Alcom-FT Technical Report Series
ALCOMFT-TR-02-73

An Adaptive Compulsory Protocol for Basic Communication in Highly Changing Ad-hoc Mobile Networks *

Ioannis Chatzigiannakis and Sotiris Nikoletseas
1. Computer Technology Institute
P.O. Box 1122, 26221 Patras, Greece
{ichatz,nikole}@cti.gr
2. Computer Engineering and Informatics Department
University of Patras, 26500 Patras, Greece

Abstract

In this paper we study the problem of basic communication in ad-hoc mobile networks where the deployment area changes in a highly dynamic way and is unknown. We call such networks highly changing *ad-hoc mobile networks.*

For such networks we investigate an efficient communication protocol *which extends the idea (introduced in [4, 6]) of exploiting the co-ordinated motion of a* small part *of an ad-hoc mobile network (the "runners support") to achieve very fast communication between any two mobile users of the network. The basic idea of the new protocol presented here is, instead of using a fixed sized support for the whole duration of the protocol, to employ a support of some initial (small) size which adapts (given some time which can be made fast enough) to the actual levels of traffic and the (unknown and possibly rapidly changing) network area by changing its size in order to converge to an optimal size, thus satisfying certain Quality of Service criteria.*

We provide here some proofs of correctness and fault tolerance of this adaptive approach and we also provide analytical results using Markov Chains and random walk techniques to show that such an adaptive approach is, for this class of ad-hoc mobile networks, significantly more efficient *than a simple non-adaptive implementation of the basic "runners support" idea.*

1 Introduction and Previous Research

The problem of communication among remote mobile hosts is one of the most fundamental problems in ad-hoc mobile networks and is at the core of many algorithms, such as for counting the number of hosts, electing a leader, data processing etc. In ad-hoc mobile networks, a node MH_S that wants to communicate with MH_R might not be within communication range but could communicate if other hosts lying "in-between" them are willing to forward packets for them.

The simplest way to establish communication in such networks is to perform flooding. By flooding we mean a distributed process of broadcasting a packet to all nodes within the transmission radius of the initiating sender and so on, until all nodes in the sender's connected component receive the packet. It is evident that such an approach will waste a large portion of the available bandwidth and thus is inefficient.

1.1 Limitations of Path Construction and Maintenance

In an attempt to overcome the inefficiency of flooding, some algorithms have been presented that try to construct a connectivity related dynamic data structure (such as a dynamic-graph) that represents the topology of the network by a series of message passes. By taking advantage of such a data structure, messages can be transmitted over a limited number of intermediate hosts that interconnect the MH_S with the MH_R, also known as *routing paths* or *routes*.

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. In such a model the motion of the mobile hosts is implied and is used, for example, in [15, 11, 16] etc.

Researchers have proposed *link-state* protocols, so named because each node gathers information about the

* This work was partially supported by the EU projects IST FET-OPEN ALCOM-FT and IMPROVING RTN ARACNE.

state of the links that have been established between the other nodes in the network. A shortest path algorithm can then be used to construct routes as needed between sources and destinations for which the link-state information is available. Another approach is that of the *distance-vector* algorithms, in which the algorithms provide some metric for a destination, which is often just the *distance* from the node to the destination, as well as the next hop (or *vector*) towards the destination. Other routing protocols have been adapted for use with ad hoc networks, notably *source-routing* (see [16]) and *link-reversal* algorithms (see [16]).

If all mobile hosts are static (for sufficiently long periods) or quasi-static (they move very slowly), establishing communication with such a routing algorithm is straight forward as long as MH_S and MH_R are within the same connected component. Static ad-hoc mobile networks are studied, for example, in [2].

However in the non-trivial case in which it is possible for a mobile host to move and change its position rapidly, some links will "fail" in the sense that some nodes will get "out of range" while it is possible for some other links to "form" as nodes will get "within range". Such an event can also occur if we consider the case of a node disconnection due to power shortage.

Thus, any protocol can be overwhelmed by link failures/formations that are too frequent or too fast. If the *rate* of topological change is very high, then such algorithms (i.e. algorithms that try to maintain data structures based on connectivity) fail to react fast enough and [16] 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, e.g. [2, 11, 15, 16].

