Alcom-FT Technical Report Series
ALCOMFT-TR-03-179

Distributed Communication Algorithms for Ad-hoc Mobile Networks[1,2]

Ioannis Chatzigiannakis[2,3] Sotiris Nikoletseas[2,3] Paul Spirakis[2,3]

September 13, 2003

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 dynamically 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* behavior of protocols for such networks, based on the *assumption* that mobile hosts (those whose motion is not guided by the protocol) conduct concurrent *random walks* in their motion space.

In particular, our work examines the fundamental problem of communication and proposes distributed algorithms for it. We provide rigorous proofs of their correctness, and also give performance analyses by combinatorial tools. Finally, we have evaluated these protocols by experimental means.

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.

[1] A preliminary version of this work has appeared in the 15th Symposium on Distributed Computing (DISC), in the 4th, 5th Workshops on Algorithmic Engineering (WAE), see [10, 12, 13].

[2] This work was partially supported by the EU projects IST FET-OPEN ALCOM-FT, FLAGS, CRESCCO and IMPROVING RTN ARACNE.

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

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

An ad-hoc mobile network ([25, 35]) is a collection of mobile hosts with wireless interfaces forming a temporary network *without the aid of any established infrastructure or centralized 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 modeled 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, 29, 35].

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 [35] also states, in ad-hoc mobile networks the topological connectivity is subject to frequent, unpredictable changes (due to the 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 [35] suggests 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 [29] 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 [25, 27, 34, 36]. 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 pair wise 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 routing 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 of 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"* [25], 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 because 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" ([38]) 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 (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 [38].

1.2 Previous and related work

In a recent paper [28], 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 simultaneous processing of many communication pairs. In their setting, they show optimality of message transmission times.

M.Adler and C.Scheideler [1] deal 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 random uniform speading of the hosts.

We note here (see also 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 [38]. 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 Election in ad-hoc mobile networks is presented in the paper of N.Malpani, J.Welch and N.Vaidya [29]. There, the authors present algorithms, based on the TORA [34] 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.

1.3 The explicit model of motions

1.3.1 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 introducing 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 centered by itself. This means that any other host inside tr can receive any message broadcast 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 1.1 *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 $\mathcal{V}(\mathcal{S})$ of space $\mathcal{S}$, and the space occupied by the transmission range of a mobile host $\mathcal{V}(tr)$. In the extreme case where $\mathcal{V}(\mathcal{S}) \simeq \mathcal{V}(tr)$ (the transmission range of the hosts approximates the space where 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 $n = O(\frac{V(\mathcal{S})}{V(tr)})$. 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 neighboring polyhedra each node is connected with a constant number of neighbors, 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.

1.3.2 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 examine the case where the motions of any host not affected by the algorithm are modeled 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. [21, 24]).

We interestingly note here a fundamental 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 [2, 5] and is captured by the famous Szemeredi's Regularity Lemma [37] 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.

2 A framework for protocol classes

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

Definition 2.1 Non-Compulsory *protocols are the ones whose execution does not affect the movement of the mobile hosts. On the other hand,* compulsory *protocols are 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) [9, 18, 33, 40, 41] facilities.

Definition 2.2 *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 pair wise communication but also for a whole set of basic problems including many-to-one communication, information spreading and multicasting.

Definition 2.3 *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 [25] 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 2.4 *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 hope that some of them will then eventually meet the receiver) that will effectively solve the basic communication 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, 22, 27, 34, 36, 39]. Indeed, this approach of exploiting pair wise 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 such cases.

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 avoids 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 visit (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, it is clear that 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 [39].

3.3 The "Snake" support management subprotocol M_Σ^S

The main idea of the protocol proposed in [10, 12] 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 breaking symmetry protocol in anonymous networks (see e.g. [23]). 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 pair wise adjacent (i.e., forming a chain 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 formally, the movement of Σ is 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 a 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 basic communication 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 transmits 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 pair wise adjacency between members of Σ, all hosts sweep the area by moving independently of 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 being within communication range of a node of the support.

