Mobility Prediction Based Neighborhood Discovery in Mobile Ad Hoc Networks

Xu Li, Nathalie Mitton, and David Simplot-Ryl

INRIA Lille - Nord Europe, Univ Lille Nord de France,
USTL, CNRS UMR 8022, LIFL, France
{Xu.Li,Nathalie.Mitton,David.Simplot-Ryl}@inria.fr

Abstract. Hello protocol is the basic technique for neighborhood discovery in wireless ad hoc networks. It requires nodes to claim their existence/aliveness by periodic 'hello' messages. Central to a hello protocol is the determination of 'hello' message transmission rate. No fixed optimal rate exists in the presence of node mobility. The rate should in fact adapt to it, high for high mobility and low for low mobility. In this paper, we propose a novel mobility prediction based hello protocol, named ARH (*Autoregressive Hello protocol*). Each node predicts its own position by an ever-updated autoregression-based mobility model, and neighboring nodes predict its position by the same model. The node transmits 'hello' message (for location update) only when the predicted location is too different from the true location (causing topology distortion), triggering mobility model correction on both itself and each of its neighbors. ARH evolves along with network dynamics, and seamlessly tunes itself to the optimal configuration on the fly using local knowledge only. Through simulation, we demonstrate the effectiveness and efficiency of ARH, in comparison with the only competitive protocol TAP (Turnover based Adaptive hello Protocol) [9]. With a small model order, ARH achieves the same high neighborhood discovery performance as TAP, with dramatically reduced message overhead (about 50% lower 'hello' rate).

Keywords: Neighborhood Discovery, Hello Protocol, MANET

1 Introduction

In mobile ad hoc networks (MANET), a fundamental issue for many network operations is *neighborhood discovery*, where each node finds out which other nodes are within its communication range (i.e., neighboring it). Having up-to-date neighborhood knowledge, a node is able to make proper networking decisions. The basic technique for neighborhood discovery is *hello protocol*. The first hello protocol was described in the Open Shortest Path First (OSPF) routing algorithm [13] for IP networks. In OSPF, nodes exchange 'hello' message carrying required information periodically at fixed frequency. When node a receives 'hello' message from node b, it creates an entry for b, or update the existing entry of b, in its neighbor table depending on whether or not b is a new neighbor. If a does not receive 'hello' message from b for a predefined amount of time, it will consider b has left its neighborhood and remove b's entry from the table. The protocol enables nodes to maintain neighborhood information in the presence of node mobility and dynamic node addition and removal.

1.1 Motivation

The usefulness of a hello protocol highly depends on the transmission rate of 'hello' message [3]. It is not a trivial task to choose proper rate. If the rate is too high with respect to node mobility, precious communication bandwidth and energy will be wasted for unnecessarily frequent transmissions. On the other hand, if it is too low, neighbor tables will quickly become out-of-date, leading to failure in other network operations and thus bandwidth waste and energy loss in those other operations. An optimal hello protocol maintains accurate neighborhood information at minimal 'hello' transmission rate. Unfortunately, no constant rate can always remain optimal in dynamic MANET. The rate should evolve together with the network along time for the best performance.

In literature, a majority of MANET protocols adopt hello protocol in one form or another as a building block. But the impact of hello protocol on the network performance has not been studied until recently in [11, 14]. Existing hello protocols all have noticeable limitations and weaknesses. They are inferior for possible applications, compared to the protocol proposed in this paper, for a variety of reasons, e.g., assumption of static networks, use of fixed 'hello' frequency, requirement of extra input parameters. A survey of these previous work will be presented later, in Sec. 2. The importance of the topic and the incompleteness and insufficiency of relevant research motivate our work presented here.

1.2 Contributions

We address the problem of neighborhood discovery in MANET by proposing a novel mobility prediction based hello protocol, named ARH (Autoregressive Hello protocol). This protocol adaptively adjusts 'hello' message rate to the optimal value according to time-varying node mobility. The idea is to let each node constantly estimate its neighbors' position using past location reports and transmit 'hello' message (reporting its current position) when its own location estimated by a neighbor is not accurate enough. For ease of presentation, terms 'predict' and 'estimate' are used interchangeably in the sequel.

