COLORADO STATE UNIVERSITY

December 13, 1998

WE HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER OUR SUPERVISION BY GEORGIOS VARSAMOPOULOS ENTITLED AN ADAPTIVE LOCATION MANAGEMENT SCHEME FOR PCS NETWORKS BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

Committee on Graduate Work

Committee Member

Committee Member

Coadviser

Coadviser

Department Head

THESIS

AN ADAPTIVE LOCATION MANAGEMENT SCHEME FOR PCS NETWORKS

Submitted by
Georgios Varsamopoulos
Department of Computer Science

In partial fulfillment of the requirements
for the Degree of Master of Science
Colorado State University
Fort Collins, Colorado
Fall 1999

ABSTRACT OF THESIS

AN ADAPTIVE LOCATION MANAGEMENT SCHEME FOR PCS NETWORKS

Location management is an essential service in mobile networks. It provides mechanisms for recording and querying location of mobile units in the network. This is needed mainly for establishing calls to mobile units. In PCS networks, location management protocols such as IS-41 and GSM use statically defined Registration Areas (RAs). A mobile unit informs its location to the network whenever it moves from one registration area to another. This thesis proposes an extension to PCS location management protocol by introducing the concept of dynamically overlapped registration areas. Also, analysis and simulation show that by dynamically adapting the registration areas to aggregate mobility pattern of the mobile units, we can greatly reduce the number of location updates by mobiles. Further, the cost of adapting the registration areas is shown to be low in terms of memory and communication requirements.

Georgios Varsamopoulos
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
Fall 1999

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Sandeep K. Gupta, heartily, who helped me do the research and write this thesis. He is a man with a lot of patience. Also I want to thank my co-advisor, Dr. Pradip Srimani, who has given me help and advice in my life in Fort Collins. Lastly, I want to thank the committee members, Dr. Anura Jayasumana and Dr. Walid Najjar, who have contributed into making one of my dreams a reality.

DEDICATION

This thesis is dedicated to my sisters, $Μαρία$ (*Maria*) and $Ανθή$ (*Anthee*).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Personal communication services (PCS) allow mobile users with wireless terminals to receive calls irrespective of their location in a seamless manner. PCS networks have a cellular architecture: the geographical area is divided into cells with one base station per cell. The mobile user's portable terminals communicate via wireless with fixed radio ports in the base station. Delivering calls to the mobile terminals requires that the current location (point of attachment) of the mobile terminal be known in order for the network to route the calls to the mobile terminal. The task of tracking the location of mobile terminals is known as location management (or tracking). Location mangament involves two basic operations: update and search. A mobile terminal updates its location periodically or otherwise. Whenever a call needs to be delivered to the mobile, the network uses the last know location of the mobile terminal to search for the mobile in the vicinity of that area. This may involve paging for the mobile terminal in certain neighborhood of the last known location of the mobile terminal. As has been demonstrated in several works, there is a tradeoff between update and search costs. Strategy which tries to reduce one cost tends to decrease the other, and vice versa. For example, if a mobile updates it's location more often then it can be searched more easily.

One of the strategy used in conventional systems to balance the cost of update and search is the use of *registration area* (RA) approach to location tracking. The geographical area is divided into several registration area, where each registration area consists of several cells. The system tracks a mobile terminals registration area instead of its cell. Whenever a mobile terminal crosses from one registration area to another it informs its new location to the system. To setup a call to a mobile terminal, the system pages all the cells in the registration area to find the current cell of the mobile terminal. A database called *location register* (LR) is associated

with each registration area to keep information about the mobiles currently registered in that registration area. There are two common standards for location management: IS-41 [3] and GSM [6]. IS-41 is used in North America and GSM is used in Europe. Both these standards use registration areas along with a two level hierarchy of location registers. The hierarchy consists of home location register (HLR) and visitor location registers (VLRs). Location information of a mobile node is kept at both its current location registrar, i.e. its VLR, and its permanent home location registrar. This facilitates locating non local mobiles during call delivery: the home location registrar of the mobile unit is contacted to find its current location. Hence, the two level hierarchy scheme increases the location update cost while reducing the location search/call delivery cost.

In this work we deal with the following problem. Statically defined registration areas cannot cover the entire movement pattern of the mobile users, sometimes, not even a good portion of it. This results in frequent inter-RA moves (location updates), which are costly. This is especially true when the mobile is far from its home RA, since this may require communication across the entire network depending upon the relative position of the mobile host and its HLR. However, if we try to use fewer RAs, each covering wider geographical area, then the load on each location registrar increases. Further, an RA configuration defined to give good performance for a certain mobility pattern may not perform well when the mobility pattern changes. Hence, the problem of identifying registration areas is very challenging.

Overlapped registration area has recently been used to address this problem [4]. In particular, overlapped RAs can achieve the following: (a) increase the area handled by each LR, effectively reducing the number of inter-RA hand-offs, and (b) keep the number of users (mobile hosts) handled by each LR same as in the case of non-overlapped RAs. In this thesis we propose a scheme that dynamically adapts to the mobility patterns and determines the area that should be covered by each LR, thus keeping the signaling overhead low. The scheme proposed in this thesis keeps track of aggregate user mobility and uses this information to decide the degree of overlapping between neighboring registration areas which will reduce the number of registration area hand-offs while keeping the call delivery cost down. The decision to adjust the registration area is done in a distributed manner and the protocol ensures that if the mobility pattern are stable then the registration areas would also stabilize. The memory and communication overhead of the proposed scheme is low making the scheme scalable. We have simulated our protocol for various mobility patterns. Our simulation results

show that the idea of overlapping is good when there is locality in the overall movement of the mobile users. However, this scheme provides little improvement when it is applied in highway-like movement patterns.

In the rest of the thesis we first discuss some related work in the literature. We then present our system model and the proposed algorithm for dynamically adjusting the registration areas. Finally, we present our simulation model and results.

Chapter 2

Related Work

Most of the previous work has put effort in reducing the registration cost by introducing caching schemes or increasing the number of layers in the hierarchy of the location scheme. They are generally of a static character, which means that they have a fixed number of partitions, their shape is fixed, and the signaling (or hierarchy) is static. We will give a brief comparison of the various flavors of location management schemes that have been proposed in the literature.

Bejerano and Cidon [1] are based on the assumption that the network distance (network delay) can be bounded by the product of the geographical distance into a (reasonable) constant: They define a logarithmic number of levels of RAs that have a radius twice as much as the ones at the level right below. When a move takes place, it traverses through a set of low level RAs. These RAs are organized into a higher layer of RA sets, or 2nd-level RAs. The second level RAs are organized into 3rd-level RAs, and so on, till we reach an abstraction to one high level RA. At each layer there is overlapping among the RAs of that layer, the overlapping part is as big as the basic radius of the RA. Thanks to the logarithmic number of layers plus the overlapping, the scheme has a very good average behavior, but it has a substantial delay in the worst case. It also has a storage space drawback, cause it wants a logarithmic number of profile replicates per user among the database infrastructure.