As presented in [13], the runners protocol does not use the idea of a (moving) backbone sub-network 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 vertex of the motion graph G, the synchronization sub-protocol P_3^R is activated. This subprotocol imposes that when runners meet on the same vertex, 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 [30] 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 are not in 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 ID.

3.5 The Hierarchical Approach

In this section we briefly introduce a *new model* of ad-hoc mobile networks, which we call *hierarchical* [11] 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.

For a detailed description and the analysis of communication in the hierarchical case see [11, ?].

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 transfer easily, see [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 3.1 *Since the motion graph G is finite and connected, the continuous Markov chain abstracting the random walk on it is automatically time-reversible.*

Definition 3.1 *$P_i(E)$ is the probability that the random walk satisfies an event E given it started at vertex i.*

Definition 3.2 *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 3.3 *For the random 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 3.4 *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 3.5 *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 3.2 *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 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 proves 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

$$P_i(X(U_i) = \sigma \mid U_i = u) = \frac{1}{e}\pi(\sigma) \tag{1}$$

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

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

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.

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

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

$$\begin{aligned} P_i(X(U_i) = \sigma \mid U_i = \lambda u) &= \sum_{\alpha \in V} P_i(X(U_i) = \alpha \mid U_i = (\lambda - 1)u) \cdot \\ &\quad \cdot P(X(U_i + u) = \sigma \mid X(U_i) = \alpha) \\ &= e^{-(\lambda-2)}\frac{1}{e}\sum_{\alpha \in V}\pi(\alpha)\frac{1}{e}\pi(\sigma) \qquad \text{from (1)} \\ &= e^{-(\lambda-1)}\frac{1}{e}\pi(\sigma) \end{aligned}$$

which ends the induction step. Then, for all σ

$$P_i(X(U_i) = \sigma) = \pi(\sigma) \tag{2}$$

and, from the geometric distribution with $q = \frac{1}{e}$,

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

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 $0 = U_0 < U_1 < U_2 < \dots$ such that for $i \geq 0$

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

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 of how 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. [19]).
From Wald's equality [4] then

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

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

3.6.3 Worst case time efficiency for the restricted motion adversary

Clearly, one intuitively expects that if $k = |\Sigma|$ is the size of the support 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 3.2, we can create a sequence of strong stationary times U_i such that $X(U_i) \in F$ where F=$\{\sigma$: 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 neighborhood 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 $\frac{k}{2m}$ (since each $\pi(\sigma)$ is at least $\frac{1}{2m}$ and there are at least k vertices σ in the induced subgraph), 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:

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

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

Lemma 3.4 *For the walk of Σ's head on the entire motion graph G, the communication time's expected value 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 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 (since in this case $d(\sigma) = r$ and $2m = rn$, for all σ, where r is the degree of any 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' and since all $\pi_{X'}(\sigma)$ are equal,

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

Now let

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

Regardless of initial positions, the chance that MH_S (or MH_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

$$P(MH_S, X' \text{ are together at time s}) = \int_0^s f(u) P(\text{together at time s} \mid M = u)\, du$$

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

$$P(MH_S, X' \text{ together at time s}) \ \leq \ \int_0^s f(u)p^*(s-u)\ du$$

But we know (see [4]) that

Lemma 3.5 *[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

$$P(MH_S, X' \text{ are together at time s}) \leq \frac{1}{n}P(M \leq s) \ + \ K\int_0^s f(u)(s-u)^{-\frac{1}{2}}du$$

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

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 $(1 \ - \ \frac{1}{e})^{-1}$) to get

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

i.e., by iteration,

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

where u is as in the construction of Theorem 3.2. Thus,

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

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

Theorem 3.6 *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 time is bounded above by $2\gamma\frac{n^2}{k} + \Theta(k)$ and by choosing $k = \sqrt{2\gamma n^2}$ we can get a best bound of $\Theta(n)$ for a support size of $\Theta(n)$.

