Trajectory Knowledge for Improving Topology Control in Mobile Ad-Hoc Networks

Jérôme Härri
haerri@eurecom.fr

Navid Nikaein
nikaeinn@eurecom.fr

Christian Bonnet
bonnet@eurecom.fr

Institut Eurécom*
Department of Mobile Communications
B.P. 193
06904, Sophia Antipolis, France

ABSTRACT

While most topology control protocols only address limited network mobility, we propose in this paper a quasi-localized topology control algorithm that considers mobility predictions in order to construct and maintain a power efficient topology without relying on periodic beacons. Indeed, a node is capable of extracting linear trajectories of its neighboring nodes based on their positions and velocities. Based on such information, a node obtains a local prediction of neighborhood evolution and can thereafter proactively adapt the topology without relying on periodic beacons. Maintenance is driven on a per-event basis. It is therefore only when a node changes course that messages are exchanged in order to adapt the structure. Our approach is able to create and keep a stable kinetic backbone at a linear message and time complexity. It also improves concurrent communications by providing a significant reduction on local power assignments, therefore reducing interferences, increasing battery life and improving the overall network lifespan.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Network**]: Network Architecture and Design—*network topology, wireless communication, distributed networks*; G.2.2 [**Discrete Mathematics**]: Graph Theory

General Terms

Algorithms, design, performance

Keywords

Trajectory, localized topology control, power assignment, stochastic mobility prediction, wireless ad hoc networks.

*Institut Eurécom's research is partially supported by its industrial members: Bouygues Télécom,Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Télécom, Hitachi, ST Microelectronics, Swisscom, Texas Instruments, Thales, Sharp

CoNEXT'05, October 24–27, 2005, Toulouse, France.

1. INTRODUCTION

A mobile ad-hoc network (Manet) consists of a collection of mobile nodes forming a dynamic autonomous network through a fully mobile infrastructure. Nodes communicate with each other without the intervention of centralized access points or base stations. In such a network, each node acts as a host and may act as a router. Due to limited transmission range of wireless network interfaces, multiple *hops* may be needed to exchange data between nodes in the network, which is why the literature often uses the term of *multi-hop* network in Manet. The *topology* of a multi-hop network is the set of communication links between nodes used by routing mechanisms. Removing redundant and unnecessary topology information is usually called *topology control*.

The importance of topology control lies in the fact that it critically affects the system performance in several ways. For once, as shown in [1], it affects networks spatial reuse. Choosing a power assignment too large results in excessive interference while choosing it too small creates a disconnected network. Power assignment in topology control also exerts influence on the energy consumption of communication, thus impacts battery life which is a critical resource in many mobile applications. In addition, topology control also improves contentions at the MAC layer.

Several topology control algorithm have been proposed [2–5,17–19] to create power efficient network topologies in MANETs, yet only considering limited mobility or no mobility at all. Some of the algorithms require explicit propagation channel (e.g., [5]), while others (e.g., [2]) incur significant message exchanges. In [5] and its extension [6], Rodoplu *et al.* introduced the notion of *relay region* and *enclosure* for the purpose of power control. It is shown that the network is strongly connected if each node maintains links with the nodes in its enclosure and the resulting topology is a minimum power topology. In [4], Ramanathan *et al.* presented two centralized algorithms that minimize the maximum power used per node while maintaining the (bi)connectivity of the network. In the same paper, the authors also proposed two distributed heuristics for mobile networks. LINT uses locally available neighbor information collected by routing protocols to keep the degree of neighbors bound. LILT further improves LINT by overriding the threshold on the node degree when topology changes indicated by routing updates result in undesirable connectivity. In [2], Narayanaswamy *et al.* developed a power control protocol, called COMPOW, that reduces the power level to a common value that reaches a maximum network connectivity. In [3], authors proposed an algorithm, called $CBTC(\alpha)$, in which each node finds the minimum power p such that transmitting with p ensures that it can reach some nodes in ev-

ery cone of degree α. Other works also exist on power efficient topology control. Following a probabilistic approach, Santi *et al.* derived the suitable common transmission range which preserves network connectivity [20]. In [21], a "backbone protocol" is proposed to manage large wireless ad hoc networks, in which a small subset of nodes is selected to construct the backbone. In [14], Li *et al.* presented a MST-based topology control algorithm, in which each node builds its local minimum spanning tree (LMST) independently and only keeps on-tree nodes that are one-hop away as its neighbors in the final topology. This approach as been improve in [17] and in [18], where a MST is quasi-locally build from the LMST structure. Finally, [19] presents a strictly local protocol, called XTC, which does not only work on Unit Disk Graphs, but also on general weighted networks graphs.

In this paper, we are focusing on a novel approach, called stochastic mobility prediction, where nodes are able to predict their neighbors' future positions. We are adapting this concept to a topology control protocol denoted as Kinetic Adaptive Dynamic topology control for Energy efficient Routing (**KADER**). We base our approach on the DDR [7] protocol which we modified to obtain a distributed self-maintained topology control strategy which tries to optimize the power assignment for multi-hop transmissions, and where modifications to the topology are announced by the respective nodes in a per-event basis. The contribution of this paper includes: (i) KADER builds a self adaptive forest which maintains the network connectivity; (ii) each tree in the forest forms a zone, in which shortest path routes are proactively maintained; (iii) the criterion to built the forest is based on the relative power needed to reach a neighbor, thus minimizing the power assignment and creating a backbone adapted to mobility; and (iv) Since KADER is based on mobility predictions, it only needs to update its structure when a node changes its trajectory. Since most of links remain valid after a localized topology changes, the updates are also kept local further minimizing the maintenance cost. The capability of forming a self-adaptive topology that is closely linked to nodes relative mobility is what make KADER achieve *linear time and message complexity, scalability and energy efficiency*.

Both CONNECT and and its extension are centralized algorithms that requires global information, thus cannot be directly deployed in the case of mobility. On the other hand, LINT and LILT cannot guarantee the preservation of the network connectivity. In opposite, KADER does not require global information and is able to ensure network connectivity. Moreover, COMPOW is known to give poor performance in the case of uneven spatial distributions, while the performance of KADER is not subject to the spatial distribution, and as a matter of fact, is especially well-suited in the case of uneven spatial distributions. Finally, what makes KADER unique compared to previous approaches, is its ability to use mobility predictions to maintain its backbone in a complete per-event way (i.e. non periodically). Indeed, most of the proposed protocols either do not consider mobility induced topology changes, or perform this task periodically although trying to adapt the frequency of topology updates to nodes limited mobility ([14]).

