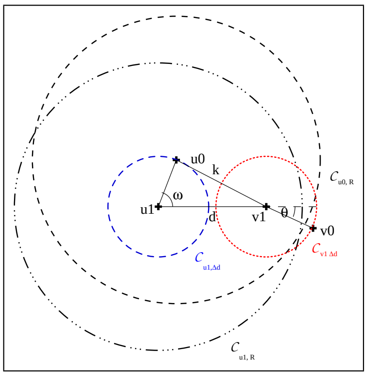

Fig. 1. A new neighbor v of a node u - Global view.

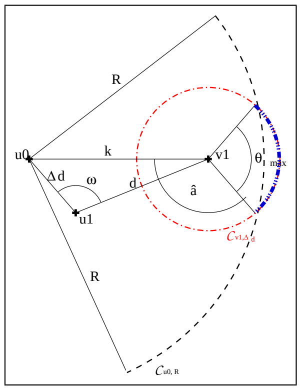

Fig. 2. A new neighbor v of a node u - Zoom.

sent), then it only depends on f_{HELLO}. For the sake of clarity, we denote the turnover by r through the rest of this paper.

The protocol we propose is based on the idea that nodes can aim at the optimal HELLO frequency f_{opt} (see (1)) by measuring the turnover. The analysis given in the next section shows the relationship between the optimal turnover r_{opt} and the optimal frequency. Indeed, if a node aims at keeping r constant, then it has to modify f_{HELLO}. Keeping the turnover close to r_{opt} should make f_{HELLO} tend toward an optimal HELLO message frequency f_{opt}. Nodes should thus compute the current turnover r each time they send a HELLO message. To do so, they just have to make a backup of their table before sending a HELLO message, in order to identify new neighbors which will appear. Once r is computed, two cases may happen:

- When r is too small ($r < r_{\text{opt}}$), this means that f_{HELLO} is too high and there are not enough changes in the table. The frequency f_{HELLO} should be decreased.
- When $r > r_{\text{opt}}$, f_{HELLO} is too low and there are too many changes. The frequency f_{HELLO} should thus be increased.

One can note that this description is generic, and does not depend on any external update function. It simply assumes that nodes are aware of the value of r_{opt} and are able to observe their neighborhood table to determine the number of new neighbors each time a new HELLO is sent. We will later provide in Section IV-C some possible functions that may be used to adapt f_{HELLO} based on the observed turnover r.

B. From optimal frequency to optimal turnover

The principal issue that remains opened is: what is the optimal frequency? Finding the correct value for f_{opt} is indeed not trivial and may depend on a lot of parameters. We propose in this section to theoretically compute r_{opt} to derive f_{opt} by setting these parameters. We thus assume a Unit Disk Graph and a mobility model where nodes move at a constant speed s in a random direction. While these assumptions may not be the most realistic ones, we used them as a first step in order to experiment our HELLO protocol.

In this analysis, we suppose that nodes are randomly deployed in a 1×1 square using a Poisson Point Process (node positions are independent) with an intensity $\lambda > 0$, λ being the mean number of nodes per surface unit. We are first interested in finding the mean number of new neighbors of a node after a time period Δt. Let v be a point at distance $d \leq R$ from node u at time $t_1 = t_0 + \Delta t$ (v is thus a neighbor of u). We need to compute the probability $\mathbb{P}(d)$ that node v is actually a *new* neighbor of u. Once $\mathbb{P}(d)$ is known, the mean number of new neighbors of a node u after a time period Δt (noted $\mathbb{E}[N]_{\Delta t}$) is simply equal to:

$$\mathbb{E}[N]_{\Delta t} = \int_{d=0}^{R} 2\lambda\pi d \times \mathbb{P}(d) dd. \tag{2}$$

