Institut Eurécom
Department of Mobile Communications
2229, route des Crêtes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-07-195

Kinetic Graphs: A Framework for Capturing the Dynamics of Mobile Structures in MANET

May 10th, 2007

Jérôme Härri, Christian Bonnet and Fethi Filali

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : {Jerome.Haerri,Christian.Bonnet,Fethi.Filali}@eurecom.fr

[1]Institut Eur´ecom's research is partially supported by its industrial members: BMW Group Research & Technology - BMW Group Company, Bouygues T´el´ecom, Cisco Systems, France T´el´ecom, Hitachi Europe, SFR, Sharp, STMicroelectronics, Swisscom, Thales.

Kinetic Graphs: A Framework for Capturing the Dynamics of Mobile Structures in MANET

Jérôme Härri, Christian Bonnet and Fethi Filali

Abstract

In Mobile Ad Hoc Networks (MANET), structures are built in order to improve network resource for broadcast or routing. Inspired by graph theory, most of those structures are built using fixed criteria, such as degree or distance, yet based only on local information. However, mobility is altering the optimality of these localized structures, as the criteria is dynamically varying with time. Since those criteria do not change homogeneously, a periodic maintenance wastes network resource, as it inefficiently acquires new values in an disorganized way. In this paper, we introduce the concept of Kinetic Graphs as a method to capture the dynamics of mobile structures and accordingly develop an efficient maintenance. Unlike the static counterpart, kinetic graphs are assumed to be continuously changing and edges are represented by time-varying weights. Kinetic graphs are a natural extension of static graphs and provide solutions to similar problems, such as convex hulls, spanning trees or connected dominating sets, but for continuously mobile networks. We therefore propose a framework for implementing Kinetic Graphs for topology management in MANET, which consists of four steps: (i) a natural representation of the trajectories, (ii) a common message format for the posting of those trajectories, (iii) a time varying weight for building the kinetic graphs, (iv) an aperiodic neighborhood maintenance. In particular, we propose two time-varying weights which could be used to directly adapt graph theory algorithms to the kinetic aspect, and also illustrate two successful applications to broadcasting and topology control for MANET.

Index Terms

Kinetic graph, mobility management, design methodologies, mobile structure, time varying weights, broadcast, MANET.

Contents

List of Figures

1 Introduction

Kinetic Graphs is a particular class of graphs aimed at maintaining the attributes of interests, such as spanner, Voronoi tesselation, or convex hull, in graphs with moving vertices. In regular graphs subject to mobility, the position of the vertices are updated, and then the attribute of interest is recomputed. Since the attributes do not change homogeneously in the whole graph, it is hard to find a refreshing rate, which optimizes the computing cost of the reconfiguration. Unlike this fixed step approach, kinetic graphs perform an event-driven simulation where only events relevant to the vertex and the attributes are generated and processed. Kinetic graphs are therefore ”mobility proactive”, as the structure is updated only when an attribute is changed, which effectively alters the graph.

Kinetic graphs are also a special application to a more generic approach called the *Kinetic Data Structures (KST)* introduced by Bash et al. [1]. In the KDS framework, it is assumed that trajectories of objects are known, but the algorithm does not know when trajectories will change. This is a direct use of mobility predictions applied to data structures, where two goals are targeted: the *optimality* with respect to the attribute, and the *maintenance efficiency*. This topic has been widely studied in various area such as mobile facility locations [2], clustering and routing [3] or shortest path [4]. A survey on KDS can be found in [5].

Mobile Ad Hoc Networks (MANETs) are an emergent concept in view of infrastructure-less communication. It appeared clear to the community that graph theory heuristics such as Connected Dominating Sets (CDS), Convex Hulls, or Minimum Spanning Trees (MST), could be applied to various objectives such as Broadcasting, Routing, or Topology Control. However, due to the limited capability of processing power, storage and energy supply, many conventional algorithms are too complicated to be implemented in wireless ad hoc networks. Thus, wireless ad hoc networks require efficient distributed algorithms with low computation complexity and low communication complexity. More importantly, distributed algorithms should also be *localized*, as each node running the algorithm could only uses the information of nodes within a constant number of hops away. However, localized algorithms are difficult to design or even sometime impossible. The community yet started to work on adapting decades of graph theory outbreaks to solve several challenging questions such as *localized Delaunay triangulation [6]*, *localized spanner [7]*, *localized spanning trees [8]* and even to broadcasting [9, 10]. A survey on Localized approaches for broadcasting and topology control may be found in [11], and for routing in [12].

When looking at the state of the art achievements in the approaches described in the previous paragraphs, we can see a straightforward interweaving aspect. *Localized Protocols* solve performance issues of *Kinetic Structures*, and conversely, *Kinetic Structures* provide solutions to handle mobility for *Localized Protocols* for MANETs. Unfortunately, these two communities have worked quite independently, and only few works [3, 4] appeared to have seen the potential benefits from joining both worlds.

In this paper, we propose to specifically regroup those two research areas and introduce the *Kinetic Graphs* framework, which consists of a neighborhood discovery process, a trajectory modeling and kinetic link weight extraction, and finally an aperiodic neighborhood maintenance. By following the framework, any localized ad hoc network protocol could be adapted to the kinetic approach, including Topology Control, Broadcasting and Routing. We first provide a general description of how the trajectories are modeled, then how the structure is initially built and finally, how it is maintained. We emphasis that our objective is to suppress the periodic beaconing process widely used by almost all protocols in order to adapt to mobility. Then, we discuss two different kinetic criteria that could be easily adapted in most of the protocols for MANETs. Finally, we show two examples where this approach has been successfully adapted.

The rest of this paper is organized as follows. In Section 2, we first describe our motivation for studying Kinetic Graphs. Then, in Section 3, we propose a possible trajectory representation, while in Section 4, we present the neighborhood discovery phase involving a common packet format and a data compression for geo-localization information. Section 5 describes two possible time varying link weights for the construction of Kinetic Graphs, and Section 6 provides heuristics in order to aperiodically maintain the neighborhood. Finally, Section 7 provides application examples, and Section 8 concludes the work.

2 Motivations and Objectives

Wireless Ad Hoc Networks are an extreme configuration of wireless networks, without a fixed or wired infrastructure, and where terminals are self-configuring in order to provide distributed multi-hop wireless communications. The lack of infrastructure or coordinator favors chaotic situations generating a large waste of already critical resources. Indeed, studies have shown that uncoordinated transmissions was not an efficient method to transmit data in wireless ad hoc networks as it creates an effect known as the ”broadcast storm” problem. Similarly to a chaotic crowd, if everyone talks at the same time, no one can listen to anyone and everyone will have wasted its energy trying to communicate, no matter how loud they tried. This remark was one of the justification for the development of structures in ad hoc networks in order to improve data diffusion and energy consumption. Algorithms creating backbones and coordination have yet to comply with two major assumptions: they must be *distributed* and *localized*. Indeed, ad hoc networks potentially being composed of a very large set of uncoordinated nodes, decisions should be taken at each node based on local information. As no omniscient coordinator exists in ad hoc networks, graph theory structures cannot be efficiently adapted. A successful new research area therefore appeared aimed at generating structures based on local information only, yet coming close to the optimal graph theory version: *Distributed Systems*. A wide range of solutions have been successfully developed to improve broadcast, transmission power or coordination.