The rest of the paper is organized as follows. Section 2 outlines the method used to obtain nodes linear trajectory knowledge, as well as our motivation to use such information for topology control. In Section 3, we present in detail the different parts of KADER's constructed topology, while in Section 4, we show the impact of trajectory knowledge on KADER's topology maintenance. Section 5 characterizes the topology created by KADER, and in Section 6, we show the time and message complexity of KADER. Finally, Section 7 highlights the benefits of KADER on routing protocols, and Section 8 draws some concluding remarks.

2. NODES TRAJECTORIES IN MANETS

We explain in this section the methods and motivations for modeling nodes trajectories in MANETs. We model nodes' positions as a piece-wise linear trajectory and show that the corresponding trajectory durations are lengthy enough to become a valuable cost for the creation of kinetic backbones.

2.1 Trajectory Knowledge

The term "Kinetic" in KADER reflects the motion aspect of our algorithm, which computes a node's trajectory based on its Location Information [8]. Such location information may be provided by the Global Positioning System (GPS) or other solutions exposed in [9] or [10]. Velocity may be derived through successive location samples at close time instants. Therefore, we assume a global time synchronization between nodes in the network and define x, y, dx, dy as the four parameters describing a node's position and instant velocity [1], thereafter called *mobility*. We describe in this section the way KADER is able to extract those trajectories. We further insist on the fact that KADER's neighbor discovery procedure distinguishes itself from regular protocols since it is neither periodically performed, nor initiated after some adaptive intervals depending on nodes mobility. It is rather triggered only when the neighborhood effectively changes.

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

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

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

$$\begin{aligned} D_{ij}^2(t) &= D_{ji}^2(t) = \|\mathbf{Pos}_j(t) - \mathbf{Pos}_i(t)\|_2^2 \\ &= \left(\begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} + \begin{bmatrix} dx_j - dx_i \\ dy_j - dy_i \end{bmatrix} \cdot t \right)^2 \\ &= a_{ij}t^2 + b_{ij}t + c_{ij}, \end{aligned} \quad (2)$$

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

2.2 Average Trajectory Durations in Ad Hoc Networks

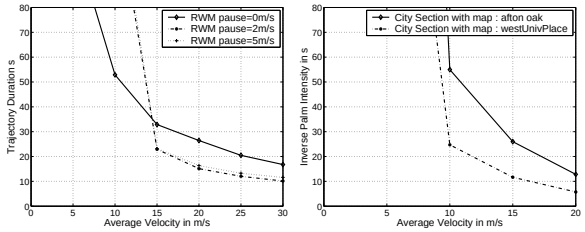
A basic assumption in KADER is to assume that nodes move following a linear trajectory, then predict to update the topology when a trajectory change occurs. Therefore, KADER scalability is highly dependent to the number of trajectory changes (or transitions) per unit of time, thereafter called β.

Part of the results obtained in [15] are depicted in Figure 1. It shows that even with an average velocity of 20m/s, nodes have an average trajectory duration of $22s$ (Figure 1(a)) for the Random

[1] 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.

Waypoint model and $10s$ (Figure 1(b)) for the City Section model. Figure 1 therefore provides a lower bound on the average trajectory duration, that is $\frac{1}{\beta} \approx 10s$ using extreme values for the configuration parameters of the mobility models. In more realistic situations, this value is rather $\frac{1}{\beta} \approx 30s$. Accordingly, it becomes conceivable to consider predictions to improve ad-hoc protocol the way we did in KADER.



(a) RWM with different pause times

(b) City Section with two suburban maps

Figure 1: Average nodes' trajectory duration ($\frac{1}{\beta}$) under the Random Waypoint mobility model and the City Section

3. KADER'S TOPOLOGY CONSTRUCTION ALGORITHM

We propose to construct a self-adapting forest from an ordinary network that consists of non-overlapping dynamic trees, thereafter called zones[3]. Each zone is kept connected with its neighboring zones through gateway nodes, thus making the whole network a set of connected zones. The size of a zone will increase or decrease dynamically without any need of periodic maintenance. Unexpected topological changes are announced by the respective nodes through a specific non-periodic message communicating its new mobility parameters. Following this event, the forest will adapt itself to the new topology (see section 4).

The algorithm described hereafter consists of five cyclic time-ordered phases: *neighborhood discovery(3.1), preferred neighbor election(3.2), forest construction(3.3), self-adaptive intra-zone clustering (3.4),* and *self-adaptive inter-zone clustering(3.5).* Since these phases are similar to DDR [7], we will only describe the major extensions.

3.1 Neighborhood Discovery

Basically, KADER's 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 $Hello$ message indicating its presence in the neighborhood, and transmitting its mobility parameters x, y, dx, dy, along with its stability parameters β and t_0. Such message is emitted using maximum 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.

[3]We will later use the term tree and zone interchangeably.

3.2 Preferred Neighbor Election

A node's *Preferred Neighbor (PN)* is a dedicated neighbor through which a node sends, receives, or forwards packets. It therefore represents the link on which a node sends its traffic. The criterion determining this neighbor depends on the application needs. For example, it could be the nodal degree in order to improve broadcast, or nodes energy level and traffic level for load balancing. It also could be a combination of all three. In KADER, we wish to obtain a criterion that is able to satisfy two objectives. The first one is to represent the energy needed to reach a neighbor. The second one is neighbor stability. A node's stability is the probability that it evolves as predicted. Since KADER does not periodically update its set of links, we want the links chosen by KADER to remain stable as possible such that routing errors could be kept low. Accordingly, KADER should not elect the closest neighbor, but might decide to choose a more distant yet a more stable one. Therefore, KADER is able to lower nodes power assignment by preferring close-by neighbors, which increases transmission concurrency and improves nodes lifespan. It is also able to lower topology maintenance and routing errors by favoring stable nodes.