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 3.7 *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 $\Theta(\rho)$, where ρ is the relative motion space size, and this time is independent of the total number of mobile hosts, and their movement.*

Note 3.8 *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 3.9 *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 $\Theta(\gamma\frac{\rho}{\sqrt{\psi}})$ where γ is an absolute constant.*

3.6.5 Time efficiency - A Lower Bound

Lemma 3.10

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

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 3.11 *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.,

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

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 3.12 *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 3.13 *Since $m = \Theta(n)$ we get a* $\Theta(n)$ lower bound *for the expected communication time. In that sense, our protocol's expected communication time is* optimal, *for a support size which is* $\Theta(n)$.

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 [10, 23], 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 [24], [21].

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_Σ^S 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 $X + Y + Z$, where X is the time for MH_S to meet Σ, Y is the time for MH_R to meet Σ (after X) and Z is the message propagation time within Σ, in order to estimate the expected values of X and Y (these 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*, π of its walk on H. We denote the expected value of this hitting time by $E_\pi T_\sigma^H$.

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

Note 3.14 *The equation above amortizes 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 3.15 *For any node σ of any graph H in a continuous-time random walk*

$$E_\pi T_\sigma^H \leq \frac{\tau_2(1 - \pi(\sigma))}{\pi(\sigma)}$$

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

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

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,

$$\pi(i) E_\pi T_i \;=\; \sum_{0}^{\infty} (P_{ii}(t) \;-\; \pi(i))\, dt$$

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

$$P_{ii}^{(t)} \;-\; \pi(i) \;=\; \sum_{m \geq 2} u_{im}^2\, exp(-\lambda_m t)$$

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

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

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

$$\pi(i) E_\pi T_i \;\leq\; (1 - \pi(i)) \sum_{0}^{\infty} exp(-\lambda_2 t)\, dt$$

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

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

■

Note 3.16 *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* $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_{n'}$, $n' = n - k + 1$ *(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. [32]) 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 $\lambda_2^H \;\geq\; \lambda_2^G$ (see *Lemma 1.15* [15], ch. 1, p.13).

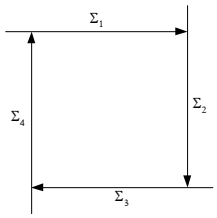

Figure 1: Deadlock situation when four "snakes" are about to be merged.

Theorem 3.17

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

Theorem 3.18 *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 3.19 *The above upper bound is minimized 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).*

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

$$E_\pi T_\sigma^H \;\geq\; \frac{(1-\pi(\sigma))^2}{q_\sigma \pi(\sigma)}$$

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 Robustness

Now, we examine the robustness of the support management subprotocol M_Σ^S and M_Σ^R under single stop-faults.

Theorem 3.20 *The support management subprotocol* M_Σ^S *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 3.2) 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 that more than one faults occur, the procedure for merging snakes described above may lead to deadlock (see Figure 1).

Theorem 3.21 *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 Experimental Evaluation

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 [31]. 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 behaviors. 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.

3.7.1 Experimental Set-up

A number of experiments were carried out modeling the different possible situations regarding the geographical area covered by an ad-hoc mobile network. We considered three kinds of inputs, unstructured (random) and more structured ones. Each kind of input corresponds to a different type of motion graph.

For all the cases that we investigated, each $h \notin \Sigma$, generates a new message with probability $p_h = 0.01$ (i.e., a new message is generated roughly every 100 rounds) by picking a random destination host. The selection of this value for p_h is based on the assumption that the mobile users do not execute a real-time application that requires continuous exchange of messages. Based on this choice of p_h, each experiment takes several thousands of rounds in order to get an acceptable average message delay. For each experiment, a total of $100,000$ messages were transmitted. We carried out each experiment until the $100,000$ messages were delivered to the designated receivers. Our test inputs are as follows.