2.1 The Burden of Mobility

Despite the efficient distributed solutions obtained in wireless ad hoc networks, a major assumption have been ignored: *mobility*. Advocates of distributed solutions argued that mobility could be simply seen as a maintenance duty, which is optimally kept local. Yet, this maintenance may also be seen as a waste of resource and a generation of instability and delays. Indeed, locality is not sufficient in order to efficiently maintain structures in wireless ad hoc networks, as mobility makes the structure adapted to past configurations, thus dooming them to inefficiency. Moreover, depending on the dynamics of the network, the local maintenance also becomes resource greedy. It has been notably observed that the OLSR routing protocol based on the Multipoint Relaying structure was not adapted to highly mobile networks such as vehicular networks, and more generally that proactive routing protocols consumed a significant energy and network resource dedicated to the maintenance of their routing tables.

Twenty years ago, the concept of *Kinetic Data Structure (KDS)* was developed as a mean to efficiently adapt data structures to mobile objects and attributes. Observing that all then-known data structures were not directly applicable to a configuration of objects moving continuously, the objective was to benefit from the coherence and continuity in the motion of the points to gain efficiency. This could be achieved by the important hypothesis of objects knowing their trajectories and those of others. One possible application of KDS was to efficiently adapt spanning trees or other graph algorithms to mobile configurations. Beside the obvious requirement for the kinetic structure to fit the real topology, another performance measure was the locality of the structure's maintenance.

Yet, at that time, the major achievements in distributed computing obtained in recent years were not available. Accordingly, those two related problems with respect to ad hoc networks were handled separately. Indeed, *mobility* was studied for centralized graph algorithms, while *localized* graph algorithms were defined for static ad hoc networks. Observing that these two fields could be complementary, we propose here to regroup both assets in a new concept we named **Kinetic Graph**. Figure 1 illustrates the two separate, yet complementary, issues of graph algorithms: central *vs.* local and static *vs.* dynamic methods.

We therefore propose to borrow the localized management to the distributed research field and the kinetic management to the Kinetic Data Structure approach. We believe that *Kinetic Graphs* could be successfully applied to all approaches depending on structures, such as topology control, connected dominating sets, routing, or even location management.

3 Trajectory Knowledge

We base our trajectory computation on Location Information, which may be provided by the Global Positioning System (GPS) or other solutions exposed in [13] or [14] and exchanged by means of beacon messages. Velocity may be derived

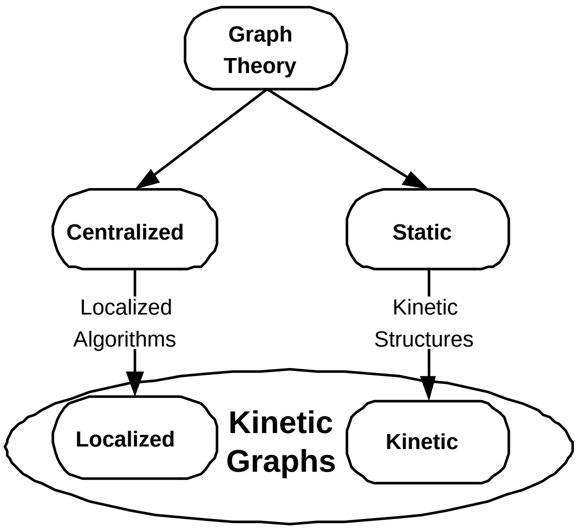


Figure 1: Illustration of the Positioning of Kinetic Graphs in Graph Theory

through successive location samples at close time instants. We also assume a global time synchronization between nodes in the network which could also be obtained by the GSP system. Accordingly, we define x, y, dx, dy as the four parameters describing a node's position and instant velocity[1], thereafter called *mobility*.

Over a relatively short period of time [2], we assume that each such node, say i, follows a linear trajectory. Its position as a function of time is then described by

$$\mathbf{Pos}_i(t) = \begin{bmatrix} x_i + dx_i \cdot t \\ y_i + dy_i \cdot t \end{bmatrix}, \tag{1}$$

where $Pos_i(t)$ represents the position of node i at time t, the vector $[x_i,\ y_i]^T$ denotes the initial position of node i, and vector $[dx_i,\ dy_i]^T$ its initial instantaneous velocity. Let us consider node j as a neighbor of i. In order to let node i compute node j's trajectory, let us define the squared distance between nodes i and j as

$$\begin{aligned} D_{ij}^2(t) &= D_{ji}^2(t) = \|\mathbf{Pos}_j(t) - \mathbf{Pos}_i(t)\|^2 \\ &= \left(\begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} + \begin{bmatrix} dx_j - dx_i \\ dy_j - dy_i \end{bmatrix} \cdot t \right)^2 \\ &= a_{ij}t^2 + b_{ij}t + c_{ij}, \end{aligned} \tag{2}$$

where $a_{ij} \geq 0$, $c_{ij} \geq 0$. Consequently, a_{ij}, b_{ij}, c_{ij} are defined as the three parameters describing nodes i and j mutual trajectories. And $D_{ij}^2(t) = a_{ij}t^2 + b_{ij}t + c_{ij}$, representing j's relative distance to node i, is denoted as j's linear relative trajectory to i. Consequently, thanks to (1), a node is able to compute the future position

[1]Unless otherwise specified, we are considered moving in a two-dimensional plane.

[2]The time required to transmit a data packet is orders of magnitude shorter than the time the node is moving along a fixed trajectory.

of its neighbors, and by using (2), it is able to extract any neighboring node's future relative distance.

Finally, considering r as nodes maximum transmission range, according to the Unit Disk Graph (UDG)[3], as long as $D^2_{ij}(t) \leq r^2$, nodes i and j are neighbors. Therefore, solving

$$\begin{aligned} D^2_{ij}(t) - r^2 &= 0 \\ a_{ij}t^2 + b_{ij}t + c_{ij} - r^2 &= 0, \end{aligned} \tag{3}$$

gives t^{from}_{ij} and t^{to}_{ij} as the time intervals during which nodes i and j remain neighbors.