In recent related work, [8, 12] study the capacity of static ad hoc networks and provide estimates on the per node capacity expected. Radios that are sufficiently distant can transmit concurrently; the total amount of data that can be simultaneously transmitted for one hop increases linearly with the total area of the ad hoc network. If the node density is constant, this means that the total one-hop capacity is $\mathcal{O}(n)$, where n is the total number of nodes. However, as the network grows larger, the number of hops between each source and destination may also grow larger, depending on communication patterns. One might expect the average path length to grow with the spatial diameter of the network, or equivalently the square root of the area, or $\mathcal{O}(\sqrt{n})$. Based on this assumption, the total end-to-end capacity is roughly $\mathcal{O}(\frac{n}{\sqrt{n}})$, and the end-to-end throughput available to each node is $\mathcal{O}(\frac{1}{\sqrt{n}})$.

The analysis provided in [8, 12] indicates that in this approach (of using a path of intermediate hosts to propagate messages) capacity is also a limiting factor; it is not encouraging that the throughput available to each node approaches zero as the number of nodes increases. These fundamental results suggest that this approach of achieving communication in ad hoc networks is inherently non-scalable.

1.2 The "Support" approach

In contrast to all such methods, in [5, 6] a new protocol framework is introduced for protocols (distributed algorithms) designed for 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). The idea of the approach is to take advantage of the mobile hosts natural movement by exchanging information whenever mobile hosts meet incidentally. Specifically, the protocol framework proposes the idea of forcing a small subset of the deployed hosts to move as per the needs of the protocol. This set of hosts is called the "support" of the network (denoted by Σ). Assuming the availability of such hosts, the designer can use them suitably *by specifying their motion* in certain times that the algorithm dictates. Thus, even if the users are spread in remote areas and they do not move beyond these areas, it is now possible for information to reach them, as the protocol takes special care of such situations.

This approach may be useful in cases of rapid deployment of a number of mobile hosts, where 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 [17] that studies 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.

Definition 1 *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* $\mathcal{P}$.

The scheme follows the general design principle of mobile networks (with a fixed subnetwork however) called the "two tier principle" [10], 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. The 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.

In a recent paper [13], Q.Li and D.Rus present a model which has some similarities to this approach. 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. In their setting [13] show optimality of message transmission times.

The "Snake" support management subprotocol M_Σ^S. The first semi-compulsory protocol using the support idea for the basic communication problem was studied in [3, 5, 6]. It uses a snake-like sequence of support stations that always remain pairwise adjacent and move in a way determined by the snake's head. The head moves by executing a random walk over the area covered by the network. We shall refer to this protocol as the *snake protocol*. In this protocol, the support Σ plays the role of a (moving) backbone subnetwork (of a "fixed" structure, guaranteed by the motion subprotocol P_1), through which all communication is routed.

The snake protocol is theoretically analyzed in the average case in [3, 6] by using interesting properties of random walks and their meeting times showing upper bounds for the average message delay or communication time of the snake protocol. The more general and realistic case is studied in [5, 6] where the efficiency of the protocol is theoretically analyzed for the arbitrary of motions where the hosts not in the support may move arbitrarily (without however trying to deliberately avoid the support).

A first implementation of the protocol was developed and experimentally evaluated in [3] with the emphasis being on confirming the theoretical analysis. Both the experiments and the theoretical analysis indicated that only a small support is needed for efficient communication. More details for the snake protocol can be found in [3]. It can be also proved (see [5, 6]) that the snake protocol is reliable and partially robust (resilient to one fault).

The HSRP support management subprotocol M_Σ^H. In [7] the *hierarchical model* of ad-hoc mobile networks is first introduced, comprised of dense subnetworks of mobile users interconnected across access ports by sparse but fast connections. In such networks, Σ is divided into a subset Σ' moving only in the upper level of the hierarchy and the hosts of $\Sigma - \Sigma'$ are split in "snakes", each randomly walking inside the lower levels of the hierarchy. 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. An alternative way ([6] to implement communication in such hierarchical networks 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. This fast backbone provides a limited number of access ports within these dense areas of mobile users.

In such hierarchical cases communication between users in different dense areas takes place in the following way: The support first gets from the sender node the messages upon meeting him and conveys these messages to the backbone system when meeting the corresponding access port. Then by exploiting the very fast communication over the backbone or the Σ', the local support in the sender's area (Σ_1) forwards the messages to some access port in the receiver area, from which subsequently the messages are picked by the local support in the receiver's area (Σ_2) and delivered to the receiver host.

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 analyze, and do not affect the correctness and the essence of this approach.