Ho and Akyildiz [4] have introduced a 3 level database hierarchy with the number of number levels dynamically adapting to the user pattern (call to mobility ratio - CMR). The hierarchy introduces the concept of Directory Areas (DA), a clustering of Registration Areas, as they are known in IS-41. Each DA has a Directory Register that logically lies between the HLR and the VLR. Usually a registration of an MT from a VLR a

to VLR b is not propagated to the HLR, as long as the RAs a and b are within the same DA. Also, usually the HLR holds the id of the DR that keeps track of the MTs mobility, not the id of the LR. Sometimes though, the HLR holds the id of the VLR, when the call and mobility patterns of the MT allow that, so the DR is skipped in a location search (call delivery), thus giving a dynamic (variable levels) property to the algorithm. They have also introduced the idea of registration area overlapping in the static sense, that is, the partitioning of the RA, as well as the overlapped portions are statically defined. Furthermore, they don't do an extended cost analysis on the overlapping version, neither do they propose a concrete algorithm on that version.

Kryukova, Massingill and Sanders, working in a theoretical level in [5], have proposed an integrated scheme of location management and routing. Their scheme is based on an acyclic undirected graph on which they maintain routing pointers. Due to the acyclicity of the graph it is easy to maintain a coherent chain of routing pointers to the mobile stations, without worrying about cycles in the route. Unrealistic assumptions are (1) the absence of cycles in the graph and (2) the assumption that the mobiles move between nodes that have a immediate network connection to each other.

Rajagopalan and Badrinath [8] have proposed a hybrid scheme on the Mobile-IP architecture that uses either the home-agent/visitor-agent paradigm or the direct location update between parties that communicate frequently. They compute a couple of values per pair of user (the computation is distributed, every host is calculating for its part) and decide what scheme to use.

Das and Sen [2] have an entirely different perspective to the problem. They don't use databases; they assume a tree-like Network organization, in which every node is a 'server'. When ever a location query reaches a server, the subtree under that server is paged. The location management here is nearly existing, as it's a scheme that rather tries to guess the location of each user than try to keep track of it.

Prakash and Singhal [7] have proposed a profile replication scheme. Their idea is to have the information of location replicated among the location servers, with each set of updated servers to intersect with each set of queried server, so that everybody can locate everybody else.

Table 2.1: **Comparison among location management schemes found in literature.**

Paper	Model	Hierarchical Database Organization	Overlapping	Replication	Caching	Comments
Bejerano [1]	network distance is depended on geographical distance	(fixed number of levels (log n))	yes (at all levels, static)	no	no	Efficient at the average case, very bad at worst case. Applicability issues.
Akyildiz [4]	yes (based on IS-41)	yes (3 levels at max, varying per user)	yes (at RA level, fixed)	no	no	no extended analysis for overlapping
Kryukova [5]	yes (undirected acyclic graph)	no	no	no	no	integrates routing and location management
Rajagopalan [8]	yes (based on IP)	yes (based on Mobile IP, varying per user)	no	no	yes	It's a hybridic scheme for either updating/invalidating location on specific pairs of users
Das [2]	An arbitrary network of MSCs	yes (a tree-like infrastructure of servers)	no	no	no	it's a search based scheme
Quorum [7]	yes (based on IS-41)	yes (two level)	no	yes, dynamic	no	Overhead to update multiple LRs might 'hurt' the network.

Chapter 3

Dynamic Overlapping Scheme

The dynamic overlapping scheme, as it is defined in this thesis, is based on cost analysis of the system model. The analysis of the system associates the calls and moves with individual cells and Registration Areas. It tries to compute the cost of the same call and mobility patterns under different system configurations and decides to use the least costly one.

We will first define the system model, introduce some system signaling constants, then derive the basic formula that determines whether there should be a change in the configuration of the system, then introduce the algorithm built on that formula.

3.1 System Model

As shown in Figure 3.1, a cellular communication network consists of an array of hexagonal communication cells. We assume that there are $\mathcal{N}$ cells in the system numbered from 1 to $\mathcal{N}$. Each cell except the boundary cells have six neighbors. A base station (BS) is in-charge of a cell. Every base station is connected to a Mobile Switching Station (MSS). The MSS are connected to an interconnection network; hence each MSS can communicate with other MSSs in the system. Each MSS is in charge of a set of cells called its Registration Area (RA). We assume that, initially, the RA of a MSS consists of all the cells whose BS are directly connected to the MSS. We call these cells as **core cells** of that MSS. Associated with each MSS are two location databases: Visitor Location Registrar (VLR) and Home Location Registrar (HLR). A VLR of a MSS keeps information about all the mobile hosts (MHs) in its registration area. To facilitate search of a MH, each MH is statically associated with a Home Location Registrar (HLR), which keeps information about the current location of the MH, i.e. the id of the MSS in whose RA the MH is currently residing. Hence, a location registrar maps an

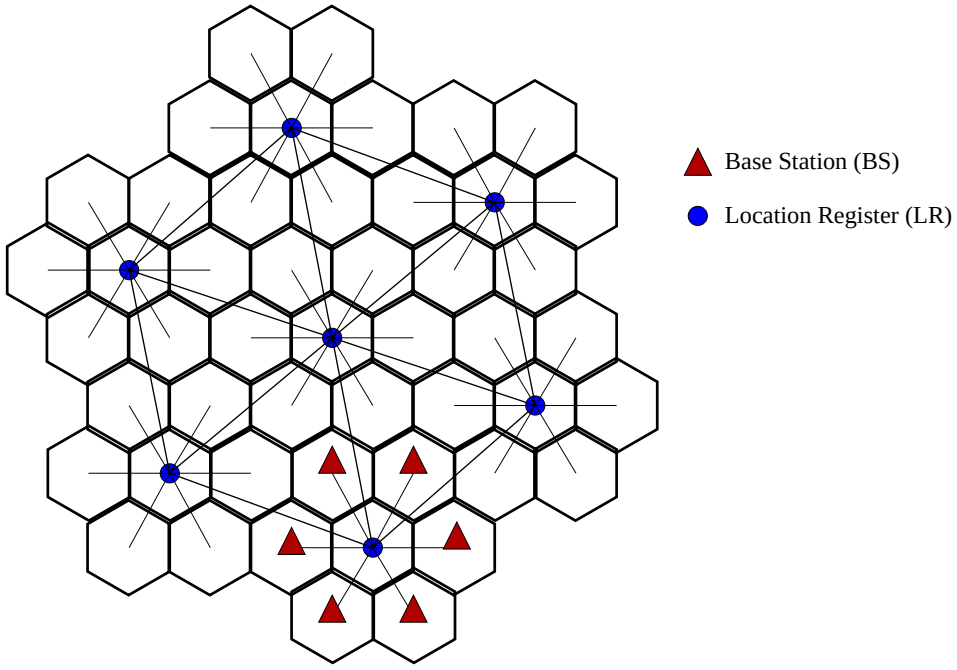

Figure 3.1: **The System Model (Cells are organized into clusters (here in clusters of 7) forming the Registration Areas (RAs). The MSSs (LRs), located at the center of each RA, are connected through a hexagonal mesh.)**

MH id to an MSS id.