4 Neighborhood Discovery

Basically, the *Kinetic Graph* neighborhood discovery procedure makes a node detect changes in its neighborhood without exchanges of periodical beacon messages. During this phase, each node broadcasts a single[4] $Hello$ message indicating its presence in the neighborhood, and transmitting its mobility parameters. Such message is emitted using maximum transmission power in order to reach the maximum number of neighbors, and is never forwarded. Thanks to mobility predictions, upon completion of this discovery procedure, nodes in the network have an accurate knowledge of their neighborhood, and as long as their neighbors keep on moving along their initial linear trajectories, there will be no need to refresh it by sending new $Hello$ messages. If such prediction becomes invalid due to an unpredicted event (i.e. trajectory changes or disconnections), the respective node spontaneously advertises its new parameters, refreshing the predictions in a event-driven way.

In the rest of this section, we will review two popular geo-localization data format, introduce a common message format for transmitting geo-localization data and finally dissert on the cost of transmitting those data.

4.1 Geo-localization Data Format

In basic simulation environments such as ns2, Qualnet or Opnet, geo-localisation data is usually based on Cartesian coordinates and the simulator's clock for time references. However, in real deployment, it is envisioned to directly use the coordinates provided by a GPS-like system (and A-GPS for indoor location), whose benefits are twofold. First, it provides a standard reference coordinates, and second, it ensures a global synchronization based on the atomic GPS clock.

[3]A Unit Disk Graph is a graph in which every two nodes are connected with an edge if and only if they are at a distance at most one. Up to normalization, a UDG corresponds to a graph where every two nodes are connected if and only if they are at a distance at most the homogenous transmission range.

[4]In order to take into account possible collision and packet losses, a $Hello$ message is sent a configurable number of times. Unless otherwise specified, we send each $Hello$ message 3 times.

4.1.1 Position Representation

Using Cartesian coordinates, nodes position is represented by the projection of the position vector onto the abscissae and the ordinate. Two values are therefore transmitted: X and Y.

According to the GPS standard, the 3D positioning provides the coordinates of a GPS device in a 3-axis referential, whose origin is the gravity center of the GPS satellite constellation. Then, the GPS terminal converts this raw data into exploitable *longitude*, *latitude*, and *elevation* in the World Geodetic System 84 (WGS84) [15], the most widely used providing a worldwide navigational system. For accurate representation, each value is provided with a 6 digits precision.

The common point of all these coordinates is that they are usually represented by a 64 bit double precision floating point. Accordingly, each geo-localization set is usually represented by 192 bits or 24 bytes. In [16], Härri, Filali and Bonnet proposed a message compression algorithm which is able to reduce the size required for the correct representation of nodes position to 48 bit instead of 192 bit.

4.1.2 Speed Representation

Using Cartesian coordinates, speed is usually represented by its projection on the abscissae and the ordinate. We therefore obtain two values dx and dy. Each of those coordinates are represented by a 64 bit double precision floating point or 8 bytes.

Using GPS coordinates, speed is obtained using the normalized velocity and the azimuth. Both are also represented by a 64 bit double precision floating point or 8 bytes.

Either with Cartesian or GPS coordinates, transmitting the speed costs 16 bytes for each target node. Similarly to the position representation, Härri Filali and Bonnet [16] proposed a speed data compression algorithm which is able to reduce the size required for the correct representation of speed to 4 bytes instead of 16 bytes for GPS data format.

4.1.3 Time Representation

As previously mentioned, the simulator clock is used to represent time in simulation environments. As the simulator is common to all nodes, this solution artificially provides time synchronization. The clock time format is usually represented by a 64 bit double precision floating point number. As time representation has already been studied in the past for routing freshness, a special mantissa/exponent representation has been developed to compress this time value to an 8 bit integer or 1 bytes.

In real deployment, as each node has its own independent clock, a clock drift is generated that avoids global synchronization between the different nodes. In order to precisely determine the position of a GPS device, its internal clock must be synchronized with the satellites atomic clocks. The GPS system is therefore able to provide a global synchronization mean to any application connected to a GPS device.

GPS time is expressed as a number of seconds since the beginning of the GPS epoch on Sunday January 6th 1980 at 0:00 UTC. Initially represented by a 32 bit integer, this value has been increased to a 64 bits long integer at the end of the last century. Accordingly, the transmission of the time in a packet requires 64 bits or 8 bytes. Finally, Härri Filali and Bonnet [16] also proposed a GPS time data compression algorithm which is able to reduce the size required for the correct representation of GPS time to 2 bytes instead of 8 bytes.

4.1.4 Discussion

Transmitting geo-localization data is a tradeoff between the potential benefits obtained by network protocols and the cost of their transmission. Indeed, it is expected that network protocols would need the geo-localization data of the sender and also of the sender immediate neighbors. Accordingly, as the network becomes dense, the overhead induced by the transmission of these geo-localization data increases significantly. Fig. 2 illustrates the cost of the transmission of geo-localization data as a function of the node degree. We can see that transmitting geo-localization without compression becomes a serious limiting factor for efficient network usage, as each packet could reach more than $1kbytes$ for dense networks. When using the compression proposed in [16], we can significantly reduce this drawback, which in turn could help improve the network protocols in general and Kinetic Graphs in particular.

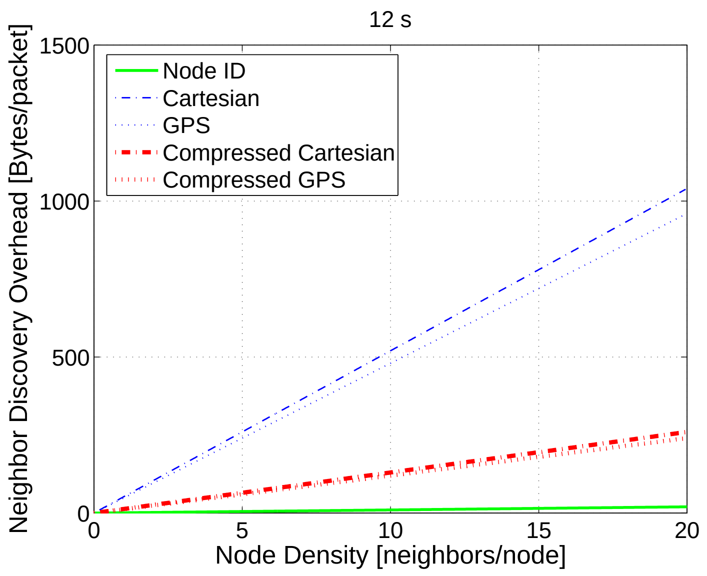


Figure 2: Illustration of the per packet overhead for geo-localization data transmission

4.2 A Common Geo-localization Message Format

This section defines the content and the structure of a mobility message containing a configurable set of geo-localization or mobility information. Unless otherwise specified, the size of each piece of data is obtained using the compression algorithms described in [16].

All `<mobility>` messages are conformed to the following specification:
`<mobility> = <value-semantic><value>`
`<value-semantic>` is an 8 bit field which describes the structure of the `<mobility>` tag.

