On the Average and Worst-case Efficiency of Some New Distributed Communication and Control Algorithms for Ad-hoc Mobile Networks *

Ioannis Chatzigiannakis[1,2] Sotiris Nikoletseas[1,2] Paul Spirakis[1,2]

Abstract

An ad-hoc mobile network is a collection of mobile hosts, with wireless communication capabilities, forming a temporary network without the aid of any established fixed infrastructure.

In such networks, topological connectivity is subject to frequent, unpredictable change. Our work focuses on networks with high rate of such changes to connectivity. For such dynamic changing networks we propose protocols which exploit the *co-ordinated* (by the protocol) motion of a *small part* of the network. We show that such protocols *can be designed to* work correctly and efficiently even in the case of arbitrary (but not malicious) movements of the hosts not affected by the protocol.

We also propose a methodology for the *analysis of the expected* behaviour of protocols for such networks, based on the *assumption* that mobile hosts (whose motion is not guided by the protocol) conduct concurrent *random walks* in their motion space.

Our work examines some fundamental problems such as pair-wise communication, election of a leader and counting, and proposes distributed algorithms for each of them. We provide rigorous proofs of their correctness, and also give their analysis by combinatorial tools and also via experiments.

*This work was partially supported by the EU projects IST FET-OPEN ALCOM-FT, IMPROVING RTN ARACNE and the Greek GSRT Project PENED99-ALKAD.

[1]Computer Technology Institute, 61 Riga Fereou Str, 26221 Patras, Greece.
E-mail: {ichatz,nikole,spirakis}@cti.gr

[2]Computer Engineering and Informatics Department, Patras University, 26500 Rion, Patras, Greece.

1 Introduction

1.1 The rate of topology changes and our approach

Mobile computing has been introduced (mainly as a result of major technological developments) in the past few years forming a new computing environment. Because of the fact that mobile computing is constrained by poor resources, highly dynamic variable connectivity and volatile energy sources, the design of stable and efficient mobile information systems is greatly complicated. Until now, two basic system models have been proposed for mobile computing. The "fixed backbone" mobile system model has been around the past decade and has evolved to a fairly stable system that can exploit a variety of information in order to enhance already existing services and yet provide new ones. On the other hand, the "ad-hoc" system model assumes that mobile hosts can form networks without the participation of any fixed infrastructure, a fact that further complicates their design and implementation.

An ad-hoc mobile network ([34, 23, 1]) is a collection of mobile hosts with wireless network interfaces forming a temporary network *without the aid of any established infrastructure or centralised administration.* In an ad-hoc network two hosts that want to communicate may not be within wireless transmission range of each other, but could communicate if other hosts between them are also participating in the ad-hoc network and are willing to forward packets for them.

Ad-hoc mobile networks have been modeled by most researchers by a set of independent mobile nodes communicating by message passing over a wireless network. The network is modelled as a dynamically changing, not necessarily connected, undirected graph, with nodes as vertices and *edges* (*virtual links*) between vertices corresponding to nodes that can currently communicate. Such a model is used, for example, in [27, 24, 34] etc.

However, the proof of correctness of algorithms presented under such settings requires a bound on the virtual link changes that can occur at a time. As [34] also states, in ad-hoc mobile networks the topological connectivity is subject to frequent, unpredictable changes (due to motion of hosts). If the *rate* of topological change is very high, then *structured* algorithms (i.e. algorithms that try to maintain data structures based on connectivity) fail to react fast enough and [34] suggest that, in such cases, the only viable alternative is flooding. If the rate is low (quasi-static networks) or medium, then adaptive algorithmic techniques can apply. In such settings many nice works have appeared like e.g. the work of [27] on leader election, the work of [1] on communication (which however examines only *static* transmission graphs) etc.

The most common way to establish communication in ad-hoc networks is to form paths of intermediate nodes that lie within one another's transmission range and can directly communicate with each other [23, 24, 33, 35]. The mobile nodes act as hosts and routers at the same time in order to propagate packets along these paths. Indeed, this approach of exploiting pairwise communication is common in ad-hoc mobile networks that cover a relatively small space (i.e. with diameter which is small with respect to transmission range) or are dense (i.e. thousands of wireless nodes) where all locations are occupied by some hosts; broadcasting can be efficiently accomplished.

In wider area ad-hoc networks with less users, however, broadcasting is impractical: two distant peers will not be reached by any broadcast as users may not occupy all intermediate locations (i.e. the formation of a path is not feasible). Even if a valid path is established, single link "failures" happening when a small number of users that were part of the communication path move in a way such that they are no longer within transmission range of each other, will make this path invalid. Note also that the path established in this way may be very long, even in the case of connecting nearby hosts.

We indeed conjecture that, in cases of high mobility rate and/or of low density of user spreading, one can even state an *impossibility result*: any algorithm that tries to maintain a global structure with respect to the temporary network will be erroneous if the mobility rate is faster than the rate of updates of the algorithm.

In contrast to all such methods, we try to avoid ideas based on paths finding and their maintenance. We envision networks with highly dynamic movement of the mobile users, where the idea of "maintenance" of a valid path is inconceivable (paths can become invalid immediately after they have been added to the directory tables). Our approach is to take advantage of the mobile hosts natural movement by exchanging information whenever mobile hosts meet incidentally. It is evident, however, that if the users are spread in remote areas and they do not move beyond these areas, there is no way for information to reach them, unless the protocol takes special care of such situations.

In the light of the above, we propose the idea of forcing a small subset of the deployed hosts to move as per the needs of the protocol. We call this set the "support" of the network. Assuming the availability os such hosts, the designer can use them suitably *by specifying their motion* in certain times that the algorithm dictates. We admit that the assumption of availability of such hosts for algorithms deviates from the "pure" definition of ad-hoc mobile networks. However, we feel that our approach opens up an area for distributed algorithms design for ad-hoc mobile networks of high mobility rate.

Our approach is motivated by two research directions of the past:

(a) The "two tier principle" [23], stated for mobile networks with a fixed subnetwork however, which says that any protocol should try to move communication and computation to the fixed part of the network. Our assumed set of hosts that are coordinated by the protocol *simulates* such a (skeleton) network; the difference is that the simulation actually constructs a coordinated *moving* set.

(b) A usual scenario that fits to the ad-hoc mobile model is the particular case of rapid deployment of mobile hosts, in an area where there is no underlying fixed infrastructure (either because it is impossible or very expensive to create such an infrastructure, or because it is not established yet, or it has become temporarily unavailable i.e. destroyed or down).

In such a case of rapid deployment of a number of mobile hosts, it is possible to have a small team of fast moving and versatile vehicles, to implement the support. These vehicles can be cars, jeeps, motorcycles or helicopters. We interestingly note that this small team of fast moving vehicles can also be a collection of independently controlled mobile modules, i.e. robots. This specific approach is inspired by the recent paper of J.Walter, J.Welch and N.Amato. In their paper "Distributed Reconfiguration of Metamorphic Robot Chains" ([39]) the authors study the problem of motion coordination in distributed systems consisting of such robots, which can connect, disconnect and move around. The paper deals with metamorphic systems where (as is also the case in our approach) all modules are identical. Note that the approach of having the support moving in a co-ordinated way, *i.e. as a chain of nodes*, has some similarities to [39].

In fact, a recent work of Q.Li and D.Rus [26] presented a model which has some similarities to ours: The authors give an interesting, yet different, protocol to send messages, which forces *all the mobile hosts to slightly deviate (for a short period of time) from their predefined, deterministic routes, in order to propagate the messages.* Their protocol thus forces the alteration of motion for any host and it works only for deterministic host routes. In their setting, [26] show optimality of message transmission times.

1.2 The explicit model of motions

1.2.1 The motion space

The set of previous research that follows the approach of slowly-changing communication graphs, models hosts motions only implicitly, i.e. via the pre-assumed upper bound on the rate of virtual link changes. In contrast, we propose an *explicit* model of motions because it is apparent that the motions of the hosts are the cause of the fragility of the virtual links.

Thus we distinguish explicitly between (a) the *fixed* (for any algorithm) *space* of possible motions of the mobile hosts and (b) the kind of motions that the hosts perform inside this space. In the sequel we have decided to model the space of motions only combinatorially, i.e. as a graph. We however believe that other research will complement this effort by introducting geometry details into the model.

In particular, we abstract the environment where the stations move (in three-dimensional space with possible obstacles) by a *motion- graph* (i.e. we neglect the detailed geometric characteristics of the motion. We expect that future research will incorporate geometry constraints into the subject). In particular, we first assume that each mobile host has a transmission range represented by a sphere tr centred by itself. This means that any other host inside tr can receive any message broadcasted by this host. We approximate this sphere by a cube tc with volume $\mathcal{V}(tc)$, where $\mathcal{V}(tc) < \mathcal{V}(tr)$. The size of tc can be chosen in such a way that its volume $\mathcal{V}(tc)$ is the maximum that preserves $\mathcal{V}(tc) < \mathcal{V}(tr)$, and if a mobile host inside tc broadcasts a message, this message is received by any other host in tc. Given that the mobile hosts are moving in the space $\mathcal{S}$, $\mathcal{S}$ is divided into consecutive cubes of volume $\mathcal{V}(tc)$.

Definition 1 *The motion graph* $G(V, E)$, $(|V| = n, |E| =$

m), which corresponds to a quantization of $\mathcal{S}$ is constructed in the following way: a vertex $u \in G$ represents a cube of volume $\mathcal{V}(tc)$. An edge $(u, v) \in G$ if the corresponding cubes are adjacent.

The number of vertices n, actually approximates the ratio between the volume of space $\mathcal{S}$, $\mathcal{V}(\mathcal{S})$, and the space occupied by the transmission range of a mobile host $\mathcal{V}(tr)$. In the extreme case where $\mathcal{V}(\mathcal{S}) \approx \mathcal{V}(tr)$ (the transmission range of the hosts approximates the space that they are moving), then $n = 1$. Given the transmission range tr, n depends linearly on the volume of space $\mathcal{S}$ regardless of the choice of tc, and $n = O\left(\frac{V(\mathcal{S})}{V(tr)}\right)$. Let us call the ratio $\frac{V(\mathcal{S})}{V(tr)}$ by the term *relative motion space size* and denote it by ρ. Since the edges of G represent neighbouring polyhedra each node is connected with a constant number of neighbours, which yields that $m = \Theta(n)$. In our example where tc is a cube, G has maximum degree of six and $m \leq 6n$.

Thus *motion graph* G is (usually) a *bounded degree graph* as it is derived from a regular graph of small degree by deleting parts of it corresponding to motion or communication obstacles. Let Δ be the maximum vertex degree of G.

1.2.2 The motion of the hosts-Adversaries

The motion that the hosts perform (we mean here the hosts that are not part of the support i.e. those whose motion is *not specified* by the distributed algorithm) is *an input* to any distributed algorithm.

(a) In the general case, we assume that the motions of such hosts are decided by an *oblivious adversary*: The adversary determines motion patterns in any possible way but independently of the part of the distributed algorithm that specifies motions of the support. In other words, we exclude the case where some of the hosts not in the support are deliberately trying to *maliciously affect* the protocol (e.g. to avoid the hosts in the support). This is a pragmatic assumption usually followed by applications. We call such motion adversaries the *restricted motion adversaries.*

(b) For purposes of studying efficiency of distributed algorithms for ad-hoc networks *on the average*, we propose that the motions of any host not affected by the algorithm are modelled by *concurrent and independent random walks.* In fact, the assumption that the mobile users move randomly, either according to uniformly distributed changes in their directions and velocities or according to the random waypoint mobility model by picking random destinations, has been used by other research (see e.g. [18, 22]).

We interestingly note here a fundumental result of graph theory, according to which, dense graphs look like random graphs in many of their properties. This has been noticed for expander graphs at least by [5, 2] and is captured by the famous Szemeredi's Regularity Lemma [38] which says that in some sense most graphs can be approximated by random-looking graphs.

In analogy, we conjecture that any set of *dense (in number)* but arbitrary otherwise motions of many hosts in the motion space can be approximated (at least with respect to their meeting and hitting times statistics) by a set of concurrent dense *random walks.* But the meeting times statistics of the hosts essentially determine the virtual fragile links of any ad-hoc network. We thus believe that our suggestion for adoption of concurrent random walks as a model for input motions, is not only a tool for average case performance analysis but it might in fact approximate well any ad-hoc network of dense motions.