More specifically, each node n samples its position at regular intervals and considers the position samples as a time series of data. From this series, it computes two associated time series, respectively for its moving direction and velocity. It applies autoregressive (AR) modeling on the two series and obtains a mobility model for itself. Each neighboring node m builds and maintains an identical mobility model for n using n's previous location reports (carried by 'hello' message) and estimates n's position. In the meantime, n predicts its own mobility (moving direction and velocity) and position using the AR-based mobility model so that it has the same location estimates for itself as its neighbors. It transmits a 'hello' message carrying its current position only when the predicted position leads to false topological change (which means its location estimates by neighboring nodes are no longer accurate). Further, AR-based 'hello' frequency prediction is suggested to detect neighborhood change caused by node removal/departure and improve the algorithm performance.

Through extensive simulation, we study the performance of ARH using real mobility trace data, in comparison with the best known competitive hello protocol TAP (Turnover based Adaptive hello Protocol) [9]. Our simulation results indicate that both protocols require a short learning curve to stabilize. Their learning curves have roughly equal length, $20 - 30$ seconds. Once passing the learning curve, they stabilize to the same high neighborhood discovery performance, about 96% accurately reflecting true neighborhood situation. In particular, ARH results in dramatically lower 'hello' frequency than TAP, i.e., around 50% less message overhead, therefore saving both bandwidth and energy.

The rest of the paper is organized as follows. We review related work in Sec. 2 and introduce autoregressive modeling in Sec. 3. We present ARH in Sec. 4 and report our simulation study in Sec. 5, followed by the closing remarks in Sec. 6.

2 Related work

In [10], the authors considered a static network whose size is known a priori, and they aimed to reduce the overall energy consumption for communication. Time is slotted. At the beginning of each time slot, a node chooses with a predefined probability p_{state} to enter one of the three states: transmitting ('hello' message), listening, and sleeping. The sum of the probabilities for the three states is 1. The authors studied the optimal values for p_{state}. In dynamic networks such as MANET, where node mobility and node addition/removal are present, finding optimal p_{state} remains to be an open problem.

Similar to [10], three protocols RP, LP, and SP are presented for static networks in [1, 15] rather than MANET. A node can be either in talking state or in listening state, and it can stay in a state for a random period. In RP, a node at each time slot enters talking state with probability p and listening state with probability $1-p$. In LP, if current state is talking, the next state will be listening; otherwise, it will make a random decision as in RP. In SP, if the current and previous states are different, the node will back off for a while. The backoff period is modeled as a uniform random variable with values in a predefined range. After this period, the node sends a message back to the original sender.

In [8], a two-state hello protocol is presented. In this protocol, there are two different 'hello' message frequencies, 1s for low-dynamic network (default frequency) and 0.2s for high-dynamic network, whose selection is however not justified. Which frequency to use depends on two factors: Time to Link Failure (TLF) and Time Without link Changes (TWC). Each node estimates TLF and TWC among its neighborhood based on past neighborhood change. If TLF is smaller than a threshold, the system switches to the High dynamics state. When TWC becomes greater than another threshold, the mechanism switches back to the Low dynamics state. This protocol alternates only between two fixed transmission rates. Its adaptation to network dynamics is obviously very limited.

In [7], three hello protocols are presented. In an adaptive protocol, a node transmits 'hello' message if its travel distance is beyond a threshold since last transmission. Transmission frequency is additionally subject to predefined a minimum value (enabling static nodes to transmit) and a maximum value (preventing 'hello' message storm). This protocol may cause unnecessary transmissions, e.g., when the node moves along a small circle. In a reactive protocol, a node starts, before sending a data packet, neighborhood discovery where it transmits 'hello' message and expects a reply from each neighbor. If no reply is received within a predefined period of time, the process is repeated, up to a maximum number of times. This protocol brings large delay into data communication and is vulnerable to high mobility. In an event-based protocol, fixed 'hello' rate is used. However, a transmission may be skipped if no communication activity is observed in previous 'hello' interval. As such, it is possible that some nodes moving from 'quiet' area to 'quiet' area are never discovered.