3.2.1 An Energy Based Election Criterion

The *power cost* function, required to transmit between nodes i and j at time t, is defined as $P_{ij}(t) = D_{ij}^{\alpha}(t) + \gamma$, where $\alpha \geq 2$ and for some constants γ. 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 KADER does not put any extra burden on any particular node[4], γ is common to all nodes and is not of great significance when comparing power costs. Therefore, without loss of generality, we assume $\gamma = 0$[5] and define

$$P_{ij}(t) = D_{ij}^{2}(t) = a_{ij}t^2 + b_{ij}t + c_{ij} \quad (3)$$

as the power cost function used by KADER. By choosing the distance between nodes as the link cost, one obtains minimum power routes that help preserve battery life (see Figure 2).

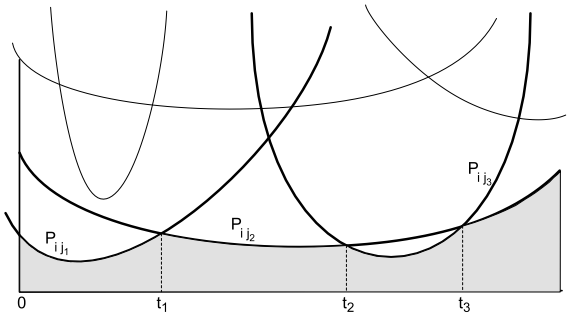


Figure 2: The power function used by KADER, where each parabola represents the energy needed to reach each neighbor of node i

[4]KADER is based on a tree and not on a cluster hierarchy

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

We then define

$$p_i(t) = e^{-\beta_i(t-t_i)} \quad (4)$$

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 3).

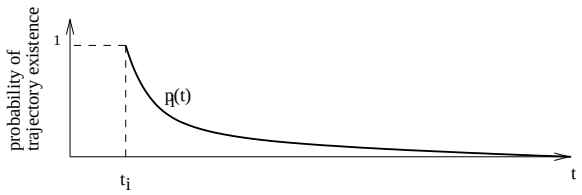


Figure 3: The stability function used by KADER, where the probability for a node i to behave as predicted decreases exponentially

Assuming independent node trajectories,

$$\begin{aligned} p_{ij}(t) &= p_i(t) \cdot p_j(t) \\ &= e^{-(\beta_i+\beta_j)(t-\frac{t_i\beta_i+t_j\beta_j}{\beta_i+\beta_j})} = e^{-\beta_{ij}(t-t_{ij})} \end{aligned} \quad (5)$$

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*.

We finally define

$$W_{ij}(t) = - \frac{p_{ij}(t)}{P_{ij}(t)} \quad (6)$$

$$= - \frac{e^{-(\beta_i+\beta_j)(t-\frac{t_i\beta_i+t_j\beta_j}{\beta_i+\beta_j})}}{a_{ij}t^2 + b_{ij}t + c_{ij}} \quad (7)$$

as the composite link cost between two neighbors (see Figure 4). A low modified power cost favors a low power cost with high stability. We have then five parameters a_{ij}, b_{ij}, c_{ij}, β_{ij}, t_{ij} describing $W_{ij}(t)$ as criteria for a preferred link between two nodes.

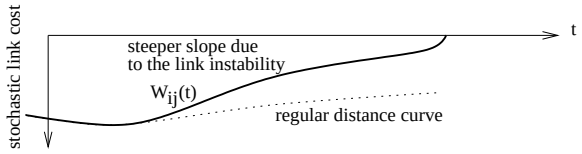


Figure 4: The composite link cost function used by KADER, where we can see the cost increase due to the link's instability.

3.2.2 Election Algorithm

Based on the criterion described above and similarly to [7], a node i can determine its PN j at time t_1, which represents the time at which j has the smallest cost function over all k other neighbors of i.

$$PN_i(t_1) = j \quad iff \quad W_{ij}(t_1) = \min_{k \in nb_i} (W_{ik}(t_1))$$

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

Yet, since we are performing mobility predictions, a PN is not elected for a single time instant t, but for a time interval $[t_1, t_2]$. During this interval, also called *activation*, the link function assigned to this PN is the smallest over all k other neighbors. An activation between node i and node j over an interval $[t_1, t_2]$ is defined as

$$act(i,j)[t_1,t_2] = \begin{cases} \min t_1 \ s.th. \ PN_i(t_1) = j \\ \max t_2 \in (t_1;\infty) \ s.th \ PN_i(t) = j \end{cases}$$

Therefore, the set (i, j,$act(i,j)[t_1,t_2]$) uniquely identifies a *preferred link* between node i and node j activated from t_1 to t_2, and will thereafter mentioned as $\overline{ij}_{[t_1,t_2]}$.

Then, by always considering the smallest link cost function for all time t, i then creates a set of actual and future preferred neighbors that always minimize the link cost.

3.3 Forest Construction

For every time instant, a dynamic forest, or a group of non overlapping dynamic trees, is constructed by connecting each node to its PN (similarly formulated as connecting the set of all preferred links). Due to mobility issues, PNs may change (along with preferred links) during the simulation. But since every node knows in advance the set of its actual and future PNs, thus actual and future preferred links, no future re-configuration or exchange of messages is needed to adapt the topology. We prove in the appendix that, whatever the network topology is, this approach always yields to a forest at each time instant.

In order to construct *preferred links* and consequently the forest, each node generates a table called *Intra-Zone table* (see Table 1). Indeed, as soon as node i determines the set of its PNs, it must notify its neighbors, especially its PNs, of its decision. It first updates its *Intra-Zone table* by adding the set of its PNs in a *PN* field along with their respective activations. Any node appearing in the *PN* field of node i's *Intra-Zone* table means it is either a PN of node i, or it elected node i as PN. Node i then sends a *PN* message $PN_i = (i, \overline{ij}_{[t_1,t_2]})$. This message indicates that node i is electing node j as its PN with the activation $act(i,j)[t_1,t_2]$. Upon reception of i's message, node j checks whether it has been chosen as the PN of i. If so, it also updates its intra-zone table regarding i. Since nodes i and j appear in the *PN* field of their respective *Intra-Zone* table, a tree branch is built between node i and its preferred neighbor j during $[t_1, t_2]$. Therefore, those edges become a preferred link, and the set of preferred links in each neighborhood generates the set of preferred paths in the network.