We also note that the theory of random walks provides tools for analytic performance estimation even for extremely fast host motions (e.g. teleportation). We comment more on this in the Section on Leader Election and Counting.

1.2.3 Non-oblivious adversaries and game-theoretic ideas

What happens if we allow the adversary which specifies the motion of some hosts to be hostile to the distributed algorithm? Such a consideration might lead to impossibility results. It might also lead to interesting analysis of the performance of the distributed algorithm, via the exploitation of some game-theoretic ideas.

Such ad-hoc systems often invoke a set of independent *selfish* and *antagonistic agent* processes trying to share a common resource. This situation evokes game theory and its main concept of rational behaviour, the *Nash equilibrium:* in an environment in which each agent is aware of the situation facing all other agents, a Nash equilibrium is a combination of choices (deterministic or randomized), one for each agent, from which no agent has an incentive to unilaterally move away. The ratio between the worst possible Nash equilibrium and the global optimum, called *coordination ratio*, was first defined in [25]. Some upper bounds for this ratio and the structure of worst-case Nash equilibria for a very simple routing problem were given in [28].

We propose here to derive "difficult" behaviours for such antagonistic ad-hoc networks by using such game-theoretic ideas. We can sometimes consider the competition between a distributed algorithm and an adversary as a game of possibly many rounds of moves of the opponents. Worst-case Nash equilibria (with respect to some optimization criteria) may then be examined and we suggest them as interesting behaviours to be tested experimentally.

We motivate the above approach by a very simple problem of pursuit-evasion. Several agents (hosts) moving along neighbor vertices of a graph (network) are looking for a fugitive. The fugitive is eliminated when it coicides with an agent at a vertex. The agents cannot "see" further away from their current location. However, the fugitive can sense the intention of a neighboring agent to come towards it, provided that the agent is not still, and then it should move away (the fugitive has a limited sense of approaching agents).

Simple randomized protocols for catching the fugitive were presented in [36, 37]. In their general structure, these protocols suggest that agents are partitioned into two sets: the *traps*, which stay immobile (hidden therefore) at some random vertices of the graph, and the *searchers*, i.e., agents continuously performing independent random walks. Note that, because of the model, the fugitive cannot sense neighboring traps since they don't move.

In this game, good strategies for the fugitive should allow it to stay at the graph and not be eliminated for as long as possible. Note that the best strategies for the fugitive are non-oblivious adversarial strategies. Fugitive motion around some chosen cycle in the graph is (together with the way agents act) a Nash equilibrium. The fugitive, while not caught, has no benefit in not following the cycle. The game ends almost surely against any non-oblivious adversary only if the traps can re-randomize their locations from time to time [37]. Since long cycles of fugitive's movement have higher probability to encounter a trap, we conclude that short cycles of such a movement are worst-case equilibria in the sense that they extend the game's duration. Note that strategies which force the fugitive to stay at some vertex forever (after some initial motion) are not good, since random walks will hit any of those positions in short expected time.

1.3 Comments on selected previous work

In a recent paper [26], Q.Li and D.Rus present a model which has some similarities to ours. The authors give an interesting, yet different, protocol to send messages, which forces *all the mobile hosts to slightly deviate (for a short period of time) from their predefined, deterministic routes, in order to propagate the messages.* Their protocol is, thus, *compulsory* for any host and it works only for deterministic host routes. Moreover, their protocol considers the propagation of only one message (end to end) each time, in order to be correct. In contrast, our support scheme allows for simoultaneous processing of many communication pairs. In their setting [26] show optimality of message transmission times.

M.Adler and C.Scheideler [1] in a previous work, dealt only with *static* transmission graphs i.e. the situation where the positions of the mobile hosts and the environment do not change. In [1] the authors pointed out that static graphs provide a starting point for the dynamic case. In our work, we consider the *dynamic case* (i.e. mobile hosts move *arbitrarily*) and in this sense we extend their work. As far as performance is concerned, their work provides time bounds for communication that are proportional to the diameter of the graph defined by a random uniform speading of the hosts.

We note here (see also section 1.1) that motion co-ordination ideas for distributed systems of metamorphic robots have been introduced by J.Walter, J.Welch and N.Amato in [39]. In their paper "Distributed Reconfiguration of Metamorphic Robot Chains" the authors study the problem of motion co-ordination in distributed systems consisting of such robots, which can connect, disconnect and move around. The paper deals with metamorphic systems where all modules are identical.

An interesting approach for Leader Electon in ad-hoc mobile networks is presented in the paper of N.Malpani, J.Welch and N.Vaidya [27]. There, the authors present algorithms, based on the TORA [33] routing method, which maintain wireless links and guarantee exactly one leader per connected component of the implied virtual links network among mobile hosts. Their proof of correctness assumes a *bounded rate* of topology changes.

2 A framework for protocol classes

Protocols (distributed algorithms) for ad-hoc mobile networks can be divided into three major categories:

Definition 2 Non-Compulsory *protocols are the ones whose execution does not affect the movement of the mobile hosts. On the other hand,* compulsory *protocols are the those that require all hosts to perform certain moves in order to ensure the correct protocol execution. Finally, the class of ad-hoc mobile networks protocols which enforce a (small) subset of the mobile hosts to move in a certain way is called the class of* semi-compulsory *protocols.*

Note that non-compulsory protocols try to take advantage of the mobile hosts natural movement by exchanging information whenever mobile hosts meet incidentally. Compulsory protocols force the mobile hosts to move according to a specific scheme in order to meet the protocol demands (i.e. meet more often, spread in a geographical area, etc.).

Note also that we can further categorize each class depending on whether or not the mobile hosts have *(individual or common) sense of orientation* in the motion space. We can strengthen this by assuming abilities of *geolocation* given to mobile hosts. Such location information (i.e complete coordinates according to some origin) can be obtained using global positioning system (GPS) [42, 43, 15, 32] facilities.

Definition 3 *The subset of the mobile hosts of an ad-hoc mobile network whose motion is determined by a network protocol P is called the* support Σ *of $\mathcal{P}$. The part of P which indicates the way that members of* Σ move and communicate *is called the* support management subprotocol M_Σ *of $\mathcal{P}$.*

This definition captures network management ideas for ad-hoc mobile networks.

Notice that our scheme defines a support (and its management subprotocol) suitable not only for pairwise communication but also for a whole set of basic problems including many-to-one communication, information spreading and multicasting.

Definition 4 *Consider a family of protocols, $\mathcal{F}$, for a mobile ad-hoc network, and let each protocol $\mathcal{P}$ in $\mathcal{F}$ have the* same *support (and the same support management subprotocol). Then Σ is called the* support of the family $\mathcal{F}$.

Our scheme follows the general design principle of mobile networks (with a fixed subnetwork however) called the "two-tier" principle [23] which says that any protocol should try to move communication and computation to the fixed part of the network. Our idea of the support Σ is a simulation of such a (skeleton) network by moving hosts, however.

In addition, we may wish that the way hosts in Σ move (maybe coordinated) and communicate is robust (i.e. that it can tolerate failures of hosts).

The types of failures of hosts that we consider here are permanent (i.e. stop) failures. Once such a fault happens then the host of the fault does not participate in the ad-hoc mobile network anymore.

Definition 5 *A support management subprotocol, M_Σ, is* k-faults tolerant*, if it still allows the members of $\mathcal{F}$ (or $\mathcal{P}$) to execute correctly, under the presence of at most k permanent faults of hosts in Σ ($k \geq 1$).*

3 Basic Communication Algorithms

3.1 Problem Definition

A *basic communication problem*, in ad-hoc mobile networks, is to send information from some *sender* user, MH_S, to another designated *receiver* user, MH_R.

One way to solve this problem is the protocol of notifying every user that the sender MH_S meets (and providing *all the information to it*) hoping that some of them will eventually meet the receiver MH_R.

Is there a more efficient technique (other than notifying every user that the sender meets, in the hope that some of them will then eventually meet the receiver) that will effectively solve the communication establishment problem without flooding the network and exhausting the battery and computational power of the hosts?

The most common way to establish communication is to form paths of intermediate nodes that lie within one another's transmission range and can directly communicate with each other [8, 19, 24, 33, 35, 40]. Indeed, this approach of exploiting pairwise communication is common in ad-hoc mobile networks that cover a relatively small space (i.e. with diameter which is small with respect to transmission range) or are dense (i.e. thousands of wireless nodes) where all locations are occupied by some hosts; broadcasting can be efficiently accomplished.

In wider area ad-hoc networks with less users, however, broadcasting is impractical: two distant peers will not be reached by any broadcast as users may not occupy all intermediate locations (i.e. the formation of a long path is not feasible). Even if a valid path is established, single link "failures" happening when a small number of users that were part of the communication path move in a way such that they are no longer within transmission range of each other, will make this path invalid. Note also that the path established in this way may be very long, even in the case of connecting nearby hosts.

In contrast to all such methods, we try to avoid ideas based on paths finding and their maintenance. We envision networks with highly dynamic movement of the mobile users, where the idea of "maintenance" of a valid path is inconceivable (paths can become invalid immediately after they have been added to the directory tables). Our approach is to take advantage of the mobile hosts natural movement by exchanging information whenever mobile hosts meet incidentally. It is evident, however, that if the users are spread in remote areas and they do not move beyond these areas, there is no way for information to reach them, unless the protocol takes special care of such situations.

In the light of the above, we propose the idea of forcing only a small subset of the deployed hosts to move as per the needs of the protocol, i.e. we propose to exploit the *support* idea. Assuming the availability of such hosts, we use them to provide a simple, correct and efficient strategy for communication between any pair of hosts in such networks that avoid message flooding.

A protocol solving this important communication problem is *reliable* if it allows the sender to be notified about delivery of the information to the receiver. Note that no distributed computing protocol can be implemented in ad-hoc mobile networks without solving this basic communication problem.

3.2 The basic protocol

In simple terms, the protocol works as follows: The nodes of the support move fast enough so that they cover (in sufficiently short time) the entire motion graph. Their motion is accomplished in a distributed way via a *support motion subprotocol* P_1. When some node of the support is within communication range of a sender, an underlying *sensor subprotocol* P_2 notifies the sender that it may send its message(s).

The messages are then stored "somewhere within the support structure". When a receiver comes within communication range of a node of the support, the receiver is notified that a message is "waiting" for him and the message is then forwarded to the receiver.

The messages received by the support are propagated within the structure when two or more members of the support meet on the same site (or are within communication range). A synchronization subprotocol P_3 is used to dictate the way that the members of the support exchange information.

In a way, the support Σ plays the role of a (moving) skeleton subnetwork (whose structure is defined by the motion subprotocol P_1), through which all communication is routed. From the above description, the size, k, and the shape of the support may affect performance.

Note that the proposed scheme does not require the propagation of messages through hosts that are not part of Σ, thus its security relies on the support's security and is not compromised by the participation in message communication of other mobile users. For a discussion of intrusion detection mechanisms for ad-hoc mobile networks see [40].

3.3 The "Snake" support management subprotocol M^S_Σ

The main idea of the protocol proposed in [9, 11] is as follows. There is a set-up phase of the ad-hoc network, during which a predefined set, k, of hosts, become the nodes of the support. The members of the support perform a leader election by running a randomized symmetry breaking protocol in anonymous networks (see section 4.6.1). This is run once and imposes only an initial communication cost. The elected leader, denoted by MS_0, is used to co-ordinate the support topology and movement. Additionally, the leader assigns local names to the rest of the support members MS_1, MS_2, ..., MS_{k-1}.

The nodes of the support move in a coordinated way, always remaining pairwise adjacent (i.e., forming a list of k nodes), so that they sweep (given some time) the entire motion graph. This encapsulates the *support motion subprotocol* P_1^S.

Essentially the motion subprotocol P_1^S enforces the support to move as a "snake", with the head (the elected leader MS_0) doing a random walk on the motion graph and each of the other nodes MS_i executing the simple protocol "move where MS_{i-1} was before". More formaly, the movement of Σ is then defined as follows:

Initially, MS_i, $\forall i \in \{0, 1, \ldots, k-1\}$, start from the same area-node of the motion graph. The direction of movement of the leader MS_0 is given by a memoryless operation that chooses *randomly the direction* of the next move. Before leaving the current area-node, MS_0 sends a message to MS_1 that states the new direction of movement. MS_1 will change its direction as per instructions of MS_0 and will propagate the message to MS_2. In analogy, MS_i will follow the orders of MS_{i-1} after transmitting the new directions to MS_{i+1}. Movement orders received by MS_i are positioned in a queue Q_i for sequential processing. The very first move of MS_i, $\forall i \in \{1, 2, \ldots, k-1\}$ is delayed by δ period of time.