In [9], the authors defined turnover ratio as the ratio of the number of new neighbors to the total number of neighbors during a time period Δt. They studied optimal (expected) turnover r_{opt}, and concluded that node velocity does not have any impact on r_{opt} and that r_{opt} is related only to 'hello' frequency and communication radius. They then suggested to adjust 'hello' rate toward the optimal value for obtaining the optimal turnover (i.e., expecting to discover all new neighbors). Based on this idea, a protocol named TAP (Turnover based Adaptive hello Protocol) is presented. In the protocol, nodes initially transmit 'hello' message at a default frequency. Turnover is checked periodically, everytime when 'hello' message is transmitted; the 'hello' rate is immediately adjusted by certain formula that takes current turnover and optimal turnover as input.

TAP is to our knowledge the only adaptive protocol comparable to our proposed ARH here. Other protocols have various obvious weaknesses as summarized above. As we will see in the sequel, ARH assumes location-awareness on each node while TAP does not. This assumption limits ARH to the scenarios where location information is available. But nevertheless, by making good and reasonable use of it ARH outperforms TAP to a great extent.

3 Preliminaries

The ARH protocol to be proposed later in the paper uses autoregressive (AR) model for mobility prediction. For a better understanding of ARH, in this section we briefly introduce AR modeling.

3.1 Autoregressive model

Autoregressive model (AR) [2] is a tool for understanding and predicting a time series of data. It estimates the current term z_k of the series by a linear weighted sum of previous p terms (i.e., observations) in the series. The model order p is generally less than the length of the series. Formally, AR(p) is written as

$$z_k = c + \sum_{i=1}^{p} \phi_i z_{k-i} + \epsilon_k \quad , \tag{1}$$

where c is a constant standing for the mean of the series, φ_i autoregression coefficients, and ϵ_k zero-mean Gaussian white noise error term. For simplicity, the constant c is often omitted. Deriving AR(p) involves determining the coefficients φ_i for $i \in [1..p]$ that give a good prediction. It can be done through computation-intensive complex calculus, for example, by the Yule-Walker equations [5] or Burg method [4]. The model can be updated continuously as new samples arrive so as to ensure accuracy, or it may be recomputed only when the predication is too far from the true measurement (beyond certain threshold).

3.2 Simplified autoregressive model

In [6], a simplified AR model is proposed. This model requires to be fed with its own estimates as samples to generate new estimates. Although its order p may be larger than 1, it can be updated using only a single sample (in contrast to the whole sample set in the traditional model). Model update needs only trivial calculus, greatly reducing the requirement on the computational power on the nodes that implement the model. Thus it becomes embeddable on tiny and computationally weak devices. We elaborate on this simplified model below.

Recall that p is the model order. A single sample is needed for initializing the model. At initialization, i.e., at time $k = 0$, we set $\phi_i = \frac{1}{p}$ for all $i \in [0..p]$ and $z_i = z_0$ for all $i < 1$. Let the estimate error be $e_k = z_k - \hat{z}_k$ at time $k > 0$. When e_k is too big, the ϕ coefficients need to be adjusted.

Assume that e_k results from using estimates as samples. We spread the error evenly over the p samples and the current estimate z_k. Each sample has error $\frac{1}{p+1}e_k$. Then the estimate with maximum error will be $\hat{z}_k = z_k - e_k + \frac{1}{p+1}e_k = z_k - \frac{p}{p+1}e_k$. We update ϕ coefficients by enforcing the following equation:

$$\sum_{i=1}^{p} \phi_i z_{k-i} = z_k - \frac{p}{p+1} e_k \quad . \tag{2}$$

We update ϕ coefficients iteratively. We first take ϕ_p as variable and the others as constant. We compute new ϕ_p (denoted by ϕ'_p) by solving the above equation. Then we plug ϕ'_p into the equation. Meanwhile, we need to increase the right side by $\frac{1}{p+1}e_k$ because the use of ϕ'_p will bring that much more error. After that, we compute a new ϕ_{p-1}, and so on. We update the coefficients one at a time in this way from ϕ_p downto ϕ_1. We compute in this order because the farther away a sample is in time, the less important its coefficient. The update sequence is shown below:

$$\phi'_p = \frac{1}{z_{k-p}}(z_k - \frac{p}{p+1}e_k - \sum_{i=1}^{p-1} \phi'_i z_{k-i} - c) \qquad ;$$

$$\vdots$$

$$\phi'_j = \frac{1}{z_{k-j}}(z_k - \frac{j}{p+1}e_k - \sum_{i=1}^{j-1} \phi'_i z_{k-i} - \sum_{i=j+1}^{p} \phi_i z_{k-i} - c) \quad ;$$

$$\vdots$$

$$\phi_1' = \frac{1}{z_{k-1}}(z_k - \frac{1}{p+1}e_k - \sum_{i=2}^{p}\phi_i z_{k-i} - c) \quad .$$

At last, value c can be updated such as $z_k = \hat{z_k}$:

$$c = z_k - \sum_{i=1}^{p}\phi_i' z_{k-1}$$

4 Autoregressive Hello Protocol

In this section, we present a novel *Autoregressive Hello protocol* (ARH) for mobile ad hoc networks. The protocol records historical location information and applies AR modeling to predict node mobility (moving direction and velocity). It determines when to transmit 'hello' message according to prediction accuracy. We start with our assumptions. Then we give an overview of the algorithm, followed by elaboration on individual algorithmic building blocks.

4.1 Assumptions

Nodes have locomotion by being attached to, for example, human being or animal, and are free to move. They may be constrained computing devices. They are aware of their own geographic location $X = [x, y]^T$ by equipped GPS devices or other localization means. For simplicity, we assume that time is synchronized. However, this time synchronization assumption is by no means necessary and can be easily relaxed, as explained at the end of Sec. 4.2.

Nodes independently divide the time domain into slots of equal length λ. The global parameter λ is a positive real number, the same for all the nodes. It defines position sampling rate. That is, each node samples its position $X_i = [x_i, y_i]^T$ at the beginning of every time slot i. Node movement is described by direction θ_i and velocity s_i. We assume that λ is selected small enough such that a node's moving direction does not change more than 2π in each time slot. Nodes have equal communication radius r_c. Each of them is associated with a unique identifier (ID) such as MAC address or manufacturer serial number by which it can be distinguished from others.

4.2 Algorithm overview

After being added into the network, each node n samples its position X at the beginning of every time slot. The position samples X_i, $i = 0, 1, 2, \cdots$ constitute a time series. n applies AR modeling to this time series to predict its own mobility and future position. Considering potential computational weakness of nodes, here we choose to use the simplified AR model [6], as detailed in Sec. 4.3. At time slot k, node n estimates its position at next time slot $k+1$, and the position estimate is denoted by $\hat{X}$. As soon as n obtains the ground truth X_{i+1}, it checks position estimate error. If the error is acceptable, i.e., $\hat{X_{i+1}}$ is close enough to X_{i+1}, it will update the AR-based mobility model by feeding $\hat{X}_{k+1}$ back to it as sample (this is the feature of simplified AR modeling); otherwise, it will update the mobility model using the real sample X_{k+1}. In the latter case, n also transmits 'hello' message that carries X_{k+1} and its ID. Section 4.4 explains how to evaluate position estimate error in detail.

Upon receiving 'hello' message from n, each neighboring node m initializes an AR-based mobility model for n using the enclosed position sample of n if it has not yet done so, and updates the model otherwise. Between two successive 'hello' messages, m uses this locally maintained mobility model to estimate n's position and keep updating the model using position estimates. As such, m makes the same position estimates for n as n does. This justifies why n transmits 'hello' message according to the accuracy of position estimate. Note that n must transmits 'hello' message right after getting its first position sample in order to enable parallel model initialization and identical position estimation by m.

In Sec. 4.1, we assumed time synchronization for simplicity. The algorithm however is not dependent of this assumption. When time is not synchronized, m and n may not have exactly the same AR model, thus identical location estimates, for n. To overcome this problem, n can simply encapsulate its latest AR mode (all the coefficients and the entire sample set) rather than just the last position sample into each 'hello' message. Then m easily synchronizes (replaces) its locally maintained AR model for n with the one maintained by n itself.