When an MH which is actively involved in communication moves from one cell, say c_{old}, to another cell, say c_{new}, a *hand-off* takes place, after which the MH receives and sends all its communication via base station of c_{new} instead of base station of c_{old}. This involves signaling between MH, BS, and MSS and resource allocation and deallocation. Further, if the two cells c_{old} and c_{new} belong to different registration areas then the HLR of the MH is informed of the new location of the MH. Further, the VLR of MHs old registration area and its new registration area also need to be updated, i.e. the MHs record in the old RA's VLR is deleted and a record for the MH in the VLR of its new RA is added. Hence, **inter-RA hand-offs** are more expensive than the hand-off which are **intra-RA**.

Here we define communication and computation costs which will be later used in the thesis.

- δ_{cv}: communication cost between a BS and its MSS.
- δ_{vv}: communication cost between two neighboring MSS.
- ρ: average number of hops between two MSSs. Hence, the average communication cost between two

arbitrary MSS is $\rho\delta_{vv}$.

- γ: average number of hops between an MSS and the HLR of an MH in question. Hence, the average communication cost to an HLR is $\gamma\delta_{vv}$.
- α_c: cost of processing a hand-off/call-delivery at a BS.
- α_v: cost of processing a hand-off/call-delivery at a MSS.

We assume that each BS and MSS are individually addressable i.e. an MSS/BS can directly send a message to another MSS/BS.

3.2 Dynamically Adjusting Size of Registration Areas

In order to facilitate orderly growth and shrinking of RAs, an MSS only includes or excludes cells from its RA's current boundary. The algorithm uses the following two types of boundaries:

Definition 1 *A cell $c \in RA$ is in the internal boundary of that RA iff $\exists d \notin RA$ such that cell d is a neighbor of cell c. The internal boundary of RA k will be denoted as $I\_Boundary(k)$.*

Definition 2 *A cell $d \notin RA$ is in the external boundary of that RA iff $\exists c \in RA$ such that cell d is a neighbor of cell c. The external boundary of RA k will be denoted as $E\_Boundary(k)$.*

All the cells in $I\_Boundary(k)$ are candidate cells for *exclusion* from the RA k and all the cells in $E\_Boundary(k)$ are candidate cells for *inclusion* into RA k. However, since core cells cannot be deleted from an RA, only cells in $I\_Boundary(k) - Core(k)$ can be excluded from an RA. The decision to include or exclude a candidate cell is based on whether the resulting configuration will have a lower overall system load. For a given system configuration C, mobility pattern $\mathcal{M}$, and call pattern $\mathcal{C}$, we define system load, $SystemLoad(C, \mathcal{M}, \mathcal{C})$ as the combined signaling load (in terms of message time complexity) as a result of all the hand-offs due to $\mathcal{M}$ and call-deliveries due to $\mathcal{C}$. For the purpose of making cell inclusion-exclusion decisions, we partition the entire system load into loads per MSS, i.e.

$$SystemLoad(C, \mathcal{M}, \mathcal{C}) \quad = \quad \sum_{k \in \mathcal{R}} Load(k, \mathcal{M}, \mathcal{C}), \tag{3.1}$$

where $Load(k, \mathcal{M}, \mathcal{C})$ is the signaling load attributed to MSS k. We note here that in case of inter_RA hand-offs and call-deliveries we split the signaling overhead equally between the two MSSs involved.

We next analyze the $Load(k)$ of MSS k for a fixed mobility pattern $\mathcal{M}$ and calling pattern $\mathcal{C}$. The load of MSS k, $Load(k)$, is defined to be the sum of follow ing five components:

1. $Cost_{intra\_RA\_calls}(k)$: signaling cost for performing intra-RA calls between MHs registered with RA k,

2. $Cost_{inter\_RA\_calls}(k)$: signaling load for performing inter-RA calls between a MH registered with RA k and another MH which is not registered with RA k,

3. $Cost_{updates\_searches}(k)$: signaling load for performing updates and searches for roaming MHs whose HLR is at MSS k.

4. $Cost_{intra\_RA\_moves}(k)$: signaling load for performing intra-RA hand-offs for an MH registered with RA k and moving from one cell to another cell in RA k , and

5. $Cost_{inter\_RA\_moves}(k)$: signaling load for performing inter-RA hand-offs for MHs moving into or out-of RA k.

That is to say,

$$\begin{aligned} Load(k) \quad = \quad & Cost_{intra\_RA\_calls}(k) + \\ & Cost_{inter\_RA\_calls}(k) + \\ & Cost_{updates\_searches}(k) + \\ & Cost_{intra\_RA\_moves}(k) + \\ & Cost_{inter\_RA\_moves}(k). \end{aligned} \tag{3.2}$$

Before analyzing each of the above five components, we define the following notations for the give configuration C, mobility pattern $\mathcal{M}$ and call pattern $\mathcal{C}$:

- $n_{calls}(k, i, m, j)$: is the total number of calls from MHs registered with RA k in cell i to MHs registered with RA m in cell j.

- $n_{moves}(k,i,m,j)$: is the total number of moves by MHs registered with RA k in cell i to cell j of RA m.

- $def(i)$: is the identifier of RA m such that cell i is in $Core(m)$. We refer to RA m as the default RA of cell i.

- $R(i)$: is the set of identifiers of all RAs which include cell i.

1. **Cost of Intra-RA calls:** Intra-RA calls have different cost for core-to-core, core-to-noncore and noncore-to-core calls. Therefore $C_{intra-ra\ calls}$ is the sum of following four components:

$$\begin{aligned} Cost_{intra\_RA\_calls}(k) \quad = \quad & \sum_{i,j \in Core(k)} n_{calls}(k,i,k,j) c_{intra-call-core-core} + \\ & \sum_{\substack{i \in Core(k) \\ j \in RA(k) - Core(k)}} n_{calls}(k,i,k,j) c_{intra-call-core-noncore} + \\ & \sum_{\substack{i \in RA(k) - Core(k) \\ j \in Core(k)}} n_{calls}(k,i,k,j) c_{intra-call-noncore-core} + \\ & \sum_{i,j \in RA(k) - Core(k)} n_{calls}(k,i,k,j) c_{intra-call-noncore-noncore} \end{aligned}$$

2. **Cost of Inter-RA calls:** Inter-RA calls have different cost, depending on whether they are incoming or outgoing,and whether they are between MH s in core and non-core cells.