Random Graphs. This class of graphs is a natural starting point in order to experiment on areas with obstacles. We used the $G_{n,p}$ model obtained by sampling the edges of a complete graph of n nodes independently with probability p. We used $p = \frac{1.05 \log n}{n}$ which is marginally above the connectivity threshold for $G_{n,p}$. The test cases included random graphs with $n \in \{1600, 3200, 6400, 40000, 320000\}$ over different values of $k \in [3, 45]$.

2D Grid Graphs. This class of graphs is the simplest model of motion one can consider (e.g., mobile hosts that move on a plane surface). We used two different $\sqrt{n} \times \sqrt{n}$ grid graphs with $n \in \{400, 1600\}$ over different values of k. For example, if $\mathcal{V}(tc) = 50m^2$ then the area covered by the ad-hoc network will be $20,000m^2$ in the case where $n = 400$.

Bipartite multi-stage graphs. A bipartite multi-stage graph is a graph consisting of a number ξ of stages (or levels). Each stage contains n/ξ vertices and there are edges between vertices of consecutive stages. These edges are chosen randomly with some probability p among all possible edges between the two stages. This type of graphs is interesting as such graphs may model movements of hosts that have to pass through certain places or regions, and have a different second eigenvalue than grid and $G_{n,p}$ graphs (their second eigenvalue lies between that of grid and $G_{n,p}$ graphs). In our experiments, we considered $\xi = \log n$ stages and chose $p = \frac{n}{\xi} \log \frac{n}{\xi}$ which is the threshold value

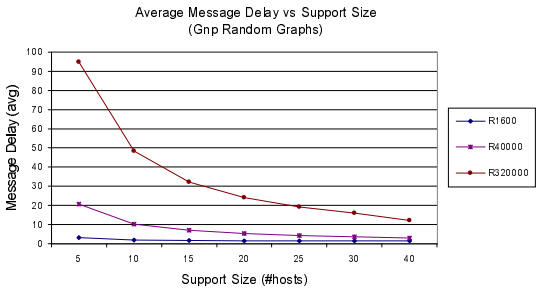

Figure 2: Average message delay for the "Snake" approach over Gnp types of motion graphs with different sizes n and varied support size k.

for bipartite connectivity (i.e., connectivity between each pair of stages). The test cases included multi-stage graphs with 7, 8, 9 and 10 stages, number of vertices $n \in \{1600, 3200, 6400, 40000\}$, and different values of $k \in [3, 45]$.

3.7.2 Validation of the "Snake" protocol's performance (average case)

In [10] 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 *average case where all users (even those not in the support) perform independent and concurrent random walks.*

In the first set of experiments, we focused on the validation of the theoretical bounds provided for the performance of the "Snake" protocol. For each message generated, we calculated the delay X (in terms of rounds) until the message has been transmitted to a member of Σ. Additionally, for each message received by the support, we logged the delay Y (in terms of rounds) until the message was transmitted to the receiver. We assumed that no extra delay was imposed by the mobile support hosts (i.e. $T_\Sigma = 0$). We then used the average delay over all measured fractions of message delays for each experiment in order to compare the results and further discuss them.

The experimental results have been used to compare the average message delays ($E(T_{total})$) with the upper bound provided by the theoretical analysis of the protocol. We observe that as the total number n of motion-graph nodes remains constant, as we increase the size k of Σ, the total message delay (i.e. $\mathrm{E}(T_{total})$) is decreased. Actually, $\mathrm{E}(T_{total})$ initially decreases very fast with k, while having a limiting behavior of no further significant improvement when k crosses a certain threshold value. Therefore, taking into account a possible amount of statistical error, the following has been experimentally validated:

$$\text{if } k_1 > k_2 \Rightarrow E_1(T_{total}) < E_2(T_{total})$$

Throughout the experiments we used small and medium sized graphs, regarding the total number of nodes in the motion graph. Over the same type of graph (i.e. grid graphs) we observed that the average message delay $\mathrm{E}(T_{total})$ increases as the total number of nodes increases. This can be clearly seen from figure 3 where the curves of the four different graph-sizes are displayed. It is safe to assume that for the same graph type and for a fixed-size Σ the overall message delay increases with the size of the motion graph. We further observe that $\mathrm{E}(T_{total})$ does not increase linearly but is affected by the expansion rate of the graph, or otherwise, as stated in the theoretical analysis,