We illustrated KADER's constructed dynamic forest in Figure 5, considered at time $t = 0$. Full lines represent actual preferred links, while dashed lines future ones. For example, as we can see on the same Figure and on Table 1(a), node k has a PN c activated between $t = 0s$ and $t = 10s$. Since the simulation time $t = 0$, the preferred link $\overline{kc}_{[0,10]}$ is active. But, node k also has a future PN d activated between $t = 10s$ and $t = 20s$. Therefore, at time $t = 0$, the preferred link $\overline{kd}_{[10,20]}$ is not yet activated and is considered as a future preferred link and depicted as a dashed line. It will be quietly activated at time $t = 10$.

3.4 Self-Adaptive Intra-Zone Clustering

KADER aims at regrouping closed-by nodes into a zone in order to provide energy efficient communications. At this phase, we illustrate how nodes obtain the best path, with respect to the electing criterion, to reach all nodes in their zone and proactively maintain them in their intra_zone table. The process is very similar to [7], and readers may refer to this paper for a more detailed description.

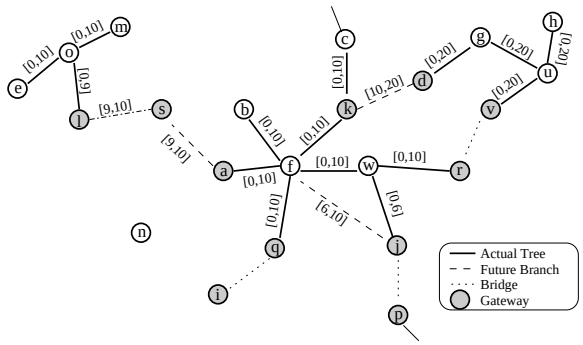


Figure 5: Constructed forest by KADER, considered at time $t = 0$. The brackets represent the links' activation intervals.

When a node i gets elected by a neighbor j, it then locally notify all its neighbors of this election. To do so, i sends a so called *Learned_PN* message $Learned\_PN_i = (i, \overline{ij}_{[t_1,t_2]})$, indicating that node j with $act(i,j)[t_1,t_2]$ has node i as its PN. Upon reception of this message, each tree member updates its intra_zone table, and re-advertises to its neighbors if it is not a leaf node[7]. For this purpose, each node generates another field in its intra_zone table called *Learned Preferred Neighbor* (*Learned_PN*, see Table 1) in order to keep nodes that have been learned to belong to the same tree. Therefore, if node i is chosen to be the PN of j over a time interval $[t_1, t_2]$, j sends a *PN* message to inform its neighborhood of its elected PN. Among the neighboring nodes of j, the PN i forwards j's decision to each node that holds a tree edge with i, say node k, activated over a time interval $[t_3, t_4]$. Then, the local view of k's tree is that, over the time interval $([t_1, t_2] \bigcap [t_3, t_4])$, j is reachable through i. For example, in Figure 5 and in Table 1(b), node j elected node w as PN for a time interval $[0, 6]$. Node j is then a *Learned_PN* of node f over the time interval $([0, 6] \bigcap [0, 10]) = [0, 6]$.

Consequently, as we can see in Table 1, on convergence of KADER, the *PN* field represents the next hop nodes to reach any node belonging to its zone (appearing in the *Learned_PN* field). This is a very interesting feature for routing since the end-to-end delay for route discovery may be limited.

PN	Learned_PN
$\mathbf{f}_{[0,10]}$	$a_{[0,10]}, b_{[0,10]}$
	$q_{[0,10]}, w_{[0,10]}$
	$r_{[0,10]}, j_{[0,10]}$
	$l_{[9,10]}, s_{[9,10]}$
$\mathbf{d}_{[10,20]}$	$g_{[10,20]}, u_{[10,20]}$
	$v_{[10,20]}, h_{[10,20]}$
$c_{[0,10]}$	-

(a) $Intra\_ZT_k$

PN	Learned_PN
$w_{[0,10]}$	$r_{[0,10]}$
	$j_{[0,6]}$
$\mathbf{k}_{[0,10]}$	$c_{[0,10]}, d_{[0,10]}$
$j_{[6,10]}$	-
$b_{[0,10]}$	-
$a_{[0,10]}$	$s_{[9,10]}, l_{[9,10]}$
$q_{[0,10]}$	-

(b) $Intra\_ZT_f$

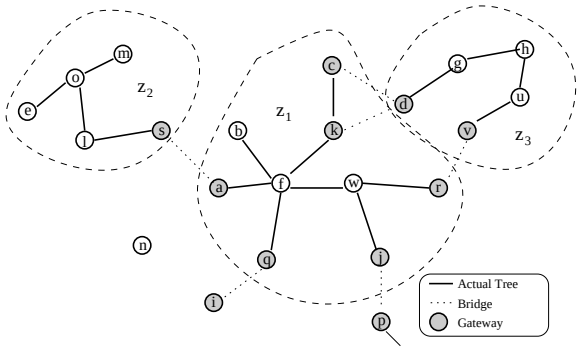
Table 1: Intra-zone table of nodes k and f regarding Figure 5

3.5 Self-Adaptive Inter-Zone Clustering

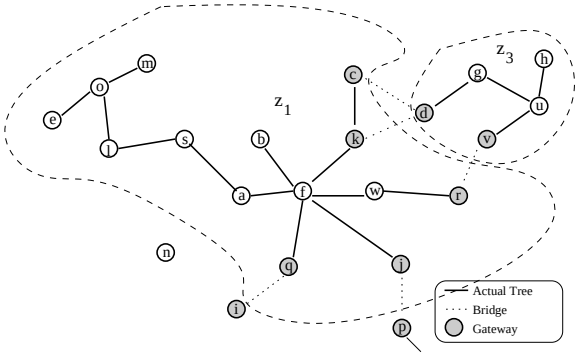
Once zones have been built, KADER's task is to keep them connected to each others at each time instant. For that matter, KADER uses a different table, called *Inter_Zone* table, that regroups connections to different surrounding zones. Each node belonging to this table is referred as *gateway* and the link connecting two zones as *bridge* (see Figure 6). Moreover, each gateway must keep a connection to each of its peer-gateways belonging to its neighboring zones. We prove in the appendix that the zones created by KADER are always connected to each others if the original graph is complete.

