Location Awareness in Wireless Ad-Hoc Networks
(or How to RLISE a TIGR?)

Guy Korland
Location Awareness in Wireless Ad-Hoc Networks
(or How to RLISE a TIGR?)

Research Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Guy Korland

Submitted to the Senate of the Technion — Israel Institute of Technology
Shevat, 5765 Haifa February 2005
The Research Thesis Was Done Under The Supervision of Dr. Roy Friedman in the Faculty of Computer Science

I would like to thank Dr. Roy Friedman for his guidance, help and support. Roy always found time and patience to hear and direct all my disordered thoughts.

I also particularly would like to thank Gabi Kliot for his useful comments and fruitful insights, and in general to all those who spare their time for me.

This work is dedicated to my parents who have been there through all my long way.

Part of this work was funded by the Israeli Science Foundation grant number 44/03.

THE GENEROUS FINANCIAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>2</td>
</tr>
<tr>
<td>1.1 Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Contribution of this Work</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 RLISE</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 TIGR</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Dissertation Road Map</td>
<td>6</td>
</tr>
<tr>
<td>2 Related Work</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Positioning</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Routing</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Location service</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Power saving</td>
<td>10</td>
</tr>
<tr>
<td>3 RLISE: Relative Location with Incomplete Stationary Emitters</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Calculating Relative Positions</td>
<td>11</td>
</tr>
<tr>
<td>3.1.1 Brief</td>
<td>11</td>
</tr>
<tr>
<td>3.1.2 System Model Assumptions</td>
<td>11</td>
</tr>
<tr>
<td>3.1.3 The Basic Protocol for Dynamic Network</td>
<td>13</td>
</tr>
<tr>
<td>3.1.4 Optimizations</td>
<td>15</td>
</tr>
<tr>
<td>3.1.5 Unsynchronized Clocks</td>
<td>18</td>
</tr>
<tr>
<td>3.1.6 Calculating Relative Positions in Static Networks</td>
<td>19</td>
</tr>
<tr>
<td>3.1.7 Handling Distance Estimation Errors</td>
<td>20</td>
</tr>
<tr>
<td>3.1.8 Analysis and Limitations</td>
<td>21</td>
</tr>
<tr>
<td>3.1.9 Utilizing Inaccurate Location Estimates</td>
<td>22</td>
</tr>
<tr>
<td>3.2 Results</td>
<td>23</td>
</tr>
<tr>
<td>3.2.1 Performance with Exact Distance Measurements</td>
<td>24</td>
</tr>
<tr>
<td>3.2.2 The Impact of Distance Measurements Errors</td>
<td>26</td>
</tr>
</tbody>
</table>
4 Timed Grid Routing (TIGR) Bites off Energy

4.1 General concept .............................................. 28
   4.1.1 Brief .................................................. 28
   4.1.2 System Model Assumptions .......................... 28
   4.1.3 The Grid Transmission Range Inequality ........... 30
   4.1.4 Scheduling Functions ................................ 31

4.2 TIGR’s Scheduling Mechanisms ............................ 33
   4.2.1 Random \( f_n \) ........................................... 34
   4.2.2 Deterministic \( f_{(k,p,n,m)} \) ......................... 34
   4.2.3 Back-to-Back \( f_{(k,p,n,m)} \) ......................... 37
   4.2.4 Representative Backbone for Multiple Nodes in a Cell 38
   4.2.5 Synchronization ....................................... 41

4.3 Location Service ............................................ 41

4.4 Routing Algorithms ......................................... 42

4.5 Analysis .................................................... 43

5 Conclusion ..................................................... 46
   5.1 Future Work ............................................... 47

Bibliography ..................................................... 49
List of Figures

1.1 Network Model .............................................................. 5

3.1 RLISE - State Machine (not a stationary emitter) .................... 12
3.2 Relative location finding - trivial cases .............................. 14
3.3 Combining two SemiAccurate states ................................... 14
3.4 Simple new state calculations - see drawings in Figure 3.2 ....... 17
3.5 More complex state calculations - see illustrations in Figures 3.3 and 3.6 17
3.6 Combining a Circle and SemiAccurate states ....................... 18
3.7 Relative location - eliminating impossible locations using the region’s border .................................................. 18
3.8 Pathological Error Propagation .......................................... 20
3.9 Accuracy Percentage vs. Network Density ............................ 20
3.10 Convergence vs. Network Density ............................. 23
3.11 Convergence vs. Transmission Range .............................. 23
3.12 Convergence per iteration ............................................ 25
3.13 The impact of the number of initial stationary emitters on convergence . 25
3.14 The Impact of Errors on Accuracy .................................... 27
3.15 Max Accumulated Error per Iteration ................................ 27

4.1 Unit Disk ........................................................................ 29
4.2 Quasi Disk ....................................................................... 29
4.3 Dynamic grid inequality .................................................... 30
4.4 Static grid inequality ....................................................... 30
4.5 The deterministic $f_{(k,p,n,m)}$ scheduling function ................... 35
4.6 $f_{(4,4,3,3)}$ during horizontal advancement ........................... 37
4.7 B2B $f_{(4,4,3,3)}$ during vertical advancement ......................... 37
4.8 $f_{(3,3,1,1)}$ ........................................................................ 38
4.9 B2B $f_{(2,2,1,1)}$ .............................................................. 38
4.10 The Back-to-Back scheduling function ............................... 39
4.11 Time line ......................................................... 40
4.12 Location Service with one direction .......................... 42
4.13 Location Service with two directions ......................... 42
4.14 Network Utilization vs. Energy Consumption ............... 44
Abstract

This work presents two loosely coupled results related to ad-hoc networks. The first result is a novel method for obtaining local positioning information, whereas the second result utilizes positioning information to devise transmission schedules that eliminate collisions and reduce energy consumption.

More specifically, position information of nodes in ad hoc networks is useful for routing, efficient service and various applications. Global location systems enable mobile nodes to discover their positions in a global coordinate system using stationary emitters. The efficacy of these systems is a function of the density of mobile nodes deployment and of the stationary emitters. In the first part of this dissertation, we explore the idea of using techniques to estimate nodes coordinates in two (or three) dimensional space, and we propose a distributed positioning method (nicknamed RLISE) based on cooperative position sharing between the mobile devices, enabling great reduction in stationary emitters deployment needed by former methods.

Energy efficiency and collisions avoidance are both critical properties to increase the lifetime and effectiveness of wireless networks. The second part of this dissertation proposes a family of protocols (nicknamed TIGR) for reducing both energy consumption and packets collisions in ad-hoc networks. In particular, this family of protocols offers a tradeoff between bandwidth utilization and power consumption. The proposed algorithms are based on geographic knowledge to form a virtual grid and on synchronized clocks in order to achieve a collision free locally computable transmission schedule. As a side effect of the above a new efficient location service that is adjusted to a grid oriented geo-routing is also presented.
Chapter 1

Introduction

Ad-hoc networking technology enables users to spontaneously form a dynamic wireless communication network [31]. Thus, an ad-hoc network can be used to provide devices with continuous network connection, even when a device is out of an infrastructure based network scope. The two most common examples of ad-hoc networks include Mobile Ad-Hoc Networks [37] (MANET) and sensor networks [32]. Examples of the possible uses of ad-hoc networks include, e.g., soldiers in a battlefield, emergency disaster relief personnel, networks of laptops, seismic sensor networks, and floor control and monitoring networks.

1.1 Objectives

Positioning: As been demonstrated, various basic services and applications can greatly benefit from positioning knowledge. For example, it was shown that geographical based routing can be efficient and scalable, e.g., in [18, 19, 39]. Moreover, in sensor networks, knowing the location of a node is vital for being able to correctly analyze and respond to sensors’ readings. There are several existing techniques that enable nodes to learn their position. The simplest approach is to equip every node with a GPS receiver. However, this option is relatively expensive, it is not always available (for instance, most laptops and PDAs sold today are still not equipped with GPS), and GPS requires line of sight, which means that it does not always work in an urban environment, and in particular inside buildings.

Another known option is to assume the existence of several base stations, also known
as *stationary emitters* or *landmarks*, who know their exact position and from which other nodes can learn their own location by using triangulation principles. These schemes, however, assume that each node is covered by the transmission range of at least three stationary emitters. This requires a large number of stationary emitters and is also sensitive to interference caused by concrete walls etc.

In order to overcome the above obstacles, several *ad-hoc positioning systems* (APS) have been proposed, e.g., [4, 12, 14, 16, 17, 21, 26, 27, 29, 33]. These schemes can be largely categorized as *DV-hop propagation* methods, *Euclidean and DV-radial* methods, and *DV-position* methods. In the DV-hop propagation method, it is assumed that each node knows the shortest path to the landmark nodes (or stationary emitters), which know their position. The landmarks publish the estimated physical distance for each hop, which allows nodes to learn their estimated distance from each landmark, thereby also compute their estimated location. In the Euclidean and DV-radial methods, nodes are assumed to be able to measure their distance or angle from landmark nodes. Once they have this information, they can use triangulation and trilateration techniques to compute their own location. Finally, in DV-position methods nodes combine both distance and angle measurements for improved accuracy. Notice, though, that in these methods, a node can only learn its location if it is within the transmission range of three other nodes that already know their location.

**Energy conservation:** There are several challenges in devising a practical ad-hoc network [37]. For example, in these networks, typically nodes are battery operated. Hence, saving energy is vital for increasing the life-time of the network. In particular, in most networking cards, being in an active listening mode (or idle) consumes roughly the same amount of energy as receiving or even sending messages. Thus, the most effective way of saving energy is by allowing most nodes to sleep. The IEEE 802.11 standard, also known as WiFi, defines a power save mode (PSM) to allow nodes that do not expect to receive any message in a given time slot to sleep throughout the time slot. However, PSM is only defined in a Wireless LAN, and does not allow any node to sleep when messages are multicasted.

**Message collision:** Message collision is another serious concern in ad-hoc networks as it wastes network bandwidth. It increases the possibility of message loss and hurts the performance. Moreover, retransmitting messages consumes energy. To reduce collisions,
WiFi uses a collision detection and avoidance mechanisms similar to Ethernet. However, wireless networks suffer from the *hidden terminal problem*. That is, suppose nodes A and B wish to send a message to C such that C is within the transmission range of both A and B but A and B are too far apart to notice each other’s signal. In this case, the collision will occur in C, yet A and B will not be aware of it. Here WiFi proposes the use of a RTS/CTS mechanism, but again, this mechanism is only defined for point-to-point messages sent within a wireless LAN.

### 1.2 Contribution of this Work

As mentioned before, in this dissertation we present two loosely coupled results related to ad-hoc networks. The first result is a novel method for obtaining local positioning information, whereas the second result utilizes positioning information to devise transmission schedules that eliminate collisions and reduce energy consumption.

The results are loosely coupled in the sense that, as discussed before, positioning information has many applications beyond the scheduling developed in the second part. Similarly, there are many ways of obtaining local positioning knowledge, and thus our second result can utilize any of the existing methods as well. We elaborate on these two main results below.

#### 1.2.1 RLISE

In the first part of this dissertation we propose a novel method for achieving positioning knowledge based on distance measurements. In our scheme, nicknamed *RLISE*, we allow for a gradual iterative learning process in which nodes can eventually learn their location even if no node is initially placed within the transmission range of three stationary emitters (or landmarks). Thus, this enables positioning with much fewer initial landmarks.