1.3 Highly Changing Ad-hoc Networks and the Need for Adaptation

In this work we consider for the first time (as far as we know) a stronger model (as implied in [15]) in the sense that the mobile hosts are now allowed to expand or shrink the area of deployment as deemed appropriate by any application(s) executed. Thus, at any time instance it is possible that the motion graph G (which combinatorially abstracts the network area, see section 3 below) will undergo certain changes by either adding or removing one or more vertices or by adding or removing one or more edges from it.

We note that such changes in the size of the motion graph capture in fact a wide variety of possible changes. Examples of such changes may be the introduction of possible obstacles as part of the area covered by the ad-hoc network or hosts' movement that congest certain areas. We here focus on the case where such changes have a *significant* effect on the size and the shape of the network's area. As an example showing the dramatic impact on the size of the area of deployment of such changes, remark that even a single connectivity change may cut off entire parts of the network area, possibly in a permanent way. Thus, any protocol used for basic communication for highly changing motion graphs should be capable of sensing and handling the formation of new components as well as the merging of two or more components, even in the case where the network size, its shape and the changes it is undergoing are *not known* to the protocol.

We remark that in such *highly changing* networks, a "static" implementation of the support approach is not satisfactory. Indeed, remark that even an initially optimal support implementation (by choosing an optimal, as possibly implied by the analysis, number of support hosts) may soon become *inefficient*, leading to big communication times (in the case where the network area grows significantly and/or obstacles appear) or employing an unnecessarily high number of support hosts (in the case where the network's area size shrinks).

1.4 Our Results

In this paper we present the design and analyze the performance of compulsory protocols for the basic communication problem in the case of highly changing ad-hoc mobile networks. We provide an adaptive algorithm extending the idea in [4, 6] where the members of Σ move like "runners", i.e., they move independently of each other sweeping the whole area covered by the network. The protocol evolves in phases of possible adaptation. In the beginning of each such phase the protocol tries to sense the need or not for an adaptation. This is done *without assuming knowledge of the network size* at any stage, since the need for an adaptation is *sensed implicitly*, by taking related measurements, such as average end-to-end delays. Based on these measurements, the protocol adapts to the (implied) network changes by accordingly changing (increasing or decreasing) the support size.

As a result, we remark that this adaptive approach *significantly outperforms* the static implementation of the runners protocol in highly changing ad-hoc mobile networks. We also estimate, by analytic means, the expected adaptation speed (i.e. the time needed for the protocol to get close to an optimal support size, as the network changes).

2 The "Runners" Support Management Subprotocol M_Σ^R

A first version of the "runners" protocol is presented in [4, 6]. Here, we first present in a more formal, justified and complete way this approach. We also discuss here mechanisms to reduce the memory requirements for the support's members. Finally, we present a limited flooding heuristic implementation of the runners support, that further reduces average message delays without having high overall communication cost in terms of additional number of message transmissions needed.

The runners support approach implements M_Σ by allowing 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.

Theorem 1 *(***Correctness***) Assuming that the motions of the mobile hosts of the networks which are not members of Σ are independent of the motion of the runners, the runners protocol is correct.*

Proof: Under this independence assumption, describing most reasonable networks, any particular mobile host will eventually meet some node of the support with probability 1. In fact, and by using the Borel-Cantelli Lemmas for infinite sequences of trials, given an unbounded period of (global) time (not necessarily known to the mobile stations) each station will meet the support *infinitely often* with probability 1 (since the events of meeting the support are mutually independent and the sum of their probabilities diverges). This guarantees correct delivery of a message onto the support Σ and, then, correct reception by a destination node when it subsequently meets the support. □

The key observation justifying this approach (and maybe its superiority, with respect to performance, compared to the "snake" approach) is that each runner will meet each other in parallel, thus accelerating the spread of information. This observation has been already reported (in the setting of a leader election problem) in [9, 6]. In [4] we performed a comparative experimental study of the "snake and the "runners protocols. These experiments showed that the "runners protocol outperforms the "snake protocol in almost all types of inputs (random, structured graphs) we considered. More precisely, the "runners protocol achieves a better average message delay in all test inputs considered.

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 an 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 sub-protocol P_3^R is partially based on the *two-phase commit* algorithm as presented in [14] 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.

Fault Tolerance of M_Σ^R. This management subprotocol provides improved reliability compared to the snake protocol in the sense that it is resilient to t faults, where $t < k$. However, note that messages may have to be re-transmitted in the case that only one copy of them exists when the faults occur.