At the beginning, neighbors of a node i are put in its *Inter_Zone* table during their full initial activation, say $[T_1, T_2]$, which is defined as the connection lifetime between the two nodes or the time two nodes remain direct neighbors. Then, as node i succeeds to add a neighbor j to its tree and updates its intra_zone table over an activation $[t_1, t_2]$, it prunes j's initial activation. The remaining activation is then $\{[T_1, T_2] \backslash [t_1, t_2]\}$. During this time, node j is still not considered part of i's tree. Node j then appears in the *Intra_Zone* table over $[t_1, t_2]$ and in the *Inter_Zone* table over $\{[T_1, T_2] \backslash [t_1, t_2]\}$.

Fore example, in Figure 6 and corresponding tables in Table 2, node s is in the *Inter_Zone* table and also in the *Intra_Zone* table of node a, yet for different time intervals. For the time interval $[0, 9]$, node a is a gateway node and the link $\overline{sa}$ is a bridge. However, from the time interval $[9, 10]$, zone $z1$ grows and zone $z2$ fusions with it. Accordingly, for this time interval, node s becomes a *PN* and nodes l, o, m, e *Learned_PN* for node a. As we can see, the size of different zones grows and shrinks over time, which makes them self-adapting to the mobility-based topology changes, yet without exchanges of messages.



(a) Constructed Forest from t=0s to t=9s



(b) Construes Forest from t=9s to t=10s

Figure 6: KADER Constructed Forest and its evolution over time

[7] A leaf node is a node which only has a single neighbor and which is never a PN.

Gate. ID	Zone ID
$s_{[0,9]}$	*z2*

(a) Inter_zone table of node a regarding Figures 6(a) and 6(b) .

PN	Learned_PN
$\mathbf{f}_{[0,10]}$	$c_{[0,10]}, b_{[0,10]}$
	$q_{[0,10]}, w_{[0,10]}$
	$r_{[0,10]}, j_{[0,10]}$
	$k_{[0,10]}$
$\mathbf{s}_{[9,10]}$	$l_{[9,10]}$

(b) Intra_zone table of node a regarding Figures 6(a) and 6(b).

Table 2: Inter-zone and Intra-zone tables of nodes a regarding Figure 6

4. MAINTENANCE OF KADER'S BACKBONE

Zone maintenance consists of several tasks, such as removal of irrelevant links, acquisition of new neighbors, notification of link errors. Most of topology control algorithms perform these tasks periodically. What makes KADER unique is its ability to do these tasks in a complete per-event manner (i.e. non periodically). Since it predicts future topology configurations, KADER triggers a zone maintenance only when those predictions appear to have failed, in other words, when nodes changed their trajectories.

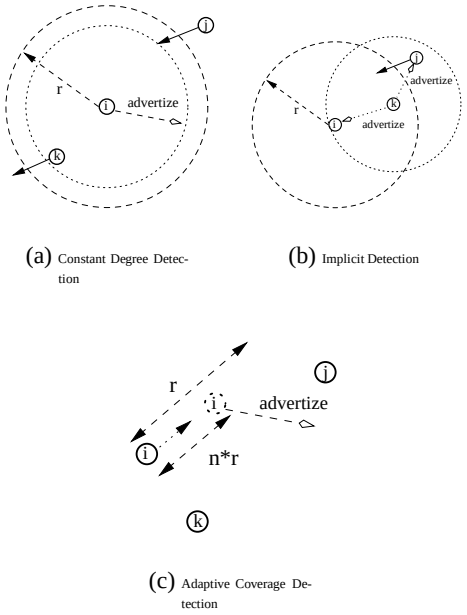
4.1 Adaptive Aperiodic Neighborhood Maintenance

A limitation in per-event topology maintenance strategies is the neighborhood maintenance. While trajectory knowledge allows to discard 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 KADER's ability to obtain up-to-date links and effective low power routes within each zone.
We developed several heuristics to help KADER detecting 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 7(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 7(b), node k is aware of both i and j's trajectories, thus is able to compute the moment at which either j or i enters each other's transmission range. Therefore, node k sends a notification 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, its velocity and its stability (see Figure 7(c));



(a) Constant Degree Detection (b) Implicit Detection

(c) Adaptive Coverage Detection

Figure 7: Three heuristics to detect incoming neighbors in a per-event basis

4.2 Quasi-local topology maintenance

In this subsection, we show how a particular message, called *New Trajectory* (NT), is used to inform neighboring nodes of any topology changes, and to trigger a quasi-local maintenance process. Therefore, KADER's zone maintenance can be seen as per-event based.

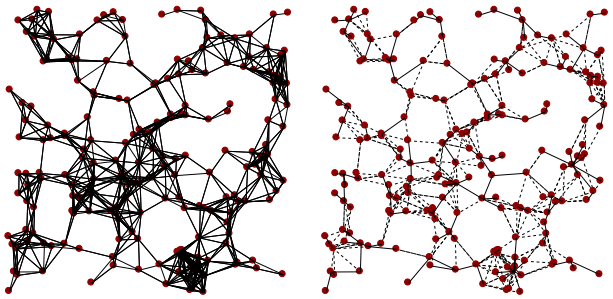
As mentioned before, node trajectories information remains valid during a short period of time. Then, since a node is unable to predict the time its neighbors will change their trajectories, it biases the link cost W to reflect the decreasing probability of the link existence, and to ensure that a good low power but unstable link could not be chosen for routing. Yet, we still consider this link valid as long as it is not otherwise notified. Consequently, when a node is changing its trajectory, it must inform its neighbors about the induced topology change. To do so, it sends a *New Trajectory* (NT) message to all its neighbors and piggybacks its new coordinates and velocity. Therefore, its neighbors are able to adapt their trees to this event. Eventually, the algorithm carries out the PN election phase again.