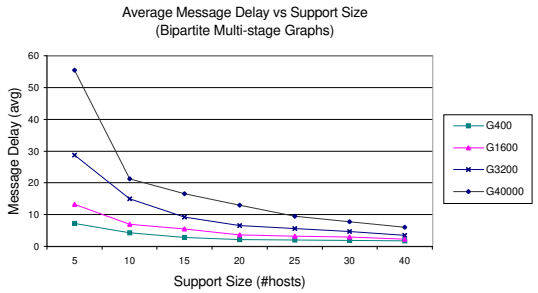

Figure 3: Average message delay of the "Snake" approach over bipartite multi-stage types of motion graphs with different sizes (n) and varied support size k.

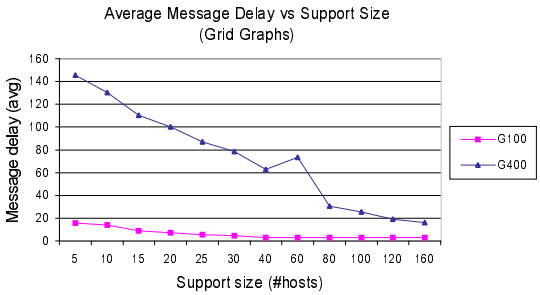

Figure 4: Average message delay of "Snake" approach over grid types of motion graphs with different sizes (n) and varied support size k.

by the second eigenvalue of the matrix of the graph. If we take into account possible amount of statistical error that our experiments are inherently prone to, we can clearly conclude the following:

$$\text{if } n_1 > n_2 \Rightarrow E_1(T_{total}) > E_2(T_{total})$$

Furthermore, we remark that $\mathrm{E}(T_{total})$ does not only depend on the actual size of the graph and the size of the support Σ but, additionally, that it is directly affected by the type of the graph. This is expressed throughout the theoretical analysis by the effect of the second eigenvalue of the matrix of the graph on communication times. If we combine the experimental results of figures 2, 3 and 4 for fixed number of motion-graph nodes (e.g. $n = 400$) and fixed number of Σ size (e.g. $k = 40$) we observe the following:

$$E_{grid}(T_{total}) > E_{multistage}(T_{total}) > E_{G_{n,p}}(T_{total})$$

which validates the corresponding results due to the analysis.

In section 3.6.6, we note that the upper bound on communication times $E(T_{total}) \leq \frac{2}{\lambda_2(G)}\,\Theta\left(\frac{n}{k}\right) + \Theta(k)$ is minimized when $k = \sqrt{\frac{2n}{\lambda_2(G)}}$. Although our experiments disregard the delay imposed by Σ (i.e. we take $\Theta(k) \to 0$) this does not significantly affect our measurements. Interestingly, the values of figures 2, 3 and 4 imply that the analytical upper bound is not very tight. In particular, for the $G_{n,p}$ graphs we clearly observe that the average message delay drops below 3 rounds (which is a very low, almost negligible, delay) if Σ size is equal to 10; however, the above formula implies that this size should be higher, especially for the medium sized graphs. Actually, it is further

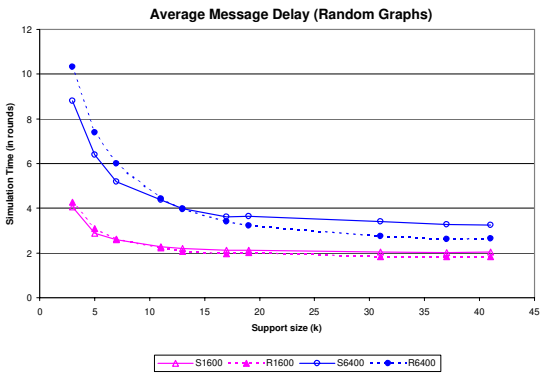

Figure 5: Average message delay over random graphs with different sizes (n) and varied support size k, for the Snake (S) protocol and Runners (R) protocol.