$$\begin{aligned} Cost_{inter\_RA\_calls}(k) \quad = \quad & \sum_{i \in Core(k)} \sum_{m \in \mathcal{R}, m \neq k} \sum_{j \in RA(m)} n_{calls}(k,i,m,j) c_{inter-outgoing-call-core} + \\ & \sum_{i \in RA(k) - Core(k)} \sum_{m \in \mathcal{R}, m \neq k} \sum_{j \in RA(m)} n_{calls}(k,i,m,j) c_{inter-outgoing-call-noncore} + \\ & \sum_{i \in Core(k)} \sum_{m \in \mathcal{R}, m \neq k} \sum_{j \in RA(m)} n_{calls}(m,j,k,i) c_{inter-incoming-call-core} + \\ & \sum_{i \in RA(k) - Core(k)} \sum_{m \in \mathcal{R}, m \neq k} \sum_{j \in RA(m)} n_{calls}(m,j,k,i) c_{inter-incoming-call-noncore} \end{aligned}$$

3. **HLR updates and searches:**

$$Cost_{updates\_searches}(k) \quad = \quad \sum_{m \in \mathcal{R}, m \neq k} n_{searches}(k,m) c_{search} + \sum_{m \in \mathcal{R}, m \neq k} n_{updates}(k,m) c_{update} \quad (3.3)$$

4. **Cost of Intra-RA hand-offs:** Inter-RA hand-offs have different cost, depending on whether they are incoming or outgoing, or whether they refer to a core or noncore cell.

$$Cost_{intra\_RA\_moves}(k) \quad = \quad \sum_{i,j \in Core(k)} n_{moves}(k,i,k,j) c_{intra-move-core-core} +$$

$$\sum_{\substack{i \in Core(k) \\ j \in RA(k) - Core(k)}} n_{moves}(k,i,k,j) c_{intra-move-core-noncore} +$$
$$\sum_{\substack{i \in RA(k) - Core(k) \\ j \in Core(k)}} n_{moves}(k,i,k,j) c_{intra-move-noncore-core} +$$
$$\sum_{i,j \in RA(k) - Core(k)} n_{moves}(k,i,k,j) c_{intra-move-noncore-noncore}$$

5. **Cost of Inter-RA hand-offs:** Inter-RA calls have different cost, depending on whether they are incoming or outgoing, or whether they refer to a core or noncore cell.

$$\begin{aligned} C_{inter-ra\ moves} = & \sum_{i \in Core(k)} \sum_{j \notin RA(k)} n_{moves}(k,i,def(j),j) c_{inter-outgoing-move-core} + \\ & \sum_{i \in RA(k) - Core(k)} \sum_{j \notin RA(k)} n_{moves}(k,i,def(j),j) c_{inter-outgoing-move-noncore} + \\ & \sum_{m \in \Re} \sum_{j \in RA(m)} (m,j,k,i) \sum_{i \in RA(k) - Core(k)} n_{moves} c_{inter-incoming-move-core} \end{aligned}$$

We define $Include(C,k,a)$ to be the configuration resulting from including cell a in to RA k of configuration C. Similarly, we define $Exclude(C,k,a)$ to be the configuration resulting from excluding cell a (where $a \notin Core(k)$) from RA k of configuration C. Let $C_{in} = Include(C,k,a)$, then

$$Load(C_{in},\mathcal{M},\mathcal{C}) = Load(C,\mathcal{M},\mathcal{C}) + Cost_{in}(k,a) - Cost_{ex}(k,a) \quad (3.4)$$

Similarly, let $C_{in} = Include(C,k,a)$, then

$$Load(C_{ex},\mathcal{M},\mathcal{C}) = Load(C,\mathcal{M},\mathcal{C}) + Cost_{in}(k,a) - Cost_{ex}(k,a) \quad (3.5)$$

3.3 Inclusion and Exclusion

When an inclusion takes place, the above values ($Cost_{intra\_RA\_calls}(k)$, $Cost_{inter\_RA\_calls}(k)$, $Cost_{updates\_searches}(k)$, $Cost_{intra\_RA\_moves}(k)$, $Cost_{inter\_RA\_moves}(k)$) change. The change in the values is the cost difference of the two configurations C, C'. As we will see below, by the inclusion of a cell into a Registration Area, there are values that increase and values that decrease. We gather the increasing quantities in the value $Cost_{in}(k,a)$ and the decreasing quantities in the value $Cost_{ex}(k,a)$. The difference $Cost_{in}(k,a) - Cost_{ex}(k,a)$ determines if the inclusion in profitable (negative values) or bad (positive values) .

1. **Cost of Intra-RA calls:** By adding the cell a to the RA k, all the calls to a from cells that were already Included to k become intra-R A calls The same applies for the calls that derive from MHs of k in a to MHs of k in the rest of the cells. So, we have an increase to the intra-RA calls by

$$\begin{aligned} Increase_{intra\_RA\_calls}(k,a)) \quad = \quad & \sum_{i \in Core(k)} n_{calls}(k,i,k,a) c_{intra-call-core-noncore} + \\ & \sum_{i \in RA(k)-Core(k)} n_{calls}(k,i,k,a) c_{intra-call-noncore-noncore} + \\ & \sum_{i \in Core(k)} n_{calls}(k,a,k,i) c_{intra-call-noncore-core} + \\ & \sum_{i,j \in RA(k)-Core(k)} n_{calls}(k,a,k,i) c_{intra-call-noncore-noncore} \end{aligned}$$

2. **Cost of Inter-RA calls:** Since all calls to users of k in cell a become intra-RA calls, we need to subtract them from the inter-RA calls. Furthermore, users of k in a make and received inter-RA calls. So here, we have both an increase and a decrease:

$$\begin{aligned} Increase_{inter\_RA\_calls}(k,a) \quad = \quad & \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{calls}(k,a,m,i) c_{inter-outgoing-call-noncore} + \\ & \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{calls}(m,i,k,a) c_{inter-outgoing-call-noncore} + \end{aligned}$$

$$\begin{aligned} Decrease_{inter\_RA\_calls}(k,a) \quad = \quad & \sum_{i \in Core(k)} n_{calls}(k,i,k,a) c_{inter-outgoing-call-core} + \\ & \sum_{i \in RA(k)-Core(k)} n_{calls}(k,i,k,a) c_{inter-outgoing-call-noncore} + \\ & \sum_{i \in Core(k)} n_{calls}(k,a,k,i) c_{inter-incoming-call-core} + \\ & \sum_{i \in RA(k)-Core(k)} n_{calls}(k,a,k,i) c_{inter-incoming-call-noncore} \end{aligned}$$

3. **HLR updates and searches:** The updates and searches are not expected to change immediately by the inclusion of a cell to the RA. Nevertheless, the reduction of the HLR updates and searches is something we are focusing on, and simulations show that. For this analysis, we consider the cost of updates and searches the same.