5. PROPERTIES OF KADER'S TOPOLOGY

In this section, we study KADER's computed backbone with parameters such as average zone diameter (i.e. in term of number of hops), average number of zones in the network, average ratio of remaining edges, average ratio of PNs in the network, and average power assignment. The following results are obtained by measuring the metrics after the population of mobile nodes was distributed uniformly on a $A \times A$ grid where $A = 2000m$, with each node having a transmission range of 250m. Moreover, each node has a different stability value, but nodes' average stability is $1/\beta = 30s$. We will compare KADER in two different cases : *variable density* and *globally constant density*[8].

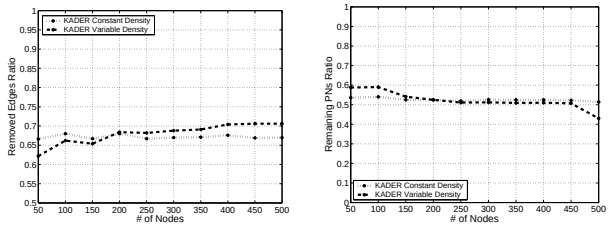
[8]Globally constant density is obtained by maintaining the ratio $\frac{\#nodes}{A^2}$ fixed

We begin by illustrating in Figure 8 the topology created by KADER from an arbitrary graph G (see Figure 8(a)) to a forest and trees (see Figure 8(b)), where solid lines are tree edges and dashed lines are bridges connecting different trees.

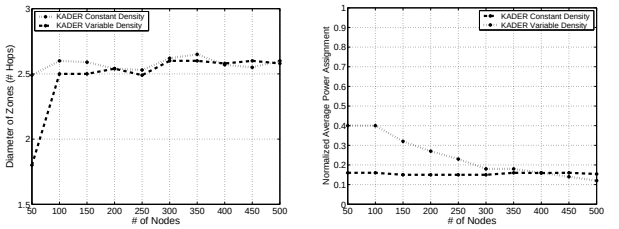


(a) An arbitrary graph (b) Constructed forest

Figure 8: KADER's Topology



(a) Removed Edges Ratio (b) Remaining PNs Ratio



(c) Diameter of Zones (d) Average Power Assignment

Figure 9: Properties of KADER's Topology

We can see in Figure 9(a) that KADER is able to remove 65% of the total number of edges. By getting rid of unnecessary links, KADER helps improve spatial reuse and concurrent communications. Then, in Figure 9(b), KADER is able to remove 45% of PNs, which helps to reduce the broadcasting overhead in the network. Therefore, by removing 65% of unecessary links and by only keeping a backbone of 55% of router nodes yet keeping the graph connected, KADER helps performing load reduction and proves to be broadcast efficient.

Then, the graph in Figure 9(c) shows the diameter of a zone versus the number of nodes in the network. The diameter of a zone is defined as the length of the path that has the longest euclidean distance. If the zone diameter is obtained, then we can place an upper bound on the Intra_zone end-to-end delay. We clearly see in Figure 9(c) that zones in KADER are relatively stable, in both a variable and a constant density. This comes from its distance parameter in the electing criterion. Since KADER always tries to create a link between nearby neighbors, neither density nor the number of nodes have a big impact on the zone's diameter. This shows another interesting property of KADER that is its suitability for dense and sparse networks on non-evenly distributed nodes.

Finally, Figure 9(d) illustrates KADER's average node power assignment ratio while keeping the graph connected. We can see that on average, KADER is able to lower the power assignment by 75%. The most important asset of a reduced power assignment is the increased battery lifespan. Yet, reducing the transmission range also causes less contention and interference for concurrent communications. This is even amplified for dense networks, where interferences reduce broadcasting efficiency. Therefore, KADER's low power assignment not only being energy efficient, also helps improve unicast and broadcast communications.

6. CONVERGENCE AND OVERHEAD COMPLEXITY OF KADER'S TOPOLOGY

It is important to compute the communication complexity of KADER for topology creation. The communication complexity describes the average number of messages required to perform a protocol operation. Note that this comparison does not include the complexity of route discovery [9]. This issue is not covered in this paper.

6.1 Message Complexity (MC)

In KADER, the network is partitioned into m zones on the average, and each zone has an average number of nodes of $\frac{n}{m}$. The amount of communication overhead to build and maintain the forest is n since by sending n PN election messages, a forest will be constructed. To construct a zone, each node generates (d-1) messages to forward the learned PN or removed PN, where d is the hop-wise zone diameter. Therefore, each zone generates $(d-1)\frac{n}{m}$ messages. Since there exists m zones in the network, the overall number of generated forward messages becomes $(d-1)n$. In conclusion, the message complexity is O($\beta \cdot n \cdot d$),where $d < 3$ (see Figure 9(c)).

6.2 Time Complexity (TC)

In order to obtain the time complexity induced by KADER, we need to analyze the broadcasting overhead in mobile ad-hoc network. Authors in [16] showed that $\Omega(\mathrm{n})$ rounds are required by any broadcasting protocol when the network nodes are mobile. Accordingly, the broadcasting overhead does not change whether nodes are arranged in lines or in mesh, but only depends on the number of nodes in the network. Therefore, since KADER converges when a *Learn_PN* message has reached every nodes in a zone, and that KADER's zones have on the average n/m nodes, the time complexity of KADER is $\Omega(\mathrm{n/m})$.

[9]KADER being zone-wise proactive, it is able to obtain intra-zone paths at no extra cost.

7. BENEFIT OF KADER'S TOPOLOGY ON ROUTING ALGORITHMS

KADER is able to derive the most stable links from a network topology such that full connectivity is always guarantied. It then becomes interesting to analyze the benefits routing protocols may obtain from it.

7.1 Efficient Routing

KADER is able to group nodes into a set of zones, which proactively maintains routes between every node belonging to the same zone. Therefore, any routing protocol using KADER may perform *Intra_Zone* routing at no extra cost. For *Inter_Zone* routing, a protocol still needs to be determined. It could be imagined that a reactive approach, such that AODV [13], may take great help of the topology created in KADER by reducing the overhead of its *route discovery* procedure. This creates an hybrid routing protocol, using proactive intra-zone routing, and on-demand zone-level routing. On the other hand, similar to OLSR [12] using MPR [11], a proactive protocol takes benefit from KADER to improve its scalability and its end-to-end delay.