We assume that the mobile support hosts move with a common speed. Note that the above described motion subprotocol P_1^S enforces the support to move as a "snake", with the head (the elected leader MS_0) *doing a random walk on the motion graph* G and each of the other nodes MS_i executing the simple protocol "move where MS_{i-1} was before". This can be easily implemented because MS_i will move following the edge from which it received the message from MS_{i-1} and therefore our protocol does not require common sense of orientation.

The purpose of the random walk of the head is to ensure a *cover* (within some finite time) of the whole motion graph, without memory (other than local) of topology details. Note that this memoryless motion also ensures fairness. The value of the *random walk principle* of the motion of the support will be further justified in the correctness and the efficiency parts of the paper, where we wish our communication establishment protocol to meet some performance requirements *regardless of the motion of the hosts not in* Σ.

A modification of M_Σ^S is that the head does a random walk on *a spanning subgraph* of G (e.g. a spanning tree). This modified M_Σ (call it M'_Σ) is more efficient in our setting since "edges" of G just represent adjacent locations and "nodes" are really possible host places.

A simple approach to implement the support *synchronization subprotocol* P_3^S is to use a forward mechanism that transmit incoming messages (in-transit) to neighboring members of the support (due to P_1^S at least one support host will be close enough to communicate). In this way, all incoming messages are eventually copied and stored in every node of the support. This is not the most efficient storage scheme and can be refined in various ways in order to reduce memory requirements.

3.4 The "Runners" support management subprotocol M_Σ^R

A different approach to implement M_Σ is to allow each member of Σ not to move in a snake-like fashion, but to perform an *independent* random walk on the motion graph G, i.e., the members of Σ can be viewed as "runners" running on G. In other words, instead of maintaining at all times pairwise adjacency between members of Σ, all hosts sweep the area by moving independently from each other. When two runners meet, they exchange any information given to them by senders encountered using a new synchronization subprotocol P_3^R. As in the snake case, the same underlying sensor sub-protocol P_2 is used to notify the sender that it may send its message(s) when within communication range of a node of the support.

As presented in [12], the runners protocol does not use the idea of a (moving) backbone subnetwork as no motion subprotocol P_1^R is used. However, all communication is still routed through the support Σ and we expect that the size k of the support (i.e., the number of runners) will affect performance in a more efficient way than that of the snake approach. This expectation stems from the fact that each host will meet each other in parallel, accelerating the spread of information (i.e., messages to be delivered).

A member of the support needs to store all undelivered messages, and maintain a list of receipts to be given to the originating senders. For simplicity, we can assume a generic storage scheme where all undelivered messages are members of a set S_1 and the list of receipts is stored on another set S_2. In fact, the unique ID of a message and its sender ID is all that is needed to be stored in S_2.

When two runners meet at the same site of the motion graph G, the synchronization subprotocol P_3^R is activated. The subprotocol imposes that when runners meet on the same site, their sets S_1 and S_2 are synchronized. In this way, a message delivered by some runner will be removed from the set S_1 of the rest of runners encountered, and similarly delivery receipts already given will be discarded from the set S_2 of the rest of runners. The synchronization subprotocol P_3^R is partially based on the *two-phase commit* algorithm as presented in [29] and works as follows.

> Let the members of Σ residing on the same site (i.e., vertex) u of G be $MS_1^u, \ldots, MS_j^u$. Let also $S_1(i)$ (resp. $S_2(i)$) denote the S_1 (resp. S_2) set of runner MS_i^u, $1 \leq i \leq j$. The algorithm assumes that the underlying sensor sub-protocol P_2 informs all hosts about the runner with the lowest ID, i.e., the runner MS_1^u. P_3^R consists of two rounds.
>
> *Round* 1*:* All $MS_1^u, \ldots, MS_j^u$ residing on vertex u of G, send their S_1 and S_2 to runner MS_1^u. Runner MS_1^u collects all the sets and combines them with its own to compute its new sets S_1 and S_2: $S_2(1) = \bigcup_{1\leq l\leq j} S_2(l)$ and $S_1(1) = \bigcup_{1\leq l\leq j} S_1(l) - S_2(1)$.
>
> *Round* 2*:* Runner MS_1^u broadcasts its decision to all the other support member hosts. All hosts that received the broadcast apply the same rules, as MS_1^u did, to join their S_1 and S_2 with the values received. Any host that receives a message at Round 2 and which has not participated in Round 1, accepts the value received in that message as if it had participated in Round 1.

This simple algorithm guarantees that mobile hosts which remain connected (i.e., are able to exchange messages) for two continuous rounds, will manage to synchronize their S_1 and S_2. Furthermore, on the event that a new host arrives or another disconnects during Round 2, the execution of the protocol will not be affected. In the case where runner MS_1^u fails to broadcast in Round 2 (either because of an internal failure or because it has left the site), then the protocol is simply re-executed among the remaining runners.

Remark that the algorithm described above does not offer a mechanism to remove message receipts from S_2; eventually the memory of the hosts will be exhausted. A simple approach to solve this problem and effectively reduce the memory usage is to construct an order list of IDs contained in S_2, for each receiver. This ordered sequence of IDs will have gaps - some messages will still have to be confirmed, and thus not part of S_2. In this list, we can identify the maximum ID before the first gap. We are now able to remove from S_2 all message receipts with smaller ID than this identified ID.

3.5 The Hierarchical Approach

In this section we introduce a *new model* of ad-hoc mobile networks, which we call *hierarchical* [10] that are comprised of dense subnetworks of mobile users interconnected across access ports by sparse but fast connections.

The *lower level* of the hierarchy may model dense ad-hoc subnetworks of mobile users that are unstructured and where there is no fixed infrastructure. To implement communication in such a case, a possible solution would be to install a very fast (yet limited) *backbone* interconnecting such highly populated mobile user areas, while using the support approach in the lower levels. We assume that this fast backbone provides a limited number of *access ports* within these dense areas of mobile users. We call this fast backbone the *higher level* of the hierarchy.

Because of the presence of the fixed infrastructure of the higher layer of the hierarchy, we assume that in each specific moment in time the availability of the higher level at a specific access port is modeled by the probability p.

Definition 6 *Let p be the probability that at any given time instance the exchange of information by the higher layer of the hierarchy is available through an access port.*

We remark that, in practice, this probability may differ between the various access ports and also vary with time. So we take, for analysis reasons, p to be a lower bound on these probabilities.

For simplicity, let's assume two such dense ad-hoc subnetworks of mobile users (A and B) that are interconnected via a very fast backbone that provides one access port to each subnetwork. Assume further that the sender MH_S is within site A and the designated receiver MH_R is located at site B. In such hierarchical case communication between users in different dense areas takes place in the following way:

a) When some mobile host of the support in the lower level A is within the communication range of the sender MH_S, the underlying sensor subprotocol P_2 notifies the sender to give its messages to the support.

b) When a node of the support arrives at the access port of A to the *higher level*, it transmits pending messages to the higher level with probability p.

c) The higher level, after having got the messages from the access port in A, propagates the messages (according to its own network management scheme) to the access port in B, where again at some time they will be delivered to a member of the other lower level support (i.e. the support of site B) within transmition range of the access port and with probability p.

d) Having received the messages from the access port B, the support forwards them to the receiver MH_R when they first meet within B.

We note that this hierarchical approach for a management subprotocol is inherently *modular* in the sense that it actually assumes, for shaping the hierarchy of supports as a whole, a basic building block: a dense area of mobile users and its access ports to some very fast fixed backbone. Thus, the changes incurred by adopting the protocol to any number of subnetworks connected across various access ports by a sparse network of very fast interconnections, are basically quantitative and easy to analyse, and do not affect the correctness and the essence of this approach.

If MH_R is allowed to change locations (i.e. moves from site A to site B within a reasonable period of time) the aforementioned scheme might fail to deliver the messages unless the protocol is extended. In reality, this problem is similar to that of mobile networks with fixed backbone, i.e. when a user enters a new cell. Instead of providing a subprotocol to determine and monitor the location of each mobile user (and complicate further our scheme), the hierarchical version of our algorithms will assume that a recipient of a message can be located anywhere within the ad-hoc network. Therefore, each mobile support propagates messages to all other lower levels creating in such a way a number of multiple copies equal to the number of dense subnetworks that make up the hierarchical ad-hoc network. These copies will be stored within the higher level for a given period of time, sufficient to meet the recipient host if it lies within the area-borders of the "biggest" lower level. This period can be set to be analogous to the cover time of each lower level's motion graph.

It is clear that the total size of interconnecting higher level, the number of access ports scattered through the lower levels of the hierarchy, and the interconnection topology of the lower levels will have an affect on the overall performance of the network.

Based on the above, it is evident that modifications only need to be made to the *synchronization subprotocol* P_3 for the support management subprotocols to adopt to the imposed hierarchy. In the following sections, we refer to the Hierarchical Support Routing Protocol as HSRP.

3.6 Analytical Considerations for the "Snake" Protocol

3.6.1 Preliminaries

In the sequel, we assume that the head of the snake does a *continuous time random walk* on $G(V, E)$, without loss of generality (if it is a discrete time random walk, all results will transfer easily, by [4]). We define the random walk of a host on G that induces a continuous time Markov chain M_G as follows: The states of M_G are the vertices of G and they are finite. Let s_t denote the state of M_G at time t. Given that $s_t = u$, $u \in V$, the probability that $s_{t+dt} = v$, $v \in V$, is $p(u,v) \cdot dt$ where

$$p(u,v) = \begin{cases} \frac{1}{d(u)} & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$

and $d(u)$ is the degree of vertex u.

We assume that all random walks are *concurrent* and that there is a global time t, not necessarily known to the hosts.

Note 1 *Since the motion graph G is finite and connected, the continuous markov chain abstracting the random walk on it is automatically time-reversible.*

Definition 7 *$P_i(E)$ is the probability that the walk satisfies an event E given it started at vertex i.*

Definition 8 *For a vertex j, let $T_j = min\{t \geq 0 : s_t = j\}$ be the first hitting time of the walk onto that vertex and let $E_i T_j$ be its expected value, given that the walk started at vertex i of G.*

Definition 9 *For the walk of any particular host, let $\pi()$ be the stationary distribution of its position after a sufficiently long time.*

We denote $E_\mu[\,]$ the expectation for the chain started at time 0 from any vertex with distribution μ (e.g. the initial distribution of the Markov chain).

We know (see [4]) that for every vertex σ, $\pi(\sigma) = \frac{d(\sigma)}{2m}$ where $d(\sigma)$ is the degree of σ in G and $m = |E|$.

Definition 10 *Let $p_{j,k}$ be the transition probability of the random walk from vertex j to vertex k. Let $p_{j,k}(t)$ be the probability that the random walk started at j will be at $k \in V$ in time t.*

Definition 11 *Let $X(t)$ be the position of the random walk at time t*

3.6.2 Correctness guarantees under the restricted motion adversary

Let us now consider the case where any sender or receiver is allowed a general, unknown motion strategy, but its strategy is provided by a restricted motion adversary.

This means that each host not in the support either executes a deterministic motion (which either stops at a node or cycles forever after some initial part) or it executes a stochastic strategy which however is *independent* of the motion of the support.

Theorem 1 *The support Σ and the management subprotocol M_Σ guarantee reliable communication between any sender-receiver (MH_S, MH_R) pair in finite time, whose expected value is bounded only by a function of the relative motion space size ρ and does not depend on the number of hosts, and is also independent of how MH_S, MH_R move, provided that the mobile hosts not in the support do not deliberately try to avoid the support.*

Proof: For the proof purposes, it is enough to show that the a node of Σ will meet MH_S and MH_R infinitely often, with probability 1 (in fact our argument is a consequence of the Borel-Cantelli Lemmas for infinite sequences of trials). We will furthermore show that the first meeting time M (with MH_S or MH_R) has an expected value (where expectation is taken over the walk of Σ and any strategy of MH_S (or MH_R) and any starting position of MH_S (or MH_R) and Σ) which is bounded by a function of the size of the motion graph G only. This then shows the Theorem since it shows that MH_S (and MH_R) meet with the head of Σ infinitely often, each time within a bounded expected duration.