- *bit 0 (position bit):* Messages with this bit cleared ('0') do not contain the position of the node. Messages with this bit set ('1') contain position information.
- *bit 1 (velocity bit):* Messages with this bit cleared ('0') do not contain the velocity of the node. Messages with this bit set ('1') contain the velocity.
- *bit 2 (azimuth bit):* Messages with this bit cleared ('0') do not contain the azimuth of the node. Messages with this bit set ('1') contain the azimuth.
- *bit 3 (stability bit):* Messages with this bit cleared ('0') do not contain the stability of the node. Messages with this bit set ('1') contain the stability.
- *bit 4 (Cartesian bit):* Message with this bit set ('1') contains Cartesian coordinates instead of GPS's. This bit is used for compatibility between simulation and deployment message formats.
- *bits 5-7 are RESERVED*

`<value>` is a field containing the mobility parameters. The length of this field may be obtained from the `<value-semantic>` field.
`<value> = <pos><azi><velo><stab><time>`
where

- `<pos>` is a 48 bit field containing the coordinates of a node following the general layout `<pos> =<Longitude> <Latitude> <Elevation>`.
- `<velo>` is an 8 bit field containing the node's velocity in m/s. If the `Cartesian bit` is set ('1'), `<velo>` is instead a 48 bit field containing the Cartesian projection of the velocity following the general layout `<velo> = <dx><dy>`.
- `<azi>` is an 8 bit field containing the node's azimuth in degree
- `<stab>` is an 8 bit field containing the node's stability. It represents the node eagerness to keep the current mobility parameters.
- `<time>` is an 16 bit field containing the time in seconds when the mobility parameters have been sampled.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     HELLO     | Resv|0|0|1|0|0|   Length      |    Address    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Address                      | Resv|0|1|1|1|1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Longitude           |            Latitude           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Elevation           |   Velocity    |    Azimuth    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Stability   |             Time              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Hello Packet Containing Geo-localization Information

The basic layout of a `<mobility>` message included in a `HELLO` packet is illustrated in Fig. 3.

In this section, we provided a framework for optimized and confi gurable transmission of geo-localization information. We believe our approach could ease interoperability and improve the performance of the neighbor discovery phase for Kinetic Graphs. This solution has also been proposed for a possible standardization within the IETF [17].

5 Time Varying Link Weights

In this section, we describe two popular link weights used in graph theory and which could be applied to kinetic graphs. Based on those weights, a graph can be build and dynamically updated based on the defi ned time-varying link weights. Most of the graph algorithms could be adapted to use those criteria, however, as mentioned in the introduction part of this chapter, it is important that graph protocols be distributed and local. Accordingly, we suggest as potential targets localized graph constructions described in [11].

5.1 Kinetic Distance Based Weight

The *power cost* function, required to transmit between nodes i and j at time t, is defi ned as $P_{ij}(t) = C \cdot D_{ij}^{\alpha}(t) + \gamma$, where $\alpha \geq 2$ and for some constants γ. As we assume free space propagation and homogenous antennas characteristics, we set $\alpha = 2$ and $C = 1$. The constant γ represents a constant charge for each transmission, including the energy needed for signal processing, internal computation, and overhead due to MAC control messages. However, since we assume perfect channel, and that the election is distributed and does not put any extra burden on any particular node, γ is common to all nodes and is not of great signifi cance when comparing power costs. Therefore, without loss of generality, we assume $\gamma = 0$[5]

[5]Therefore, Power and Distance will later be interchangeably used.

and define

$$P_{ij}(t) = D_{ij}^2(t) = a_{ij}t^2 + b_{ij}t + c_{ij} \tag{4}$$

as the power cost function for the weight of the *Kinetic Graphs*. By choosing the distance between nodes as the link cost, one obtains minimum power routes that help preserve battery life (see Fig. 4).

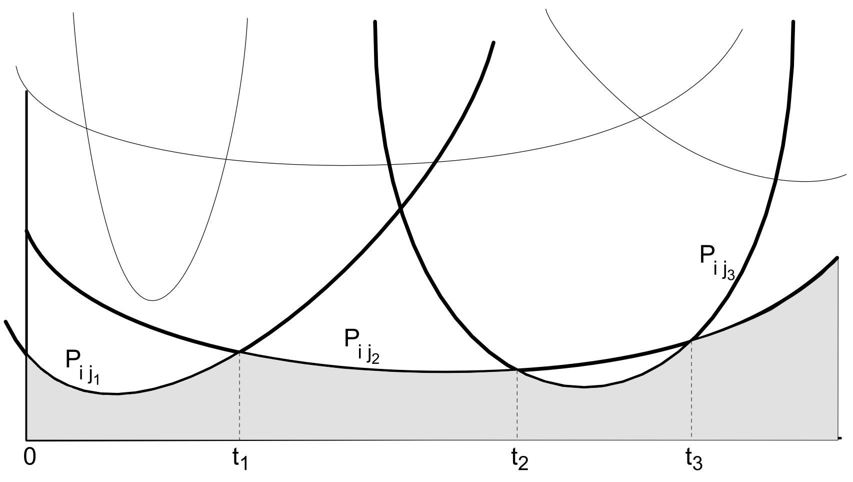


Figure 4: The power function, where each parabola represents the energy needed to reach each neighbor of node i as a function of time

We then define

$$p_i(t) = e^{-\beta_i(t-t_i)} \tag{5}$$

as the probability that a node i is continuing on its present trajectory, where the Poisson parameter $\frac{1}{\beta_i}$ indicates the average time the node follows a trajectory, and t_i the time its current trajectory has begun (see Figure 5).

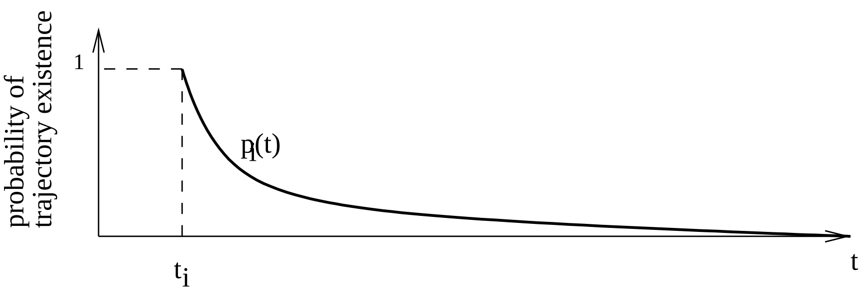


Figure 5: The stability function, where the probability for a node i to behave as predicted decreases exponentially

Assuming independent node trajectories,

$$\begin{aligned} p_{ij}(t) &= p_i(t) \cdot p_j(t) \\ &= e^{-(\beta_i+\beta_j)(t-\frac{t_i\beta_i+t_j\beta_j}{\beta_i+\beta_j})} = e^{-\beta_{ij}(t-t_{ij})} \end{aligned} \tag{6}$$

describes the probability that nodes i and j are continuing on their respective courses at time t, which will be considered as the *stability*[6] of link $\overline{ij}$. The modified power cost below probabilistically weights the power cost $P_{ij}(t)$ to reflect the link's *stability*.