As in existing techniques, *RLISE* is based on simple geometric principles, but it also employs partial location gossiping in order to enhance each node’s perception of its own location in an iterative process. Thus, nodes can learn their position much faster than in the Euclidean methods, and moreover, there are situations in which the Euclidean method does not converge while our method does. The distance measurement capabilities we require can be obtained by employing *Time of Arrival* (ToA) [21] and *Received Signal*
Strength Indicator (RSS) [4] techniques. Thus, RLISE does not require any special hardware.

Also, we would like to stress that all computations are local, and do not rely on any central entity. Another feature of RLISE is that when the network connectivity is not sufficient, nodes can still maintain partial information on where they might be. For example, for a while, a node might only be able to tell that it is located somewhere on a circle, or that it is definitely in one of two possible locations. This information might still be useful for many applications, and in particular, once the network connectivity improves, it allows for quick learning of the exact location. These aspects are elaborated later in this dissertation.

The dissertation also includes an analysis of our protocol, RLISE. In particular, we discuss in detail the benefits and the limitations of RLISE, and present extensive simulations. In these simulations we explore the performance of RLISE vs. the number of stationary emitters, the density of the network, the impact of mobility, and its sensitivity to errors in distance measurements. These measurements generally confirm our assumptions about RLISE’s behavior.

1.2.2 TIGR

In the second part of this dissertation we propose a new scheme that both allows many nodes in the system to sleep for large fraction of the time and eliminates collisions, including in multiple hop networks and when messages are multicast. Our scheme assumes
that nodes know their locations and have synchronized clocks. With these assumptions, we divide the network area into a logical grid, based on geographic location. Given this grid, we then devise our family of protocols. Each protocol defines a scheduling mechanism that assigns for each cell in the grid in which time slots it should send its messages and in which time slots it should be willing to receive messages. We then devise locally computable routing protocols for these schedulers, and explore the impact of the various scheduling mechanisms on power saving, latency, and throughput.

Finally, we introduce a new efficient location service aimed at such grid oriented geo-routing. The location service utilizes the observation that all that a node needs to know in order to route a message on a grid is the direction of the target node (up/down/right/left). Based on this observation, the location service can greatly reduce the number of messages that a node needs to send when moving from one grid cell to another. Moreover, with our location service, the lookup is always served locally. This enables the routing protocol to adapt its routing to movement at no additional cost.

Figure 1.1 summaries our work as an extend version of the network layers model. The first column stands for the standard network layers while the other modules represent our work. The Geo-Routing (Chapter 4), the Location service (Chapter 4) and the Positioning (Chapter 3) modules provide a complete routing service for the Network/IP layer, based on geographic knowledge. The Scheduler (Chapter 3) provides an extension for the Mac layer providing message collision avoidance and energy conservation.

1.3 Dissertation Road Map

The rest of this dissertation is organized as follows. Chapter 2 summarizes the relevant related work. In Chapter 3 we describe and analyze the RLISE algorithm. The TIGR scheme is introduced in Chapter 4. Chapter 5 concludes this dissertation.
Chapter 2

Related Work

2.1 Positioning

There are several methods for obtaining geographic localization. The Global Positioning System (GPS) [15] is probably the most commonly used for outdoor localization. GPS relies on satellites to achieve triangulation. Yet, line of sight access to satellites is often not available indoors, which largely limits the use of GPS to being outside.

Several attempts to mimic the use of satellites indoor were made using full cover of stationary antennas. For example, the RADAR system [4] is a radio-frequency (RF) based system for locating and tracking users inside buildings. RADAR uses multiple base stations positioned to provide overlapping coverage in the area of interest, using received radio signal strength (RSS) to determine an object’s location. Another example is the Location Estimation Assisted by Stationary Emitters (LEASE) [21] system. LEASE is composed of three different stationary devices that should be deployed across the building. In LEASE, all positioning computations are done by a global server.

There are also several works on learning location information using cooperative sharing between mobile nodes. For example, Savarese, Rabaey and Beutel proposed the Assumption Base Coordinates (ABC) algorithm to calculate relative positions of nodes based on range measurements between the nodes in [33]. This algorithm determines the location of unknown nodes one at a time in the order they establish communication. Yet, this approach makes an assumption that all nodes are aware of each other, and therefore can apply messages in a specific order.

A variation of the ABC protocol called TERRAIN is also presented in [33]. The
main differences between TERRAIN and RLISE is that in TERRAIN the computation cannot proceed with partial information. Also, in TERRAIN, anchor nodes start calculating their locations in semi-virtual coordinates. When the computation discovers another coordinate system, a correction phase is started until eventually the system converges to a single coordinate system.

SpotON [14] provides absolute and relative location capability for sensors networks using tags based on RSS. However, in SpotON, all computations are done by a global server.

Another method that helps determine nodes positions is the Angle of Arrival (AOA), which is used in several Ad Hoc Positioning System (APS) [39, 26]. These protocols offer a hybrid mixture of two major concepts: distance vector (DV) routing, and beacon based positioning (GPS). A mechanism for positioning learning that is based on region intersections was proposed in [11]. By this scheme, nodes maintain an estimate on the maximal transmission range, and the region(s) it might be in. Nodes exchange this information, and when a node receives such a message, it calculates the intersection between the sender’s regions and it’s own, based on the maximal transmission range. This method also allows for gradual learning of the location, with improved accuracy as more messages are received. On the other hand, it is sensitive to errors in the assumed maximal transmission range.

2.2 Routing

A good survey of geographic ad-hoc routing protocols appears in [25]. Three link reversal routing algorithms are analyzed with respect to how the algorithms behave on different classes of networks in [34]. Greedy geographic routing algorithms were studied in [40] in an important class of wireless sensor networks that provide sensing coverage over a geographic area (e.g., surveillance or object tracking systems). That paper demonstrates that existing greedy geographic routing algorithms can successfully find short routing paths based on local states in sensing-covered networks. They have also proposed a new greedy geographic routing algorithm called Bounded Voronoi Greedy Forwarding (BVGF).

A solution for two fundamental problems of geographic routing is presented in [10]. Specifically, geographic routing requires that all nodes know their locations, and it has
trouble routing around local dead-ends. The solution offered in [10] is based on utilizing a nearby location aware node that acts as a proxy for geographic forwarding.

The Location-Aided Routing (LAR) route discovery algorithm is based on limited flooding in a certain angle [20], which greatly reduces the number of nodes to whom a route request is propagated. Rooftop networks [6] is a good example of static ad hoc networks. In such networks, the position of a node may not change once it has become part of the network.

Many proactive unicast routing protocols have been studied in the literature. Examples include Distance-Vector routing (DSDV) [30], Optimized Link State Routing (OLSR) [9] and Topology Broadcast Based on Reverse-Path Forwarding (TBRPF) [28].

2.3 Location service

GLS [24] is a distributed location service that when combined with geographic forwarding allows the construction of ad hoc mobile networks that scale to a large number of nodes. Each node sends its position updates to its location servers without knowing their actual identities, assisted by a predefined ordering of node identifiers and a predefined geographic hierarchy.

In GAF [42], the sensor field is divided into small squared cells such that two nodes in two neighboring cells are connected. One node in each cell will be active while all others are in idle mode. Active nodes are supposed to communicate data among themselves. This scheme increases the lifetime of the network by allowing excessive idle of the non-active nodes. SPAN [8] is a distributed, randomized algorithm where nodes make local decisions on whether to sleep, or to join a forwarding backbone as a coordinator. With SPAN, the system lifetime increases as the ratio of idle-to-sleep energy consumption grows, and increases as the network becomes more dense.

The Virtual Grid Architecture Protocol (VGAP), introduced in [2, 3], operates on a fixed virtual rectilinear architecture (virtual grid), which is devised by using location information obtained from Global Positioning System (GPS). The virtual grid consists of a few, but possibly more powerful, mobile nodes known as ClusterHeads (CHs) that are elected periodically.
2.4 Power saving

Our work can be seen as generalizing some concepts of TDMA to multiple hop networks. Differently, some contention-free MAC protocols for sensor and ad-hoc networks are presented in [7, 13]. These protocols do not rely on global time and are self-stabilizing. This is achieved by allocating dynamically transmission time slots for each node. But these algorithms do not allocate any sleeping time, which forces each node to be in either listen or transmit state.

The concept of turning off nodes in order conserve energy is also explored in the literature. For example, PAMAS [35] presents a MAC protocol that conserves energy by turning off radios overhearing cross traffic. An Adaptive Fidelity Energy-Conservation Algorithm (AFECA) that gives a trade off between energy dissipation and data delivery quality according to application requirements is presented in [41]. Using the observation that in densely-populated network many nodes are interchangeable for routing purposes, nodes can sleep while other nodes are available. But, due to the fact that this algorithm is based on estimation on the network density, messages can get lost and retransmitting is still needed once in a while.
Chapter 3

RLISE: Relative Location with Incomplete Stationary Emitters

3.1 Calculating Relative Positions

3.1.1 Brief

In this section, we present the basic algorithm for calculating nodes’ position nicknamed RLISE. RLISE in general can be seen as a gossip based version of the well known triangulation location method with a few shortcuts added. These shortcuts help the process and improve its results and are in fact the essence of this section.

3.1.2 System Model Assumptions

For clarity of presentation, the presentation deals with two dimensions only. However, extending the protocol to three dimensions is obvious.

We assume a collection of nodes (also referred to as devices), each equipped with a wireless transmitter and receiver. We assume that nodes communicate with each other only by exchanging messages through their wireless transmitters and receivers. We refer to two nodes that are in the transmission range of each other (can successfully exchange messages without interference) as neighbors.

We further assume that when a node \( q \) receives a message from another node \( p \), then \( q \) can calculate its distance from the sender \( p \), e.g., by examining the received signal strength indicator (RSS) [4] or by Time of Arrival (TOA) [21]. In RSS, if the energy
Figure 3.1: RLISE - State Machine (not a stationary emitter)
in which a signal is sent is known, then the distance can be computed based on the energy drop between sending and receiving the event. In ToA, on the other hand, the distance is sent based on the latency of a ping message. Such capabilities are considered reasonable \[4, 21\].

Moreover, we assume that some of the nodes have precise a-priori knowledge of their geographical location. These nodes are called stationary emitters. Other nodes have no a-priori knowledge of their position. However, nodes can transmit to each other messages containing their location, if it is known, or a summary of possible locations for them. For example, \( p_i \) might know that it is somewhere on a circle, or that it is in one of two locations (but not in any other location), etc. Also, nodes can indicate an error estimate for their location.

We assume for the simplicity of the presentation the existence of a collision detection/avoidance layer, therefor we assume that messages are never lost. Notice that this assumption is not essential in most of the cases since our algorithm periodically restart the convention process (see section 3.1.3). In special cases this assumption can also be replaced by message retransmission. We elaborate on this issue below in section 3.1.6.

Also, as mentioned before, initially we assume that each node can detect its distance from a sender of a message it receives accurately (e.g. TOA [21] and RSS [4]) and the clocks of all the stationary emitters (nodes with preliminary accurate position knowledge) are tightly synchronized (we do not force any clocks’ synchronization on non stationary emitters’ nodes). Later, we relax these two assumptions and investigate the impact of this relaxation.