So, let EM be the expected time of the (first) meeting and $m^* = supEM$, where the supremum is taken over all starting positions of both Σ and MH_S (or MH_R) and all strategies of MH_S (one can repeat the argument with MH_R).

We proceed to show that we can construct for the walk of Σ's head a *strong stationary time sequence* V_i such that for all $\sigma \in V$ and for all times t

$$P_i(X(V_i) = \sigma \mid V_i = t) = \pi(\sigma)$$

Remark that strong stationary times were introduced by Aldous and Diaconis in [3].

Notice that at times V_i, MH_S (or MH_R) will necessarily be at some vertex σ of V, either still moving or stopped.

Let u be a time such that for X,

$$p_{j,k}(u) \geq \left(1 - \frac{1}{e}\right)\pi(k)$$

for all j, k. Such a u always exists because $p_{j,k}(t)$ converges to $\pi(k)$ from basic Markov Chain Theory.

Note that u depends only on the structure of the walk's graph, G. In fact, if one defines separation from stationarity to be

$$s(t) = max_j s_j(t)$$

where

$$s_j(t) = sup\{s : p_{ij}(t) \geq (1 - s)\pi(j)\}$$

then

$$\tau_1^{(1)} = min\{t : s(t) \leq e^{-1}\}$$

is called the *separation threshold time*. For general graphs G of n vertices this quantity is known to be $\mathcal{O}(n^3)$ ([7]).

Now consider a sequence of stopping times $U_i \in \{u, 2u, 3u, \dots\}$ such that

$$P_i(X(U_i) = \sigma \mid U_i = u) = \left(1 - \frac{1}{e}\right)\pi(\sigma) \quad (1)$$

for any $\sigma \in V$. By induction on $\lambda \geq 1$ then

$$P_i\big(X(U_i) = \sigma \mid U_i = \lambda u\big) = e^{-(\lambda-1)}\left(1 - \frac{1}{e}\right)\pi(\sigma)$$

This is because of the following: First remark that for $\lambda = 1$ we get the definition of U_i. Assume that the relation holds for $(\lambda - 1)$ i.e.

$$P_i\big(X(U_i) = \sigma \mid U_i = (\lambda-1)u\big) = e^{-(\lambda-2)}\left(1 - \frac{1}{e}\right)\pi(\sigma)$$

for any $\sigma \in V$. Then $\forall \sigma \in V$

$$\begin{aligned}
P_i & \big(X(U_i) = \sigma \mid U_i = \lambda u\big) = \\
= & \sum_{\alpha \in V} P_i\big(X(U_i) = \alpha \mid U_i = (\lambda-1)u\big) \cdot P_{\alpha,\sigma}(u) \\
= & e^{-(\lambda-2)}\left(1 - \frac{1}{e}\right) \sum_{\alpha \in V} \pi(\alpha)\frac{1}{e}\pi(\sigma) \quad \text{from (1)} \\
= & e^{-(\lambda-1)}\left(1 - \frac{1}{e}\right)\pi(\sigma) \sum_{\alpha \in V} \pi(\alpha) \quad \text{from definition of } u \\
= & e^{-(\lambda-1)}\left(1 - \frac{1}{e}\right)\pi(\sigma)
\end{aligned}$$

which ends the induction step. Then, for all σ

$$P_i\big(X(U_i) = \sigma\big) = \pi(\sigma) \quad (2)$$

and

$$E_i U_i = u\left(1 - \frac{1}{e}\right)^{-1}$$

Now let $c = u\frac{e}{e-1}$. So, we have constructed (by (2)) a *strong stationary time sequence* U_i with $EU_i = c$. Consider the sequence $0 = U_0 < U_1 < U_2 < \ldots$ such that for $i \geq 0$

$$E\big(U_{i+1} - U_i \ \mid\ U_j,\ \ j \leq i\big) \leq c$$

But, from our construction, the positions $X(U_i)$ are *independent* (of the distribution $\pi()$), and, in particular, $X(U_i)$ *are independent* of U_i.

Therefore, regardless of the strategy of MH_S (or MH_R) and because of the independence assumption, the support's head has chance at least $\min_\sigma \pi(\sigma)$ to meet MH_S (or MH_R) at time U_i, independently as i varies.

So, the meeting time M satisfies $M \leq U_T$ where T is a stopping time with mean

$$ET \leq \frac{1}{min_\sigma \pi(\sigma)}$$

Note that the idea of a stopping time T such that $X(T)$ has distribution π and is independent of the starting position is central to the standard modern Theory of Harris - recurrent Markov Chains (see e.g. [16]).

From Wald's inequality [4] then

$$EU_T \ \leq\ c \cdot ET$$

$$\Rightarrow \quad m^* \ \leq\ c\,\frac{1}{\min_\sigma \pi(\sigma)}$$

Note that since G is produced as a subgraph of a regular graph of fixed degree Δ we have

$$\frac{1}{2m}\ \leq\ \pi(\sigma)\ \leq\ \frac{1}{n}$$

for all σ ($n = |V|$, $m = |E|$), thus

$$ET\ \leq\ 2m$$

hence

$$m^* \leq 2mc\ =\ \frac{e}{e-1}2mu$$

Since m, u only depend on G, this proves the Theorem. □

Corollary 1 *If Σ's head walks randomly in a* regular *spanning subgraph of G, then $m^* \leq 2cn$.*

3.6.3 Time efficiency for the (worst) case of a restricted motion adversary

Clearly, one intuitively expects that if $k = |\Sigma|$ then the higher k is (with respect to n), the best the performance of Σ gets.

By working as in the construction of the proof of Theorem 1, we can create a sequence of strong stationary times U_i such that $X(U_i) \in F$ where F={σ: σ is a position of a host in the support}. Then $\pi(\sigma)$ is replaced by $\pi(F)$ which is just $\pi(F) = \sum \pi(\sigma)$ over all $\sigma \in F$. So now m^* is bounded as follows:

$$m^*\ \leq\ c\,\frac{1}{\min_{\sigma\in J}\left(\sum \pi(\sigma)\right)}$$

where J is any induced subgraph of the graph of the walk of Σ's head such that J is the neighbourhood of a vertex σ of radius (maximum distance simple path) at most k. The quantity

$$\min_J\left(\sum_{\sigma\in J}\pi(\sigma)\right)$$

is then at least $\frac{k}{2m}$ and, hence, $m^*\ \leq\ c\,\frac{2m}{k}$.

The overall communication is given by:

$$T_{total}\ =\ X\ +\ T_\Sigma\ +\ Y \tag{3}$$

where:

- X is the time for the sender node to reach a node of the support.
- T_Σ is the time for the message to propagate inside the support. Clearly, $T_\Sigma = \mathcal{O}(k)$, i.e. linear in the support size.
- Y is the time for the receiver to meet with a support node, after the propagation of the message inside the support.

We have for all MH_S, MH_R:

$$E(T_{total})\ \leq\ \frac{2mc}{k}\ +\ \Theta(k)\ +\ \frac{2mc}{k}$$

(since $Z = \Theta(k)$)

The upper bound achieves a minimum when $k = \sqrt{2mc}$.

Lemma 1 *For the walk of Σ's head on the entire motion graph G, the communication establishment time's expected time is bounded above by $\Theta(\sqrt{mc})$ when the (optimal) support size $|\Sigma|$ is $\sqrt{2mc}$ and c is $\frac{e}{e-1}u$, u being the "separation threshold time" of the random walk on G.*

Remark that other authors use for u by the symbol $\tau_1^{(1)}$ for a symmetric Markov Chain of continuous time on n states.

3.6.4 Tighter Bounds for the worst case of motion

To make our protocol more efficient, we now force the head of Σ to perform a random walk on a *regular spanning graph* of G. Let $G_R(V, E')$ be such a subgraph.

Our improved protocol versions assume that (a) such a subgraph exists in G and (b) is given in the beginning to all the stations of the support.

Then, for any $\sigma \in V$, and for this new walk X', we have for the steady state probabilities

$\pi_{X'}(\sigma)\ =\ \frac{1}{n}$ for all σ.

Let now M be again the first meeting time of MH_S (or MH_R) and Σ's head. By the stationarity of X'

$$\int_0^t P\big(S, X' \text{ are together at time s}\big)\ ds\ =\ \frac{t}{n} \tag{4}$$

Now let

$$p^*(t)\ =\ \max_{x,v} p_{x,v}(t)$$

Regardless of initial positions, the chance that R,Σ's head are together (i.e. at the same vertex) at time u is *at most* $p^*(u)$. So, by assuming first that Σ's head starts with the stationary distribution, we get

$$P\big(\text{meet at time s}\big) = \int_0^s f(u) P\big(\text{meet at time s} \,|\, M = u\big) du$$

where $f(u)$ is the (unknown) density function of the meeting time M. So

$$P\big(\text{meet at time s}\big) \;\leq\; \int_0^s f(u) p^*(s-u)\; du$$

But we know [4] that

Lemma 2 *[4] There exists an absolute constant* K *such that*

$$p^*(t) \;\leq\; \frac{1}{n} \;+\; Kt^{-\frac{1}{2}} \quad \textit{where } 0 \leq t < \infty$$

Thus

$$P\big(\text{meet at time s}\big) \leq \frac{1}{n} P(M \leq s) + K \int_0^s f(u)(s-u)^{-\frac{1}{2}} du$$

So, from (4), we get

$$\begin{aligned}
\frac{t}{n} \;&\leq\; \frac{1}{n}\int_0^t P(M \leq s) ds \;+\; K\int_0^t f(u) du \int_0^t (s-u)^{-\frac{1}{2}} ds \\
&=\; \frac{t}{n} \;-\; \frac{1}{n}\int_0^t P(M > s) ds \;+\; 2K\int_0^t f(u)(t-u)^{-\frac{1}{2}} du \\
&\leq\; \frac{t}{n} - \frac{1}{n} Emin(M,t) \;+\; 2Kt^{-\frac{1}{2}}
\end{aligned}$$

So, the expected of the minimum of M and t is

$$Emin(M,t) \;\leq\; 2Knt^{-\frac{1}{2}}$$

Taking $t_0 = \big(4Kn\big)^2$, from Markov's inequality we get $P\big(M \leq t_0\big) \;\geq\; \frac{1}{2}$.

When Σ starts at some arbitrary vertex, we can use the notion of *separation time* $s(u)$ (a time to approach stationarity by at least a fraction of $\big(1 \;-\; \frac{1}{e}\big)^{-1}$ to get

$$P\big(M \leq u + t_0\big) \;\geq\; \frac{1 \;-\; s(u)}{2}$$

i.e., by iteration,

$$EM \;\leq\; \frac{2(u \;+\; t_0)}{1 \;-\; s(u)}$$

where u is as in the construction of Theorem 1. Thus,

$$m^* \leq \frac{2}{\big(1 \;-\; \frac{1}{e}\big)}(t_0 + u)$$

but for regular graphs it is known (see e.g. [4]) that $u = \mathcal{O}\big(n^2\big)$ implying (together with our remarks about Σ's length and set of positions) the following theorem:

Theorem 2 *By having* Σ*'s head to move on a regular spanning subgraph of* G*, there is an absolute constant* $\gamma > 0$ *such that the expected meeting time of* MH_S *(or* MH_R*) and* Σ *is bounded above by* $\gamma\,\frac{n^2}{k}$.

Again, the total expected communication establishment time is bounded above by $2\gamma\frac{n^2}{k} \;+\; \Theta\big(k\big)$ and by choosing $k = \sqrt{2\gamma n^2}$ we can get a best bound of $\Theta\big(n\big)$ for a support size of $\Theta\big(n\big)$.