Theorem 2 *(***Fault-Tolerance***) The support management subprotocol M_Σ^A is t-fault tolerant, where $t < k$ and k the size of the support. Hosts need to re-transmit messages until a receipt of delivery is received by a member of Σ.*

Proof: Let's assume that a sender node MH_S transmits some messages to the first member of Σ that it encounters and let's assume further that this member of Σ is distant from the rest of the support. Thus copies of the original messages are only stored on this runner and they will not propagate within the support structure for some time τ. Then in such a worst case, if a fault occurs on this member during this period τ, the only copies of the messages will be lost and they will not be delivered to their final destination.

Thus message retransmission until finally a receipt of delivery is received by a member of Σ is needed and it ensures the fault-tolerance of the protocol. □

3 The Adaptive "Runners" Support Management Subprotocol M_Σ^{AR}

We use the graph theoretic model introduced in [9] and also used in [3, 4, 5, 6, 7] that maps the movement of the mobile users in the three-dimensional space S to a so-called *motion graph* $G = (V, E)$ (its precise definition will be given shortly). Let $|V| = n$. The environment where the stations move (in three-dimensional space with possible obstacles) is abstracted by a graph by neglecting the detailed geometric characteristics of the motion. We first assume that each mobile host has a transmission range represented by a sphere tr having the mobile host as its center. 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 integral value 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 S, S is quantized into cubes of volume $\mathcal{V}(tc)$.

The *motion graph* $G = (V, E)$, which corresponds to the above quantization of S, is constructed in the following way. A vertex $u \in V$ represents a cube of volume $\mathcal{V}(tc)$. Two vertices u and v are connected by an edge $(u, v) \in E$ if the corresponding cubes are adjacent. Let $\mathcal{V}(S)$ be the volume of space S. Clearly, $n = \mathcal{O}\big(\mathcal{V}(S)/\mathcal{V}(tc)\big)$ and consequently n is a rough approximation of the ratio $\mathcal{V}(S)/\mathcal{V}(tr)$. Let us call the ratio $\frac{\mathcal{V}(S)}{\mathcal{V}(tr)}$ by the term *relative motion space size* and denote it by ρ.

The semi-compulsory and compulsory protocols studied in [3, 4, 5, 6, 7, 9] assume that the area of deployment (the *motion graph*) remains fixed through out the execution of the protocol, i.e. the protocol execution and the performance analysis provided assume a fixed graph G. Also, selection of the optimal support size implied by the analysis, assumes that the network area size (i.e. n) is known in advance.

The *adaptive-compulsory version* of the runners approach for communication in ad-hoc mobile networks is motivated by the fact that in many cases such networks undergo a variety of *significant changes*. Additionally, these changes are unpredictable and *not known* in advance and happen in an area of deployment which is not fixed. In this sense, these extra assumptions further strengthen the existing model of ad-hoc mobile networks.

Examples of such changes in the network's area topology can result by introducing possible obstacles or by creating new paths for hosts' movement. These changes may have significant effect on the size and the shape of the network's area. As an example showing the dramatic impact on the size of the area of deployment of such changes, remark that even a single connectivity change may cut off entire parts of the network area in a permanent way.

In this work, we study such changes by basically using the number of vertices n in the motion graph, modeling adjacency (and thus possible motions) in the quantized network area. We expect that future work will also introduce geometry details in the modeling and the analysis.

We remark that in such *highly changing* networks, a "static" implementation of the runners approach is not satisfactory, i.e. having a fixed k will not suffice. Indeed, remark that even an initially optimal runners implementation (by choosing an optimal, as possibly implied by the analysis, number of runners) may soon become *inefficient*, leading to big communication times (in the case where the network area grows significantly and/or obstacles appear) or employing an unnecessarily high number of runners (in the case where the network's area size shrinks). This situation is further complicated when we assume (as is often the case in practice) that the size of the area covered by the network (and thus the optimal support size implied by analytical considerations) is not known in advance.

The reasons above call for an adaptive extension of the support management subprotocol.