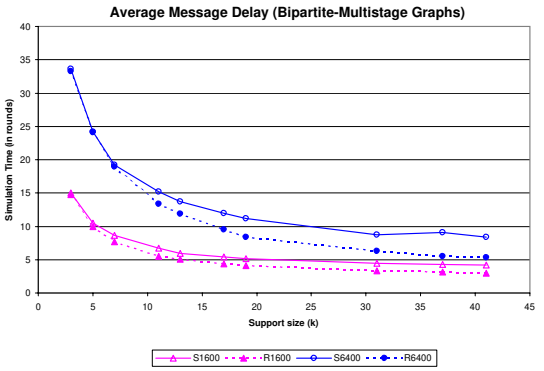

Figure 6: Average message delay over bipartite multi-stage motion graphs with different sizes (n) and varied support size k, for the Snake (S) protocol and Runners (R) protocol.

observed that there exists a certain threshold value (as also implied by the theoretical analysis) for the size of Σ above which the overall message delay does not further significantly improve. The threshold size for the support validated experimentally is smaller than the threshold implied by the analytical bound.

3.7.3 Experimental Comparison of the "Snake" and the "Runners" approach

In [13] we performed a comparative experimental study of the "snake" and the "runners" protocols. Our test inputs were similar to that used in section 3.7.2. We measured the total delay of messages exchanged between pairs of sender-receiver users. For each motion graph constructed, we injected $1,000$ users (mobile hosts) at random positions that generated 100 transaction message exchanges of 1 packet each by randomly picking different destinations. For each message generated, we calculated the overall delay (in terms of simulation rounds) until the message was finally transmitted to the receiver. We used these measurements to experimentally evaluate the performance of the two different support management subprotocols.

The reported experiments for the three different test inputs we considered are illustrated in Figures 5–7. In these figures, we have included the results of two instances of the input graph

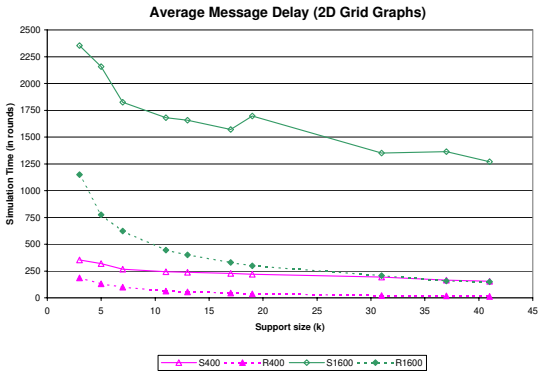

Figure 7: Average message delay over grid motion graphs with different sizes (n) and varied support size k, for the Snake (S) protocol and Runners (R) protocol.

w.r.t. n, namely a smaller and a larger value of n (similar results hold for other values of n). Each curve in the figures is characterized by the name 'Px', where P refers to the protocol used (S for snake and R for runners) and x is a 3 or 4 digit number denoting the value of n.

The curves reported for the snake protocol confirm the theoretical analysis in section 3.6.6. That is, the average message delay drops rather quickly when k is small, but after some threshold value stabilizes and becomes almost independent of the value of k. This observation applies to all test inputs we considered. We also note that this behavior of the snake protocol is similar to the one reported in 3.7.2, although we count here the extra delay time imposed by the synchronization subprotocol P_3.