Recall that $n = \mathcal{O}\left(\frac{V(\mathcal{S})}{V(tr)}\right)$ i.e. n is linear to the ratio ρ of the volumes of the space of motions and the transmission range of each mobile host. Thus,

Corollary 2 *By forcing the support's head to move on a regular spanning subgraph of the motion graph, our protocol guarantees a total expected communication time of* $\Theta\big(\rho\big)$*, where* ρ *is the relative motion space size, and this time is independent of the total number of mobile hosts, and their movement.*

Note 2 *Our analysis assumed that the head of* Σ *moves according to a continuous time random walk of total rate 1 (rate of exit out of a node of* G*). If we select the support's hosts to be* ψ times faster *than the rest of the hosts, all the estimated times, except of the inter-support time, will be divided by* ψ*. Thus*

Corollary 3 *Our modified protocol where the support is* ψ *times faster than the rest of the mobile hosts guarantees an expected total communication time which can be made to be as small as* $\Theta\big(\gamma\frac{\rho}{\sqrt{\psi}}\big)$ *where* γ *is an absolute constant.*

3.6.5 Time efficiency - A Lower Bound

Lemma 3

$$m^* \;\geq\; \max_{i,j} E_i T_j$$

Proof: Consider the case where MH_S (or MH_R) just stands still on some vertex j and Σ's head starts at i. □

Corollary 4 *When* Σ *starts at positions according to the stationary distribution of its head's walk then*

$$m^* \;\geq\; \max_j E_\pi T_j$$

for $j \in V$*, where* π *is the stationary distribution.*

From a Lemma of ([4], ch. 4, pp. 21), we know that for all i

$$E_\pi T_i \;\geq\; \frac{(1 \;-\; \pi(i))^2}{q_i \pi(i)}$$

where $q_i = d_i$ is the degree of i in G i.e.,

$$E_\pi T_i \;\geq\; \min_i \frac{\big(1 \;-\; \frac{d_i}{2m}\big)^2}{d_i \frac{d_i}{2m}} \;\geq\; \min_i \frac{1}{2m}\, \frac{(2m \;-\; d_i)^2}{d_i^2}$$

For regular spanning subgraphs of G of degree Δ we have $m = \frac{\Delta n}{2}$, where $d_i = \Delta$ for all i. Thus,

Theorem 3 *When Σ's head moves on a regular spanning subgraph of G, of m edges, we have that the expected meeting time of MH_S (or MH_R) and Σ cannot be less than* $\frac{(n-1)^2}{2m}$.

Corollary 5 *Since $m = \Theta\big(n\big)$ we get a $\Theta\big(n\big)$* lower bound *for the expected communication time. In that sense, our protocol's expected communication time is* optimal *when the support size is $\Theta\big(n\big)$.*

3.6.6 Time efficiency on the average

Time-efficiency of semi-compulsory protocols for ad-hoc networks is not possible to estimate without a scenario for the motion of the mobile users not in the support (i.e. the non-compulsory part). In a way similar to [9, 20], we propose an "on-the-average" analysis by assuming that the movement of each mobile user *is a random walk on the corresponding motion graph G*. We propose this kind of analysis as a necessary and interesting first step in the analysis of efficiency of any semi-compulsory or even non-compulsory protocol for ad-hoc mobile networks. In fact, the assumption that the mobile users are moving randomly (according to uniformly distributed changes in their directions and velocities, or according to the random waypoint mobility model, by picking random destinations) has been used in [22], [18].

In the light of the above, we assume that any host, not belonging in the support, conducts a random walk on the motion graph, independently of the other hosts.

The communication time of M^S_Σ is the total time needed for a message to arrive from a sending node u to a receiving node v. Since the communication time, T_{total}, between MH_S, MH_R is bounded above by $X + Y + T_\Sigma$, where X is the time for MH_S to meet Σ, Y is the time for MH_R to meet Σ (after X) and T_Σ is the message propagation time in Σ, in order to estimate the expected values of X and Y (they are random variables), we work as follows:

(a) Note first that X, Y are, statistically, of the same distribution, under the assumption that u, v are randomly located (at the start) in G. Thus $E(X) = E(Y)$.

(b) We now replace the meeting time of u and Σ by a hitting time, using the following thought experiment:
(b1) We fix the support Σ in an "average" place inside G.
(b2) We then collapse Σ to a single node (by collapsing its nodes to one but keeping the incident edges). Let H be the resulting graph, σ the resulting node and $d(\sigma)$ its degree.
(b3) We then estimate the hitting time of u to σ assuming u is somewhere in G, according to the *stationary distribution*, $\vec{\pi}$, of its walk, on H. We denote the expected value of this hitting time by $E_\pi T^H_\sigma$.

Thus, now $E(T_{total}) = 2E_\pi T^H_\sigma + \mathcal{O}(k)$.

Note 3 *The equation above amortises over meeting times of senders (or receivers) because it uses the stationary distribution of their walks for their position when they decide to send a message.*

Now, proceeding as in [4], we get the following lemma.

Lemma 4 *For any node σ of any graph H in a continuous-time random walk*

$$E_\pi T^H_\sigma \;\le\; \frac{\tau_2\big(1-\pi(\sigma)\big)}{\pi(\sigma)}$$

where $\pi(\sigma)$ is the (stationary) probability of the walk at node (state) σ and τ_2 is the relaxation time of the walk.

Proof: Let $Z_{i,j} = \sum_{t=o}^{\infty}\big(P_{i,j}(t) - \pi(j)\big)\,dt$, be the ij^{th} element of the *recurrent potential* $Z{=}\{Z_{i,j}\}$ of the random walk on H. Here,

$$P^{(t)}_{i,j} = \Pr\{walk\ at\ vertex\ j\ on\ time\ t\ given\ it\ starts\ at\ vertex\ i\}$$

A corollary of renewal identities relates the mean hitting time $E_\pi T_i$ to Z_{ii} (see [4], Chapter 3) as follows: For all vertices i,

$$\pi(i)E_\pi T_i \;=\; Z_{ii}$$

hence,

$$\pi(i)E_\pi T_i \;=\; \sum_0^\infty \big(P_{ii}(t) \;-\; \pi(i)\big)\,dt$$

Now, $f(t) = P_{ii}(t) - \pi(i)$ is a *completely monotone* function, because, by the spectral representation of the transition matrix, it follows that

$$P^{(t)}_{ii} \;-\; \pi(i) \;=\; \sum_{m\ge 2} u^2_{im}\,exp(-\lambda_m t)$$

where u is the orthonormal matrix of the spectral theorem for Markov chains. But then, λ_2 controls the behaviour of $f(t)$ as $t \to \infty$ in the sense

$$f(t) \;\le\; f(0)\,exp(-\lambda_2 t)$$

But $f(0) = 1 - \pi(i)$. Then we get for all vertices i,

$$\pi(i)E_\pi T_i \;\le\; \big(1-\pi(i)\big)\sum_0^\infty exp(-\lambda_2 t)\,dt$$

i.e. (for $i = \sigma$)

$$E_\pi T_\sigma \;\le\; \frac{1}{\lambda_2}\,\frac{1\;-\;\pi(\sigma)}{\pi(\sigma)}$$

□

Note 4 *In the above bound, $\tau_2{=}\frac{1}{\lambda_2}$ where λ_2 is the second eigenvalue of the (symmetric) matrix $S = \{s_{i,j}\}$ where $s_{i,j} = \sqrt{\pi(i)}\,p_{i,j}\,(\sqrt{\pi(i)})^{-1}$ and $P = \{p_{i,j}\}$ is the transition matrix of the walk. Since S is symmetric, it is diagonalizable and the spectral theorem gives the following representation: $S = U\Lambda U^{-1}$ where U is orthonormal and Λ is a diagonal real matrix of diagonal entries $0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_{n'}$, $n' = n - k + 1$ (where $n' = |V_H|$). These λ's are the eigenvalues of both P and S. In fact, λ_2 is an indication of the* expansion *of H and of the asymptotic rate of convergence to the stationary distribution, while relaxation time τ_2 is the corresponding interpretation measuring time.*

It is a well-known fact (see e.g. [31]) that $\forall v \in V_H$, $\pi(v) = \frac{d(v)}{2m'}$ where $m' = |E_H|$ is the number of the edges of H and $d(v)$ is the degree of v in H. Thus $\pi(\sigma) = \frac{d(\sigma)}{2m'}$.

By estimating d(σ) and m' and remarking that the operation of locally collapsing a graph does not reduce its expansion capability and hence $\lambda^H_2 \;\ge\; \lambda^G_2$.

Theorem 4

$$E(X) \; = \; E(Y) \; \leq \; \frac{1}{\lambda_2(G)}\,\Theta\left(\frac{n}{k}\right)$$

Theorem 5 *The expected communication time of our scheme is bounded above by the formula*

$$E(T_{total}) \; \leq \; \frac{2}{\lambda_2(G)}\,\Theta\left(\frac{n}{k}\right) \; + \; \Theta(k)$$

Note 5 *The above upper bound is minimised when* $k = \sqrt{\frac{2n}{\lambda_2(G)}}$, *a fact also verified by our experiments (see [9] and also section 3.7 in this paper).*

We remark that this upper bound is tight in the sense that by [4], Chapter 3,

$$E_\pi T^H_\sigma \; \geq \; \frac{\big(1-\pi(\sigma)\big)^2}{q_\sigma \pi(\sigma)}$$

where $q_\sigma = \sum_{j \neq \sigma} p_{\sigma j}$ in H_j is the total exit rate from σ in H. By using martingale type arguments we can show sharp concentration around the mean degree of σ.

3.6.7 The Average Time Efficiency of the Hierarchical Approach

The time needed for two mobile users in different sites of the lower levels of the hierarchy to communicate is:

$$T_{total} = X + X_\Sigma + X_{AP} + T_{backbone} + Y_{AP} + Y_\Sigma + Y$$

where X, Y represent the times (which are random variables) needed for a mobile user to meet its local support, respectively, X_Σ, Y_Σ are the times for the messages to propagate within each local support, respectively. X_{AP} and Y_{AP} are random variables representing the times needed for the randomly moving support's head of each site to deliver (respectively, receive) the messages to (respectively, from) the corresponding access port. $T_{backbone}$ is the time needed for the higher layer to propagate the messages from one access port to the other.

Now, as before, by the analysis of the average case for the basic protocol,

$$E(X) \; = \; E(Y) \; \leq \; \frac{1}{\lambda_2(G)}\,\Theta\left(\frac{n}{k}\right)$$

where n, k, $\lambda_2(G)$ are, respectively, the number of vertices of the motion graph of each lower level site, the support size and the second eigenvalue of the adjacency matrix of the motion graph of each site.

Note 6 *To simplify the analysis the lower level sites are identical. The extension of the results to the general case is straightforward.*

Note 7 *The above upper bound is minimised when* $k = \sqrt{\frac{2n}{\lambda_2(G)}}$, *a fact also verified by our experiments (see [10] and also section 3.7 in this paper).*

This analysis indicates the important fact that *only a small sized support Σ is needed in each lower level site to achieve very efficient times for reaching the support.* This size (actually of order equal to the square root of the number of vertices in the local motion graph) is also verified experimentally.

We now proceed with the analysis of the X_{AP} and Y_{AP} times. Remark that, X_{AP} and Y_{AP} have statistically the same distribution, thus their expected value, because of Lemma 4 and the fact that $\pi(AP) = \frac{d_{AP}}{2m}$, is given by:

$$E(X_{AP}) \; = \; E(Y_{AP}) \; = \; \frac{1}{p}\,\frac{1}{\lambda_2(G)}\,\Theta\left(\frac{2m}{d_{AP}}\right)$$

(where m is the number of edges in the motion graph and d_{AP} is the degree of the access port vertex), since the expected number of visits of the support's head to an access port until a successful availability of the higher layer is geometrically distributed with success probability p, and a visit of the support to the access port can be in fact viewed as a hitting time of a mobile user starting from a random point to the access port (since the support's head performs a random walk).

Now, we may naturally assume that the degree d_{AP} of the vertex corresponding to the access port is (because of its critical with respect to connectivity position in the network) at least $\overline{d}$, where $\overline{d} = \frac{2m}{n}$ is the average degree in the graph. Thus, we get

$$E(X_{AP}) \; = \; E(Y_{AP}) \; = \; \frac{1}{p}\,\frac{1}{\lambda_2(G)}\,\Theta\big(n\big)$$

Finally $T_{backbone}$ is a function, as we have already said, of the highway traffic parameters and we consider this to be a given parameter of the protocol having constant size.

Thus, we finally get:

$$E(T_{total}) \; = \; \frac{2}{\lambda_2(G)}\left(\Theta\left(\frac{n}{k}\right) \; + \; \frac{1}{p}\Theta\left(n\right)\right) \; + \; 2\Theta\big(k\big)$$

which gives a linear average message delay $E(T_{total}) = O\big(n\big)$, where n is the number of vertices of the motion graph of each site.