Let u_0 (resp. v_0) be the position of node u (resp. v) at time t_0 and u_1 (resp. v_1) its position at time t_1. Let also be $\Delta d = S \times \Delta t$. We are thus interested in finding the probability that $|u_0 v_0| > R$ knowing that $|u_1 v_1| < R$. We denote by ω the direction $\overrightarrow{u_0 u_1}$ and θ the direction $\overrightarrow{v_0 v_1}$. ω and θ are chosen in a uniformly random fashion in $[-\pi, \pi]$ (each direction has the same probability to be chosen). Fig. 1 illustrates our model. $C_{u,R}$ is the circle centered at u with radius R and $k = |u_0 v_1|$. The blue dashed circle $C_{u_1,\Delta d}$ (resp. red dotted circle $C_{v_1,\Delta d}$) represents the possible position of u_0 (resp. v_0). We can first notice that the shorter Δt, the less likely v is to be a new neighbor of u. This leads to the fact that there exists $d_{\text{min}} = \min(0, R - 2\Delta d)$ s.t. if $d < d_{\text{min}}$, $\mathbb{P}(d) = 0$ whatever ω and θ.

Given d and ω, computing the probability $\mathbb{P}(d, \omega)$ that node v is a new neighbor of node u amounts to computing the probability that node v comes from the dotted blue angular sector of Fig. 2, Thus, we have:

$$\mathbb{P}(d, \omega) = \int_{0}^{\theta_{\text{max}}} \frac{d\theta}{2\pi}. \tag{3}$$

By using notations on Fig. 2, we have $2\pi = \theta_{\text{max}} + 2\hat{a}$ and $\hat{a} = \arccos(\frac{\Delta d^2 - R^2 + k^2}{2k\Delta d})$. We deduce :

$$\theta_{\text{max}} = 2\arccos(\frac{R^2 - \Delta d^2 - k^2}{2k\Delta d}). \tag{4}$$

Given d and ω, node v can be a new neighbor iff $C_{u_0,R}$ and $C_{v_1,\Delta d}$ intersect (see Fig. 2). If not, that means that nodes v and u were already neighbors at t_0. $C_{u_0,R}$ and $C_{v_1,\Delta d}$ intersect only if:

$$R - \Delta d \leq k \leq R + \Delta d. \tag{5}$$

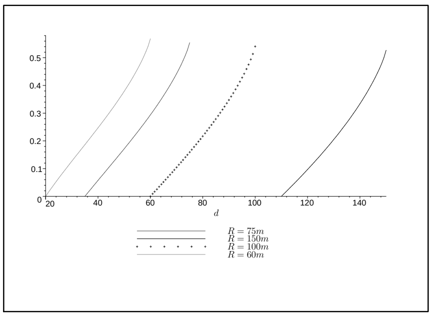

Fig. 3. Probability to be a new neighbor as a function of $d = |uv|$ - $S = 2\,m.s^{-1}$.

From (5), we can deduce the angular sector $\Omega = [\omega_{\text{min}}, \omega_{\text{max}}]$ of ω for which node v is a new neighbor.

We have $k = \sqrt{\Delta d^2 + d^2 - 2d\Delta d\cos(\omega)}$ and $k < d + \Delta d\, \forall \omega \in \{0, \pi\}$. Since $d < R$, we have $k < R + \Delta d\ \forall \omega \in \{0, \pi\}$ and so $\omega_{\text{max}} = \pi$. We can also notice that $k > R - \Delta d$ for $\omega > \omega_{\text{min}}$ such that $k(\omega = \omega_{\text{min}}) = R - \Delta d$.

$$R - \Delta d = k \Leftrightarrow \omega_{\text{min}} = \arccos(\frac{d^2 + 2R\Delta d - R^2}{2d\Delta d})$$

Thus, given a distance d, node v is a new neighbor of u iff $\omega \in \Omega = [\omega_{\text{min}}, \omega_{\text{max}}]$:

$$\omega_{\text{min}} = \arccos(\frac{d^2 + 2R\Delta d - R^2}{2d\Delta d}) \leq \omega \leq \pi.$$

Now, we can compute the probability $\mathbb{P}(d)$ that a node v is a new neighbor of u:

$$\begin{aligned} \mathbb{P}(d) &= 2 \times \int_0^\pi \frac{\mathbb{P}(d,\omega)}{\pi} d\omega \\ &= \begin{cases} \frac{1}{\pi^2}\int_{\omega_{\text{min}}}^{\pi} \theta_{\text{max}} d\omega & \text{if } d_{\text{min}} \leq d \leq R, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

Fig. 3 plots $\mathbb{P}(d)$ for several values of R, $\Delta t = 0.2s$, $S = 2\,m.s^{-1}$ and $\lambda = \frac{100}{1000 \times 1000}$. As expected, $\mathbb{P}(d)$ increases when d or S increases and is proportional to R.