We do not assume anything about the nodes’ position, their distribution or their mobility, although their distribution can greatly impact the protocol results, especially when there is no full connectivity. Similarly, the nodes velocity can greatly impact RLISE results. We discuss this issue in more detail below.

### 3.1.3 The Basic Protocol for Dynamic Network

In RLISE, a node can be in one of the four states, as illustrated in the state machine diagram in Figure 3.1: Accurate, SemiAccurate, Circle, and Unknown. Accurate indicates

\[1\]

Yet, let us note that neither method is perfect. For example, the amount of energy used to send a signal varies over time in WiFi, often without advanced warning. Also, ToA’s accuracy depends on the clocks drift between nodes.
that the node knows its exact location represented by coordinates \((X,Y)\). SemiAccurate indicates that the node knows that it must be in one of two optional locations, represented by two coordinates \((X_1,Y_1)(X_2,Y_2)\). Circle indicates that the node knows that it is somewhere on a continuous circle represented by the middle coordinates and radius \((X,Y,R)\).

Each node that is not a stationary emitter starts a round in the Unknown state, which is maintained in the STATE variable, and tries to gradually upgrade its state during the round. This is facilitated by having each node maintains a variable \(cur\_round\), which is initialized to 0. The \(cur\_round\) and the STATE variables are updated as described below, see figure 3.1 for full state machine, the underlined inputs summaries in general this work extension and will be explained further (for clarity only the inputs that cause a state change were indicated).

Stationery emitters start in the Accurate state while all other nodes start in the Unknown state. Each stationery emitter periodically increases its round number, broadcast its state and round number to all it neighbors, and then sleeps for a given time period, which should be adapted according to the network mobility and can be considered as a refresh rate. Non-stationery emitters wait for incoming messages: Whenever a node \(p_i\) receives a state message from another node \(p_j\), compares the newly received round number to its \(cur\_round\). If it larger is, then \(p_i\) sets its state to Unknown and update its
cur\_round accordingly; if the newly received round number is smaller, then it’s smaller the message is dropped.

Following this, if its cur\_round equals the newly received one, \( p_i \) computes its new state according to the geometric possible solution. For example, some trivial cases are showed in Figure 3.2. Here, if a node \( p_i \) that is in the Unknown state receives an Accurate message from another node \( p_j \), then \( p_i \) can upgrade its state to Circle with \( p_j \)’s location as the center of the circle and the distance between them as the radius. Finally, if indeed the state of the node was changed as a result of processing this message, then it broadcasts it updated STATE and cur\_round variables to its neighbors. The full pseudo-code is listed under Algorithm 1 and Algorithm 2.

### 3.1.4 Optimizations

Often, in a sensor network, and even in a MANET, the region’s boundaries are known. In these cases, it is possible to eliminate potential locations that are outside the network region. This helps improving the accuracy of the location calculations, as illustrated in Figure 3.7.

An important variant on the basic protocol is allowing a process to hold more than one potential circle in the Circle state and more than two points in the SemiAccurate state. That is, suppose a node \( p \) that is in the Unknown state hears a message from another node \( q \) that is in the SemiAccurate state. In this case, \( p \) can tell that it is in one of the circles whose centers are the two possible positions of \( q \), but it does not know which of these circles. Thus, if we allow \( p \) to maintain both of them, then eventually by interacting with additional nodes, it might be able to rule out one of these circles. In turn, this can later even help \( q \) to rule out one of its possible locations and upgrade to the Accurate state. On the other hand, as we would like to keep the space and computation requirements of RLISE small, it is recommended to allow each node to keep at most 2-3 circles in the Circle state and at most 4-6 potential locations in the SemiAccurate state.

This optimization is not incorporated into Algorithm 1 as presented above for clarity of presentation reasons. Yet, adding this to the code is obvious. In particular, our simulation measurements have explored this optimization, as reported below in Section 4.5.
Algorithm 1 Relative Location Algorithm for non-Stationary Emitters

OnReceive( r_state, round)
1:   STATE ∈ 
2:     \{Accurate(x,y),SemiAccurate(x_1,y_1,x_2,y_2), 
3:       Circle(x,y,r),Unknown\}
4:   if cur_round < round then
5:     STATE := Unknown
6:     cur_round := round 
7:   else if cur_round > round then 
8:     drop message
9:   end if
10: if r_state = Accurate(-) then
11:   if STATE = SemiAccurate(-) then
12:     STATE := Semi2Accurate (STATE,r_state,d) \{See Figure 3.4\}
13:   else if STATE = Circle(-) then
14:     STATE := Circle2Semi (STATE,r_state,d) \{See Figure 3.4\}
15:   else if STATE = Unknown then
16:     STATE := Unknown2Circle (STATE,r_state,d) \{See Figure 3.4\}
17:   end if
18: else if r_state = SemiAccurate then
19:   if STATE = SemiAccurate(+) then
20:     STATE := SemiPlusSemi(STATE,r_state,d) \{See Figure 3.5\}
21:   else if STATE = Circle(+) then
22:     STATE := CirclePlusSemi(STATE,r_state,d) \{See Figure 3.5\}
23:   end if
24: else if r_state = Circle ∧ STATE=SemiAccurate then
25:     STATE := SemiPlusCircle(STATE,r_state,d) \{See Figure 3.5\}
26: end if
27: if STATE changed then
28:   Broadcast STATE, cur_round
29: end if

 Algorithm 2 Relative Location Algorithm for Stationary Emitters

1:   STATE := Accurate(x,y)
2:   round := 1;
3: loop
4:   Broadcast STATE, round
5:   SLEEP Δt 
6:   round := round + 1 
7: end loop
Figure 3.4: Simple new state calculations - see drawings in Figure 3.2

```plaintext
SemiPlusSemi (SemiAccurate(x1,y2,x2,y2),Accurate(x',y'),d)
if dist((x1,y1),(x1',y1')) = d or dist((x2,y2),(x2',y2')) = d then
    return Accurate(x1,y1)
else if dist((x2,y2),(x1',y1')) = d or dist((x1,y1),(x2',y2')) = d then
    return Accurate(x2,y2)
else
    return SemiAccurate(x1,y1,x2,y2)
endif

Circle2Semi (Circle(x,y,r),Accurate(x',y'),d)
if the circles (x,y,r) and (x',y',d) intersect then
    let (x1,y1) and (x2,y2) be the two intersection points of (x,y,r) and (x',y',d)
    return SemiAccurate(x1,y1,x2,y2)
else
    return Circle(x,y,r)
endif

Unknown2Circle (UnknownAccurate(x',y'),d)
return Circle(x',y',d)
```

Figure 3.5: More complex state calculations - see illustrations in Figures 3.3 and 3.6
Another possible optimization on the basic protocol is to collect several messages before applying them in Line 4 of the protocol. This way, it is possible to choose messages that provide the best location information. For example, ones that bring the state to Accurate within the least number of steps. If such messages do not exist, then it may be possible to choose messages that result in the two closest points in the SemiAccurate state. If the latter is not achievable, then one can pick the messages that produce the Circle state with the smallest radius. Another variant of this idea is to apply the first message that arrives, yet to keep the last few messages. Then, on each message arrival, to try to choose the best set of messages among the last saved ones, and apply these messages to recalculate the best possible location.

### 3.1.5 Unsynchronized Clocks

It is possible to eliminate the synchronized clocks assumptions simply by making the round numbers behave like logical timestamps [23]. That is, each time a stationary emitter hears about a round $r$ that is larger than its own by more than one, it assigns its current round number to be $r$ and sends a message with the new round.
3.1.6 Calculating Relative Positions in Static Networks

The same algorithm can be applied to static networks but for such networks there is no need to periodically rebuild the position information. Specifically, the original algorithm can be reduced, removing lines 2-5 from Algorithm 1 and minimizing Algorithm 2 to line 4 (see Algorithm 3 and 4). But as mentioned before (section 3.1.2) using this reduction without the assumption of a collision detection/avoidance layer can greatly hurt the protocol results. In our simulations we found out that in most cases a simple retransmission can cover up on this weakness.

Algorithm 3 Relative Location Algorithm for Static non-Stationary Emitters

```
OnReceive( r_state, round)
1: STATE ∈ {Accurate(x,y),SemiAccurate(x1,y1,x2,y2),
              Circle(x,y,r),Unknown}
2: if r_state = Accurate(-) then
3:   if STATE = SemiAccurate(-) then
4:     STATE := Semi2Accurate (STATE,r_state,d) {See Figure 3.4}
5:   else if STATE = Circle(-) then
6:     STATE := Circle2Semi (STATE,r_state,d) {See Figure 3.4}
7:   else if STATE = Unknown then
8:     STATE := Unknown2Circle (STATE,r_state,d) {See Figure 3.4}
9:   end if
10: else if r_state = SemiAccurate then
11:   if STATE = SemiAccurate(-) then
12:     STATE := SemiPlusSemi (STATE,r_state,d) {See Figure 3.5}
13:   else if STATE = Circle(-) then
14:     STATE := CirclePlusSemi (STATE,r_state,d) {See Figure 3.5}
15:   end if
16: else if r_state = Circle ∧ STATE=SemiAccurate then
17:     STATE := SemiPlusCircle (STATE,r_state,d) {See Figure 3.5}
18: end if
19: if STATE changed then
20:   Broadcast STATE, cur_round
21: end if
```

Algorithm 4 Relative Location Algorithm for Static Stationary Emitters

```
1: STATE := Accurate(x,y)
2: Broadcast STATE
```

19
3.1.7 Handling Distance Estimation Errors

As can be noticed from the code of the algorithm, the correctness of the location estimates depends on the ability to accurately measure the distance from a sender. In this section, we explore the impact of errors in measuring this distance on the outcome of the protocol, and discuss heuristics to alleviate this problem. The most obvious problem in inaccurately measuring the distance from a sender is that a node will think that it is on a circle with a different radius than the real one. This can lead to finding incorrect circle intersection points, and eventually to an incorrect position.

In most situations, this process should lead to a bounded error propagation. Yet, there is one pathological case in which the error can be quite large. Interestingly, this happens when a node is in the Circle state and it receives a message from another node, who is very close to the circle’s center. In this case, as illustrated in Figure 3.8, the calculated intersection points can be very far from the real ones.

A way to overcome this problem is to ignore such messages. That is, if a node is in the Circle state and it receives a message with a position that is very close to the center of its circle, it simply ignores this message. The rational here is that it is better to remain in a less accurate state, with the hope of later receiving another message that can upgrade the state to a more accurate one, than to adopt a message that is likely to result in a large position error. Another optimization in order to mitigate the propagation...
of errors is based on the assumption that the distance measurement error is proportional to the distance itself [12]. Thus, a node can collect a few messages before reacting to any of them, and then decide to adopt the ones that are sent from the closest nodes, while avoiding the false circle intersection mentioned in the previous paragraph.