3.6.8 Robustness

Now, we examine the robustness of the support management subprotocol M^S_Σ and M^R_Σ under single stop-faults.

Theorem 6 *The support management subprotocol M^S_Σ is 1-fault tolerant.*

Proof: If a single host of Σ fails, then the next host in the support becomes the head of the rest of the snake'. We thus have two, independent, random walks in G (of the two snakes) which, however, will meet in expected time at most m^* (as in Theorem 1) and re-organize as a single snake via a very simple re-organization protocol which is the following:

When the head of the second snake Σ_2 meets a host h of the first snake Σ_1 then the head of Σ_2 follows the host which is "in front" of h in Σ_1, and all the part of Σ_1 after and including h waits, to follow Σ_2's tail. □

Note that in the case when more than one faults occur, the procedure for merging "snakes" described above may lead to deadlock, as figure 1 graphically depicts.

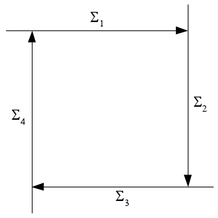

Figure 1: Deadlock situation arising when four "snakes" are about to merge.

Theorem 7 *The support management subprotocol M_{Σ}^{R} is resilient to t faults, for any $0 \leq t < k$.*

Proof: This is achieved using redundancy: whenever two runners meet, they create copies of all messages in transit. In the worst-case, there are at most $k - t$ copies of each message. Note, however, that messages may have to be re-transmitted in the case that only one copy of them exists when some fault occurs. To overcome this limitation, the sender will continue to transmit a message for which delivery has not been confirmed, to any other members of the support encountered. This guarantees that more than one copy of the message will be present within the support structure. □

3.7 Experiments

To experimentally validate, fine-tune and further investigate the proposed protocols, we performed a series of algorithmic engineering experiments.

All of our implementations follow closely the support motion subprotocols described above. They have been implemented as C++ classes using several advanced data types of LEDA [30]. Each class is installed in an environment that allows to read graphs from files, and to perform a network simulation for a given number of rounds, a fixed number of mobile users and certain communication and mobility behaviours. After the execution of the simulation, the environment stores the results again on files so that the measurements can be represented in a graphical way.

In [9] we performed extensive experiments to evaluate the performance of the "snake" protocol. A number of experiments were carried out modeling the different possible situations regarding the geographical area covered by an ad-hoc mobile network. We only considered the *restricted case where all users (even those not in the support) perform independent and concurrent random walks.*

In [12] we performed a comparative experimental study of the "snake" and the "runners" protocols based on the generic framework (as presented above) to implement protocols for mobile computing. We also have extended the experimental setup in [9] to include more pragmatic test inputs regarding motion graphs, i.e., graphs which model the topology of the motion of the hosts. Our test inputs included both random as well as more structured graphs. In addition, we considered also the case where the users (only those not in the support) perform less random (arbitrary) motions.

Our experiments showed that: (i) for both protocols only a small support is required for efficient communication; (ii) the "runners" protocol outperforms the "snake" protocol in almost all types of inputs we considered. More precisely, the "runners" protocol achieve a better average message delay in all test inputs considered, except for the case of random graphs with a small support size. The "runners" protocol achieves a higher delivery rate of messages right from the beginning, while the "snake" protocol requires some period of time until its delivery rate stabilizes to a value that is always smaller than that of the "runners". Finally, the "runners" protocol has smaller requirements for the size of local memory per member of the support.

In [10] a set of experiments were carried out to evaluate, further investigate and comparatively study the performance of the Hierarchical Support Routing Protocol (HSRP) and the "snake" algorithm in the new model of hierarchical ad-hoc networks. The experiments indicate that, although the pattern of the "snake" algorithm's performance remains the same, it does not provide satisfactory communication times even if the size of the support is big. This implies the fact (also remarked analytically) that the average message delay is affected (in fact dominated) by the time (whose expectation is very high) required for a single support to move between lower level sites in order to deliver the messages to their destination. In contrast to the above, the communication times achieved by HSRP are indeed by far more efficient than those of the original algorithm even in the case of a small probability p. The fact that HSRP is more efficient than the "snake" protocol in hierarchical networks, implies also an inherently modular and hierarchical nature of the support idea.

4 Leader Election and Counting

4.1 Problem Definition

Suppose that m mobile hosts are moving in a space $\mathcal{S}$. The hosts want to elect a unique leader. This means that starting from a configuration where all mobile hosts are in the same state, a configuration should be reached where exactly one host is the "leader" while all other hosts are in a state "lost". Furthermore, the leader should know the size of the mobile network, m.

More formally, the standard definition of the leader election problem for static networks [29, 6] is that

- eventually there is a leader and
- there should never be more than one leader

One way to perform this is to utilize an underlying routing protocol (see [1]), which delivers (if possible) a message from one mobile host to another, regardless of their position. This scheme, in the case of increased mobility of the hosts, could lead to a situation where most of the computational and battery power of the hosts is consumed for the routing protocol.

An interesting approach is presented in the paper of N.Malpani, J.Welch and N.Vaidya [27]. There, the authors present algorithms, based on the TORA [33] routing method, which maintain wireless links and guarantee exactly one leader per connected component of the implied virtual links network among mobile hosts. Their proof of correctness assumes a *bounded rate* of topology changes.

In contrast, our approach follows our general scheme of explicitly modelling motions in the motion space $\mathcal{S}$, which

```
boolean my_state=participate; the state of the mobile
host regarding the protocol (participate or inactive)
int my_id; the identity of the mobile host
int counter=1; the local counter of the host

on arriving at a new vertex u of G
if (my_state ≠ inactive) then
 begin
   broadcast < host, my_id >;
   on receive < host, i > decide_on(i);
 end

on being on a vertex u and receive < host, i >
if (my_state ≠ inactive) then
 begin
   broadcast < host, my_id >;
   decide_on(i);
 end

procedure decide_on(id)
 if (id > my_id) then
  begin
    broadcast < my_counter, counter >;
    my_state=inactive;
  end
 else
  begin
    receive < my_counter, k >;
    counter=counter+k;
  end;
```

Figure 2: The Non-Compulsory leader election protocol executed by the mobile hosts in the ad hoc network

we consider to be quantized in order to get the motion graph G.

4.2 The Algorithm

The proposed protocol executed by the mobile hosts is presented in Figure 2. This protocol is *Non-Compulsory*.

Each mobile host keeps a local counter which counts the other mobile hosts that it has met. At the beginning this counter is equal to one (the mobile host knows the existence of itself only). When two or more mobile hosts meet on a vertex u of G they exchange their identities. The winner is the one with the higher identity. The winner receives the counter of the loser and adds this to its local counter. It continues to participate in the protocol execution (it remains in state *participate*). The mobile host that lost, changes its state into *inactive* and no longer responds to messages concerning the protocol execution.

The protocol uses messages of size $O(\log\ m)$ bits since the only information carried by each message is a counter. If the given bandwidth of radio communication enables the transmission of messages of size $O(m\ \log\ m)$ bits, the hosts may also exchange lists of identities. Each mobile host may keep a local list of mobile hosts that it has defeated (initially this list contains only itself). When two hosts meet, the winner concatenates its local list with the list transmitted by the loser. In this way, the final winner will know not only the size of the mobile network but also the identities of the hosts that move inside $\mathcal{S}$.

4.3 Proof of Correctness

As can be seen from the previous section, in order for the protocol to be executed correctly the mobile hosts are required to meet and exchange information. If this is not the case, the protocol may never elect a unique leader. Another observation is that the protocol does not include any type of termination detection mechanism. Note that these weaknesses are common for all Non-Compulsory protocols. For example, suppose that the hosts are spread in different remote areas of $\mathcal{S}$ in such a way that communication among them is impossible. If they do not move beyond these areas, there is no way to execute globally any protocol and provide termination detection mechanisms. In a later section we present a *Las Vegas* variation of this simple protocol and a variation which includes termination detection if the size of space $\mathcal{S}$ is known to the mobile hosts.

In this section we assume that the hosts move in such a way that the total number of mobile hosts that participate in the protocol execution decreases in time and there exists a time instance t where only one mobile host is still in state *participate* (the one with the highest identity). This mobile host is the final winner and the size of the mobile network is contained in its *counter* variable.

A mobile host transfers its knowledge about the network size whenever it meets another host and loses (if its identity is smaller than the winner's identity). This is done by transmitting its counter to the winner. This happens only once, since after that, the host is *inactive* and no longer responds to messages concerning the protocol. Thus, a host cannot be counted twice. The *counter* variable of a mobile host contains its current knowledge about the size of the network (i.e. the sum of the counters of all other hosts it has met plus itself).

Lemma 5 *If at some time instance t, m_k is the only mobile host in state "participate" then the counter of m_k contains the size of the mobile network.*

4.4 Analysis - Average Case

We follow the analysis of the *voter model* as described by *Aldous and Fill* in [4]. We order the vertices of G as $i_1, ..., i_n$ by giving the lower label i_1 to the vertex occupied by the mobile host which is the final winner and by assigning the other labels arbitrarily to the remaining vertices. Recall that when two hosts meet only one continues to participate in the protocol execution. According to this construction,

$$C_f \leq max_j M_{i_1,j}$$

since the time C_f is less than the maximum time required for the winner to meet any other host. Let $m^* \equiv max_{i,j} E_i[M_{i,j}]$.

Lemma 6 $Pr[M_{i,j} > t] \leq exp(-\lfloor \frac{t}{em^*} \rfloor)$

Proof: Starting from time 0, divide the time axis into equal time intervals of size s. For any initial distribution μ, any $s > 0$ and any integer $k \geq 1$,

$$Pr_\mu[M_{i,j} > ks \mid M_{i,j} > (k-1)s] = Pr_\theta[M_{i,j} > s]$$

for some initial distribution θ (due to the memoryless property of the Markov chain). By definition

$$Pr_\theta[M_{i,j} > s] \leq max_i Pr_i[M_{i,j} > s]$$

By applying the Markov inequality $Pr[x > t] \leq \frac{E[x]}{t}$,

$$max_i Pr_i[M_{i,j} > s] \leq \frac{max_{i,j} E_i[M_{i,j}]}{s}$$

and finally

$$max_i Pr_i[M_{i,j} > s] \leq \frac{m^*}{s}$$

By induction on k,

$$Pr_\mu[M_{i,j} > \lambda s] \leq (\frac{m^*}{s})^\lambda$$

implying (by substituting $\lambda s = t$) that

$$Pr_\mu[M_{i,j} > t] \leq (\frac{m^*}{s})^{\lfloor \frac{t}{s} \rfloor}$$

Using the above observations, the tail of the distribution can be bounded in the following way:

$$\begin{aligned} Pr_\mu[M_{i,j} > t] &\leq \left(\frac{m^*}{s}\right)^{\lfloor \frac{t}{s} \rfloor} \\ &\leq e^{\lfloor \frac{t}{s} \rfloor \log \frac{m^*}{s}} \\ &\leq e^{\lfloor \frac{t}{m^* e} \rfloor \log \frac{1}{e}} \quad \textit{(by substituting } s = m^* e\textit{)} \\ &\leq e^{-(\log e) \lfloor \frac{t}{m^* e} \rfloor} \\ &\leq e^{\lfloor -\frac{t}{m^* e} \rfloor}, \quad 0 < t < \infty \end{aligned}$$

□

Theorem 8 *The expected value of C_f, $E[C_f]$, is bounded according to the inequality $E[C_f] \leq e(\log n + 2) max_{i,j} E_i[T_j]$.*

Proof: Recall that

$$C_f \leq max_j M_{i_1,j} \tag{5}$$

$$Pr[max_j M_{i_1,j} > t] \leq \sum_j Pr[M_{i_1,j} > t] \tag{6}$$

By definition,

$$\begin{aligned} E[C_f] &= \int_0^\infty Pr[C_f > t] dt \\ &\leq \int_0^\infty Pr[max_j M_{i_1,j} > t] dt \qquad (from \quad (5)) \\ &\leq \int_0^\infty min(1, \sum_j Pr[M_{i_1,j} > t]) dt \qquad (from \quad (6)) \\ &\leq \int_0^\infty min(1, n\, e^{-\frac{t}{em^*}}) dt \qquad (from \quad Lemma\ 6) \\ &\leq \int_0^\infty min(1, n\, e^{-\frac{t}{em^*}+1}) dt \\ &\leq \int_0^\infty min(1, n\, e\, e^{-\frac{t}{em^*}}) dt \quad (3) \end{aligned}$$

By applying the formula

$$\int_0^\infty min(1, Ae^{-\alpha t}) dt \equiv \alpha^{-1}(1 + \log A), A \geq 1$$

(3) yields

$$\begin{aligned} \int_0^\infty min(1, n\, e\, e^{-\frac{t}{em^*}}) dt &= em^*(2 + \log n) \\ &= e\, max_{i,j} E_i[M_{i,j}](2 + \log n) \end{aligned}$$

Since $max_{i,j} E_i[M_{i,j}] \leq max_{i,j} E_i[T_j]$,(see [4]), we finally conclude that

$$E[C_f] \leq e(\log n + 2) max_{i,j} E_i[T_j]$$

□

Corollary 6 *For the expected value of C_f, $E[C_f]$, $E[C_f] \leq e(\log n + 2)2\epsilon$ where ϵ is the number of edges of the graph G.*

4.4.1 A tighter upper bound on the execution time of the protocol

A key observation that leads to a tighter bound on the execution time of the protocol is that the basic inequality of the proof of Theorem 8 , $C_f \leq max_j M_{i_1,j}$, is quite rough. Specifically, this inequality corresponds to the extreme case where the winner host meets all other hosts one at a time, while in the average case, the processes exclude each other in parallel and thus accelerate the counting procedure.