4. **Cost of Intra-RA hand-offs:** By adding cell a to the RA k, all the hand-offs from the rest of the cells in k become intra-RA hand-offs. The same applies to the hand-offs from cell a to the rest cells in k.

$$\begin{aligned} Increase_{intra\_RA\_moves}(k,a) &= \sum_{i \in Core(k)} n_{moves}(k,i,k,a) c_{intra-move-core-noncore} + \\ &\sum_{i \in All(k)-Core(k)} n_{moves}(k,i,k,a) c_{intra-move-noncore-noncore} + \\ &\sum_{i \in Core(k)} n_{moves}(k,a,k,i) c_{intra-move-noncore-core} + \\ &\sum_{i \in RA(k)-Core(k)} n_{moves}(k,a,k,i) c_{intra-move-noncore-noncore} \end{aligned}$$

5. **Cost of Inter-RA hand-offs:** For Inter-RA hand-offs, we have to subtract the hand-offs between a and k, cause they are intra-RA now. Furthermore, we need to add the inter-RA hand-offs that happen by users of k in a that move out of k.

$$Increase_{inter-ra\ moves}(k,a) = \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{moves}(k,a,m,i) c_{inter-outgoing-move-noncore}$$

$$\begin{aligned} Decrease_{inter-ra\ moves}(k,a) &= \sum_{i \in Core(k)} n_{moves}(k,i,def(a),a) c_{inter-outgoing-move-core} + \\ &\sum_{i \in RA(k)-Core(k)} n_{moves}(k,i,def(a),a) c_{inter-outgoing-move-noncore} \end{aligned}$$

Where $def(i)$ is the default RA of cell i.

We put the values together:

$$\begin{aligned} Cost_{in}(k,a) &= Increase_{intra\_RA\_calls}(k,a) + Increase_{inter\_RA\_calls}(k,a) + \\ &\quad Increase_{intra\_RA\_moves}(k,a) + Increase_{inter-ra\ moves}(k,a) \end{aligned}$$

$$Cost_{ex}(k,a) = Decrease_{inter\_RA\_calls}(k,a) + Decrease_{inter-ra\ moves}(k,a)$$

For the exclusions process, it is not hard to see that those two values switch roles.

3.4 Algorithm

The basic idea behind the algorithm is to compute the $Cost_{in} - Cost_{ex}$ difference for every applicable pair of (k,a) and decide whether to do inclusions, exclusions or maintain the same configuration. The computation

is distributed among the RAs; each RA computes the set of neighboring cells that are applicable for inclusion and the set of cells that are applicable for exclusion, then computes the differences and decides which cells to include and which cells to exclude.

In order the algorithm to compute the above costs, the system has to keep track of the n_{calls} values. Then each Registration Area LR queries for the values it needs and makes the expansion process independently of the other LRs decisions. In our distributed algorithm, the n_{calls} values are kept in a distributed manner by the BSs. At fixed intervals, the LRs send out DATA_REQ messages to all the candidate cells asking for the n_{calls} values. The BSs respond with the values in a DATA_ACK message. Then the LRs compute the $Cost_{in}, Cost_{ex}$ values for each candidate cell. If there is a change to take place, the LR sends an INCLUSION_NOTIF or an EXCLUSION_NOTIF message to the appropriate cells – note that in the same *reconfiguration period* the same RA might include some cells and exclude others. The queried cells respond with a INCLUSION_NOTIF or an EXCLUSION_NOTIF respectively. Lastly, before sending DATA_REQ to the candidate cells, each LR sends DATA_REQ to all the cells of its RA in order to compute the load of the RA. Below, we see the 3 components of the algorithm:

- **Module_MT**: In the MT, we use subscription and calling protocols similar to the IS-41 and GSM standards, with a few additions: when there is a registration, the MT sends some extra information, its current VLR id and its HLR id. The VLR id is used by the receiving BS to decide which LR to contact. If the VLR id that is sent is not one of the LR id's that service the BS, then we have a case of inter-RA registration. Otherwise, we have a simple intra-RA hand-off. The HLR id is used by the contacted LR to speedup the lookup procedure. A Pascal-like pseudocode is given by Figure 3.2

- **Module_BS**: In the BS, we use the same subscription and call-delivery protocols, with the subscription algorithm slightly altered to work with the multiplicity of servicing LRs. Also, also the BS carries an algorithm for the dynamic overlapping protocol. In the dynamic overlapping algorithm, there are two data structures and three functions. The data structures are a) DATA, that holds the id, the vector of servicing LR and the calling and hand-off rates for each serving LR. b) VALUES, which is like an "instance" of the DATA structure, which holds values specific to a servicing LR. This structure is sent back to the LR, when it queries for the rate values. The functions are: a) receive_DATA_REQ()

```
Module MT

send_REGISTRATION_NOTIF(new_cell,my_data);


Module BS

procedure receive_REGISTRATION_NOTIF(i,MT_data) is
begin
    if (MT_data.VLR_id ∈ f(My_id)) then
        send_REGISTRATION_NOTIF(MT_data.VLR_id,My_id,i);
    else
        send_REGISTRATION_NOTIF(DEFAULT_VLR,My_id,i);
end receive_REGISTRATION_NOTIF;
```

Figure 3.2: **Registration Algorithms for Base Station and Mobile Terminal**

which handles the DATA_REQ messages from a querying LR. It sends back and instance of VALUES. b) receive_INCLUSION_NOTIF() which handles the INCLUSION_NOTIF message, which notifies the BS that is included into an RA. c) receive_EXCLUSION_NOTIF() which handles the EXCLUSION_NOTIF message, which notifies the BS that is excluded from an RA. There is also one extra constant, the default LR id. The default LR is contacted in the case of an inter-RA registration.

- **Module_LR**: In the LR, the reconfiguration protocol has one data structure, DATA, and three real variables, cur_Cost, Threshold and Limit, that are used to decide whether the LR should include or exclude a candidate BS. The new functions are: a) call_reconfigure(), which acts like a background daemon process and calls the reconfigure() function b) reconfigure(), which computes the sets of the candidate cells (one set for inclusion. one set of exclusion), queries the candidate cells for a VALUE instance by sending a DATA_REQ message to each candidate cell, collects the DATE_REQ_ACK responses and computes which cells should be included, which ones should be excluded and which should retain their status. It then calls the excluded_marked_cells() and include_marked_cells() functions that sent the the exclusion and inclusion notifications respectively, and collects the acknowledgments of the notifications.

A Pascal-like pseudo-code of the algorithm is shown in Figure 3.4.