3.1.8 Analysis and Limitations

One of the main benefits of our RLISE protocol is that all computations are simple and local. They rely on simple, efficiently computable, geometric principles. Also, all exchanged messages carry only few integer values. These messages are exchanged only between immediate neighbors, and there is no need to forward messages along multiple hops. Moreover, the memory requirements of this protocol are also minimal, i.e., remembering the current state of the protocol, which can be represented as a handful of integers. Even in the extensions mentioned above to handle errors, at most a small number of recent messages need to be remembered. As becomes evident in our simulations detailed below, the vast majority of nodes learn their position within a very small number of message exchanges. Yet, RLISE does not require that any node will be within the transmission range of at least three stationary emitters.

RLISE works very well when nodes are either static, or when nodes move at a walking speed, typically up to 1-2 meters per second, which is much slower than the rate in which information propagates and the added error resulting by the use of obsolete location information is negligible. However, when nodes move much faster, e.g., at the speed of cars and faster, their physical position changes rapidly. In this case, in order to prevent nodes from sending obsolete location information to other nodes, thereby contaminating the system with false location information, we need to increase the refresh rate by setting the $\Delta t$ timer of Algorithm 2 to a small value. Note also that most ad-hoc positioning systems suffer from the same problem so in that sense of accuracy RLISE is no worse than other ad-hoc positioning approaches.

If the number of fast moving nodes is very small, and they are aware that they are moving very fast, we can simply disallow such nodes from sending their location information to others. Otherwise, in order for RLISE to work well with many very fast moving nodes, these fast moving nodes must be aware of the direction and speed that they are moving (e.g., using a compass and a speedometer). In this case, mobile nodes can simply add the offset from the position information they receive based on their movement since
obtaining that knowledge. Interestingly, with this assumption in place, our protocol in fact lends itself easily to rapid mobility in the sense the all calculations are local, and thus even when a node moves fast, this does not require any type of global re-computation. Moreover, due to the gradual learning process of RLISE, rapid mobility can be used in these cases to obtain accurate location information even when the network is not fully connected!

Another limitation of RLISE is that not all nodes are guaranteed to eventually learn their exact location (although as shown in Section 4.5 below, above a certain density, this happens only to a tiny fraction of nodes). Similarly, when the distance estimates are not perfect, the computed location of a small fraction of the nodes might be far from their physical location (although most nodes compute a location close to their real one). At the same time, it always performs better then Euclidean methods.

### 3.1.9 Utilizing Inaccurate Location Estimates

As discussed before, many works have explored how to utilize exact location information. In this section we discuss how the *SemiAccurate* and *Circle* states can also be used in location based routing algorithms and services.

When considering geographical routing protocols, there are several alternatives. For example, if a node is in the *Circle* state and the radius of the circle is small, a node can publish its circle. Then, the message will be routed to the correct geographical region of the network. Once the message arrives near the actual target node, most geographical routing protocols maintain proactive neighborhood information that is sufficient to complete the forwarding to its destination. The same is true when a node is in the *SemiAccurate* state and both possible locations are close to each other, as they are likely to fall in the same region of the routing protocol. During our simulations we discovered that very often this was indeed the case.

If a node is in the *SemiAccurate* state and the possible locations are far, then the message can be routed to both locations. This way messages will still arrive quickly to the node, and will only require a redundancy factor of 2. When a node is in the *Circle* state with a moderate radius, it is possible to initially route the message to the closest point on the circle. From there, the message can be routed along the circle edge, until it reaches the vicinity of the target node.
When considering location based services such as location dependent information, again, if the possible locations of the *SeimAccurate* state are near each other, then the information can be generated, e.g., according to the middle position on the straight line connecting them. If the positions are far, then it is possible to generate the information relevant to both. If a node is in the *Circle* state with a small radius, then the information can be sent based on the middle of the circle. This way the information might not be as accurate as with perfect location knowledge, but still at a much higher quality than without any knowledge.

### 3.2 Results

Our performance evaluation was carried out by simulations. In each simulation, we assumed a simulation area of 100x100 meters, in which nodes are spread in a random manner with uniform distribution. Also, a certain percentage of the nodes act as stationary emitters, i.e., they know their exact position right from the start of the simulation. All other nodes are unaware of their location at the beginning of each simulation, but gradually learn it while executing Algorithm 1. In the simulations we varied the number of nodes between 20 and 160, and the transmission range between 10 and 40 meters. Each data point is the average of 10 runs. We initially present simulations in which nodes are assumed to be able to measure the exact distance from a sender, and later explore how
errors in these measurements affect the accuracy of the learned positions. The former indicates how well our scheme, RLISE, can do in optimal situations, while the latter serves as a guideline for how it behaves in actual environments, and how advances in distance measurements technology will affect RLISE’s accuracy.

### 3.2.1 Performance with Exact Distance Measurements

Figure 3.9 explores the percentage of nodes that end up in each of the protocol states, after it reaches steady state, as a function of the number of nodes in the system. Since the area is fixed, this in fact indicates the relation between the density of the network and the accuracy of the system. In this case we fixed the number of stationary emitters to 10 in all runs, and the transmission range to 20 meters. As can be expected, the accuracy obtained by RLISE is proportional to the density of the network. Yet, this accuracy grows very fast until it reaches the point where with only 100 nodes, more than 90% of the nodes end up in the *Accurate* state. From there on, the improvement in accuracy grows much slower, until with 160 nodes we have 99% of the nodes in *Accurate* state.

Figure 3.10 presents how many protocol iterations (or local message exchanges) are needed until the system reaches a steady state, in which all nodes are in the best accuracy state as a function of the network density. As can be seen, initially, the number of iterations for convergence increases dramatically with the number of nodes. However, this is somewhat misleading. Recall from Figure 3.9 that with fewer than 80 nodes, the number of nodes that reach the *Accurate* state is small. Moreover, the number of *Accurate* nodes grows dramatically until we reach 80 nodes (after which is levels). However, once we bypass 80 nodes, in which most nodes end up *Accurate*, the number of iteration for convergence drops in a concave manner and eventually goes down to fewer than 6 iterations.

Figure 3.11 presents how many protocol iterations are needed until the system reaches a steady state as a function of the transmission range. In this graph we fixed the number of nodes to 100. As can be seen, here we have a similar phenomenon. When the transmission range is below 15 meters, a relatively small number of nodes reach the *Accurate* state, and thus the converges speed in that part of the graph grows very quickly as a function of the range. Beyond 15 meters, we have a significant concaved drop that levels at

---

2 Each protocol round includes multiple iterations defined by the number of message exchanges experienced by each node within this round.
just above 2 rounds (on the average). Notice that when nodes and SEs are placed using uniform distribution, the probability that a node is in transmission range of at least 3 SEs is roughly equivalent to $\pi R^2 \times \text{number-of-stationary-emitters}/(3 \times \text{size-of-area})$, where $R$ is the transmission range. Thus, with a transmission range of 35 meters and above, each node is likely to be within the transmission range of at least 3 stationary emitters.

Figure 3.12 illustrates the percentage of nodes that reach the Accurate state during each protocol iteration. Here again we have 100 nodes, 10 stationary emitters, and a transmission range of 20 meters. Also, in Figure 3.12 we compare three variants of RLISE to pure Euclidean positioning [27]. The RLISE variants are the most basic scheme, denoted RLISE, a variant that allows storing up to 2 potential circles and 4 points, denoted RLISE2, and a variant that allows storing up to 4 possible circles and 8 points, denoted RLISE4. As can be seen, RLISE converges much faster than the Euclidean method. This is because in the Euclidean method, nodes cannot make any use of partial location information and must wait until they have at least three neighbors that already established their location. Also, in this comparison, 98.5% of the nodes in RLISE reached the Accurate state, whereas in the Euclidean method only 93.5% eventually found their location. Thus, not only RLISE is faster, but indeed RLISE also enables more nodes to find their location.
Additionally, as can be seen in Figure 3.12, when considering all variants of RLISE, the second iteration is the most effective. After that, the number of additional nodes that become Accurate drops exponentially with each iteration. This indicates that iterations 2-4 are the most effective and most nodes converge during these iterations. Moreover, RLISE4 converges faster than RLISE2, which converges faster than RLISE. However, as can be seen, the main difference between RLISE2 and RLISE4 is in the final iterations. This indicates that there is not much point in maintaining more than 4 circles, and even 2 is probably good enough. As for nodes that reach the Accurate state in the first iteration, these are nodes that are placed initially within the transmission range of at least three stationary emitters.

For the Euclidean method, in the first three iterations, a growing number of nodes find their location. After that, a slowly decreasing number continues to find their location. This highlights the fact that in Euclidean positioning, a node does not learn anything about its location until at least three of its neighbors know their accurate location.

Figure 3.13 explores the convergence rates of RLISE and the standard Euclidean method with different numbers of stationary emitter (with 100 nodes and a transmission range of 20 meters). As can be seen, with RLISE many more nodes reach the Accurate state than with the Euclidean.

3.2.2 The Impact of Distance Measurements Errors

In order to study the impact of distance measurements errors on the accumulated accuracy error, we run simulations in which distance measurements errors varied in a random manner, with a gaussian probability, up to a given maximum [12]. The measurements we present here are on the basic algorithm, without the improvements detailed in Section 3.1.7. Figure 3.14 indicates the impact of the maximal error in each single distance measurements on the accumulated error in the calculated position of each node vs. its true location. In this graph we present the percentage of nodes whose accumulated error is below a given distance from their true location. Moreover, we present four lines, representing maximal single distance measurement error of 0.1 meters, 0.3 meters, and 0.7 meters. The number of nodes was fixed to 100, the number of stationary emitters set to 10, and the transmission range at 20 meters. As can be seen, in all cases the location calculated by the vast number of nodes is less than 10 meters.

Figure 3.15 exhibits the maximum accumulated error obtained in each protocol itera-
Figure 3.14: The Impact of Errors on Accuracy

Figure 3.15: Max Accumulated Error per Iteration

tion. As can be expected, the maximum accumulated error grows polynomially with the iteration number. This is highlights the propagation of errors in subsequent calculations.
Chapter 4

Timed Grid Routing (TIGR) Bites off Energy

4.1 General concept

4.1.1 Brief

In this second part of this dissertation we propose a new scheme aimed to reduce energy consumption and eliminates messages collisions. This scheme is a scheduling scheme based on geographic location, that both allows many nodes in the system to sleep for large fraction of the time and eliminates collisions thanks to the time division.

4.1.2 System Model Assumptions

As in Chapter 3, we assume a collection of nodes, each equipped with a wireless transmitter and receiver. We assume that nodes communicate with each other only by exchanging messages through their wireless transmitters and receivers. We assumes that the transmitter of each node \( p \) has a bounded transmission range \( t_p \). Thus, when a node \( p \) sends a message, all other nodes whose distance from \( p \) is less then \( t_p \) may potentially receive this message (Figure 4.1). It is known that this simplistic transmission Unit Disk model is not completely realistic. However, it simplifies the model and presentation.

Moreover, our protocols do not depend on the Unit Disk model and our simulation results also use a more realistic model known as Fuzzy Unit Disk [5] or Quasi Unit Disk [22]. In these models not all the nodes within \( t_p \) always receive each message sent
by $p$ (Figure 4.2), e.g., due to obstacles such as concrete walls, etc. But all the nodes whose distance from $p$ is less than $r_p$ receive all the messages sent by $p$.