From (6), we can deduce the number of new neighbors that node u encounters during a time period Δt:

$$\mathbb{E}[n]_{\Delta t} = 2\pi\lambda \int_{d_{\text{min}}}^{R} d\mathbb{P}(d)dd = \frac{\lambda}{\pi}\int_{d_{\text{min}}}^{R}\int_{\omega_{\text{min}}}^{\pi} d * \theta_{\text{max}} d\omega dd. \tag{7}$$

Equation (7) allows us to theoretically compute r_{opt} (see Section IV-A). Indeed, r_{opt} is the ratio obtained between two HELLO packets sent at frequency f_{opt}. Thus:

$$r_{\text{opt}} = \frac{\mathbb{E}[n]_{\Delta t = \frac{1}{f_{\text{opt}}}}}{\lambda \pi R^2}.$$

The speed parameter S appears in the result of $\mathbb{E}[n]_{\Delta t}$ only through $\Delta d = S \times \Delta t$. For $\Delta t = \frac{1}{f_{\text{opt}}}$, we have $\Delta d = S * \frac{aR}{2S} = \frac{aR}{2}$. Thus, since Δd does not depend on S anymore, r_{opt} is independent of the speed and only depends on R.

Fig. 4 and 5 plot the theoretical values of r_{opt} as a function of R for different values of a. Note that with the help of

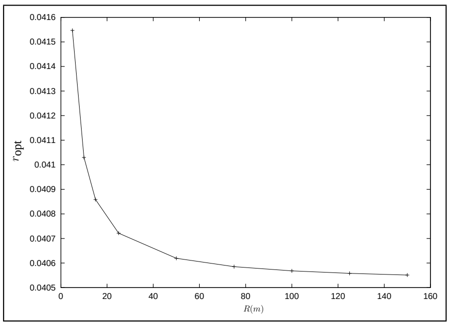

Fig. 4. Theoretical values of r_{opt} as a function of R for $a = 0.1$.

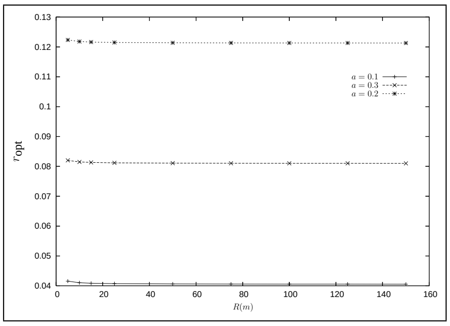

Fig. 5. Theoretical r_{opt} as a function of R for different values of a.

this study, by watching its neighborhood during Δt, a node can count the number of new neighbors and from this quantity, deduces its relative speed S (from correspondence tables). This can help for many applications.

C. Implementation

There are still some remaining issues regarding how our TAP protocol may be implemented. One of them is about how a node may obtain a correct turnover value. In the previous section we indeed showed that this value may be very small (*e.g.,* 0.04) while it is nearly impossible for a node to observe a so small turnover between two successive HELLO messages. A solution to this problem is to let nodes archive more than one table into a history of size X: if X is sufficiently large, then a correct value may be expected. The turnover may then be computed by counting neighbors present in the most recent table that are not present in the oldest one and by using the current HELLO delay as:

$$r = \text{nb new neighbors} \times \frac{\text{current } d_{\text{HELLO}}}{\text{elapsed time}}.$$