Finally, since we aim at suppressing periodic beacon messages, a node that will shortly leave the neighborhood must be automatically removed from the neighboring table. We use t_{ij}^{to} as a *timeout* counter. Upon expiration, it will remove the corresponding neighbor from the table. The link weight computed so far is able to dynamically represent the energy cost between two mobile nodes. However, it does not represent the actual capability to reach the neighbor, more specifically if two nodes stop being within mutual transmission range. For that matter, we must add a function which invalidates a link weight as soon as two neighbors stop being neighbors in the Unit Disk Graph sense. Accordingly, to represent the node's finite range, we use an inverse sigmoid function

$$Sigm_i(t) = \frac{1}{1 + e^{a\cdot(t-t_i^{to})}} \tag{7}$$

whose value is equal to 1 as long as $t < t_i^{to}$ and thereafter drops to 0, where t_{ij}^{to} is computed as described in Section 3.

We finally define

$$\begin{aligned} W_{ij}(t) &= -\frac{p_{ij}(t)}{P_{ij}(t)} \cdot Sigm_{ij}(t) && (8) \\ &= -\frac{e^{-(\beta_{ij})(t-t_{ij})}}{a_{ij}t^2 + b_{ij}t + c_{ij}} \cdot \frac{1}{1 + e^{a\cdot(t-t_{ij}^{to})}} \\ W_{ij}(t) &= -\frac{e^{-(\beta_i+\beta_j)(t-\frac{t_i\beta_i+t_j\beta_j}{\beta_i+\beta_j})}}{a_{ij}t^2 + b_{ij}t + c_{ij}} && (9) \\ &\cdot \frac{1}{1 + e^{a\cdot(t-t_{ij}^{to})}} \end{aligned}$$

as the composite link cost between two neighbors (see Figure 6). A low modified power cost favors a low power cost with high stability. We have then six parameters a_{ij}, b_{ij}, c_{ij}, β_{ij}, t_{ij}, and t_{ij}^{to} describing $W_{ij}(t)$ as the time varying weight of a link between two nodes in a *Kinetic Graph*.

In order to clarify our approach, let's consider the situation depicted in Fig. 7-C. Node i tries to find the best next hop node to reach a far destination node. To do so,

[6]The probability that the mutual trajectory between two nodes remains identical after both nodes have changed course at the same time is negligible

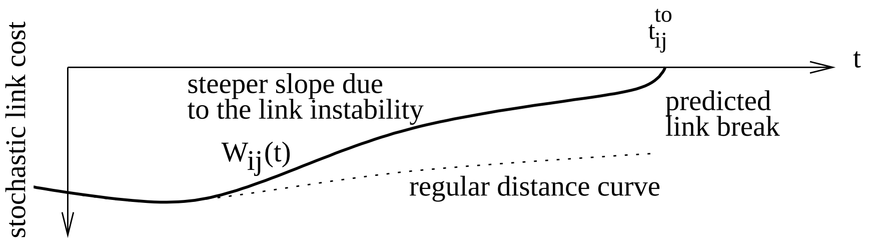


Figure 6: The composite link cost function, where we can see the cost increase due to the link's instability.

it will consider the distance separating it from its neighbors, and the stability of the respective links, in other words, the expected length of its neighbors' trajectories. Fig. 7-A reflects the probabilities nodes j_1 and j_2 are not to have changed their trajectories. t_{j_1} and t_{j_2} are the time they actually began. As it can be seen, at time t_0-t_0 representing the execution time-the probability node j_1 has not to have changed its trajectory is bigger than j_2. Therefore, as depicted in Fig. 7-B, even though node j_2 is closer to node i and has a similar trajectory, this link is less reliable than j_1's. However, at time t_{trans}, node j_2 has a relatively more reliable link and follows a similar trajectory that node i. Therefore, at this time, node i automatically changes its next hop neighbor, and this, without any exchange of messages.

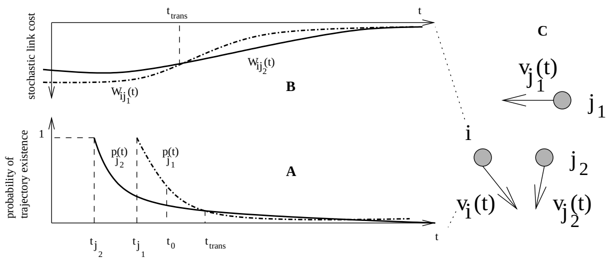


Figure 7: Topology example

5.2 Kinetic Nodal Degree Weight

In Graph theory, besides the Euclidian distance, the nodal degree is also widely used, as it provides high spreading effi ciency instead of low weight structure. While the former is popular as a criterion for routing protocol (i.e. Distance Vector), the latter is very popular for broadcast protocols, as a node with a high nodal degree has a larger diffusion potential.

Similar to the euclidian distance, the nodal degree may also be applied to Kinetic Graphs as a time varying link weight. We explain in this section, the method for modeling *Kinetic Nodal Degrees* in MANETs.

As defined in Section 3, we model two nodes i and j mutual trajectory as

$$D^2_{ij}(t) = a_{ij}t^2 + b_{ij}t + c_{ij} \tag{10}$$

Consequently, thanks to (10), a node is able to compute the future position of its neighbors and is able to extract any neighboring node's future relative distance.

Considering r as nodes maximum transmission range, as long as $D^2_{ij}(t) \leq r^2$, nodes i and j are neighbors. Therefore, we obtain t^{from}_{ij} and t^{to}_{ij} as the time intervals during which nodes i and j remain neighbors. Consequently, we can model nodes' kinetic degree as two successive sigmoid functions, where the first one jumps to one when a node enters another node's neighborhood, and the second one drops to zero when that node effectively leaves that neighborhood (see Fig. 8).