7.2 Energy Efficiency

In KADER, during the construction of the forest, every node elects its *Preferred Neighbor* partly depending on the energy needed to reach it. Indeed, the transmission range of the Intra_zone routing is always adapted to reach only the desired PN. Hence, by using the power assignment, KADER performs topology control and makes proactive *Intra_Zone* routes optimal in terms of energy data flow generated and forwarded by each node, further reducing the energy used for routing and increasing the channel capacity. It also improves concurrent communications by reducing interferences and contentions. Since KADER does not use beacons, routing protocols using KADER reach routing energy efficiency.

8. CONCLUSION

In this paper, we have presented a novel approach to topology control protocols, called Kinetic Adaptive Dynamic topology control for Energy efficient Routing (KADER). It employs nodes' trajectory knowledge to get rid of periodic beacons and to fit its structure to nodes mobility patterns. The major properties of KADER are *Linear Complexity, Scalability, and Energy Efficiency*.

We showed that by using trajectory knowledge to predict nodes future positions, KADER is able to dynamically create and maintain a connected backbone without using periodic beacons. Results pointed out that the structure created and maintained by KADER was composed of only 45% of nodes, and 65% of links composing the original network, while always being able to keep a full connectivity between every node. KADER also lowers the power assignment at each node by 75%. Then, KADER is also able to provide each node with the next hop node to reach any neighbor belonging to the same zone. Therefore, best intra-zone paths with respect to the link cost using in 3.2.1 are obtained at no extra cost.

Finally, using a similar denomination than those in routing protocols, KADER can be classified as the first totally reactive topology control protocol, since it is only triggered when an event occurs. Therefore, by initiating topology maintenance only when a node is changing course and not periodically, KADER reaches energy efficiency and scalability for topology control.

9. REFERENCES

[1] P. Gupta and P. R. Kumar, "The capacity of wireless networks", in *IEEE Trans. Inform. Theory*, vol. 46, no. 2, pp. 388-404, March 2000.

[2] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar, "Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the compow protocol", in *Proc. of the European Wireless'02, ,* pp. 156-162, Florence, Italy, February 2002.

[3] L. Li, J. Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer, "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks", in *Proc. ACM Symposium on Principles of Distributed Computing*, pp. 264-273, Newport, Rhode Island, United States, August 2001.

[4] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment", in *Proc. IEEE INFOCOM 2000*, pp. 404-413, Tel Aviv, Israel, March 2000.

[5] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks", in *IEEE J. Select. Areas Commun.*, pp. 1333-1344, vol. 17, no. 8, August 1999.

[6] Li Li and Joseph Y. Helpern, "A Minimum-energy Path-preserving Topology-control Algorithm", in *IEEE Transactions on Wireless Communications*, pp. 910-921, vol. 3, no. 3, 2004.

[7] Navid Nikaein, Houda Labiod , and Christian Bonnet, "Distributed Dynamic Routing Algorithm for Mobile Ad-Hoc Networks," *Proc. MobiHOC 2000*, USA/Boston.

[8] C. Gentile, J. Haerri, and R. E. Van Dyck, "Kinetic minimum-power routing and clustering in mobile ad-hoc networks," *IEEE Proc. Vehicular Technology Conf. Fall 2002*, pp. 1328-1332, September 2002.

[9] R.J. Fontana and S.J. Gunderson, "Ultra-wideband precision asset location system", *IEEE Conf. on Ultra Wideband Systems and Technologies*, pp. 147-150, 2002.

[10] C. Gentile and Luke Klein-Berndt, "Robust Location using System Dynamics and Motion Constraints", in *Proc. of the IEEE International Conference on Communications (ICC'04)*, Paris, June 2004.

[11] A. Laouiti *et al*, "Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks", *35th Annual Hawaii International Conference on System Sciences (HICSS'2001)*, Hawaii, USA, 2001.

[12] T. Clausen *et al*, "Optimized Link State Routing Protocol", *IEEE INMIC*, Pakistan, 2001.

[13] C.E. Perkins *et al.* "Ad Hoc On-Demand Distance Vector Routing", `http://www.ietf.org/rfc/rfc3561.txt`.

[14] N. Li, J.C. Hou, L. Sha, "Design and analysis of an MST-based topology control algorithm", in *Proc. of the IEEE Infocom Conference*, pp. 1702-1712, Vol. 3, San Francisco, 2003.

[15] J. Haerri and Christian Bonnet, "A Lower Bound for Vehicles' Trajectory Duration", in *Proc of the IEEE 62nd Semiannual Vehicular Technology Conference (VTCFall'05)*, Dallas, USA, September 2005.

[16] Ravi Prakash *et al.*, "A Lower Bound for Broadcasting in Mobile Ad Hoc Networks", in *EPFL Technical Report IC/2004/37*, EPFL, Switzerland, 2004.

[17] Xiang-Yang Li *et al.*, "Applications of k-local MST for topology control and broadcasting in wireless ad hoc networks", in *IEEE Transactions on Parallel and Distributed Systems*, pp. 1057-1069, vol. 15, no. 12, December 2004.

[18] F.J. Ovalle-Martinez, I. Stojmenovic, F. Garcia-Nocetti, J. Solano-Gonzalez, " Finding minimum transmission radii and constructing minimal spanning trees in ad hoc and sensor networks", in *Journal of Parallel and Distributed Computing*, pp. 132-141, vol. 65, no. 2, February 2005.

[19] Roger Wattenhofer, Aaron Zollinger "XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks", in *4th IEEE Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Santa Fe, April 2004.

[20] P. Santi, D. M. Blough, and F. Vainstein, A probabilistic analysis for the range assignment problem in ad hoc networks, in *Proc. ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2001)*, pp. 212-220, Long Beach, California, United States, August 2000.

[21] S. Basagni, D. Turgut, and S. K. Das, "Mobility-adaptive protocols for managing large ad hoc networks", in *Proc. IEEE International Conference on Communications (ICC 2002)*, pp. 1539-1543, Helsinki, Finland, June 2001.