4.3 Mobility modeling and position estimation

At each node n, the position samples $X_0, X_1, X_2, \cdots$ constitutes a time series. From it, two associated time series are computed, one for direction θ and the other for velocity s, describing the mobility of n over time. At time slot $i > 1$, direction is computed as

$$\theta_i = \begin{cases} \arctan(\frac{y_i - yi{-}1}{x_i - x_{i-1}}) & \text{if } x_i > x_{i-1} \\ \arctan(\frac{y_i - yi{-}1}{x_i - x_{i-1}}) + \pi & \text{if } x_i < x_{i-1} \\ \frac{1}{2}\pi & \text{if } x_i = x_{i-1} \text{ and } y_i > y_{i-1} \\ -\frac{1}{2}\pi & \text{if } x_i = x_{i-1} \text{ and } y_i \leq y_{i-1} \end{cases} \tag{3}$$

and speed

$$s_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \quad . \tag{4}$$

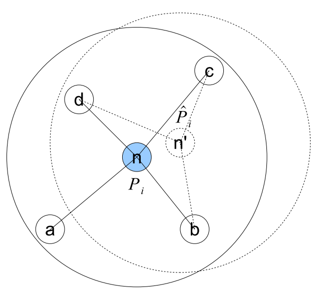
Future direction $\hat{\theta}_{i+1}$ and speed $\hat{s}_{i+1}$ are predicted in order to deduce a position estimate of

$$\hat{X}_{i+1} = \begin{bmatrix} \hat{x}_{i+1} \\ \hat{y}_{i+1} \end{bmatrix} = \hat{X}_i + \begin{bmatrix} cos(\hat{\theta}_{i+1}) \\ sin(\hat{\theta}_{i+1}) \end{bmatrix} \times \hat{s}_{i+1} \quad . \tag{5}$$

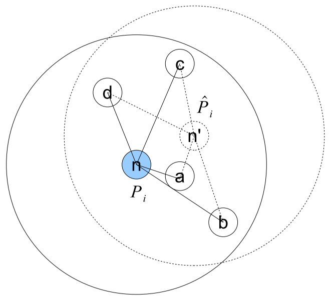
The s series can be directly applied in Eqn. 1; whereas, θ is a cyclic value in $[0, 2\pi]$ and needs an adaptation. We thus translate the θ series into $[0, \infty[$ by introducing a t series computed as follows:

$$t_i = \begin{cases} \theta_1 & \text{if } i = 1 \\ \theta_i + 2k\pi & \text{otherwise, with } k \in \mathbb{N} \text{ s.t.} \\ & |t_i - t_{i-1}| \text{ is minimum.} \end{cases} \tag{6}$$

Note that the sampling rate $1/\lambda$ is chosen such that between two successive samples, the moving direction of node n can not vary more than 2π. Equation 1



(a) False edge deletion



(b) False ordered adjacency

Fig. 1. Evaluating location estimate

is thus applied on the t series for estimating $\hat{t}_{i+1}$. Finally, $\hat{\theta}_{i+1}$ is obtained from $\hat{t}_{i+1}$ as follows:

$$\hat{\theta}_{i+1} = \hat{t}_{i+1} - \lfloor \frac{\hat{t}_{i+1}}{2\pi} \rfloor \times 2\pi \quad . \tag{7}$$

Summarizing, the mobility model of node n is composed of two simplified AR models, one for s series and one for t series. The two series are processed independently. Their corresponding AR models evolve along time by being input with estimates produced by themselves and are periodically corrected by being fed with new position samples, as described in Sec. 3.2.

4.4 Evaluating position estimates

When evaluating position estimates, a naive way is to use a predefined error threshold δ. That is, node n transmits 'hello' message at time slot i if $|\hat{X}_i - X_i| > \delta$. This method introduces additional parameter δ, and renders the algorithm performance subject to the selected parameter value. If δ is too small, 'hello' message frequency may unnecessarily increase; if it is too large, each node could have a stale or ineffective neighborhood map, which degrades the performance of other networking protocols, for example, routing protocols. The best value of δ depends on global network conditions, which are usually beyond the knowledge available to each node. To enable ARH to work in all mobility scenarios and adjust itself seamlessly to one of optimal protocols for any particular mobility, we propose a parameterless evaluation method.