However, a node may not receive a massage from $p$ even though its distance from $p$ is less than $r_p$ due to interference with other messages. Therefore, we also define an interference range $R_p$, which has the following meaning. If two nodes $p$ and $q$ send messages simultaneously, then for any third node $A$ whose distance from $p$ and $q$ is less than $R_p$ and $R_q$, respectively, $A$ does not receive these messages. Clearly, for any node $p$, $r_p \leq R_p$. Yet, it is common to assume that $R_p$ is not much larger than $r_p$. The specific relation required for our scheme is discussed below.

The TIGR protocol also relies on the assumption that all the nodes know their location using one of the reviewed methods or RLISE (notice that the RLISE method is optional, therefore we do not assume any distance measure capability), and can find out the location of each other. The second demand can be satisfied either using a static allocation, or by using a location service. We further assume that nodes have synchronized clocks.

For the clarity we first assume that the nodes are deployed on a virtual grid formation such that each grid cell includes at least one node, and that each node knows the coordinates of the grid cell in which it is placed. Given two grid cells $(x,y)$ and $(x',y')$ we say that these two cells are neighbors, denoted by $(x,y) \leftrightarrow_n (x',y')$, if $|x-x'| \leq 1 \land |y-y'| \leq 1$. Moreover, we say that $(x,y)$ and $(x',y')$ are level neighbors, denoted by $(x,y) \leftrightarrow_{ln} (x',y')$. 
if $|x - x'| + |y - y'| \leq 1$. We call the cells at the edges of the grid border cells while the other cells are called non-border cells.

### 4.1.3 The Grid Transmission Range Inequality

As elaborated below, our aim is to allow nodes to transmit messages such that no other concurrent transmission will interfere with these messages at the intended recipients. In particular, our scheme is based on having a node from one cell transmitting a message to another node in a level neighboring cell. Thus, if we do not restrict the locations of nodes within each cell, then the transmission range must be long enough so that a transmission anywhere in one cell would reach anywhere in the other cell.

However, we would also like to schedule as many concurrent transmissions as possible, meaning that we must also assume a bound on the interference range. This leads to the following inequality, called the grid transmission range inequality (see also Figure 4.3):

$$\forall p, \sqrt{5}d \leq r_p \leq R_p < c \cdot d$$

Here, $r_p$ is the transmission range, $R_p$ is the interference range, $d$ is the widthlength of each cell, and $c$ is some constant. Obviously, $c > \sqrt{5}$. This model without the reference to $R_p$ is called fuzzy disk graph in [5] and quasi disk graph in [22].

In a special case where the network is static, e.g., a sensor network, and holds a mesh formation as shown in Figure 4.4, we can obtain an even tighter inequality, called the mesh transmission range inequality:

$$\forall p, d \leq r_p \leq R_p < c \cdot d$$
4.1.4 Scheduling Functions

The goal of the scheduler is to associate with each grid cell, for each time slot $t$, a value from the activity domain, which includes $ACTIVE$, $PASSIVE$, and $SLEEP$. Here, $ACTIVE$ means that the grid cell is allowed to send messages during the corresponding time slot $t$, $PASSIVE$ means that it should listen for messages, and $SLEEP$ means that it should be in sleeping mode. The scheduler is thus a mapping from time slots and grid coordinates to the activity domain, or formally, $f(t,X,Y) \rightarrow \{ACTIVE, PASSIVE, SLEEP\}$.

Interestingly, it may happen that an arbitrary scheduling function causes disconnection in the network. For example, if there exists a cell $(x,y)$ for which $f(t,x,y) = SLEEP$ for all values of $t$. Similarly, the behavior of the scheduling function may control the minimal number of hops (cells) that a message must travel in the grid in order to arrive to its destination. This motivates specifying good properties that one can expects from a scheduling function.

Below, we define several such desired properties that we will require from the schemes we present. But first, for a given scheduling function we define the interference distance between two grid cells $(x,y)$ and $(x',y')$ to be the shortest distance between any node in $(x,y)$ to any other node in $(x',y')$. Also, in the definitions below we assume that the grid transmission range inequality holds with a constant $c$.

**Definition 1 (Validity).** A scheduling function $f$ is valid iff: for any time $t > 0$ and grid cell $(x,y)$, whenever $f(t,x,y) = ACTIVE$, then there is exactly one level neighboring cell $(x',y')$ of $(x,y)$ for which $f(t,x',y') = PASSIVE$ and for any other cell $(x'',y'')$ whose interference distance from $(x',y')$ is shorter than $x \cdot d$ we have $f(t,x'',y'') \neq ACTIVE$.

**Definition 2 (Well Connected).** A scheduling function $f$ is well connected if for any cells $(x,y)$, $(x',y')$ and $t$ there exists a finite series of time slots $\{t_i\}$ where $t < t_1 < \ldots < t_n$ and cells $\{(x_i,y_i)\}$ respectively, such that $(x,y) = (x_1,y_1) \leftrightarrow_{In} (x_2,y_2) \leftrightarrow_{In} \ldots \leftrightarrow_{In} (x_n,y_n) = (x',y')$, where $f(t_i,x_i,y_i) = ACTIVE$ and $f(t_i,x_{i+1},y_{i+1}) = PASSIVE$.

The combination of the validity and well connected properties form the basics for any desirable scheduling function in any grid formation obeying the grid/mesh transmission range inequality.\(^1\) Validity ensures that there will be no collisions on one hand, and that whenever a node sends a message, some other node will be listening for these messages.

\(^1\)It is also possible to relax the validity property to allow multiple $PASSIVE$ cells for each $ACTIVE$ one, adding a slight complication to the proofs of the lemmas.
On the other hand, well connected means that it is possible to find a routing scheme that will deliver messages between any pair of nodes.

Notice also that valid scheduling functions restrict the advancement of messages in the grid along horizontal and vertical lines only (no diagonal movements). Thus, we may define the grid distance between any two grid cells \((x, y)\) and \((x', y')\), denoted by \(d_G((x, y), (x', y'))\) as \(|x - x'| + |y - y'|\). That is, the grid distance is the minimal number of hops separating two grid cells when movement is restricted to horizontal and vertical advancements only. Next, define the notion of a fair scheduling function.

**Definition 3 (Fairness).** A scheduling function \(f\) is fair if for each \((x, y) \leftrightarrow_{ln} (x', y')\) and a time slot \(t\) there exists a time slot \(t' > t\) for which \(f(t', x, y) = \text{ACTIVE} \land f(t', x', y') = \text{PASSIVE}\).

**Lemma 4.** Every fair scheduling function is well connected.

**Proof.** Consider any pair of cells \((x, y)\) and \((x', y')\) and any time slot \(t\). Let \((x_1, y_1) \leftrightarrow_{ln} (x_2, y_2) \leftrightarrow_{ln} \ldots (x_n, y_n)\) be a sequence of cells such that \((x_1, y_1) = (x, y)\), \((x_n, y_n) = (x', y')\) (such a sequence always exists in a grid). Since \(f\) is fair, there exists \(t < t_1 \ldots < t_{n-1}\) for which \(f(t_i, x_i, y_i) = \text{ACTIVE} \land f(t_i, x_{i+1}, y_{i+1}) = \text{PASSIVE}\). Hence, \(f\) is well connected. \(\square\)

**Definition 5 (Strong Fairness).** A scheduling function \(f\) obeys strong fairness if \(f\) is fair and for any triplet of grid cells \((x, y), (x', y')\), and \((x'', y'')\) such that \((x, y) \leftrightarrow_{ln} (x', y')\) and \((x, y) \leftrightarrow_{ln} (x'', y'')\), if there are two time slots \(t_1\) and \(t_2\), \(t_1 < t_2\) for which \(f(t_1, x, y) = \text{ACTIVE}\) and \(f(t_1, x', y') = \text{PASSIVE}\), there exists a time slot \(t\), \(t_1 < t < t_2\) for which \(f(t, x, y) = \text{ACTIVE}\) and \(f(t, x'', y'') = \text{PASSIVE}\).

Thus, fairness simply ensures that eventually, it is possible to advance from every node to any of its level neighbors. Clearly, this is helpful for devising local routing policies. The notion of strong fairness adds to this by ensuring that between any two time slots in which it is possible to advance in some direction, there is a time slot in which it is possible to advance in any other direction. The benefit of strong fairly scheduling functions, as discussed below, is that with such a scheme, locally computed greedy routing yields paths that are both shortest in terms of distance and fastest is terms of latency.

**Definition 6 (Global Strong Fairness).** A scheduling function \(f\) obeys global strong fairness if \(f\) is fair and for any grid cells quartette \((x_1, y_1), (x_2, y_2), (x_3, y_3)\) and \((x_4, y_4)\)
such that \((x_1, y_1) \leftarrow_{ln} (x_2, y_2)\) and \((x_3, y_3) \leftarrow_{ln} (x_4, y_4)\) (not the same couples of cells), if there are two time slots \(t_1\) and \(t_2\), \(t_1 < t_2\) for which \(f(t_1, x_1, y_1) = ACTIVE\) and \(f(t_1, x_2, y_2) = PASSIVE\), there exists a time slot \(t\), \(t_1 < t < t_2\) for which \(f(t, x_3, y_3) = ACTIVE\) and \(f(t, x_4, y_4) = PASSIVE\).

**Definition 7 (Δ-timely).** A scheduling function \(f\) is Δ-timely if there exists a constant \(Δ\) such that for any time interval \([t_1, t_2]\) such that \(t_2 - t_1 \geq Δ\), for every pair of grid cells \((x, y)\) and \((x', y')\), \((x, y) \leftarrow_{ln} (x', y')\) such that there exists a time slot \(t'\) in which \(f(t', x, y) = ACTIVE \land f(t', x', y') = PASSIVE\), there is a time slot \(t'' \in [t_1, t_2]\) in which \(f(t'', x, y) = ACTIVE \land f(t'', x', y') = PASSIVE\).

**Lemma 8.** Every globally strong fair scheduling function \(f\) such that for each time slot \(t\) there is at least one cell \((x, y)\) for which \(f(t, x, y) = ACTIVE\) is Δ-timely for \(Δ = 4n\), where \(n\) is the number of cells.

**Proof.** If \(f\) is a globally strong fair scheduling function, then by definition, for each couple of cells \((x_1, y_1)\) and \((x_2, y_2)\) such that \((x_1, y_1) \leftarrow_{ln} (x_2, y_2)\), at least once every \(4n\) time slots \(f\) evaluates to \(f(t, x_1, y_1) = ACTIVE \land f(t, x_2, y_2) = PASSIVE\). Otherwise, there is another couple of level neighbors cells \((x_3, y_3)\) and \((x_4, y_4)\) for which \(f\) returns at least twice in the same \(4n\) time slots \(f(t, x_3, y_3) = ACTIVE \land f(t, x_4, y_4) = PASSIVE\) (according the Pigeonhole Principle). Yet, this is a contradiction to the definition of \(f\) as a globally strong fair function. Consequently, \(f\) is Δ-timely for \(Δ \leq 4n\).

The globally strong fairness property promises equal sharing of the bandwidth between the cells across the grid. On the other hand, strong fairness only promises for each cell a fair scheduling to each direction. Finally, the Δ-timely property gives a time bound that can be used to calculate a bound on the routing latency.