In this work, we present such an adaptive approach by incorporating the protocol P_{ADAPT} in the "runners" support management subprotocol. Our basic idea is as follows: The protocol P_{ADAPT} evolves in phases of possible adaptation. In the beginning of each such phase the protocol tries to sense the need or not for an adaptation. This is done without assuming knowledge of the network size at any stage, since the need for an adaptation is sensed implicitly, by taking related measurements, such as communication times. Let the time needed to measure the performance of the network and sense the adaptation need be t_{meas}. When such measurements indicate a certain behavior (i.e. communication times that progressively become significantly bigger or smaller, respectively), the protocol adapts to the (implied) new network size by respectively changing the support size (increasing or reducing it). The adaptation is done progressively, by adding (or removing) support members in each step of the adaptation procedure in a geometrically growing fashion. This progressive adaptation allows to sense reaching a new optimal support size, when the changes in the measured communication times exhibit a certain limiting behavior, in the sense that further change does not incur a significant effect on communication times anymore. Let t_{steps} be the number of adaptation steps. Then the overall time to adapt is

$$T_{ADAPT} = t_{steps} \cdot (t_{meas} + t_{change\_size})$$

where t_{change_size} is the time needed to increase or decrease the size of the support.

Note that previous related research, both analytical and experimental, on the performance of the support approach indeed reveals such a threshold behavior for the support size and its effect on communication times. However, since the network size at any time is not known to the protocol, we manage to have fast enough adaptation, by selecting a modification factor for the support size which is related to the amount of implicitly sensed change in the network. Remark that this "sensitivity factor" for adaptation can be chosen by the protocol's implementer.

Remarks on Termination. Termination of a distributed protocol is a stable property that once it occurs, it persists forever ([14]). Since P_{ADAPT} is required to sense the need for adaptation during the whole evolution of the ad-hoc mobile network, it is executed continuously as an underlying process (i.e. as a daemon) and in this sense, the protocol does not terminate but runs forever.

However, in the case that the network size and shape eventually become constant (i.e. the performance of the network remains constant) P_{ADAPT} will enter a stable state during which no activity (due to P_{ADAPT}) will take place. Note that the time to enter this "stable" state of inactivity is based on the "sensitivity factor" for adaptation.

Thus using a similar way of thinking as in [16], if all changes to the network shape and size *seize at some time*, then P_{ADAPT} will eventually reach a stable state of *inactivity* and remain in such a state for as long as the network shape and size remain fixed.

Fault Tolerance of M_{Σ}^{AR}. We remark that the adaptive runners protocol further improves the reliability of the support. In the case where t runners fail, there will be a decrease on the performance of the network. Therefore, during the next adaptation phase, the protocol will sense this based on the new performance measures and thus it will add at least t new runners provided that t hosts are available (or at least 1, in the case of at least 1 non-faulty host). Therefore, the adaptive version of the runners protocol can now tolerate an increased number of faults, which is now limited only by the total number of users that participate in the network.

Theorem 3 *(***Fault-Tolerance***) The support management subprotocol M_{Σ}^{AR} is t-fault tolerant, where $t < n$ and n the* *total number of users of the network, and at least 1 "runner" is non-faulty at any time instance. Hosts need to re-transmit messages until a receipt of delivery is received by a member of* Σ*.*

Proof: Let's assume that at a particular time instance, $t < n$ runners participate in the support and all t fail simultaneously. Then, the P_{ADAPT} protocol will seize to run and therefore the degradation in the performance of the network will not be sensed. Thus, in the case where no host in the support survives, the protocol is not t-fault tolerant.

Now, let's assume that only $t-1$ runners fail of the originally available t and that the P_{ADAPT} protocol is still active due to this 1 non-faulty "runner". Therefore, in the next adaptation phase, the degradation in network performance (due to the failure of the $t-1$ "runners") will be sensed and the non-faulty "runner" will increase the size of the support. The increase in the size of the support will be limited by the total number of users n available, even if more "runners" are requested.

The proof of the need for re-transition of messages is similar to that of theorem 2. □

Remarks on Compulsory Classification. At any time instance only a sub-set of the users will be controlled by the support management subprotocol. However, based on the assumption that the network area is changing in size and shape, some mobile hosts that are not part of the support will need to join it while others that are already part of the support will abandon it. Given a sufficiently long period of time, it is possible that all mobile hosts will be part of the support and thus it is compulsory, in the sense that it may affect the movement of all hosts (see [6] for more details on protocol classification).

3.1 QoS Guarantees

In order to provide to the users of the highly changing network certain Quality of Service (QoS) guarantees, we study in this work two important performance measures: the average message delay and the message delivery rate. Note that the average message delay is more interesting from each specific user's point of view, while the delivery rate provides important information to the network administrator for the overall network performance.

Measuring the Avg. Message Delay. In order for the members of the support to measure the average message delay to sense the changes imposed on the shape and size of the network, certain modifications need to be made on the protocol. A possible way to achieve this is by using a new synchronization subprotocol P_3^{AR} that extends the original P_3^R by providing measures on the performance of the network.