Module BS

structure $Data$ **is**
    integer my_id;
    integer set $Vlrs$;
    real $N\,[|V|]\,, Ntot$;
    real $Rc\,[|V|]\,, Rctot$;
    real A;
end $Data$;

structure $Values$ is
    int Nv, Nt;
    real Rcv, Rct;
    real $Rhov, Rhot$;
    real $Rhv, Rcinv$;
end $Values$;

procedure receive_DATA_REQ(**integer** $action\_id$, **integer** k) **is**
begin
    $Values.Nv = Data.N[k]$;
    $Values.Nt = Data.Ntot$;
    send(DATA_ACK($action\_id, Data.my\_id, Values$),$k$);
end receive_DATA_REQ;

procedure receive_INCLUSION_NOTIF(**integer** $action\_id$, **integer** k) **is**
begin
    $Data.Vlrs := Data.Vlrs \cup \{k\}$;
    send(INCLUSION_ACK($action\_id, Data.my\_id$),k);
end receive_INCLUSION_NOTIF;

procedure receive_EXCLUSION_NOTIF(**integer** $action\_id$, **integer** k) **is**
begin
    $Data.Vlrs := Data.Vlrs - \{k\}$;
    send(EXCLUSION_ACK($action\_id, Data.my\_id$),k);
end receive_EXCLUSION_NOTIF;

Figure 3.3: **Reconfiguration Algorithm for Base Station**

```
Module LR

real cur_Cost;
real threshold;
real limit;

structure Data is
    integer my_id;
    integer set serviced_BSs;
end Data;

procedure call_reconfigure(sleep_time) is
begin
    sleep(sleep_time);
    compute the I and E sets;
    foreach cell i in (M ∪ I) do
        request for and receive cost values;
    reconfigure();
end call_reconfigure;

procedure reconfigure() is
begin
    recompute cur_Cost as Σ_{i∈M}(cost_i);
    foreach i in (I) do
        begin
            if (Cost_in(my_id,i)/Cost_ex(my_id,i) < threshold)
                    ∧(Cost_in + cur_Cost < limit) then
                begin
                    mark i for inclusion;
                    cur_Cost = cur_Cost + Cost_in;
                end
        end

    foreach i in (E) do
        begin
            if (Cost_in(my_id,i)/Cost_ex(my_id,i) > threshold) then
                begin
                    mark i for exclusion;
                end
        end
    exclude_marked_cells();
    include_marked_cells();
end reconfigure;

procedure exclude_marked_cells() is
    foreach marked cell i for exclusion
        begin
            get_distinct(action_id);
            send(EXCLUSION_NOTIF(action_id, k),i);
        end

    foreach marked cell i
        begin
            get_distinct(action_id);
            receive(EXCLUSION_ACK(action_id, k),i);
            serviced_BSs ← serviced_BSs − {i};
            unmark all the neighboring cells to i
                that are marked for inclusion;
        end
end exclude_marked_cells;

procedure include_marked_cells() is
begin
    foreach marked cell i for inclusion do
        begin
            get_distinct(action_id);
            send(INCLUSION_NOTIF(action_id, k),i);
        end

    foreach marked cell i do
        begin
            get_distinct(action_id);
            receive(INCLUSION_ACK(action_id, k),i);
            serviced_BSs ← serviced_BSs ∪ {i};
        end
end include_marked_cells
```

Figure 3.4: **Reconfiguration Algorithm for LR**

3.5 Inclusion and Exclusion messaging

The reconfiguration is divided into *reconfiguration period*. Within that period, the LR updates the value of the Load variable, which is used in cell inclusion. Then it asks for traffic contributions from external cells and decides which ones to include. In the next period, it will have some non-core cells in the RA, which can be candidates for exclusion.

Consider the example in Figure 3.5. In the first reconfiguration period, the LR sends to all the core cells a DATA_REQ. It then combines the information from the DATA_ACKs returned and computes the current load. Then, it computes the I and E sets. It sends a DATA_REQ to each of the cells in I and E and according to the info in the DATA_ACK, the current load and the connectivity of the RA, it decides to include the external_cell_1 only.

In the second reconfiguration period, the LR asks again all the cells in the RA (including the external_cell_1) and computes the load. Then it asks the external_cell_2 as well and then decides to exclude external_cell_1 and include external_cell_2.

3.6 Call Delivery and Registration messaging

The call delivery and Registration are similar to the standard (IS-41) message sequence, with the difference that the messages hold some more information, and that the nodes (BSs,LRs) keep load information with every control message. The call delivery happens as shown in Figure 3.6 The calling MT sends a call request (CALL_REQ) to the BS it is in. The ”calling” BS forwards the CALL_REQ to the VLR of the MT. If the called MT is registered with the same VLR, then the call is local (first part of Figure 3.6). Otherwise, the VLR asks the HLR of the called MT for the location of it. Then the CALL_REQ is sent to the appropriate LR (the VLR of the called MT) and then forwarded to the BS where the called MT is in. Then a sequence of acknowledgments are sent back to the calling MT to signal the admission or not of the call.

The registration algorithm is also enhanced to handle the multiplicity of the servicing LRs for each BS. When a new registration comes to a BS from an MT, the BS checks whether the current VLR of the MT also services the cell of the new BS. if so, then the MT remains in the same RA. If the servicing VLR is not in the set of servicing LRs of the BS, then the BS sends a REG_REQ to the ”default” LR which is going to be the