Definition 12 *Let $X, X_1, X_2 \ldots X_m$ be independent random variables on the same distribution $F(x)$, probability density function $f(x)$ and mean value $E[f(x)]$. Let $F_{min(m)}(x)$ denote the distribution of the minimum of these variables, $f_{min(m)}(x)$ its probability density function and $E\left[f_{nim(m)}(x)\right]$ its mean value.*

Lemma 7 *There exists a constant $m_0 < m$ for which*

$$\forall\, m > m_0,\ E\left[f_{min(m)}(x)\right] \leq \frac{1}{m} E[f(x)]$$

Proof:

$$\begin{aligned} F_{min(m)}(x) &= Pr[min X_i < x, \quad 0 < i \leq m] \\ &= 1 - Pr[min X_i > x] \\ &= 1 - Pr[\forall i \ \ X_i > x] \\ &= 1 - Pr[X > x]^m \quad (since\ X_i\ independent) \\ &= 1 - (1 - Pr[X < x])^m \\ &= 1 - (1 - F(x))^m \end{aligned}$$

Derivation of both sides of the equation results in

$$\begin{aligned} \left[F_{min(m)}(x)\right]' &= [1 - (1 - F(x))^m]' \ \Rightarrow \\ f_{min(m)}(x) &= m\ f(x)\ (1 - F(x))^{m-1} \end{aligned}$$

By definition, $E[f(x)] = \int_0^\infty x f(x) dx$. Therefore,

$$\begin{aligned} E[f_{min(m)}(x)] &= \int_0^\infty x f_{min(m)}(x) dx \\ &= \int_0^\infty x f(x) m (1 - F(x))^{m-1} \end{aligned}$$

Since $E[f_{min(m)}(x)] \neq \infty$ there exists a real λ for which

$$\begin{aligned} E[f_{min(m)}(x)] &= 2\int_{\lambda}^{\infty} xf(x)\; m(1-F(x))^{m-1}dx \\ &\leq 2\int_{\lambda}^{\infty} xf(x)\; m(1-F(\lambda))^{m-1}dx \\ &= 2m(1-F(\lambda))^{m-1}\int_{\lambda}^{\infty} xf(x)dx \\ &\leq 2m(1-F(\lambda))^{m-1}E[f(x)] \end{aligned}$$

In order to get $E\left[f_{min(m)}(x)\right] \leq \frac{1}{m}E[f(x)]$ it suffices to have $(1-F(\lambda))^{m-1} \leq \frac{1}{2m^2}$. This holds for every $m \geq m_0$, where m_0 is a constant depending on $F(\lambda)$, since $F(x) \leq 1$ always. □

Lemma 8 *Let S_1 and S_2 be two (not necessarily) independent random variables on the same distribution and $S = S_1 + S_2$. Then, $E[S] = 2E[S_1] = 2E[S_2]$.*

Proof: By linearity of expectation. □

Definition 13 *Let $C_f(k)$ be the time required by the processes that still participate in the protocol to have defeated k other processes. Obviously, $C_f(m) = C_f$. Furthermore, let $M_{i,j}(k)$ be the time required for the first interaction of two processes (i.e. minimum of meeting times) that still participate in the protocol, when k processes still participate in the protocol.*

According to Lemma 8, each time two mobile processes meet, the counter of the winner process is doubled on the average. This yields that the final winner is the product of the interaction of two mobile processes that, on the average, have counted (and excluded) $m/2$ mobile processes each. The average duration of this final phase is $E[M_{i,j}]$. Furthermore, each one of the last two processes is a product of the interaction of two processes, each of which has counted $m/4$ processes on the average, while the duration of this phase is $E[M_{i,j}(2)]$ on the average.

Theorem 9 $E[C_f] \leq (2+\log(m_0))2\,\epsilon$*, where m_0 is a constant.*

Proof:

$$\begin{aligned} E[C_f] &= E[C_f(m)] \\ &= E[M_{i,j}(2)] + E\left[C_f\left(\frac{m}{2}\right)\right] \\ &= E[M_{i,j}(2)] + E[M_{i,j}(4)] + E\left[C_f\left(\frac{m}{4}\right)\right] \\ &= E[M_{i,j}(2)] + E[M_{i,j}(4)] + \ldots + E[M_{i,j}(m)] \\ &= \sum_{k\leq\log(m_0)} E\left[M_{i,j}(2^k)\right] + \\ &+ \sum_{\log(m_0)<k<\log(m)} E\left[M_{i,j}(2^k)\right] \\ &\leq \log(m_0)E[M_{i,j}] + \sum_{\log(m_0)<k<\log(m)} \frac{2^k}{m}E[M_{i,j}] \\ &\leq \log(m_0)E[M_{i,j}] + E[M_{i,j}] \sum_{\log(m_0)<k<\log(m)} \frac{2^k}{m} \\ &\leq \log(m_0)E[M_{i,j}] + 2\;E[M_{i,j}] \\ &\leq (2+\log(m_0))E[M_{i,j}] \\ &\leq (2+\log(m_0))2\epsilon \end{aligned}$$

□

4.4.2 The case of $m \neq n$

Up to this point we assumed that the number of mobile hosts equals the number of nodes of the underlying graph G. In this section we will prove the rather counter-intuitive fact that the execution time of the protocol is not affected seriously by the number of mobile hosts or their initial distribution on G.

The case of $m < n$

Let us first consider the case where the initial placement of the mobile hosts is a configuration chosen uniformly and at random out of all configurations where the mobile hosts do not overlap, occupying exactly m vertices. It is easy to observe that this configuration can be regarded as an instance of the execution of the protocol with n initial mobile hosts. Therefore, on the average, the execution time of the algorithm in this case is less than C_f, as described in the previous sections. Furthermore, by following the arguments of Section 4.4.1, only the first stages of the protocol execution are omitted. These stages have little contribution to the total execution time ($O\left(\frac{1}{n}E[M_{i,j}]\right)$) and thus, the protocol execution time is not seriously affected. The extreme case of $m = 2$ can provide a clear insight on the previous argument. In this case, the protocol execution time is on the average $E[M_{i,j}]$, which differs from the case of $m = n$ only by a constant factor. The chosen initial configurations can be shown to be the worst case, since in all other cases the overlapping mobile processes will exclude each other in one step, leading to a configuration with even less participating mobile hosts. In fact, suppose that m mobile hosts, $m \leq n$ are placed on m vertices of G with initial distribution μ. This case can be reduced to the case of $m = n$ by using the following construction: We consider $m - n$ new "virtual" mobile hosts. These hosts are identified with $vm_1 = -1, vm_2 = -2, \ldots, vm_{m-n} = -(m-n)$ and are placed on the remaining $m - n$ vertices of G with an initial distribution μ'. These "virtual" hosts are also participating in the protocol. Obviously they do not affect the execution and the final winner since they lose and become inactive the first time they meet one of the m hosts. According to Theorem 9, $E[C_f]$ is at most equal to $(2+\log(m_0))2\epsilon$ in this case too. The chosen initial configurations can be shown to be the worst case, since in all other cases the overlapping mobile processes will exclude each other in one step, leading to a configuration with even less participating mobile hosts.

The case of $m > n$

In this case, the mobile hosts that reside on the same node exclude each other in the first step, leading thus to a sub-case of the previous paragraph.

4.4.3 Discussion of the result

The result of Sections 4.4.1 and 4.4.2 are implying that the execution time of the protocol is linear on the average to the ratio between the volume of space $\mathcal{S}$, $V(\mathcal{S})$, and the space occupied by the transmission range of a mobile host $V(tr)$. This is a direct consequence of the construction of G (see Section 1.2.1) since $\epsilon = O(n)$ and $n = O(\frac{V(S)}{V(tr)})$. This result holds regardless of the choice of the polyhedron that is used for the quantization of $\mathcal{S}$ and the initial positions of the hosts. It also leads to the (rather intuitive) fact that powerful transmitters can speed up the protocol execution.

4.5 A variation of the basic protocol with termination detection

The calculation of the average protocol execution time allows us to construct a new protocol that provides a unique leader with a given probability of success. We consider that the execution of the protocol fails if the mobile hosts stop the execution of the protocol without having a unique leader. Note that in this case there is no mobile host whose counter contains the correct network size. Let p_f denote the protocol failure probability ($p_f \equiv Pr[C_f > t]$).

Lemma 9 *For any given failure probability p_f, running the protocol for time $t = \frac{c\epsilon}{p_f}$ where c is a constant, suffices to ensure that the result will be correct with failure probability at most p_f.*

Proof:

$$\begin{aligned} Pr[C_f > t] &< \frac{E[C_f]}{t} \Rightarrow \\ t &\leq \frac{E[C_f]}{Pr[C_f > t]} \\ &\leq \frac{c\epsilon}{p_f} \end{aligned}$$

□

Lemma 9 implies that the mobile hosts may decide on the duration of the protocol execution only if they know the total number of edges ϵ of the underlying graph G. Recall that in G, the number of edges ϵ is linear to the number of vertices n (in the case of a cube-based grid ϵ is related to n by the inequality $\epsilon \leq 6n$). Intuitively and by taking into consideration the discussion in Section 1.2.1, a mobile host must have a feeling of the available space in order to be able to determine whether the protocol has terminated or not. Consider, for example, a mobile host moving in the area of Manhattan, that at time t remains in state *participate* having defeated k other mobile hosts. Its estimation of being a unique winner would be different if the total area in question is Manhattan, New York or North America.

Finally, note that in the case of time constraints (i.e. the mobile hosts have to execute the protocol for a specific time period), Lemma 9 may be applied by the mobile hosts to calculate the probability of success for a given run time t, in order to find out how much they can rely on the result.

4.6 Las Vegas Compulsory protocols for leader election and counting in an ad hoc network without sense of orientation

Since a major weakness of the basic Non-Compulsory protocol is that it may never elect a unique leader, the previous results lead to the specification of a *Las Vegas Compulsory* protocol that elects a unique leader in the ad hoc mobile network in time linear to n with a given probability of success. This protocol behaves as the one described in Section 4.2, but in addition, it forces the mobile hosts to perform a random walk on G. The protocol includes also termination detection as it is presented in the previous section. The host can decide on a failure probability p_f and use it in order to find the required time t to run the protocol. If the host after time t is still in state participate, it is (with probability at least $1 - p_f$) the unique leader (on the other hand, if it is in state inactive it knows that a unique leader has been probably elected). The proposed Las Vegas Compulsory protocol can be applied even in the case where the mobile hosts have no sense of orientation. We further conjecture that this algorithm is *optimal in time* in the case of mobile hosts with no sense of orientation. A straightforward extension of this protocol is a Compulsory counting protocol. After time t, the leader initiates a flooding phase as follows: Every time it meets another mobile host it transmits its counter. Each mobile host that has been informed about the counter behaves in exactly the same way. The duration of this phase is obviously bounded by $(2 + \log(m_0))2\epsilon$ too.