In addition to the information already stored by the members of the support, a performance measure on the current delivery delay is attached to each still en route message by indicating the overall time period required for the message's travel time. When the message is delivered, this information measures the total communication time experienced by this message. In other words, the set S_2 is modified in order to store the unique ID of a message, the sender ID and the performance measure.

Thus, by exchanging information on their S_1, S_2 sets, the measurements on performance are now made common to all runners. However, it is crucial to make some further assumptions so that P_3^{AR} is correct. Therefore, we assume that all members of Σ have precise clocks although it is not necessary for their clocks to be synchronized. In other words, the protocol will not work correctly if some of the clocks are drifting, as the clocks must measure the same time interval for the same period of time.

Measuring the Delivery Rate. Measuring the delivery rate is simpler and in contrast to measuring the average message delay, no modifications need to be made on P_3^R. The original synchronization subprotocol P_3^R already provides mechanisms to notify the member of the support on messages delivered. Thus it is very simple for the leader of Σ to calculate the delivery rate by performing a single operation on its own S_1 and S_2 sets.

3.2 The Adaptation Protocol P_{ADAPT}

We below provide a high-level description of the adaptation procedure. Because of lack of space, in the full paper (located at *http://faethon.cti.gr/adhoc/ipdps02.html*) we also provide a pseudo-code description of P_{ADAPT}.

Definition 2 *Let* $x_0, x_1, \ldots, x_i$ *be the performance measure (such as the average communication time) at the end of the step* i*, where* x_0 *is an initial value and step* i *is the current execution step.*

Definition 3 *Let* $\delta x_i = |x_i - x_{i-1}| \geq \phi \cdot x_{i-1}$ *be the "sensed" alteration in the performance measure implying corresponding changes in the network and the need for adaptation, where* ϕ *is the "sensitivity factor" of the adaptation procedure and is set by the implementer to a small fixed percentage constant (i.e.* $\phi = 0.1$*), according to the expected network evolution (i.e.* ϕ *serves as a kind of network specification).*

Using the sensitivity factor to trigger adaptation, we try to avoid non-necessary adaptation in cases of trivial changes in the network. Thus, when the sensitivity threshold $\phi \cdot x_{i-1}$ is crossed, a new adaptation phase is initiated. Figure 1 (in the full paper) describes the P_{ADAPT} procedure executed by the leader of the mobile support, where t is the t^{th} step of the P_{ADAPT} procedure and $c(\xi) = c \cdot \xi$ is the additive adaptation factor, where c a small constant used for initialization purposes and $\xi = \frac{|x_i - x_{i-1}|}{x_{i-1}}$ is a measure of the actual change in performance.

Note that ϕ can be adjusted appropriately to affect the end of procedure P_{ADAPT} according to how close to the next optimal support size we wish to go.

Definition 4 *Let t_{sense} be the last i such that $\delta x_i \geq \phi \cdot x_{i-1}$, i.e. t_{sense} is the last step of the last adaptation.*

Thus at the end of the adaptation phase, the mobile host stores $x_{t_{sense}}$ for further use, and for any sensing of subsequent adaptation phases the following rule is used

Definition 5 *Let $\Delta x_i = |x_i - x_{t_{sense}}| \geq \phi \cdot x_{t_{sense}}$, be the* next *"sensed" alteration in the performance measure implying corresponding changes in the network and the need for adaptation since the last adaptation phase t_{sense}.*

Remark that δx_i measurers performance changes over short time intervals. To prevent our protocol from not detecting a sequence of small changes that do not cross the sensitivity factor criterion but whose cumulative effect over a long time period leads to an adaptation need, we use Δx_i to measure performance changes with respect to the performance measures taken at the end of the last adaptation.

We interestingly note that P_{ADAPT} protocol has very limited memory requirements, since we only need to maintain the values of $x_{t_{sense}}$, x_{i-1} and the counter t.

3.3 Analytic Considerations for P_{ADAPT}

Analysis of the Adaptation Speed. We below provide some analysis for the adaptation speed, i.e. for the time T_{ADAPT} needed to get close to the new optimal support size. The overall time to adapt is $T_{ADAPT} = t_{steps} \cdot (t_{meas} + t_{change\_size})$, where t_{steps} is the number of adaptation steps, t_{meas} is the time needed to measure the performance of the network (i.e. the x_i values) and t_{change_size} is the time needed to increase or decrease the size of the support, accordingly.