Regarding the "runners" protocol, we firstly observe that its curve follows the same pattern with that of the snake protocol. Unfortunately, we don't have yet a theoretical analysis for the new protocol to see whether it behaves as should expected, but from the experimental evidence we suspect that it obeys a similar but tighter bound than that of the snake protocol. Our second observation is that the performance of the "runners" protocol is slightly better than that of the "snake" protocol in random graphs (except for the case of random graphs with small support size; see Figure 5) and bipartite multi-stage graphs (Figure 6), but is substantially better in the more structured case (grid graphs; cf. Figures 7). This could be partly explained by the fact that the structured graphs have a smaller second eigenvalue than that of bipartite multi-stage and random graphs. A small value for this eigenvalue makes the average message delay to be more dependent on the hidden constant in the asymptotic analysis of the "snake" protocol (see section 3.6.6) which is apparently larger than the corresponding constant of the runners protocol.

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. 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.

[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.

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

[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, December, 1998.

[9] A. Boukerche: A Source Route Discovery Optimization Scheme in Mobile Ad hoc Networks. In *Proc. 8th European Conference on Parallel Processing* – EUROPAR'02, 2002.

[10] 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. Lecture Notes in Computer Science, Volume 1982 (Springer-Verlag, 2000).

[11] 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*, satellite workshop of *15th Annual International Parallel & Distributed Processing Symposium* – IPDPS'01, 2001.

[12] 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, pp. 285–299, 2001. Also Brief Announcement in *Proc. 20th Annual Symposium on Principles of Distributed Computing* – PODC'01, pp. 320–322, 2001. Lecture Notes in Computer Science, Volume 2180 (Springer-Verlag, 2001).

[13] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis: Experimental Evaluation of Basic Communication for Ad-hoc Mobile Networks. In *Proc. 5th Annual Workshop on Algorithmic Engineering* – WAE'01, pp. 159–171, 2001. Lecture Notes in Computer Science, Volume 2141 (Springer-Verlag, 2001).

[14] I. Chatzigiannakis and S. Nikoletseas: Design and Analysis of an Efficient Communication Strategy for Hierarchical and Highly Changing Ad-hoc Mobile Networks. In ACM/Baltzer Mobile Networks Journal (MONET), Special Issue on Parallel Processing Issues in Mobile Computing, Guest Editors: A. Zomaya and M. Kumar, to appear in 2002.

[15] F.R.K. Chung: Spectral Graph Theory. Regional conference series in mathematics, ISSN 0160-7642; no. 92. CBMS Conference on Recent Advances in Spectral Graph Theory, California State University at Fesno, 1994. ISBN 0-8218-0315-8

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

[17] 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* – STOC'00, pp. 369–378, 1990.

[18] 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.

[19] R. Durret: *Probability: Theory and Examples.* Wadsworth, 1991.

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

[21] 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.

[22] 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.

[23] 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 ACM Symposium on Parallel Algorithms and Architectures* – SPAA'99, 1999.

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

[25] T. Imielinski and H.F. Korth: *Mobile Computing.* Kluwer Academic Publishers, 1996.

[26] R. Iyer, D. Karger, H. Rahul, and M. Thorup. An experimental study of poly-logarithmic fully-dynamic connectivity algorithms. In *Proc. 2nd Workshop on Algorithm Engineering and Experiments* – ALENEX'00, pp. 59–78, 2000.

[27] 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.

[28] 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.

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

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

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

[32] R. Motwani and P. Raghavan: *Randomized Algorithms*. Cambridge University Press, 1995.

[33] B. Parkinson and S. Gilbert: NAVSTAR: global positioning system ten years later In *Proc. IEEE*, pp. 1177-1186, 1983.

[34] 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, October, 1999.

[35] C.E. Perkins: *Ad Hoc Networking*. Addison-Wesley, Boston, USA, January, 2001.

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

[37] 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.

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

[39] 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.

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

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