### 4.2 TIGR’s Scheduling Mechanisms

In this section we present TIGR’s basic scheduling protocol. For clarity of presentation, initially we assume that each of the grid cells contains one node. We later relax this assumption.

In order to eliminate collisions and save energy, TIGR employs only valid well connected scheduling functions. Moreover, at any given moment some cells of the grid sleep.
The challenge is to find such scheduling functions that provide a good balance between sleeping time, propagation latency, and network utilization.

4.2.1 Random \( f_n \)

The first scheduling function we consider is a randomized function \( f_{\text{rand}} \) that randomly chooses the values \( f(t, x, y) \) while maintaining the validity property. Notice that in order to allow local calculation of such random function, it is possible to start all nodes with the same seed and the same pseudo random generator.

**Lemma 9.** Any random function \( f_{\text{rand}} \) is well connected and fair.

**Proof.** Note that by definition, any such function \( f_{\text{rand}} \) is valid. Also, randomness also ensures the fairness property. Thus, by Lemma 4, this scheduling function is also well connected.

4.2.2 Deterministic \( f_{(k,p,n,m)} \)

**General Case** \( (k \geq c + 2 \land p \geq c + 2 \land n \geq c + 1 \land m \geq c + 1) \)

The problem with random functions is that there is no bound on how long it can take before a message can advance from a given node to any of its level neighbors. Thus, it is hard if not impossible to develop schemes that obtain minimal routing latencies. Using a deterministic method can eliminate this disadvantage. We list the code for such a function in Figure 4.5. Notice that for clarity of the algorithm, the special treatment for the border cells was discarded. These cells should be specially checked to ensure that any \text{ACTIVE} cell will have a counter \text{PASSIVE} and vice versa.

This scheduling function schedules cells to send and listen for messages in a manner that allows for information to first flow along rows to the right, then along columns upwards, then along rows to the left, and then along columns downwards. Yet, due to interference, the information cannot flow on all rows/columns at the same time, but rather only rows/columns that are further apart than the interference distance can operate in parallel. Thus, the function rotates between the rows and columns that can forward messages until all of them have been given a chance to forward their information.

To understand this function intuitively, consider the following simple sequential case. Here, the scheduling function assigns cell \((0,0)\) to first send messages to \((0,1)\), then
function \( f(k,p,n,m) \):(t,x,y):
\[ T := t \mod (k+p) \]
\[ r := \frac{t}{k+p} \]
if \( T < k \) then
\[ R := T \]
\[ \text{case} \]
\[ x \equiv \mod n \land ((y \equiv \mod k \land r \equiv \mod 2 0) \lor (Y - y - 1 \equiv \mod k \land r \equiv \mod 2 1)) : \]
\[ \text{return } ACTIVE \]
\[ x \equiv \mod n \land ((y - 1 \equiv \mod k \land r \equiv \mod 2 0) \lor (Y - y - 2 \equiv \mod k \land r \equiv \mod 2 1)) : \]
\[ \text{return } PASSIVE \]
\[ \text{default :} \]
\[ \text{return } SLEEP \]
\[ \text{endcase} \]
elseif \( k \leq T \) then
\[ R := T \mod k \]
\[ \text{case} \]
\[ y \equiv \mod m \land ((x \equiv \mod p \land r \equiv \mod 2 0) \lor (X - x - 1 \equiv \mod p \land r \equiv \mod 2 1)) : \]
\[ \text{return } ACTIVE \]
\[ y \equiv \mod m \land ((x - 1 \equiv \mod p \land r \equiv \mod 2 0) \lor (X - x - 2 \equiv \mod p \land r \equiv \mod 2 1)) : \]
\[ \text{return } PASSIVE \]
\[ \text{default :} \]
\[ \text{return } SLEEP \]
\[ \text{endcase} \]
endif
end

Figure 4.5: The deterministic \( f(k,p,n,m) \) scheduling function
(0,1) to (0,2), etc., until a message can propagate all the way from (0,0) to (0,n−1). Once this has been established, the function assigns cell (1,0) to forward messages to (1,1), afterwards (1,1) to (1,2), etc. This corresponds to rows sending information to the right.

Once all rows have been given a chance to forward information to the right, we start a process of allowing columns to send information upward. So, first cell (0,0) forwards a message to (1,0), then (1,0) to (2,0), and so forth.

Now, in order to enable parallelism, we can allow several rows to forward information at the same time, if their distance is greater than the interference distance. Moreover, even within the same row, we can allow multiple cells to transmit if they are far enough. The same applies to allowing multiple columns and multiple nodes within a column to forward information at the same time. The distance between rows that act concurrently is the value of the k parameter, while the distance between columns that act concurrently is controlled by the p parameter, which means the every 2(k+p) time units, we complete a full cycle, as each cell needs to be active twice horizontally (right and left) and twice vertically (up and down). Also, the parameter n stands for the distance between cells inside the same row that can transmit concurrently, while m is the distance between cells within the same column that can transmit concurrently. Notice that the example above, in which there is no parallelism, corresponds to k = n = X and p = m = Y.

Figure 4.6 exemplifies horizontal advancement along rows for f(4,4,3,3) while Figure 4.7 illustrates the same function with vertical advancement along the columns. In summary, the scheme of the deterministic f(k,p,n,m) offers a tradeoff between energy consumption and network utilization and latency: more parallelism can result in lower latencies and better network utilization, but also more power consumption since fewer nodes are sleeping in each time slot.

**Lemma 10.** The deterministic function f(k,p,n,m) is valid, Δ-timely and obeys strong fairness for odd values of n and m.

**Proof.** By the definition of f, when T < k, between any two concurrently ACTIVE cells on different rows there is a distance of at least c because n ≥ c + 1, which maintains the validity property. For cells within the same row, the distance between any concurrently ACTIVE cells is at least c + 1 because p ≥ c + 2, which again ensures the validity property. The same validity property is preserved when T ≥ k. f is Δ-timely for Δ ≥ 2(k·n + m·p). By the definition of f, after each 2(k·n + m·p) time slots f completes a full cycle such
that for every $t' \equiv_{mod2(k,n+m,p)} t''$, $f(t',X,Y) = f(t'',X,Y)$. $f$ obeys strong fairness by the same reason. Thus, $f$ is cyclic and in each cycle (for odd $n$ and $m$) for each couple $(x,y) \leftrightarrow ln (x',y')$, $f$ returns $f(t',x,y) = ACTIVE \land f(t',x',y') = PASSIVE$ once.

Notice that due to Lemma 4, $f$ is also well connected for odd $n$ and $m$. This makes $f$ a good scheduling function. Also, $f$ can be easily adapted to be well connected for even values of $n$ and $m$ when the matrix contains an even number of cells.

**Meshes**

In a special case where the network is static and holds a mesh formation (one node in the middle of each cell) as shown in Figure 4.4, the constants $k$, $p$, $n$ and $m$ can be lowered to $k \geq c+1$, $p \geq c+1$, $n \geq \sqrt{c^2-1}$, and $m \geq \sqrt{c^2-1}$. Figure 4.8 shows the case in which $c \leq 2$, $k=p=3$ and $m=n=1$.

### 4.2.3 Back-to-Back $f_{(k,p,n,m)}$

#### General Case ($k \geq c+1 \land p \geq c+1 \land n \geq c+1 \land m \geq c+1$)

The previous scheduling function is parameterized by the four parameters $k$, $p$, $n$ and $m$, which allow to change the network throughput, determined by the number of ACTIVE nodes during each time slot. But, this function cannot yield the maximum possible network throughput for all values of $c$. For this, we present in Figure 4.10 our last deterministic scheduling function, which can always obtain the maximum throughput. This
function is nicknamed Back-to-Back due to the way it selects the ACTIVE and PASSIVE cells for each time slot. That is, during horizontal (vertical) advancement, nodes of the same row (column) flip their functionality between ACTIVE and PASSIVE on each time slot, as illustrated in the example in Figure 4.9.

**Meshes**

When the network obeys the mesh transmission range inequality, the lower bounds for the constants $k$, $p$, $n$ and $m$ can be lowered to $k \geq c$, $p \geq c$, $n \geq \sqrt{c^2-1}$, and $m \geq \sqrt{c^2-1}$.

Given that $c \leq \sqrt{2}$, the Back-to-Bach function $f_{(2,2,1,1)}$ (as illustrated in Figure 4.9) is a valid scheduling function that yields the maximum number of ACTIVE cells ($n/2$ of the cells) for each time slot under the mesh transmission range inequality ($d \leq r_p \leq R_p < c \cdot d$).

*Remark 11.* The deterministic function presented in Section 4.2.2 yields a maximum $n/3$ ratio of ACTIVE cells for $c \leq \sqrt{2}$.

### 4.2.4 Representative Backbone for Multiple Nodes in a Cell

Until now we assumed that each area in the grid contains one and only one node. This assumption corresponds to static networks. However, in a MANET, mobile nodes can move from one area to another and several nodes may share the same area. To ensure the advantages of the using TIGR, a representative node should be elected in each area, and these elected nodes should act as gateways for the other nodes.
function B2B \( f(k,p,n,m)(t,x,y) \):

\[
T := t \mod (k+p)
\]
\[
r := \frac{x}{k+p} \text{ //in integers}
\]

if \( T < k \) then

\[
R := T
\]

\[
\text{case}
\]
\[
x \equiv \mod n \text{ } r \land (((y \equiv \mod 2k \text{ } R \lor \equiv \mod 2k \text{ } R + k + 1) \land r \equiv \mod 2 \text{ } 0) \}
\]
\[
\lor ((Y - y - 1 \equiv \mod 2k \text{ } R \lor \equiv \mod 2k \text{ } R + 1 \equiv \mod 2 k + 1) \land r \equiv \mod 2 \text{ } 1)) : \]
\[
\text{return } \text{ACTIVE}
\]
\[
x \equiv \mod n \text{ } r \land (((y \equiv \mod 2k \text{ } R + 1 \lor \equiv \mod 2k \text{ } R + k) \land r \equiv \mod 2 \text{ } 0) \}
\]
\[
\lor ((Y - y - 1 \equiv \mod 2k \text{ } R + 1 \lor \equiv \mod 2k \text{ } R + k) \land r \equiv \mod 2 \text{ } 1)) : \]
\[
\text{return } \text{PASSIVE}
\]

\[
\text{default :}
\]
\[
\text{return } \text{SLEEP}
\]

endcase

endif

elseif \( k \leq T \) then

\[
R := T \mod 4
\]

\[
\text{case}
\]
\[
y \equiv \mod n \text{ } r \land (((x \equiv \mod 2p \text{ } R \lor \equiv \mod 2p \text{ } R + p + 1) \land r \equiv \mod 2 \text{ } 0) \}
\]
\[
\lor ((X - x - 1 \equiv \mod 2p \text{ } R \lor \equiv \mod 2p \text{ } R + 1 \equiv \mod 2 p + 1) \land r \equiv \mod 2 \text{ } 1)) : \]
\[
\text{return } \text{ACTIVE}
\]
\[
y \equiv \mod n \text{ } r \land (((x \equiv \mod 2p \text{ } R + 1 \lor \equiv \mod 2p \text{ } R + p) \land r \equiv \mod 2 \text{ } 0) \}
\]
\[
\lor ((X - x - 1 \equiv \mod 2p \text{ } R + 1 \lor \equiv \mod 2p \text{ } R + p) \land r \equiv \mod 2 \text{ } 1)) : \]
\[
\text{return } \text{PASSIVE}
\]

\[
\text{default :}
\]
\[
\text{return } \text{SLEEP}
\]

endcase

endif

end

Figure 4.10: The Back-to-Back scheduling function
This solution, which is often referred to as a *dominating group*, offers many advantages and was extensively researched, e.g., in [2]. Below, we describe a simple election protocol that implements this notion, although other election methods can also be considered. In the election protocol, every node declares itself, and one node is elected using some deterministic function, e.g., the node with the smallest identifier, or the node with the highest level of energy, to be the cell’s representative. The most widely used election method in homogenous mobile networks is to choose the node that has the most energy. This way, the network burden is divided among the nodes and the network lifetime is increased. This is implemented in TIGR in the following manner. An election process takes place periodically once every $\gamma$ time units, where $\gamma$ is some constant, in a time slot in which the cell is *ACTIVE*.