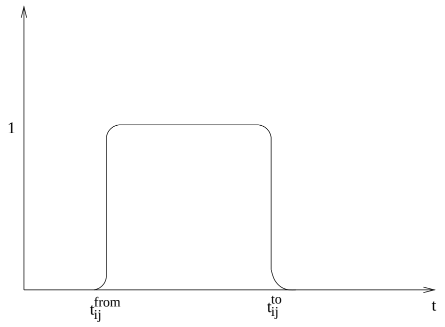
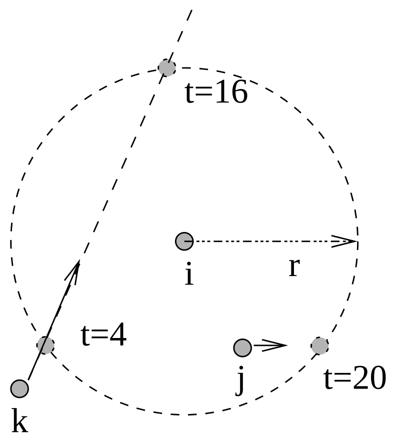


Figure 8: Double sigmoid function modeling a link lifetime between node i and node j

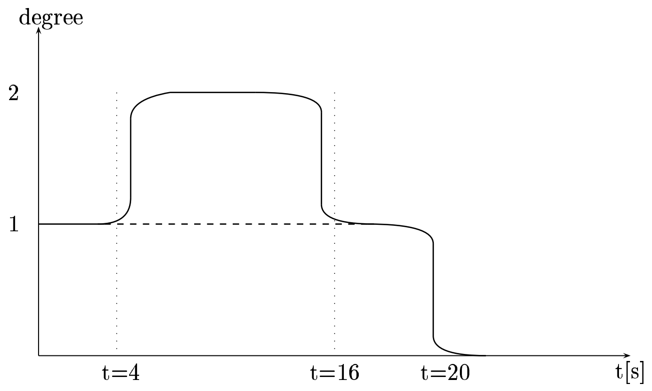
Considering $nbrs_i(t)$ as the total number of neighbors detected in node i's neighborhood at time t, we define

$$\begin{aligned} Deg_i(t) \;\; = \;\; & \sum_{k=0}^{nbrs_i(t)} \left(\frac{1}{1+\exp(-a\cdot(t-t^{from}_k))} \right. \\ & \left. \cdot \frac{1}{1+\exp(a\cdot(t-t^{to}_k))} \right) \end{aligned} \tag{11}$$

as node i's kinetic degree function, where t^{from}_k and t^{to}_k represent respectively the time a node k enters and leaves i's neighborhood. Thanks to (11), each node is able to predict its actual and future degree and thus is able to proactively adapt its coverage capacity. Figure 9(a) illustrates the situation for three nodes. Node k enters i's neighborhood at time $t = 4s$ and leaves it at time $t = 16s$. Meanwhile, node j leaves i's neighborhood at time $t = 20s$. Consequently, Fig. 9(b) illustrates the evolution of the kinetic degree function over t.



(a) Node i kinetic neighborhood



(b) Node i kinetic nodal degree

Figure 9: Illustration of nodes kinetic degrees

Finally, the kinetic degree is obtained by integrating (11)

$$\widehat{Deg}_i(t) = \int_t^{\infty} \left(\sum_{k=0}^{k=nbrs_i} \left(\frac{1}{1+\exp(-a\cdot(t-t_k^{from}))} \right.\right.$$
$$\left.\left. \cdot \frac{1}{1+\exp(a\cdot(t-t_k^{to}))} \right) \right) \tag{12}$$

For example, in Fig. 9(b), node i kinetic degree is ≈ 32.

Similarly to the previous section 5.1, the kinetic nodal degree may also be stochastically weighted by the probability of the existence of the link. The last task is therefore to consider the uncertainty of a predicted degree by adding the stability function (6). Accordingly, we obtain a criterion reflecting nodes actual and future degree, yet biased by the uncertainty of the link between all respective neighbors.

By using substituting (6) to (12), we defi ne

$$\begin{aligned}\widehat{Deg}_i(t) \quad = \quad & \int_t^{\infty} \left(\sum_{k=0}^{k=nbrs_i} \left(\frac{1}{1+\exp(-a\cdot(t-t_k^{from}))} \right.\right. \\ & \cdot \frac{1}{1+\exp(a\cdot(t-t_k^{to}))} \\ & \left.\left. \cdot \exp(-(\beta_i+\beta_k)(t-\frac{t_i\beta_i+t_k\beta_k}{\beta_i+\beta_k})) \right) \right)\end{aligned} \tag{13}$$

Using the same topology as Fig. 9 and applying the uncertainty of predicted degrees, we obtain a stochastically predicted nodal degree depicted in Fig. 10. Initially, node i has a degree equal to 1 since node j is in its neighborhood and both initiated their trajectories at the same time. Yet, as time elapses, so does the probability both nodes have to keep their trajectories. Therefore, the stochastically predicted degree decreases. Then, at time $t = 4$, node i detects a new neighbor k and computes the time during which both nodes will be in range. However, node k initiated its trajectory before nodes i and j, consequently node k's Poisson function is smaller than node j's (see Fig. 10 bottom part). Thus, during the interval node i and k are in range, the nodal degree of node i does not increase as much as it did in Fig. 9. Worse, its decreasing curve is sharper than the one between nodes i and j taken alone. Similarly to Fig. 9, at time $t = 16$ and $t = 20$, nodes k and j leave i's neighborhood thus making i's nodal degree decrease abruptly. The main difference here between the two fi gures, is that the degree is not stable during the time two nodes are in range but decreases following the probability both nodes are still following their initial trajectories.

6 Aperiodic Neighborhood Maintenance

A limitation in per-event maintenance strategies is the neighborhood maintenance. While mobility prediction and the kinetic graph approach allow to discard

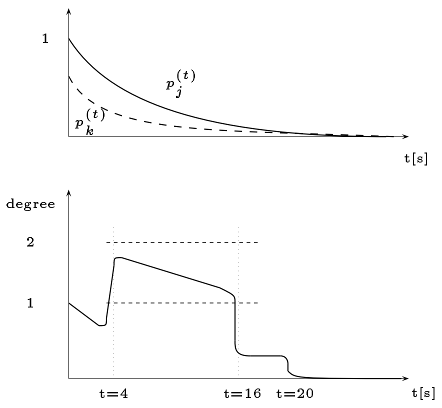


Figure 10: Stochastically Predicted Nodal Degree

invalid links or unreachable neighbors, it remains impossible to passively acquire new neighbors reaching some other nodes' neighborhood. The lack of an appropriate method to tackle this issue would limit Kinetic Graphs' ability to obtain up-to-date links and effective kinetic weights.
We developed several heuristics to help Kinetic Graphs detect nodes stealthily entering some other nodes transmission range in a non-periodic way.