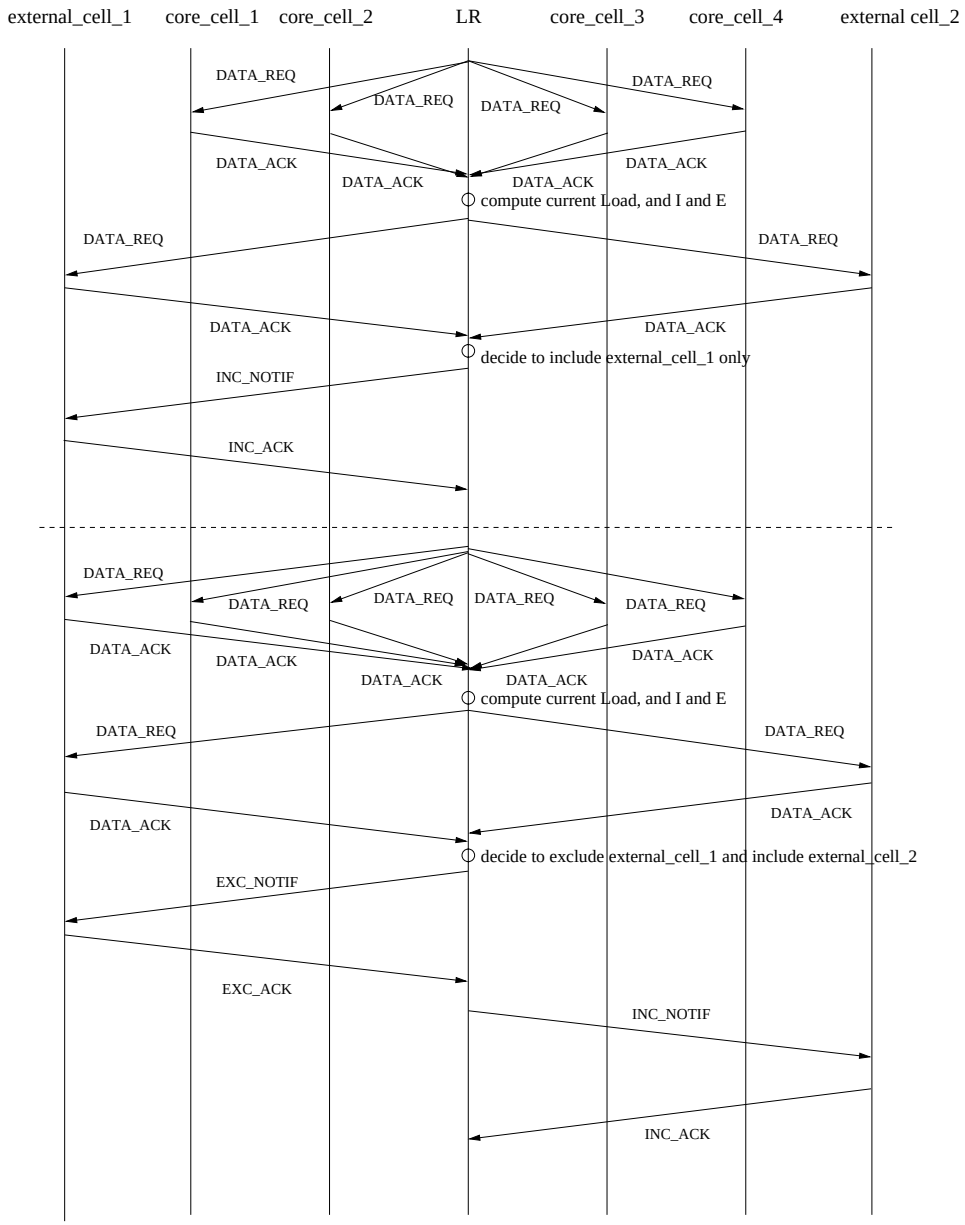

Figure 3.5: **An example of inclusion and exclusion messaging sequences. The configuration periods are divided by a dashed line.**

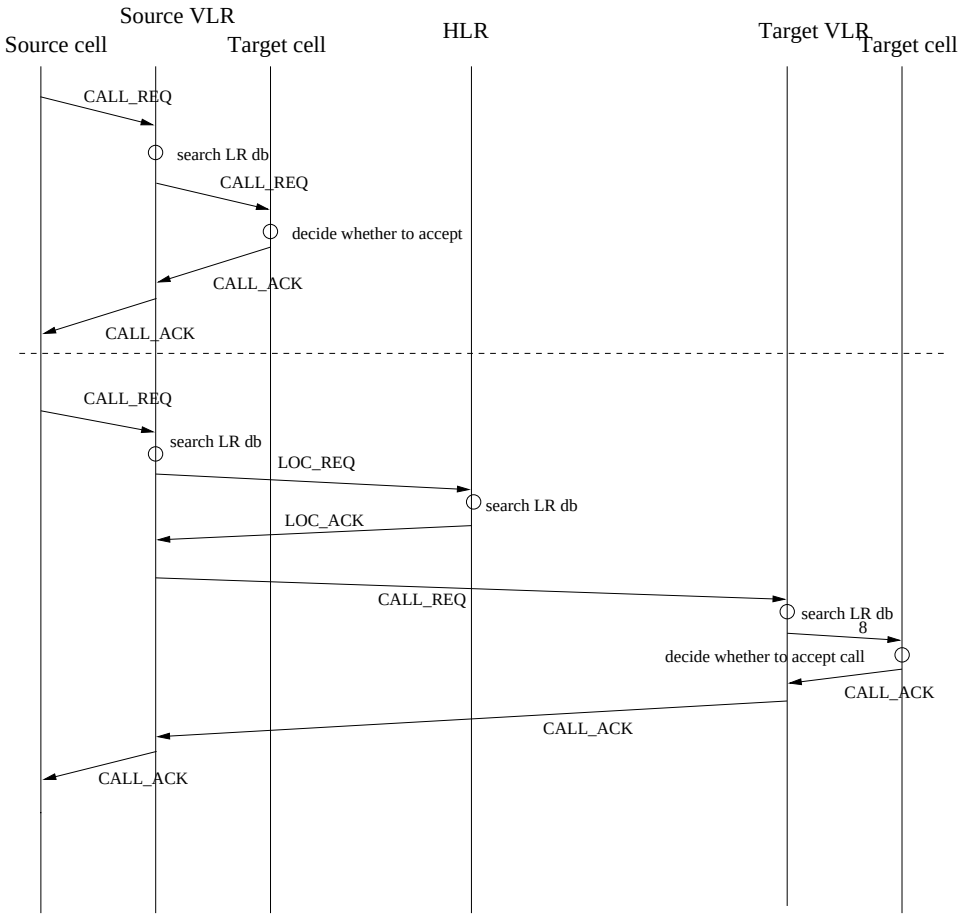

Figure 3.6: **Call delivery messaging sequence. The upper part shows call delivery within the same RA, the lower part shows the messaging when the MTs are in different RAs.**

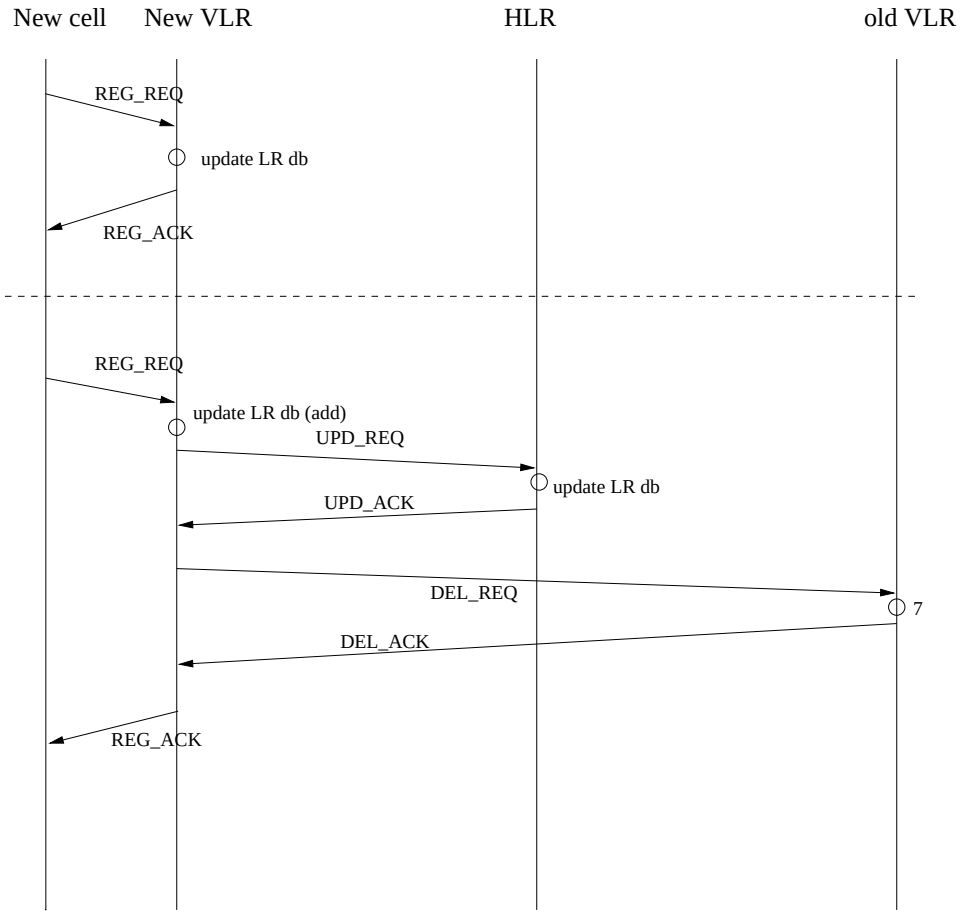

Figure 3.7: **Registration messaging sequence. The upper part shows the case when there is no inter-RA hand-off. The lower part shows the case when there is a inter-RA hand-off**

new VLR of the MT. The new VLR updates the location of the MT with the HLR and then sends a REG_DEL message to the old VLR. The process is demonstrated in Figure 3.7.