Specifically, at each time slot i, node n maintains two neighborhood subgraphs G_n and $\hat{G}_n$ of its neighbors and itself using its true location X_i and location estimate $\hat{X}_i$, respectively. In both graphs, neighbors' locations are estimates. If an edge between n and neighbor m in G_n disappears in G'_n, we say there is a *false edge deletion* in G'_n. Node n constructs a circular order among its incidental edges according to their appearance sequence in certain direction, either clockwise or counter-clockwise. If the order is different in the two graphs, we say there is a *false ordered adjacency* in G'_n. Node n transmits 'hello' message if G'_n contains either false edge deletion or false ordered adjacency. This topology-based evaluation method does not require any pre-set parameter and provides a great degree of flexibility.

Figure 1 illustrates the two false situations. Node n has 4 neighbors. Big circles indicate its communication range. G_n is represented by solid links, while G'_n is shown by dashed links. In Fig. 1(a), false edge deletion happens to edge an; in Fig. 1(b), false ordered adjacency involves edges an and bn.

4.5 Detecting neighborhood change

If a node does not transmit 'hello' message, its neighbors will consider their location estimates for that node are correct. They are not able to distinguish this situation from node failure, where the node is malfunctioning and will never transmit 'hello' message. If a new neighbor arrives without sending 'hello' message (it is possible when the current set of neighbors of that node all have acceptable location estimates), the node will not be able to know it or update its neighborhood. Additional mechanism is needed for detecting neighborhood change and improving the algorithm performance.

A straightforward method is to use constant low frequency 'hello' message, regardless of node mobility. While 'hello' message loss implies node departure, forced 'hello' transmission increases the chance of new neighbor discovery. However, this method reserves our efforts so far and brings us back to the original problem – how to determine the best 'hello' frequency. It is not a solution.

Here we propose a 'hello'frequency predication based solution, similar to the main body of ARH. We let each node n use an AR model to predict the inter-arrival time of 'hello' message of each neighbor m. If 'hello' message does not arrive on time for more than a threshold number α of times, then n considers that m has left its neighborhood. Node m builds an AR model for itself to monitor the inter-departure time of its own 'hello' message. The model is approximately equal to the model that n builds for it. So, m will follow this model for 'hello' message transmission. It transmits next 'hello' message within α number of successive predicated intervals, even if it does not need to do so by the main algorithm.

With some additional computation overhead on each node, this solution adapts to node mobility and ensures detection of leaving neighbors. It still does not guarantee new neighbor detection since the new neighbor may possibly not transmit any message before moving away. But nevertheless, this is a problem for any 'hello' protocol. In fact, it is not possible to guarantee new neighbor detection because 'hello' message is transmitted at discrete time instants.

5 Performance evaluation

In this section, we evaluate our new hello protocol ARH through simulation, in comparison with TAP [9] that is to date the most efficient adaptive hello protocol known. In contrast to ARH, TAP supposes that nodes are not equipped with GPS-like devices and thus are not aware of their position.

As the objective of a hello protocol is neighborhood discovery, the protocol must be able to keep the consistency of neighborhood tables among nodes at minimal 'hello' frequency (i.e., message overhead). Thus in addition to 'hello' frequency, we use two evaluation metrics: *neighborhood accuracy* and *neighborhood error*. Assuming that $N(u)$ is the set of *actual* neighbors of a node u, and

$N'(n)$ the set of neighbors *known* to u (*i.e.* whose identifier is present in its neighborhood table), these two metrics are defined below. From their definition, notice that $\mathrm{acc}(u) + \mathrm{err}(u)$ is not necessarily equal to 1.