Additionally, when multiple nodes may populate the same cell, time should be allocated for the communication between these nodes and their elected representative. This *intra-cell* communication time is allocated at the beginning of the *ACTIVE* phase of the slot. Figure 4.11 outlines the time-line for the entire interaction. It exhibits how this integrates with the inter-cell communication without causing any interference there.
4.2.5 Synchronization

So far, we assumed tight clocks synchronization across the network. This demand can be satisfied using GPS, for example. But, due to GPS limitations in urban areas and inside buildings an alternative solution should be adapted. This solution can come in the form of periodic cooperative clock synchronization. Yet, no clock synchronization technique can result in perfect synchronization. To compensate for that, we introduce the *Inter-communication Listening* phase as shown in Figure 4.11. In this phase, nodes start listening for messages slightly before their *Inter-communication Transmitting* phase, in order to compensate for any possible clocks drift.

4.3 Location Service

Most of the geo-routing algorithms are based on the existence of some kind of a location service layer. To the best of our knowledge, none of the existing location service algorithms is explicitly tailored to a grid.

Interestingly, when one uses a grid oriented geo-routing such as TIGR, it is not necessary to know the exact geographic location of nodes. Rather, it is enough that each node knows the direction in which all other nodes are with respect to it (Up, Down, Right, and left). This observation helps reducing the number of update messages caused by nodes’ movements. This is because as long as a node remains in the same cell, no updates need to be sent. Moreover, even when a node moves between neighboring cells, this only affects the directional information in at most $O(\sqrt{n})$ other cells.

One implementation, nicknamed LS1, that uses this property is shown in Figure 4.12. Each node (cell) holds for each node in the network one of the possible one-hop directions for routing a message towards this node (Right, Up, Left & Down). In this scheme, all cells for which the destination node is on their right side and not above them are marked Right; all those for which the destination node is above and not to their left are marked Up, etc’. Figure 4.12 illustrates the updated network after a node movement to the left. The marked cells are the only ones that were updated. The cost of such an update is $\sqrt{n}$ messages, where $n$ is the number of cells in the network. However, lookup is completely
local and does not generate any messages. Notice that a cost of $O(n)$ updates should be paid once on startup or when a node joins the network.

A similar location service, nicknamed LS2, can be built in which each node holds the two possible directions to forward a message towards its destination (Figure 4.13). Yet, it requires twice the amount of update messages compared to LS1 on each node movement between cells. On the other hand, this location service can offer better flexibility for geo-routing algorithms.

4.4 Routing Algorithms

Given the location service LS1 and the deterministic scheduling function $f(k,p,n,m)$, as described above, it is simple to define a greedy geo-routing algorithm. Each node simply consults the local location service information before forwarding the message to the direction pointed by the location service. The latency for each message from $(x, y)$ to $(x', y')$ is bounded by $2(nk + mp)(\frac{|x-x'|}{k} + \frac{|y-y'|}{p} + C)$ time slots (where $C$ is a constant resulted by the first and last steps on each directions). This result follows from the sequential nature of $f$, in which $k$ sequential steps can be done on the same round on the $X$ axis and $p$ step on the $Y$ axis, while each round takes $2(nk + mp)$ time slots.

Lemma 12. For every scheduling function $f$ that is both $\Delta$-timely and obeys strong fairness, a strictly greedy routing scheme (not necessarily using LS1) delivers every message
sent between any pair of cells \((x,y)\) and \((x',y')\) within \(\Delta \times d_{G((x,y),(x',y'))}\) time slots.

**Proof.** The lemma follows directly from the fact that \(f\) is both \(\Delta\)-timely and obeys strong fairness. In other words, it is promised that at least once in \(\Delta\) time a greedy routing algorithm will be able to forward the message one step towards the destination. \qed

This property cannot be promised for a random scheduling function. Instead, in the case of probabilistic scheduling functions, it is only possible to give a similar probabilistic bound. Also, the scheduling function \(f_{(k,p,n,m)}\) is \(\Delta\)-timely for \(\Delta \geq 2(nk + mp)\). Moreover, the bound achieved using the geo-routing algorithm that was presented above is lower than the upper bound promised by this Lemma.

Regarding multicast, the simplest scheme is that whenever a node receives a message \(m\) for the first time (or the representative node in the case of multiple nodes per cell), it marks this message for forwarding. For a given cell \((x,y)\), based on the scheduling function, the corresponding node in \((x,y)\) sends the message in each time slot \(t\) for which (i) \(f(t,x,y) = ACTIVE\), (ii) for some level neighboring cell \((x',y')\) \(f(t,x',y') = PASSIVE\), (iii) it has never received \(m\) from this cell, and (iv) it has not forwarded \(m\) to that cell yet. Once \(m\) is either received from or disseminated to all level neighboring cells, then \(m\) can be eliminated. This scheme ensures no collisions, eventual delivery to all nodes, and power saving by sleeping due to the reliance on the scheduling function.

### 4.5 Analysis

The use of the deterministic \(f_{(k,p,n,m)}\) described in Section 4.2.2 presents a tradeoff between network utilization and energy consumption. The network utilization for the deterministic scheduling function is given by \(\frac{1}{nk}\) when messages are transmitted horizontally and \(\frac{1}{mp}\) when messages are transmitted vertically. This function defines a transmission cycle that repeats itself every \(2(k \cdot n + p \cdot m)\) time slot, out of which \(k\) are for horizontal advancements and \(p\) for vertical. We define the overall network utilization as the fraction of cells for which the scheduling function evaluates to \(ACTIVE\). From this, we get a utilization factor of \(\frac{k}{(k+p) \cdot nk} + \frac{p}{(k+p) \cdot mp} = \frac{n+m}{n \cdot m \cdot (k+p)}\). If we assign \(n=m=1\) and \(k=p=3\), for example, we get a utilization factor of \(\frac{1}{3}\) (Figure 4.8).

Measurements of WaveLan radios [36] have shown that the energy consumption ratios between listening for messages to receiving a message to sending a message are
1:1.05:1.4, respectively. As can be seen, the different between listening for a message and receiving in terms of energy consumption is negligible. This ratio implies a clear linear connection between the total energy consumption and network utilization, as showed in figure 4.14. Also, in the case of a mesh, the best that can be hoped for is a ratio of $\frac{1}{2}$, which is attainable using the Back to Back function that is given in Figure 4.9.

Table 4.1 compares the three scheduling functions we presented in terms of complexity, the maximal network utilization they offer, and the latency they incur when using greedy routing. The computational complexity of Random is $O(n)$ due to the need to ensure the validity property. On the other hand, the other two functions are easily computable in $O(1)$ time.

Both the Random and Deterministic methods can obtain maximum network utilization of $\frac{1}{3}$ (i.e., $\frac{1}{3}$ of the nodes are transmitting in each time slot), whereas Back-to-Back can reach $\frac{1}{2}$. Yet, notice that when a $\frac{1}{3}$ of the nodes are transmitting, another $\frac{1}{3}$ are sleeping, thereby saving energy. On the other hand, when $\frac{1}{2}$ of the nodes are transmitting, all others are listening. Of course, it is possible to reduce the network utilization of Back-to-Back to $\frac{1}{3}$ (or even lower) and increase its power efficiency proportionally, but in such cases it has no advantage over the Deterministic function. On the other hand, the
Deterministic function is more intuitive. Moreover, unless all nodes constantly generate messages, Back-to-Back is more likely to suffer from unused time slots (in which some nodes are in active mode, but in fact they have nothing to send). Finally, recall that the latency was analyzed in Section 4.4.
Chapter 5

Conclusion

We presented in this dissertation two loosely coupled protocols which are in general orthogonal network layers but can be combined together in a same network stack or used separately.

In the first part (Chapter 3) of this dissertation we have demonstrated a local distributed scheme for geographic localization, which uses relative measurements and cooperative knowledge. The algorithm assumes the existence of a small number of nodes that know their exact location a-priori, e.g., through GPS or cellular triangulation. It also assumes that each node can determine its distance from a sender of a message, e.g., using known range measurements techniques such as ToA and RSSI.

With these two reasonable assumptions, our protocol allows nodes to gradually learn their location without relying on any central authority and while only exchanging local information with their immediate neighbors. Unlike standard Euclidean methods, our protocol allows nodes to maintain partially accurate states, in which they only know about a potential circle they might be on, or a couple of potential exact positions. This, in turn, allows our protocol to converge faster, and moreover, to converge in situations in which pure Euclidean methods do not.

We have presented extensive simulations that investigate the performance of our scheme, and explored its behavior as well as its limitations, including the impact of mobility and of errors in distance measurements on the accuracy of the location estimations. These simulations have shown that indeed when the network connectivity is reasonable, or better, the vast majority of nodes learn their accurate location very quickly. Also, despite reasonable potential errors in distance measurements, the vast majority of nodes
compute a location that is within 10 meters of their true location (and in fact, even within 1 meter of their correct position).

The purpose of the second part (Chapter 4) of this work was to demonstrate that by dividing the area of an ad-hoc network into a logical grid, we can devise simple locally computable transmission schedules that avoid collisions and enable energy savings by allowing some nodes to sleep. This is even when the network is multiple hop, and even when message are multicast. As mentioned before, all these results rely on each node knowing its position and assuming well synchronized clocks (both properties are satisfied when using a GPS). We have presented specific scheduling functions and routing algorithms that demonstrate this ability. Moreover, the scheduling functions we presented are parameterized in a way that enables trading off network utilization levels vs. power saving and latency.

As an artifact, we have also shown that the use of a logical grid structure enables devising a highly efficient location service. In such a service, other than the initial publishing of a node, movement either does not generate any update messages, or generate $O(\sqrt{n})$ messages. Furthermore, lookup is completely local, meaning that the routing protocol can reacts to movement as it happens at no extra cost.

5.1 Future Work

Our work can still be extended in several ways. An interesting open topic is to exploit the past in order to improve the convergence and accuracy of our RLISE scheme. For example, in order to reduce the impact of errors in distance estimations, it is possible to try computing the location based on multiple messages, and then take the average. Similarly, it may be possible to use multiple such messages to rule out gross errors, etc. Here again, the trade-off is how to keep memory consumption, message sizes, and computation reasonably small.