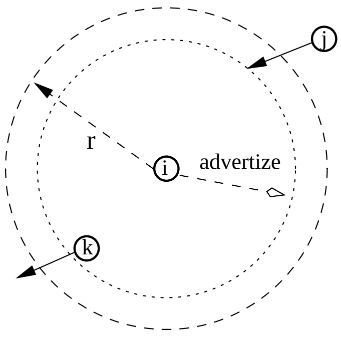
- ***Constant Degree Detection***— Every node tries to keep a constant neighbor degree. Therefore, when a node i detects that a neighbor actually left its neighborhood, it tries to acquire new neighbors by sending a small advertising message. (see Figure 11(a));

- ***Implicit Detection***— A node j entering node i transmission range has a high probability to have a common neighbor with i. Considering the case depicted in Figure 11(b), node k is aware of both i and j's movement, thus is able to compute the moment at which either j or i enters each other's transmission range. Therefore, node k sends a notifi cation message to both nodes. In that case, we say that node i implicitly detected node j and vice versa;

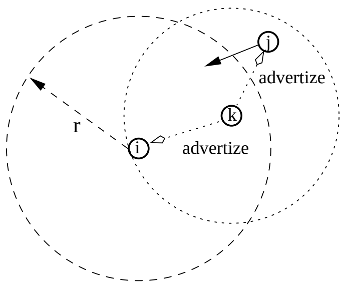
- ***Adaptive Coverage Detection***— We require each node to send an advertising message when it has moved a distance equal to a part of its transmission range. An adjusting factor which vary between 0 and 1 depends on the node's degree and its velocity (see Figure 11(c));

A second approach is identical to the information exchange period proposed in [8]. The idea is to determine the refreshing rate by a probabilistic model with the following assumptions:

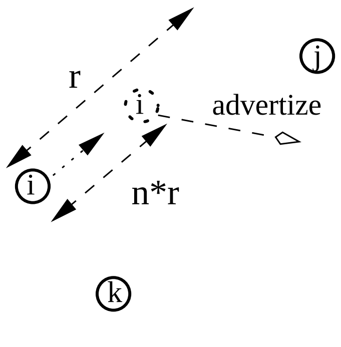
- All nodes are randomly distributed within a disk of area $S0$ and the total number of nodes in G,N, is known.



(a) Constant Degree Detection



(b) Implicit Detection



(c) Adaptive Coverage Detection

Figure 11: Three heuristics to detect incoming neighbors in a per-event basis

- For a short time interval of length t, each node moves independently toward a random direction in $(0, 2\pi)$ with a constant speed v that is uniformly distributed in $[0, vmax]$.
- The maximum transmission range of a node is $d = dmax$.

Under these assumptions, Li [8] calculated the probabilities that a new neighbor moves into the transmission range of node u within a time interval of t. We ignore the case of existing neighbors moving out of the transmission range of node u since we already know this intervals.

The probability, p_{join}, that node w moves into transmission range of node u within time t is

$$\begin{cases} p_{join} &= \int_{d}^{d+r} \frac{2xS_1}{S_0 r^2} & for\, 0 < r < 2d \\ p_{join} &= \frac{\pi d^2 (r-2d)}{S_0 r} \int_{d-r}^{d+r} \frac{2xS_1}{S_0 r^2} & for\, r \geq 2d \end{cases}$$

Then, given that node u has n neighbors and the total number of nodes is N. the probability that no new neighbor enters the visible neighborhood of node u is

$$p_1 = (1 - p_{join})^{N-n-1}$$

Therefore, the probability that the visible neighborhood of node u changes is

$$p_{change} = 1 - p_1$$

Given a predetermined probability threshold p_{th}, we can determine the neighborhood update interval t such that $p_{change} < p_{th}$.

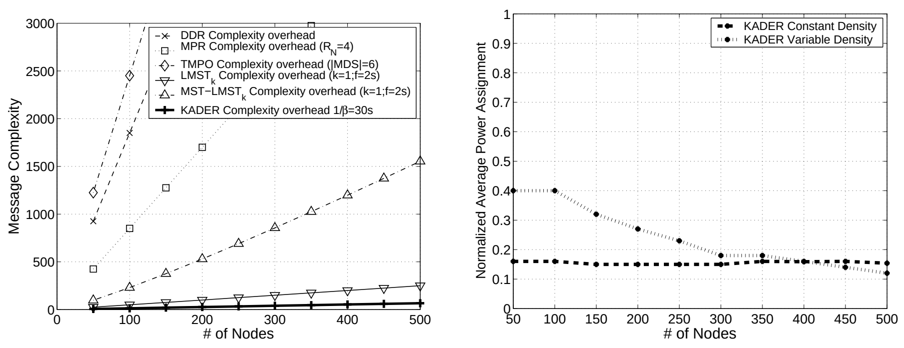
7 Application Examples

In this section, we will describe two successful application of the Kinetic Graph framework. The signifi cant difference between both solutions is the two different kinetic link weights used to build the different structures.

7.1 Kinetic Graphs applied to Topology Control

While most topology control protocols only address limited network mobility, authors in [18] presented a novel approach, called Kinetic Adaptive Dynamic topology control for Energy effi cient Routing (KADER). It employs the kinetic graph framework to get rid of periodic maintenance beacons and to fi t its structure to nodes mobility patterns. The major properties of KADER are *Linear Complexity, Scalability, and Energy Efficiency*.

The complete description is this protocol is out of scope of this paper and we refer the interested reader to [18]. KADER is a typical example of kinetic graphs applied to topology control as it is based on a localized protocol, DDR [19], and changes the original decision criterion by the kinetic distance link weight. The authors initially seeking to improve DDR for energy effi ciency, they used this kinetic distance instead of the instantaneous nodal degree. However, a similar approach could also have been performed by simply replacing the instantaneous by the kinetic degree. Fig. 12 illustrates KADER's properties, where Fig. 12(a) compares the maintenance overhead of different topology control with KADER, while Fig. 12(b) depict KADER's energy effi ciency.



(a) Message Complexity (b) Average Power Assignment

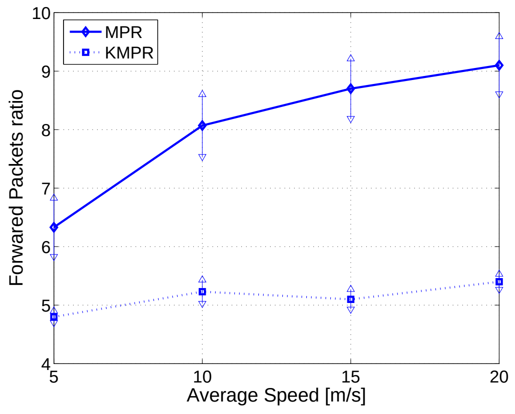
Figure 12: KADER's Properties

7.2 Kinetic Graphs Applied to Broadcasting

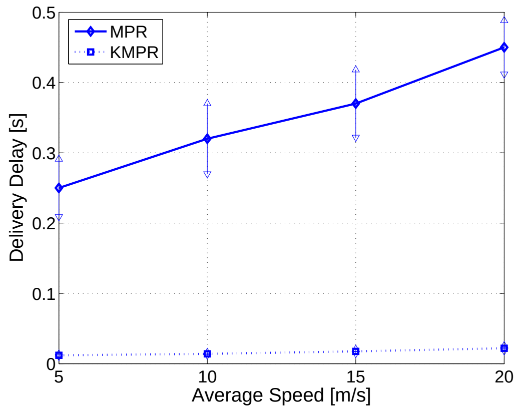
By using the kinetic nodal degree link weight, it is also possible to improve broadcasting in mobile ad hoc networks. For example, authors in [20] presented an approach for improving the well-known MPR protocol by using kinetic graphs. The authors showed that the Kinetic Multipoint Relaying (KMPR) protocol was able to reduce the flooding by up to 40%, and this by reducing the MPR channel access by 75% and MPR broadcast delay by 90%. Fig. 13 illustrates KMPR and MPR's broadcast properties, and the benefi t from using kinetic graphs in broadcasting is straightforward. Authors also performed tests under vehicular and pedestrian mobility with a similar success. We refer the interested reader to [20] for a complete description of the KMPR protocol.