To adapt f_{HELLO}, one also needs to define some functions based on the turnover r. For instance, the amplitude at which f_{HELLO} should be modified (either increased or decreased) should be determined by the difference between r and r_{opt}: the higher the difference, the more likely f_{HELLO} and f_{opt} are really different from each other. To compute this amplitude, we propose to use the following function $g(x)$:

$$g(x) = \begin{cases} (\frac{r - r_{\text{opt}}}{r_{\text{opt}}})^2 & \text{if } r < 2 \times r_{\text{opt}}, \\ 1 & \text{otherwise.} \end{cases}$$

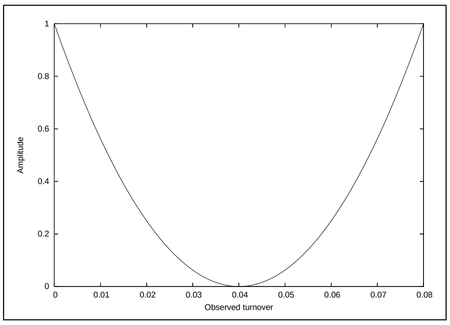

Fig. 6. Function $g(x)$ used to adapt f_{HELLO} ($r_{\text{opt}} = 0.04$).

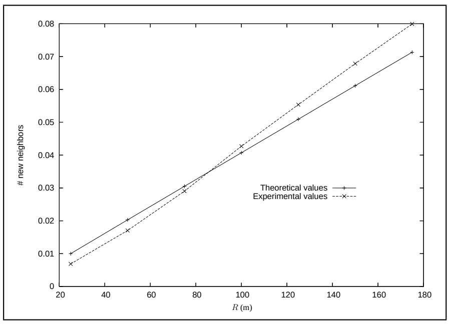

Fig. 8. New neighbors per node - $\Delta t = 5s$ - $S = 2m.s^{-1}$ - Increasing R.

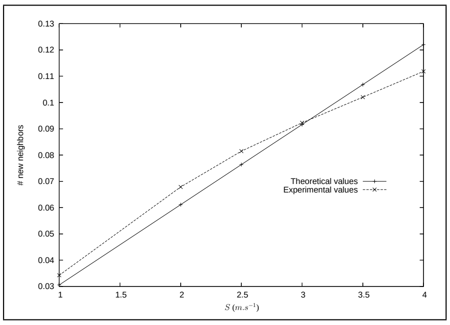

Fig. 7. New neighbors per node - $R = 150$ - $\Delta t = 5s$ - Increasing S.

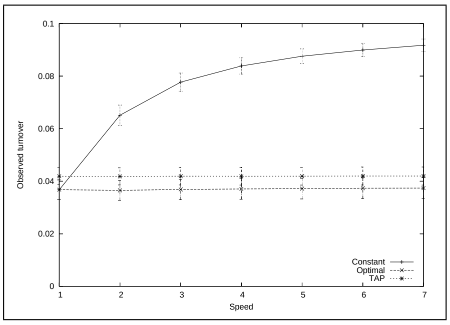

Fig. 9. Observed turnover.

This function is illustrated by Fig. 6, where r_{opt} is fixed to 0.04 in accordance with theoretical results (see Fig. 4). With this function, when r and r_{opt} are only slightly different, then f_{HELLO} is only slightly modified. Opposed to this case, the higher the difference between r and r_{opt}, the fastest f_{HELLO} should converge to the optimal frequency. Of course, any other interesting function $g(x)$ such that $g(r_{opt}) = 0$ may be used for this purpose since our protocol does not depend on this particular one.

Finally, to adapt the delay between two HELLO messages and thus the resulting turnover, we propose to use the following series:

$$d_{\text{HELLO}} = \left\{ \begin{array}{ll} d_{\text{HELLO}} + \frac{d_{\text{HELLO}}}{4} \times g(r) & \text{if } r \leq r_{\text{opt}}, \\ d_{\text{HELLO}} - \frac{d_{\text{HELLO}}}{4} \times g(r) & \text{otherwise.} \end{array} \right\}$$

Once again, any other function may be used to adapt the delay between HELLO messages. For instance, it may be possible to adjust the functions in order to favor certain aspects, like the bandwidth usage. Evaluating what kind of functions may be used to favor such or such behavior is actually beyond the scope of the current paper.