Let n be the number of vertices in the motion graph at the beginning of running procedure P_{ADAPT}, i.e. at time t_{sense} and let n' be the number of vertices at the end of the adaptation phase. Remark that both n and n' are not known by the protocol but implied by the performance measures taken.

Let k, k' be the "optimal" support size in each case, respectively. We have

$$\begin{aligned} k' &= k \pm c(\xi)^{t_{steps}} \\ c(\xi)^{t_{steps}} &= |k' - k| \\ t_{steps} &= \frac{\ln |k' - k|}{\ln c(\xi)} \end{aligned}$$

Note that $c(\xi) = c \cdot \xi$ where c, ξ are of constant size, thus the denominator is a relatively small constant, chosen by the implementer.

Thus, the required steps to adapt is logarithmic in the absolute difference of the corresponding support sizes. These sizes depend on the network area. In the snake implementation of the support approach, we have, roughly, that the optimal support size is $k = \Theta\big(\sqrt{n}\big)$. The analysis for the runners implementation is much harder to do. However, we expect (because of the concurrent random walks) an acceleration in expected communication times, thus a smaller support will suffice in the case of runners.

Thus the expected number of steps to adapt is upper bounded by $\Theta\big(\ln|\sqrt{n'} - \sqrt{n}|\big)$, and thus we manage to achieve an adaptation speed which is logarithmic in the network size.

Analysis of the Time to Increase the Size of Σ. As we have seen in the previous sections, based on certain quality of service guarantees and the corresponding performance measures taken, the P_{ADAPT} protocol will detect the changes on the performance of the network and decide to act by increasing or decreasing the size of the support accordingly. In particular, at the end of each adaptation phase of P_{ADAPT}, the leader of Σ will modify the size of the support based on the calculated k_{i+1}.

The support size is increased by including c more mobile hosts to it (obviously, out of those that are already runners) and similarly it is decreased by removing c runners from the support. To ensure correctness by avoiding adding more than c hosts to the support the c new runners are chosen in a centralized manner by the (unique) leader runner of the support: The first c hosts that accidentally meet with the leader runner become the c new runners. The procedure is similar in the case that we want to reduce the size of the support, however now we are interested on the first c runners that accidentally meet with the leader runner.

Note 1 *The analysis below holds only in the case where the hosts not in the support perform concurrent and independent random walks. In this case, the relative effect of the walk by the leader runner and the user's walk, can be seen as a single random walk by the leader runner onto the position of the user which is "fixed" in an average place of the motion graph (roughly, this "walk" has speed twice as big as the two original walks). This assumption allows us to investigate the average case performance of our protocol.*

We assume that the runners perform concurrent and independent random walks on the motion graph. We also assume that there is a global time t, not necessarily known to the hosts. We define the random walk of a mobile host (runner) on G that induces a continuous time Markov Chain M_G as follows: The states of M_G are the vertices of G. Let

s_t denote the state of M_G at time t. Given that s_t=u, u∈V, the probability that s_{t+dt}=v, v∈V, is $p(u,v)$·dt where

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

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

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

Let $T_i = min\{t \geq 0 : s_t = i\}$ be the first hitting time on state i (the first time that the mobile host visits vertex i).

Theorem 4 *In the case of adapting by increasing the support size, the expected time needed is:*

$$E(t_{change\_size}) \leq c \cdot \frac{1}{\lambda_2(G)} \cdot \Theta(n)$$

Proof: Let X_i be the time needed for the leader to meet a node i ($1 \leq i \leq c$) not in the support. Clearly, $t_{change\_size} = \sum_{i=1}^{c} X_i$.

The times X_i (and clearly t_{change_size} too) are random variables. By linearity of expectation, the expected time to modify the support size is

$$E(t_{change\_size}) = E\left(\sum_{i=1}^{c} X_i\right) = \sum_{i=1}^{c} E(X_i)$$

Now remark that because c is a small constant (corresponding to a very small part of hosts in the network) the times X_i are roughly statistically identical and thus, $E(t_{change\_size}) = c \cdot E(X_i)$

Furthermore, note that time X_i is in fact a first hitting time of the random walk performed by the leader runner u onto a node σ not in the support, according to the stationary distribution π of u's walk on graph G. Thus $E(X_i) = E_\pi T_\sigma^G$.

Note 2 *Since the motion graph G is finite and connected, the continuous time Markov Chain abstracting the random walk on it is automatically time reversible.*

Recall from [1] that,

Lemma 1 *[1] For any node σ of any graph G in a continuous-time random walk*

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

where π_σ is the (stationary) probability of the walk at node (state) σ and τ_2 is the relaxation time of the walk.