8 Conclusion

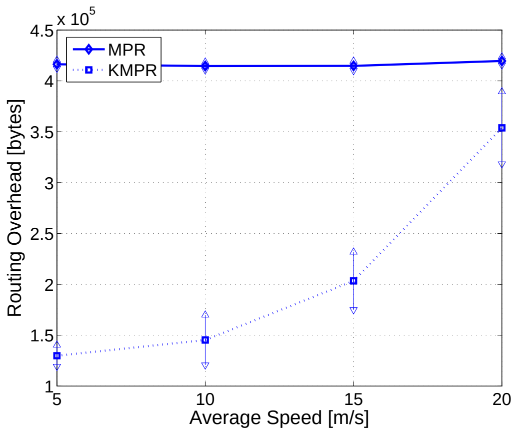
In this paper, we presented an original approach for applying mobility predictions to Mobile Ad Hoc Networks (MANET) called the *Kinetic Graphs*. The objective was to construct and maintain a mobile topology or routing structure without relying on periodic maintenance. For that matter, we provided guidelines



(a) Relay Ratio



(b) Broadcast Delay



(c) Maintenance Overhead

Figure 13: KMPR's Properties

for adapting any localized MANET protocol to the kinetic approach. The kinetic graph approach requiring geo-localization information, we described a common packet format for their exchange during a neighborhood discovery process. Then, a trajectory representation must be defined, based on which kinetic link criteria are generated. Finally, we proposed different solutions to aperiodically maintain the neighborhood.

The interesting feature of the proposed framework is that the approach is independent of the criteria chosen to build the backbone, or the localized MANET protocol used, and various approaches or combinations may be tested. As an example, we provided two possible kinetic link weights: the *kinetic distance* and the *kinetic degree*, and illustrated two approaches, KADER and KMPR, successfully employing them to maintain a structure subject to mobility.

We therefore showed that by regrouping localized algorithms and kinetic structures, two apparently separated yet complementary research fields, we could successfully improve mobility management in Mobile Ad Hoc Networks.

References

[1] J. B. *et al.*, “Data structures for mobile data,” *Journal of Algorithms*, vol. 31, no. 1, pp. 1–28, 1999.

[2] S. B. *et al.*, “Mobile facility location,” in *Proc. of the 4th international Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2000.

[3] J. Herschberger, “Smooth kinetic maintenance of clusters,” in *Proc. of the 19th Annual Symposium on Computational Geometry*, 2003.

[4] C. Gentile, J. Härri, and R. E. V. Dyck, “Kinetic minimum-power routing and clustering in mobile ad-hoc networks,” in *Proc. of the IEEE Vehicular Technology Conference (VTC’02 Fall)Conference*, 2002.

[5] L. Guibas, “Kinetic data structures: A state of the art report,” in *Proc of the 3rd Workshop of Algorithmic Foundations of Robotics*, 1998.

[6] X.-Y. L. *et al.*, “Localized delaunay triangulation with application in wireless ad hoc networks,” *IEEE Trans on Parallel and Distributed Processing*, vol. 14, no. 10, pp. 1035–1047, 2003.

[7] K. A. *et al.*, “Geometric spanners for wireless ad hoc networks,” *IEEE Transactions on Parallel and Distributed Processing*, vol. 14, no. 4, pp. 408–421, 2003.

[8] N. Li, J. Hou, and L. Sha, “Design and analysis of an mst-based topology control algorithm,” in *Proc. of the IEEE INFOCOM*, 2003.

[9] N. Li and J. Hou, “Blmst: A scalable, power-effi cient broadcast algorithm for wireless networks,” in *Proc. of the 1st International Conference on Quality of Service in Heterogenous Wired/Wireless Networks (QSHINE’04)*, 2004.

[10] J. Cartigny, F. Ingelrest, and D.Simplot, “Lmst and rng based minimum-energy broadcast protocols in ad hoc networks,” *Ad Hoc Networks*, vol. 3, no. 1, pp. 1–16, 2005.

[11] X. Li and I. Stojmenović, “Broadcasting and topology control in wireless ad hoc networks,” in *Handbook of Algorithms for Mobile and Wireless Networking and Computing*, A. Boukerche and I. Chlamtac, Eds. CRC Press, 2006, ch. 11, pp. 239–264.

[12] I. S. H. Frey, “Geographic and energy aware routing in sensor networks,” in *Handbook of Sensor Networks: Algorithms and Architectures*, I. Stojmenović, Ed. Wiley, 2006, ch. 12, pp. 381–415.

[13] R. Fontana and S. Gunderson, “Ultra-wideband precision asset location system,” in *Proc. of the IEEE Conference on Ultra Wideband Systems and Technologies*, 2002.

[14] C. Gentile and L. Klein-Berndt, “Robust location using system dynamics and motion constraints,” in *Proc. of the IEEE International Conference on Communications (ICC)*, 2004.

[15] Wikipedia - World Geodetic System, http://en.wikipedia.org/wiki/WGS84.

[16] J. Haerri, F. Filali, and C. Bonnet, “Rethinking the overhead of geo-localization information for vehicular communications,” Institut Eurécom, Technical Report 07-194, 2007.

[17] J. H. *et al.*, “Manet position and mobility signaling format,” February 2007, internet Draft, http://tools.ietf.org/id/draft-haerri-manet-position-signaling-00.txt (work in progress).

[18] J. Härri, N. Nikaein, and C. Bonnet, “Trajectory knowledge for improving topology control in mobile ad-hoc networks,” in *Proceedings of the 1st ACM/e-NEXT International Conference on Future Networking Technologies (Co-NEXT’05)*, 2005.

[19] N. Nikaein, H. Labiod, and C. Bonnet, “Distributed dynamic routing algorithm for mobile ad-hoc networks,” in *The ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2000.

[20] J. Härri, F. Filali, and C. Bonnet, *On the application of mobility predictions to multipoint relaying in MANETs: kinetic multipoint relays*, ser. Lecture Notes in Computer Science. Springer, 2005, vol. 3837, pp. 143–155.