V. Experimental Results

Our experimental results were obtained thanks to a homemade simulation tool, using the Unit Disk Graph model. In our simulations, we used a Poisson point process of intensity $\lambda = \frac{100}{A}$ in a square area of size $A = 1000 \times 1000$. The results obtained are within a 95% confidence interval. For each iteration, a new network is generated. We used the functions described in Section IV-C with a history size $X = 10$. Regarding TAP, the targeted turnover is fixed to 0.04. In these experiments, we do not consider the problem of determining the lifetime of an entry of the neighborhood table: an entry is removed as soon as the corresponding node has not sent a HELLO at the right time. We chose to discard all other protocols presented in Section III since they rely on very specific assumptions (*e.g.*, slotted network, dedicated hardware). To evaluate our TAP protocol, we chose to compare it to two other schemes:

- **Constant**: f_{HELLO} is fixed to 0.2 and never varies.
- **Optimal**: f_{HELLO} is set to f_{opt} (refer to Section IV-B) based on the current speed.

We first provide results that demonstrate the correctness of the theoretical analysis performed in Section IV-B. We thus give the expected average number of new *real* neighbors per node for an observation period $\Delta t = 5s$. The neighbors referenced here are the real physical neighbors, and no neighborhood table is used. On Fig. 7, R is fixed to $150m$ and S increases. On Fig. 8, S is fixed to $2m.s^{-1}$ and R increases. In both cases, one may notice that experimental and theoretical results perfectly match.

On Fig. 9 the observed turnover for all selected protocols is provided. As expected, the turnover increases for the Constant scheme since the observation periods have a fixed duration (d_{HELLO} is set to $5s$) while the speed increases. A higher number of new neighbors is thus observed at each HELLO, and the turnover increases accordingly. The Optimal scheme adjusts f_{HELLO} by using the exact value of the current speed, and provides a constant turnover. The observed turnover is close to 0.04, thus validating the theoretical value computed in Section IV-B. Regarding TAP, it is interesting to note that it is effectively able to aim at a given ratio (set to 0.04 here),

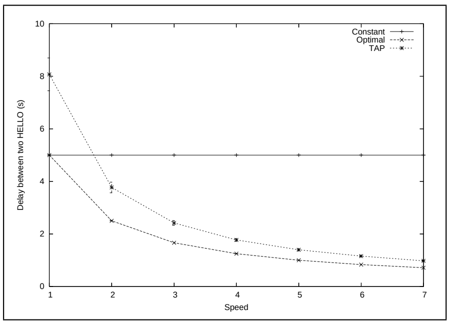

Fig. 10. Delay between two HELLO messages.

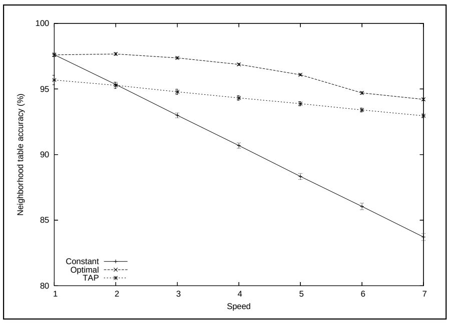

Fig. 11. Accuracy of neighborhood tables

providing a constant turnover. The observed value is slightly higher than expected, but this may be corrected by using different adjustment functions and/or by targeting at a slightly lower turnover.

On Fig. 10, we give the observed value of d_{HELLO} for varying speed. Of course, the delay of the Constant scheme does not vary since it does not take the speed into account. The Optimal scheme computes f_{HELLO} based on the real speed and should thus provide an optimal value of d_{HELLO}. Regarding TAP, the real speed value is not used, since it is not available. As explained in Section IV, nodes only observe the turnover and adjust the delay based on this observation. One can note that it is very effective. When the "optimal" turnover value of 0.04 is targeted, the delay is not the same as with the Optimal scheme because, as observed in Fig. 9, the real turnover is slightly higher than expected.