Definition 1. *Neighborhood accuracy $acc(u)$ is the proportion of actual neighbors of node u that have been indeed detected by u.*

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

Definition 2. *Neighborhood error $err(u)$ measures both how many neighbors of node u have not been detected, and how many "false neighbors" remain in its neighborhood table (*i.e. *old neighbors that have not been removed).*

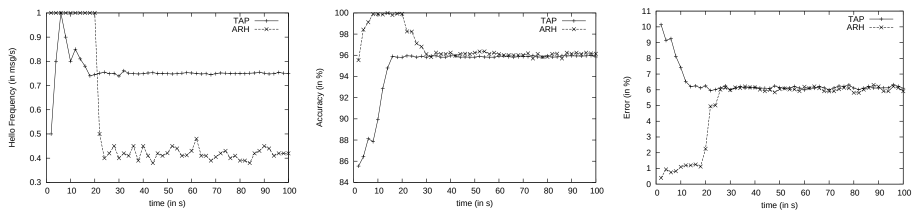
$$err(u) = \frac{|N(u) \backslash N'(u)| + |N'(u) \backslash N(u)|}{|N(u)|} \times 100.$$

We implemented ARH and TAP using WSNet/Worldsens [16] event-driven simulator, with IEEE 802.11 DCF being implemented at MAC layer and free space propagation model at physical layer. Packet collision and contention were also implemented. In ARH, we set AR model order $p = 5$, position sampling interval $\lambda = 2s$ and $\alpha = 5$ (the order of the AR model for 'hello' frequency predication; see Sec. 4.5). We generated nodal mobility trace based on logs obtained from real experiments on pedestrian runners. Node moving speed was thus spread around a mean value of 1 meter per second. Mobility trace data can be found in [12]. A varying number of nodes (from 100 to 600) were uniformly randomly deployed in a 1000×1000 square region. These nodes have the same transmission range $r_c = 100m$. For each setting, we conducted 50 simulation runs to obtain average results.

5.1 Performance in relation with time

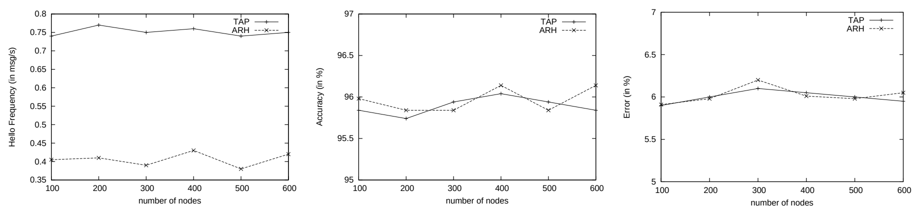
We first evaluate the performance of ARH and TAP along time with a fixed number (100) of nodes. Figure 2(a) plots the message overhead, i.e., 'hello' frequency, of both protocols. In ARH, each node first sends 'hello' message with higher frequency so as to train its mobility model on neighboring nodes. After the training period ($20 - 30$ seconds in our simulation), the number of 'hello' messages sent greatly decreases to stabilize at about the half of messages sent by nodes with TAP. This indicates that ARH is much less costly than TAP in both bandwidth usage and energy consumption.

Figures 2(b) and 2(c) compare neighborhood discovery performance for ARH and TAP. Starting from a low performance point, TAP achieves increasingly better accuracy (resp. lower error) since it adapts 'hello' frequency till achieving a stabilized turnover and better performance. Contrarily, in ARH each node at first sends a lot of 'hello' messages for mobility model training. Frequent 'hello' transmission leads to very good neighborhood accuracy. After the mobility model is fully trained, satisfactory location prediction can be computed, and the number of 'hello' messages sent decreases, resulting in a slight decrease in accuracy and a slight increase of error. After the initial short training period, both protocols stabilize with the same excellent performance, around 96% of accuracy and 6% error in neighborhood tables.



(a) ‘Hello’ frequency. (b) Accuracy. (c) Error.

Fig. 2. Performance in relation with time.



(a) ‘Hello’ frequency. (b) Accuracy. (c) Error.

Fig. 3. Performance in relation with number of nodes.

5.2 Performance in relation with number of nodes

Figure 3 plots the simulation results obtained with regards to different number of nodes after the protocols ARH and TAP pass their initial learning curve. We observe consistent performance as the number of nodes changes. That is to say, the number of nodes has no impact on the protocol performance. Indeed, in TAP, the adaptation of ‘hello’ rate is related to the mean speed of nodes and thus is not impacted by the node density. In ARH, ‘hello’ rate depends on the error that a node detects between its position estimate and its real position, both independent from the number of neighbors and the total number of nodes.