It would be interesting to extend our RLISE scheme to three dimensions, to explore how the initial stationary emitters placement affect the accuracy of our scheme, and try to find an optimal such placement. Finally, it might be possible to use a hybrid AOA and TOA [26] scheme for improved accuracy. Additional optimizations include incorporating TTL based techniques for improving the accuracy of the learned positions along the lines of [12]. Also, utilizing mobility, and in particular the fact that movement is continuous,
to eliminate impossible locations, thereby converging faster to accurate locations [38].

A point that we overlooked in TIGR is that the load in networks in general, and in sensor networks in particular, is often imbalanced a fact that was . By the nature of sensor networks, local events can create local network overloads. It might be possible to better handle such cases with random scheduling functions. Specifically, such scheduling functions would give priority to loaded areas until they return to normal operation. Yet, one needs to ensure that such functions will not create routing deadlocks.

In case of deterministic scheduling functions, nodes failure or empty grid cells can turn much of the network disconnected. Handling these cases is left for future work. It involves devising adaptable scheduling functions and modifications to the routing algorithm to overcome such failures.

Few of the new emerging devices support multiple frequencies communication and protocols (e.g. 802.11b and 802.11g) which can be used as a tool for more sophisticated parallel scheduling functions.

As for the Location Service scheme, it may be possible to apply it to existing grid based Location Services such as LLS [1] and GLS [24].
Bibliography


מודעות למעיקום ברשויות אד-הוק

גיא קורלנצ

Technion - Computer Science Department - M.Sc. Thesis MSC-2005-01 - 2005
מידע על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעי המחשב

גיא קורלנדי

הOrth לשטט התכניין – מרכז טכנולוגיה לישראל
המחקר罡ע הבחינהו ו.راعי פרידמן בפנדלות המ démarche.
אני מודה לטענויות על התמיכות הטכניות והדרכה בשיתוף פעילתי.
תוכל אוניברסלי

tekir

CHAPTER 1

1.1 המטרות של המחקר
1.2 אלגוריתמים בסיסיים למש动生成
1.2.1 אלגוריתם בסיסי למש动生成
1.2.2 אלגוריתם בסיסי למש动生成
1.3 מבנה העיבוד של המחקר

CHAPTER 2

טעויות הקשורים
2.1 מיקום מיקום
2.2 ניתוב
2.3 שירת מיקום
2.4 שימור אגרונומי

CHAPTER 3

MIKOM יישום עם인데יפיזים קבוציים (RLISE)
3.1 תקציר
3.1.1 {}
3.1.2 {}
3.1.3 {}
3.1.4 {}
3.1.5 {}
3.1.6 {}
3.1.7 {}
3.1.8 {}
3.2 טעויות
3.2.1 {}
3.2.2 {}

CHAPTER 4

4.2.1 {}
4.2.2 {}

Technion - Computer Science Department - M.Sc. Thesis  MSC-2005-01 - 2005

( RLISE )
4.1 General overview

4.1.1 TIGR

4.1.2 Model of operation

4.1.3 Analysis and results

4.1.4 Functions of the system

4.2 TIGR

4.2.1 Random functions

4.2.2 Functions of the system

4.2.3 Functions of the system

4.2.4 Functions of the system

4.3 Algorithms

4.4 Conclusions

4.5 Session summary

5 Bibliography
רשימה תבליאות ויזורים

1.1 מודל רשת

12 מכנון מצבים – RLISE
14 מטריאנס ווקר יזוי – המקרה הטריוויאלי
14 מטריאנס ווקר יזוי – המקרה הטריוויאלי
17 היוון פ스וט של מסב חדר (ראשת שירוט)
17 היוון מטבבים של מדד (ראה שירוט)
18 היוון מטבבים של מדד (ראה שירוט)
20 קורח יזוי – ביטוי מצבי אל אפערית עיון שיפור צבעי אזור
20 קורח יזוי – ביטוי מצבי אל אפערית עיון שיפור צבעי אזור
23 חתך לתופס במלון תורשת
23 חתך לתופס במלון תורשת
25 חתך לתופס במלון תורשת
25 חתך לתופס במלון תורשת
27 חתך לתופס במלון תורשת
27 חתך לתופס במלון תורשת
29 דיסק החידה
29 דיסק החידה
30 עין דיסק
30 עין דיסק
35 פונקציות תומר
35 פונקציות תומר
37 פונקציות תומר
37 פונקציות תומר
38 פונקציות תומר
38 פונקציות תומר
39 פונקציות תומר
39 פונקציות תומר
42 פונקציות תומר
42 פונקציות תומר
4.1 f(4,4,3,3)
4.1 f(4,4,3,3)
4.2 f(4,4,3,3)
4.2 f(4,4,3,3)
4.3 f(4,4,3,3)
4.3 f(4,4,3,3)
4.4 f(4,4,3,3)
4.4 f(4,4,3,3)
4.5 f(4,4,3,3)
4.5 f(4,4,3,3)
4.6 f(4,4,3,3)
4.6 f(4,4,3,3)
4.7 f(4,4,3,3)
4.7 f(4,4,3,3)
4.8 f(3,3,1,1)
4.8 f(3,3,1,1)
4.9 B2B f(3,3,1,1)
4.9 B2B f(3,3,1,1)
4.10 מיקוד תומר
4.10 מיקוד תומר
4.11 מקוד תומר
4.11 מקוד תומר
4.12 שירוט מיקוד עם ציון את
4.12 שירוט מיקוד עם ציון את
4.14 ניצולת הרשת מול פריקת אנרגיה
תקציר

רשות א-אוד-ווק מאפרעות ומגמות לצרור בואו ספוטטים תקירות אלחוטיות ידניות. ...

רשות א-אוד-ווק וכלל הספקלקיציון וחיבור רצי מתכון, סג כאר המיתר מצロックל צולח של רשת...

בעלת תשתית קובעה, סכודת תקשורת השכחת ביוור של רשות א-אוד-ווק כלל רשתות א-אוד-ווק.

ニュート (MANET): הוק כללו תוארו דוגמא-דוגמאות אופעות Vân שרייה לשי מושל שלי רשת אוד-ש פלשתית...

חיבור רצי לטווח של רשות אוד-ש, הוק יכולה לספק למכשורי אוסטר, רשתות מחשבים ידניים, חיבורים ט }))טיסטים ורשתות...

שליטה ופיקוח.

מikım: קים ממון של יורים בסיסים ואפליקציות הכליות טלוריות ממידת על מיקום גאוגרפי.

לודגנאות, ישות מבוסס עד יוןורפיג יובל לוחות על יער פנימי ולא יער, אוסף נייר הפרחת

רוב יער. יערעל יער, ברשותﻴיתוישנא, יידעו המיקום של זומתי הורות על מונת לפעבר יער.

טיפוף נון של קירות התשישה. ידע ממספר בכיוון קיים ממעטפת לקיMocks לקוי מילמוד את

מikımם. הפרשות ביור היה ליידי את הצמות על מקלת GPS. ואכזף האופציה אם הקור

וחיסה, אל תומרי יימה (למשל, רוב המחשבים הנישאים ומוחשב וכדי אל מיידי ב

GPS דרuxe קרreira, זכלו את עדב תמייב ארבעים וביתם במרום. אופשרר דועות

(Israellands או Stationary emitters) (Landmarks או Stationary emitters) האורח איה להנין קימיה של מפסטר תחת הסיס

מikımם המודים ובוגרונות צמתיו ארציים יכלו למזרח את מיקומים על שיווק מצולח הטויאנגלציה

(Israellands או Stationary emitters) (Landmarks או Stationary emitters) האורח איה להנין קימיה של מפסטר תחת הסיס

בביס, דרישה ידריש ממספר ברי להנין ביס ופסאי להפרעות חתונה מיקומים אוד-ווק.

(APS) תחת המיקום של בקיה לא-כ, הבכלה שלהם ישות אל-כ, דרישה ידריש ממספר מטריקות מיקום אוד-ווק.

.DV-position 1  DV-radial - Euclidean ,DV-hop propagation-בכלה של תארו ביס, מוניות כל זום יידע את המסלל הקצר-ביוור להננות בפסי,.

вшיתות לשון שיתות אל-כ, מוניות כל זום יידע את המסלל הקצר-ביוור להננות בפסי.
Asher Zuritau, M.Sc. Thesis

Technion - Computer Science Department - M.Sc. Thesis
MSC-2005-01 - 2005

Abstract:

The basic proposition of this work is to present a method for calculating the distance between two points using the Euclidean distance. The method is called DV-radial - Euclidean, and its main advantage is that it is fast and accurate. The method is based on the concept of trilateration, which is a technique for determining the position of an object using the distances from three known points.

In this work, we propose a new method called DV-position, which is an improvement over the existing method. Our method is more accurate and efficient, and it can be used in real-time applications. We compare our method with the existing methods and show that it outperforms them in terms of accuracy and speed.

We also propose a new method for calculating the position of an object using a network of sensors. Our method is based on the concept of PSM (Power Saving Mode) and it can be used in wireless networks. We compare our method with the existing methods and show that it is more efficient and accurate.

In addition, we present a new method for calculating the position of an object using a network of beacons. Our method is based on the concept of trilateration and it is more accurate than the existing methods. We compare our method with the existing methods and show that it outperforms them in terms of accuracy.

Finally, we present a new method for calculating the position of an object using a network of cameras. Our method is based on the concept of computer vision and it is more accurate than the existing methods. We compare our method with the existing methods and show that it outperforms them in terms of accuracy.
The information-bearing: A task of determining the best decision is to be made to achieve a better performance in the network. The best decision is achieved when the WiFi network is used with multiple antennas, and the decision-making is performed using the Received Signal Strength Indicator (RSSI).

In the information-bearing task, the WiFi network is used to determine the best decision. The decision is achieved when the WiFi network is used with multiple antennas, and the decision-making is performed using the Received Signal Strength Indicator (RSSI).

Technion - Computer Science Department - M.Sc. Thesis  MSC-2005-01 - 2005

3
הسوق, ואנ מחיצים שרות מיקום עילה chaud המותאמ ליתוף מבטסיס סרי. שירת המיקום
מועכל את האבחנה ככ כל איש זeam צרי לדעט על מות לתוך הדעת על סירג או תכון של צומת
 {. המטימה (מולעולטאוםויומוספמא). בחקסס על האבחנה ככ, שירת המיקום יכל לחקסן בכרעה
משמעתיית את מספר ההדיעה את כל זאמ אגון תלת למקרה של תנועה הם אד בכרעה
לשמוח. כי לע כל, ישthritis המיקום שלן, החיפוש והמי נמקום תמי. דיבר עם מפואר למגנום
הערה לעך את משולג היותו בחתמה להיעשות לא עלה נספה.

 צריך את מוגר ממסאת את עובד בבדול לחם של המווט חורבהת למודל ושכבות של
הרשת. המממד החפש医学院 את מודל השכבות הבסיסית דוב שאורח מחוקים מיציגים את
הערת והרשת. מחוקים: Positioning 1 Location Service ,Geo-Routing ,Scheduler
봇 Nhật וממסאת Mac Scheduler- הממסת והרחבת לשכבות 1 השכרת 하
ולשנת החפש医学院 ישמור אורגיית.