4.6.1 Anonymous networks

The protocols described in the previous section can also be applied when the mobile network is *anonymous*, i.e. the mobile hosts do not have distinct identities. In the latter case the proposed variation leads to *Las Vegas Compulsory* protocols for *leader election* and *counting in anonymous ad hoc networks without sense of orientation*. The proposed variation is as follows. Whenever a mobile host meets another host on a vertex of G they choose random identities over a predefined set and then exchange them. If there is a conflict, they repeat the procedure. The winner is finally the one that selected the higher identity. We remark that the time required for two hosts to select unique identities when they meet (regardless of the number of conflicts) is negligible compared to the time required for a mobile host to move from one vertex of G to another and thus does not affect the protocol execution time.

4.6.2 Comparison theorems and the steady-state of the random motion

Let us note that a related "many objects" motion process is the so called Symmetric Exclusion (SE) process. Let G be an undirected graph of n vertices. To start, r unlabeled particles are placed in an initial configuration, $1 \leq r \leq n$. At each step, a particle is chosen at random. Then one of the neighboring sites of this particle is chosen at random. If the neighboring site is unoccupied, the chosen particle moves there; if the neighboring site is occupied the system stays as it was. This is a reversible Markov chain on the r-sets of $1, 2, \ldots, n$. The reader can easily recognize that this is the discrete analog of our proposed model of motions of the mobile hosts, provided that the "time" in the concurrent movement case is replaced by rounds of steps (i.e. a step sequence in which all particles move at least once). Let us denote our continuous model by RM. Fill (1991) gave bounds on the second eigenvalue of SE (but for labeled particles) on the finite circle of n vertices.

Diaconis and Saloff-Coste [17] study the chain SE by comparison with a second Markov chain on r-sets that proceeds by picking an unoccupied site at random (not necessarily a neighboring site) and moving the particle to the unoccupied site. This second process which corresponds to mobile hosts moving "very fast" or e.g. by teleportation (or other means), is a well-studied chain (the Bernoulli-Laplace model for diffusion). Its eigenvalues are known. We denote this process by TP.

The method of comparison of [17] is based on the minimax characterization of eigenvalues based on the Dirichlet forms of the chains, viewed as a self-adjoint operator because of reversibility (see e.g. Horn and Johnson, 1985).

Diaconis and Saloff-Coste developed a method for comparison of the Dirilecht forms, which provides tight upper and lower bounds on the eigenvalues of SE (and thus on the eigenvalues of RM) by using the eigenvalues of TP and structural (path) properties of the graph G. The method gives bounds for the mixing properties of RM on any particular network G. A crude but universal estimate for the second largest eigenvalue β_1 of SE (the first is unity) is

$$\beta_1 \leq 1 - 1/rn^2 d_0$$

where d_0 is the maximum degree of the graph. This bound is easily obtained by the above method.

4.7 Experiments

The basic protocol was implemented and tested using the Distributed Systems Platform (DSP) [1](see [41]). The DSP tool allows the specification, implementation and testing of distributed algorithms for fixed and mobile networks. The simulations focused on the case of $m = n$, with the mobile processes performing random walks on the underlying graph. An example of the simulation results for 10 runs and three different topologies can be found in [21].

In general, the simulations confirmed the theoretical results. The number of participating hosts decreased exponentially in time. In fact, on the average, one third of the mobile hosts were defeated during the first step of the protocol. The protocol execution time was on the average less than the theoretical result, because of the fact that the analysis holds for every underlying graph G, while the quantization of the space leads usually to regular graphs, for which it is known that the random walks mix faster. As expected, the average protocol execution time increased for more irregular topologies as compared to the execution time for regular topologies of the seme size. Finally, the protocol execution time increased linear to the number of nodes for regular topologies of the same degree.

5 Future Work

We are currently exploring the direction of impossibility results for any distributed algorithm that attempts to maintain structural information of the implied fragile network of virtual links.

We are also exploring the Physics paradigm of *interacting particles* and their modeling, since we feel that mobile hosts interactions may behave in a similar way.

References

[1] M. Adler and C. Scheideler: Efficient Communication Strategies for Ad-Hoc Wireless Networks. In Proc. 10th Annual Symposium on Parallel Algorithms and Architectures (SPAA'98)(1998).

[2] M. Ajtai, J. Komlos and E. Szemeredi: Deterministic Simulation in LOGSPACE. In Proc. of 19th Annual ACM Symposium on Theory of Computing (STOC'87) (1987).

[3] D. Aldous and P. Diaconis: Strong Stationary Times and Finite Random Walks. Adv. Appl. Math., 8:69-97, 1987 (1987).

[4] D. Aldous and J. Fill: *Reversible Markov Chains and Random Walks on Graphs.* Unpublished manuscript. http://stat-www.berkeley.edu/users/aldous/book.html (1999).

[5] N. Alon and F. R. K. Chung: Explicit Construction of linear sized tolerant networks. Discrete Mathematics, vol 72, pp 15-19, 1998 (1998).

[6] H. Attiya and J. Welch: *Distributed Computing: Fundamentals, Simulations and Advanced Topics* McGraw-Hill Publishing Company, 1998 (1998).

[7] G. Brightwell and P. Winkler: Maximum Hitting Times for Random Walks on Graphs. Journal of Random Structures and Algorithms, 1:263-276, 1990 (1990).

[8] J. Broch, D. B. Johnson, and D. A. Maltz: The dynamic source routing protocol for mobile ad-hoc networks. IETF, Internet Draft, draft-ietf-manet-dsr-01.txt, Dec. 1998. (1998).

[9] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis: Analysis and Experimental Evaluation of an Innovative and Efficient Routing Approach for Ad-hoc Mobile Networks. In Proc. 4th Annual Workshop on Algorithmic Engineering (WAE'00)(2000).

[10] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis: An Efficient Routing Protocol for Hierarchical Ad-Hoc Mobile Networks In Proc. 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, 15th Annual International Parallel & Distributed Processing Symposium (IPDPS'01)(2001).

[11] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis: An Efficient Communication Strategy for Ad-hoc Mobile Networks In Proc. 15th International Symposium on Distributed Computing (DISC'01)(2001). Also as a Brief Announcement in Proc. 20th Annual Symposium on Principles of Distributed Computing (PODC'01)(2001).

[12] I. Chatzigiannakis, S. Nikoletseas, N. Paspalis, P. Spirakis and C. Zaroliagis: An Experimental Study of Basic Communication for Ad-hoc Mobile Networks. In Proc. 5th Annual Workshop on Algorithmic Engineering (WAE'01)(2001).

[13] D. Coppersmith, P. Tetali and P. Winkler: Collisions among Random Walks on a Graph. SIAM J. Disc. Math., 6:363-374, 1993. (1993).

[14] D. Coppersmith, P. Doyle, P. Raghavan and M. Snir: Random Walks on Weighted Graphs and Applications to On-line Algorithms. In Proc. 22nd ACM Symposium on Theory of Computing, pages 369-378, 1990 (1990).

[1] The DSP tool was designed and implemented during the EU ESPRIT LTR ALCOM-IT project

[15] G. Dommety and R. Jain: Potential networking applications of global positioning systems (GPS). Tech. Rep. TR-24, CS Dept., The Ohio State University, April 1996 (1996).

[16] R. Durret: *Probability: Theory and Examples.* Wadsworth. (1991).

[17] P. Diaconis and L. Saloff-Coste: Comparison Theorems for reversible Markov Chains. The Annals of Applied Probability, Vol. 3, No. 3, 1993, pp. 696-730 (1993).

[18] Z. J. Haas and M. R. Pearlman: The performance of a new routing protocol for the reconfigurable wireless networks. In Proc. of the IEEE International Conference on Communications (ICC'98) (1998).

[19] Z. J. Haas and M. R. Pearlman: The zone routing protocol (ZRP) for ad-hoc networks. IETF, Internet Draft, draft-zone-routing-protocol-02.txt, June 1999. (1999).

[20] K. P. Hatzis, G. P. Pentaris, P. G. Spirakis, V. T. Tampakas and R. B. Tan: Fundamental Control Algorithms in Mobile Networks. In Proc. 11th Annual Symposium on Parallel Algorithms and Architectures (SPAA'99) (1999).

[21] K. Hatzis, G. Pentaris, P. Spirakis and V. Tampakas: Counting in Mobile Networks: Theory and Experimentation. In Proc. 3rd Annual Workshop on Algorithmic Engineering (WAE'99) (1999).

[22] G. Holland and N. Vaidya: Analysis of TCP Performance over Mobile Ad Hoc Networks. In 5th Annual ACM/IEEE International Conference on Mobile Computing (MOBICOM'99) (1999).

[23] T. Imielinski and H. F. Korth: *Mobile Computing.* Kluwer Academic Publishers. (1996).

[24] Y. Ko and N. H. Vaidya: Location-Aided Routing (LAR) in Mobile Ad-hoc Networks. In Proc. 4th Annual ACM/IEEE International Conference on Mobile Computing (MOBICOM'98) (1998).

[25] E. Koutsoupias and C. Papadimitriou: Worst-Case Equilibria. In Proc. 16th Symposium on Theoretical Aspects of Computer Science (STACS'99) (1999).

[26] Q. Li and D. Rus: Sending Messages to Mobile Users in Disconnected Ad-hoc Wireless Networks. In Proc. 6th Annual ACM/IEEE International Conference on Mobile Computing (MOBICOM'00) (2000).

[27] N. Malpani, J.L. Welch and N. Vaidya: Leader Election Algorithms for Mobile Ad Hoc Networks. In Proc. 3rd Annual Symposium on Discrete Algorithms and Models for Mobility (DIALM'00) (2000).

[28] M. Mavronicolas and P. Spirakis: The Price of Selfish Routing. In Proc. 33rd Symposium on Theory of Computing (STOC'01) (2001).

[29] N. A. Lynch: *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., 1996 (1996).

[30] K. Mehlhorn and S. Näher: *LEDA: A Platform for Combinatorial and Geometric Computing.* Cambridge University Press, 1999 (1999).

[31] R. Motwani and P. Raghavan: *Randomized Algorithms.* Cambridge University Press. (1995).

[32] B. Parkinson and S. Gilbert: NAVSTAR: global positioning system ten years later In Proceeding of IEEE, pp. 1177-1186, 1983 (1983).

[33] V. D. Park and M. S. Corson: Temporally-ordered routing algorithms (TORA) version 1 functional specification. IETF, Internet Draft, draft-ietf-manet-tora-spec-02.txt, Oct.1999. (1999).

[34] Charles E. Perkins: *Ad Hoc Networking.* Addison-Wesley, Boston, USA, Jan'2001 (2001).

[35] C. E. Perkins and E. M. Royer: Ad-hoc on demand distance vector (AODV) routing. IETF, Internet Draft, draft-ietf-manet-aodv-04.txt (IETF'99). (1999).

[36] P. Spirakis and B. Tampakas: Distributed Pursuit-Evasion: Some Aspects of Privacy and Security of Distributed Computing (Brief Announcement). In Proc. 13th Annual Symposium on Principles of Distributed Computing (PODC'94) (1994).

[37] P. Spirakis, B. Tampakas and H. Antonopoulou: Distributed Protocols against Mobile Eavesdroppers. In Proc. 9th International Workshop on Distributed Algorithms (WDAG'95), Lecture Notes in Computer Science, vol. 972 (Springer-Verlag, 1995), pp. 160-167 (1995).

[38] E. Szemeredi: Regular Partitions of graphs. Colloques Internationaux C.N.R.S., Nr 260, Problemes Combinatoires et Theorie des Graphes, Orsay, pp. 399-401 (1976).

[39] J. E. Walter, J.L. Welch and N. M. Amato: Distributed Reconfiguration of Metamorphic Robot Chains In Proc. 19th Annual Symposium on Principles of Distributed Computing (PODC'00) (2000).

[40] Y. Zhang and W. Lee: Intrusion Detection in Wireless Ad-hoc Networks. In Proc. 6th Annual ACM/IEEE International Conference on Mobile Computing (MOBICOM'00) (2000).

[41] DSP Web Site, http://helios.cti.gr/alcom-it/dsp

[42] Iowa State University GPS page. http://www.cnde.iastate.edu/ gps.html.

[43] NAVSTAR GPS operations. http://tycho.usno.navy.mil/gpsinfo.html.