6 Conclusions

We proposed a novel *Autoregressive Hello protocol* (ARH) for neighborhood discovery in mobile ad hoc networks (MANET). Each node predicts its neighbors mobility and position by autoregressive (AR) modeling, based on historical location reports; it also predicts its own mobility and position using position samples in the same way. The node updates its location among neighbors when the its own location estimate leads to false topology change in its neighborhood. Each location update corresponds to a ‘hello’ message transmission. Simulation results indicate that ARH achieves as high neighborhood discovery performance as the best-known algorithm TAP [9], at dramatically reduced ‘hello’ rate (about 50% smaller). This is a great advantage in wireless communications since more message transmissions indicate more bandwidth usage and more energy consumption. It is at the cost of an additional requirement for location-awareness on each node. We conclude that ARH is a highly-efficient alternative to TAP when location information is readily available.

In our current work, the AR model order p was set to a small value 5. It is possible that a smaller p (thus yet smaller storage overhead at individual nodes) leads to similar neighborhood discovery performance. On the other hand, if a larger-valued p (thus increased model complexity and accuracy) is used, ARH would possibly have higher neighborhood discovery performance. In addition, the selection of p could be subject to the mobility nature of nodes. For example, it is probably necessary to select p differently when nodes are vehicles rather than human. In the future, we will study the trade-off between model complexity and algorithm performance in different mobility scenarios.

Acknowledgements

This work was partially supported by CPER Nord-Pas-de-Calais/FEDER Campus Intelligence Ambiante and the ANR SensLAB project.

References

1. Alonso, G., Kranakis, Wattenhofer, G., E., Widmayer, P.: Probabilistic protocols for node discovery in ad hoc, single broadcast channel networks. In: Proc. IEEE IPDPS (2003)
2. Box, G., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control (4th edition), Wiley (2008)
3. Chakeres, I., Belding-Royer, E.: The utility of hello messages for determining link connectivity. In: Proc. WPMC, pp. 504-508 (2002)
4. Collomb., C.: Burg's Method, Algorithm and Recursion. Internet resource.
5. Eshel, G.: The Yule Walker Equations for the AR Coefficients. Internet resource.
6. Ghaddar, A., Razafindralambo,T., Simplot-Ryl, I., Simplot-Ryl, D., Tawbi, S.: Towards Energy-Efficient Algorithm-Based Estimation in Wireless Sensor Networks. In: Proc. MSN (2010).
7. Giruka, V., Singhal, M.: Hello protocols for ad hoc networks: Overhead and accuracy trade-offs. In: Proc. IEEE WoWMoM, pp. 354-361 (2005)
8. Gomez, C., Cuevas, A., Paradells, J.: A two-state adaptive mechanism for link connectivity maintenance in AODV. In: Proc. REALMAN, pp. 98-100 (2006)
9. Ingelrest, F., Mitton, N., Simplot-Ryl, D.: A Turnover based Adaptive HELLO Protocol for Mobile Ad Hoc and Sensor Networks. In: Proc. IEEE MASCOTS, pp. 9-14 (2007)
10. McGlynn, M., Borbash, S.: Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc networks. In: Proc. ACM MobiHoc, pp. 137-145 (2001)
11. Medina, A., Bohacek, S.: A performance model of neighbor discovery in proactive routing protocols. In: Prof. ACM PE-WASUN, pp. 66-70 (2010)
12. MobilityLog. `http://researchers.lille.inria.fr/~mitton/mobilitylog.html`
13. Moy, J.: OSPF - Open Shortest Path First. RFC 1583 (March 1994)
14. Razafindralambo, T., Mitton, N.: Analysis of the Impact of Hello Protocol Parameters over a Wireless Network Self-Organization. In: Proc. ACM PE-WASUN, pp. 46-53 (2007)
15. Sawchuk, G., Alonso, G., Kranakis, E., Widmayer, P.: Randomized protocols for node discovery in ad hoc multichannel networks. In: Proc. AdHoc-Now (2003)
16. WSNet/Worldsens Simulator. `http://wsnet.gforge.inria.fr/`