APPENDIX

DEFINITION 1. *An arbitrary undirected time dependent graph $G(t)$ is defined as $G(t) = (V, E(t))$, where V is the set of vertice, and $E(t)$ is the set of edges at time t.*

THEOREM 1. *For any graph $G(t)$, let $G'(t) = (V, E'(t))$ be the subgraph obtained by connecting each vertex V to its preferred links $E'(t)$. Then $G'(t)$ is a forest.*

PROOF. Let $G(t)$ be the original graph at time t, and let $G'(t)$ be the graph obtained by executing the KADER algorithm for each vertex $v \in V$ at time t. We first recall that the main idea is to select for each node $v \in G(t)$, a neighbor that has the maximum link function *W*. In order to prove that $G'(t)$ does not contain any cycle $C = v_i, v_{i+1}, \ldots, v_{i-1}, v_i$, let us suppose the contrary, and let v_i be the vertex of C with the biggest W.

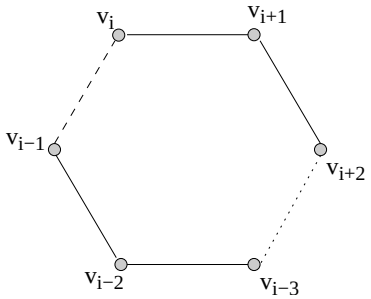


Figure 10: The proof of theorem 1

Let us consider two vertice of v_{i-1} and v_{i+1} adjacent to v_i in C (Figure 10). Without loss of generality, assume that the algorithm on v_i chosen an adjacent vertex v_{i+1} (if neither v_{i-1} nor v_{i+1} had been chosen, C is not a cycle). Consider now the execution of the algorithm on v_{i-1}. We will show that such node will not choose v_i, thus implying that C is not a cycle.

Lets define $f(v_i, v_{i+1})$ as the weight function W between v_i and v_{i+1}. Since v_i chooses v_{i+1}, $f(v_i, v_{i+1}) > f(v_i, v_{i-1})$. And by v_{i-2}'s decision to choose v_{i-1}, $f(v_{i-2}, v_{i-1}) > f(v_{i-2}, v_{i-3}) > f(v_i, v_{i+1})$, f being an monotone increasing function. Therefore, $f(v_i, v_{i-1}) < f(v_{i-1}, v_{i-2})$. This proves that v_{i-1} will not choose v_i as PN, and C will not be a cycle. □

THEOREM 2. *For any PN activation $act(v_i, v_{i+1})[t_1, t_2]$, and any graph $G'(t_1, t_2) = (V, E'(t_1, t_2))$ obtained by connecting each vertex to its preferred links $E'(t_1, t_2)$ activated during $[t_1, t_2]$, $G'(t_1, t_2)$ is then always a forest at every time instant included in $[t_1, t_2]$.*

PROOF. When a node v_i elects a PN v_{i+1} during an activation $act(v_i, v_{i+1})[t_1, t_2]$, it means that $\forall t \in [t_1, t_2]$, $f(v_i, v_{i+1})(t) > f(v_i, v_k)(t)$, $\forall k$. Since nodes share a common clock, all their current left activation are equal to the current time and will thereafter be considered as 0, past activations being irrelevant.

If a node v_i elects a PN v_{i+1} during an activation $act(v_i, v_{i+1})$ $[0, t_1]$, without loss of generality, v_{i+1} can elect a node v_{i+2} as PN during an activation $act(v_{i+1}, v_{i+2})[0, t_2]$. Since the algorithm prunes the activation between v_i and v_{i+2} as $([0, t_1] \bigcap [0, t_2])$, we must consider two cases. In the first case, the initial activation $act(v_i, v_{i+1})$ $[0, t_1]$ is less or equal than $act(v_{i+1}, v_{i+2})[0, t_2]$, thus it is kept unaltered during the forwarding steps. In the second case, the algorithm prunes the initial activation.The forwarded activations are two separated and mutually exclusive activations.

Let us consider t_2 smaller or equal to t_1. Then, following the development in the proof of Theorem 1, at some point, node v_{i-1} could elect node v_i during an activation $act(v_i, v_{i-1})[0, t_3]$, as $t_3 <= t_1$. $[0, t_3]$ is the remaining activation after multiple pruning at each node in the path. Then, it means that $\forall t \in [0, t_3]$, v_{i-1} could elect v_i as PN, thus creating a cycle during this time. Theorem 1 prove that this situation is not possible, since $\forall t \in [0, t_3]$, we obtain a stable tree which is not a function of t. Then, during the activation $act(v_i, v_{i-1})[0, t_3]$, C does not contain any cycle.

Since the initial activation has been pruned, we still need to consider the case of the remaining activation ($[t_3, t_1]$). Without loss of generality, let us consider that this activation has been pruned at a single node v_{i+2}. This node has the possibility to elect v_{i+1} as PN (mutual election), updating the mutual activation as the union of their respective ones. Note that this case does not create a cycle. v_{i+2} can otherwise elect another node, say v_{i+3}. Since $[t_3, t_1]$ $\bigcap$ $[0, t_3] = \emptyset$, $\overline{v_{i+2}v_{i+3}}$ is then a branch of a different and independent tree and the situation is independent to the previous one. Therefore, this neither creates a cycle, which concludes the proof. □

THEOREM 3. *$\forall$ G, let G' be the subgraph obtained by connecting each node to its preferred links during their respective activations. Then G' is a forest at every time instant.*

PROOF. $\forall$ node v_i, since all its PNs activation intervals are mutually exclusive ($\bigcap (act(v, v_1), \ldots, \ act(v, v_n)) = \emptyset$), from Theorem 2, we can conclude that KADER always yield to a forest at every time instant. □

THEOREM 4. *For any complete graph G(t), the subgraph $G'(t) = (V, E'(t))$ created by connecting each vertex to its preferred neighbor creates a set of connected zones.*

PROOF. Let us consider the contrary and take two unconnected zones A and B. At any time instant, a neighbor of a node either belongs to its *Intra_Zone* table or to its *Inter_Zone* table. The former means both nodes belong to the same zone, and the latter means both nodes belong to two adjacent zones which they are connecting. Therefore, if a link between two nodes do not exist between two zones in $G'(t)$, it also could not exist in the original graph $G(t)$. This is a contradiction to the hypothesis of a complete graph $G(t)$. □