We finally give in Fig. 11 the accuracy of the neighborhood tables, which is equal to the percentage of nodes present in the table of a node that are really physical neighbors of this node: if f_{HELLO} is too low, then tables are not up to date and the accuracy drops. The Constant scheme does not adapt f_{HELLO} with the speed, and the accuracy thus quickly drops. Both the Optimal and TAP schemes are able to keep a correct accuracy of the tables because of the adjustment of d_{HELLO} observed in Fig. 10. The accuracy provided by TAP is slightly lower than with the Optimal scheme because d_{HELLO} is slightly higher than expected. While this may be easily corrected by using different adjustment functions, the provided accuracy is still sufficient for most applications.

VI. Conclusion and Open Issues

We presented TAP, an adaptive HELLO protocol for mobile ad hoc and sensor networks that simply estimates the neighborhood turnover and adjusts the HELLO frequency based on this observation, to aim at a given optimal frequency. Besides the fact that our protocol is simple to implement, it is especially well-tailored to standard mobile ad hoc and sensor networks since it does not rely on any specific hardware to achieve the adjustment. We theoretically computed the optimal turnover based on given assumptions, and experimentally showed that the TAP protocol provides a good accuracy when aiming at that optimal turnover under these assumptions.

There are some remaining open issues that we did not consider in this paper. One of them is about the timeout that should be used to remove deprecated neighbors from the table. We indeed focused on the frequency of HELLO messages and did not consider the problem of determining the optimal lifetime of a table entry. This problem is important since considering deprecated entries may lead to serious problems. Using a constant lifetime is not a good solution since it should depend on the speed of the nodes, just as the HELLO frequency. Another issue that we did not address is about networks with heterogeneous speed or driven by more realistic mobility models. A better physical layer model (*e.g.*, the lognormal shadowing model [9]) might also be considered since HELLO messages may then get lost before being received. We would like to further study the consequences of these more realistic assumptions and adapt our TAP protocol consequently.

Acknowledgments

This work was partially financed by the European Commission under the Framework 6 IST Project "Wirelessly Accessible Sensor Populations (WASP)".

References

[1] I. Chakeres and E. Belding-Royer. The utility of hello messages for determining link connectivity. In *WPMC*, 2002.
[2] B. Clark, C. Colbourn, and D. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1–3):165–177, 1990.
[3] T. Clausen, P. Jacquet, A. Laouiti, P. Mühlethaler, A. Qayyum, and L. Viennot. OLSR - Optimized Link State Routing Protocol, October 2003. RFC 3626.
[4] V. Giruka and M. Singhal. Hello protocols for ad hoc networks : Overhead and accuracy trade-offs. In *WoWMoM*, 2005.
[5] C. Gomez, A. Cuevas, and J. Paradells. A two-state adaptive mechanism for link connectivity maintenance in AODV. In *REALMAN*, 2006.
[6] M. McGlynn and S. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc networks. In *Mobihoc*, 2001.
[7] J. Moy. OSPF - Open Shortest Path First, March 1994. RFC 1583.
[8] C. Perkins, E. Belding-Royer, and S. Das. AODV - Ad hoc On-Demand Distance Vector Routing, July 2003. RFC 3561.
[9] L. Quin and T. Kunz. On-demand routing in MANETs: The impact of a realistic physical layer model. In *Ad Hoc Now*, 2003.
[10] G. Sawchuk, G. Alonso, E. Kranakis, and P. Widmayer. Randomized protocols for node discovery in ad hoc multichannel networks. In *Ad Hoc Now*, 2003.
[11] A. Troël. *Prise en compte de la mobilité dans les interactions de mobilité entre terminaux à profils hétérogènes*. PhD thesis, Université de Rennes 1, France, 2004. In french.
[12] G. Wattenhofer, G. Alonso, E. Kranakis, and P. Widmayer. Randomized protocols for node discovery in ad hoc, single broadcast channel networks. In *IPDPS*, 2003.