Note 3 *In the above bound, τ_2=$\frac{1}{\lambda_2}$ where λ_2 is the second eigenvalue of the motion graph G. In fact, λ_2 is an indication of the* expansion *of G and of the asymptotic rate of convergence to the stationary distribution, while relaxation time τ_2 is the corresponding interpretation measuring time.*

Note 4 *The above expected time is needed to hit* any node *σ of the graph. Thus, it really represents a "worst case", where all hosts not in the support are moving (and even stop there) in network areas far way from the leader runner.*

It is a well-known fact (see e.g. [1]) that $\forall v \in V_G$, $\pi_v = \frac{d(v)}{2m}$ where $m = |E_G|$ is the number of the edges of G and $d(v)$ is the degree of v in G. Thus $\pi_\sigma = \frac{d(\sigma)}{2m}$. In our case, $m \leq 6n$ (because we quantize the space into cubes and because of possible obstacles) and $d(\sigma) \leq 6$ (for the same reasons). Thus $\pi_\sigma = \frac{1}{n}$ and we get the theorem by plugging this value into (1). □

As already mentioned above, this is a very crude overestimation (especially in the case of networks of many users). To get a more realistic picture of the time needed, we assume a rather uniform spread of mobile hosts in the network. In such a case there is always a host within $\Theta\left(\frac{n}{h}\right)$ adjacent cubes, where h is the number of hosts. By using simple properties of random walks we can easily prove the following theorem:

Theorem 5 *Assuming uniform spread of the h hosts into the n cubes of the network area, it is*

$$E(t_{change\_size}) = c \cdot \Theta\left(\frac{n}{h}\right)$$

Analysis of the Time to Reduce the Size of Σ.

To calculate the time needed to reduce the support size, we work using similar arguments as in the case of increasing the support size, and we get the following two theorems:

Theorem 6 *In the case of adapting by decreasing the support size, the expected time is:*

$$E(t_{change\_size}) \leq c \cdot \frac{1}{\lambda_2(G)} \cdot \Theta(n)$$

Theorem 7 *By assuming uniform spread of the k runners in the support into the n cubes of the network area, we get*

$$E(t_{change\_size}) = c \cdot \Theta\left(\frac{n}{k}\right)$$

We note that the estimation in Theorem 7 above is crude. A more tight analysis is much more difficult, it may use notions and techniques from Physics (like interacting particles) and we currently work on this.

Overall P_{ADAPT} Performance. The overall expected time needed by P_{ADAPT} can be easily calculated by the above Theorems, the relation $T_{ADAPT} = t_{steps} \cdot (t_{meas} + t_{change\_size})$ and linearity of expectation.

4 Closing Remarks and Future Work

We have presented a new adaptive, compulsory protocol for a basic communication problem in a highly changing ad-hoc mobile network. We provided correctness and fault-tolerance proofs for this protocol and also investigated analytically its performance.

There are several directions for future work. An obvious one is to analyze theoretically the runners protocol by using advanced analytic techniques from Physics, such as theory of interacting particles. Another one is to enforce a specific motion pattern to the support (e.g., moving along a spanning subgraph of the motion graph) instead of doing random walks, or by forcing it to perform the random walks of the support on this spanning subgraph. Finally, we plan to implement our protocol, experimentally validate its performance and its superiority over the static implementation of the runners protocol.

References

[1] 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).

[2] 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).

[3] 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).

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

[5] 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 Brief Announcement in 20th Annual Symposium on Principles of Distributed Computing (PODC'01)(2001).

[6] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis: On the Average and Worst-case Efficiency of Some New Distributed Communication and Control Algorithms for Ad-hoc Mobile Networks. Invited Paper in Proc. 1st International Workshop on Principles of Mobile Computing, (POMC'01)(2001).

[7] 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).

[8] P. Gupta, P. R. Kumar: The Capacity of Wireless Networks. IEEE Transactions on Information Theory, 46(2):388-404, March 2000 (2000).

[9] 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).

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

[11] 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).

[12] J. Li, C. Blake, D.S.J. De Couto, H.I. Lee and R. Morris: Capacity of Ad-hoc Wireless Networks. In Proc. 7th Annual ACM/IEEE International Conference on Mobile Computing (MOBICOM'01) (2001).

[13] 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).

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

[15] 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).

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

[17] 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).