3.7 Case Analysis

Following the analysis from Chapter 4 we will show under witch conditions an expansion takes place. It also gives a good reason for showing that the algorithm will move to a RA expansion, when the expansion is going to reduce the load of the LR.

$$Cost_{in} - Cost_{ex} \quad = \quad Increase_{intra\_RA\_calls}(k,a) - Decrease_{inter\_RA\_calls}(k,a) +$$

$$Increase_{inter\_RA\_calls}(k,a) + Increase_{intra\_RA\_moves}(k,a)$$

$$-Decrease_{inter-ra\ moves}(k,a) + Increase_{inter-ra\ moves}(k,a)$$

$$\begin{aligned}
= \quad & \Big(\sum_{i \in Core(k)} n_{calls}(k,i,k,a) c_{intra-call-core-noncore} + \\
& \sum_{i \in RA(k)-Core(k)} n_{calls}(k,i,k,a) c_{intra-call-noncore-noncore} + \\
& \sum_{i \in Core(k)} n_{calls}(k,a,k,i) c_{intra-call-noncore-core} + \\
& \sum_{i,j \in RA(k)-Core(k)} n_{calls}(k,a,k,i) c_{intra-call-noncore-noncore} \Big) \\
- \quad & \Big(\sum_{i \in Core(k)} n_{calls}(k,i,k,a) c_{inter-outgoing-call-core} + \\
& \sum_{i \in RA(k)-Core(k)} n_{calls}(k,i,k,a) c_{inter-outgoing-call-noncore} + \\
& \sum_{i \in Core(k)} n_{calls}(k,a,k,i) c_{inter-incoming-call-core} + \\
& \sum_{i \in RA(k)-Core(k)} n_{calls}(k,a,k,i) c_{inter-incoming-call-noncore} \Big) \\
+ \quad & \Big(\sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{calls}(k,a,m,i) c_{inter-outgoing-call-noncore} + \\
& \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{calls}(m,i,k,a) c_{inter-outgoing-call-noncore} \Big) \\
+ \quad & \Big(\sum_{i \in Core(k)} n_{moves}(k,i,k,a) c_{intra-move-core-noncore} + \\
& \sum_{i \in All(k)-Core(k)} n_{moves}(k,i,k,a) c_{intra-move-noncore-noncore} + \\
& \sum_{i \in Core(k)} n_{moves}(k,a,k,i) c_{intra-move-noncore-core} + \\
& \sum_{i \in RA(k)-Core(k)} n_{moves}(k,a,k,i) c_{intra-move-noncore-noncore} \Big) \\
- \quad & \Big(\sum_{i \in Core(k)} n_{moves}(k,i,def(a),a) c_{inter-outgoing-move-core} + \\
& \sum_{i \in RA(k)-Core(k)} n_{moves}(k,i,def(a),a) c_{inter-outgoing-move-noncore} \Big) \\
+ \quad & \Big(\sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{moves}(k,a,m,i) c_{inter-outgoing-move-noncore} \Big)
\end{aligned}$$

$$= \sum_{i \in Core(k)} n_{calls}(k,i,k,a) c_{intra-call-core-noncore}$$

$$
\begin{aligned}
&- \sum_{i \in Core(k)} n_{calls}(k,i,k,a) c_{inter-outgoing-call-core} \\
&+ \sum_{i \in RA(k)-Core(k)} n_{calls}(k,i,k,a) c_{intra-call-noncore-noncore} \\
&- \sum_{i \in RA(k)-Core(k)} n_{calls}(k,i,k,a) c_{inter-outgoing-call-noncore} \\
&+ \sum_{i \in Core(k)} n_{calls}(k,a,k,i) c_{intra-call-noncore-core} \\
&- \sum_{i \in Core(k)} n_{calls}(k,a,k,i) c_{inter-incoming-call-core} \\
&+ \sum_{i,j \in RA(k)-Core(k)} n_{calls}(k,a,k,i) c_{intra-call-noncore-noncore} \\
&- \sum_{i \in RA(k)-Core(k)} n_{calls}(k,a,k,i) c_{inter-incoming-call-noncore} \\
&+ (\sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{calls}(k,a,m,i) c_{inter-outgoing-call-noncore} \\
&+ \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{calls}(m,i,k,a) c_{inter-outgoing-call-noncore}) \\
&+ \sum_{i \in Core(k)} n_{moves}(k,i,k,a) c_{intra-move-core-noncore} \\
&- \sum_{i \in Core(k)} n_{moves}(k,i,def(a),a) c_{inter-outgoing-move-core} \\
&+ \sum_{i \in All(k)-Core(k)} n_{moves}(k,i,k,a) c_{intra-move-noncore-noncore} \\
&- \sum_{i \in RA(k)-Core(k)} n_{moves}(k,i,def(a),a) c_{inter-outgoing-move-noncore} \\
&+ \sum_{i \in Core(k)} n_{moves}(k,a,k,i) c_{intra-move-noncore-core} \\
&+ \sum_{i \in RA(k)-Core(k)} n_{moves}(k,a,k,i) c_{intra-move-noncore-noncore} \\
&+ \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{moves}(k,a,m,i) c_{inter-outgoing-move-noncore} \\
= &\sum_{i \in Core(k)} n_{calls}(k,i,k,a)(c_{intra-call-core-noncore} - c_{inter-outgoing-call-core}) \\
+ &\sum_{i \in RA(k)-Core(k)} n_{calls}(k,i,k,a)(c_{intra-call-noncore-noncore} - c_{inter-outgoing-call-noncore})
\end{aligned}
$$

$$+ \sum_{i \in Core(k)} n_{calls}(k,a,k,i)(c_{intra-call-noncore-core} - c_{inter-incoming-call-core})$$

$$+ \sum_{i,j \in RA(k)-Core(k)} n_{calls}(k,a,k,i)(c_{intra-call-noncore-noncore} - c_{inter-incoming-call-noncore})$$

$$+ \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{calls}(k,a,m,i) c_{inter-outgoing-call-noncore}$$

$$+ \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{calls}(m,i,k,a) c_{inter-outgoing-call-noncore}$$

$$+ \sum_{i \in Core(k)} n_{moves}(k,i,k,a)(c_{intra-move-core-noncore} - c_{inter-outgoing-move-core})$$

$$+ \sum_{i \in All(k)-Core(k)} n_{moves}(k,i,k,a)(c_{intra-move-noncore-noncore} - c_{inter-outgoing-move-noncore})$$

$$+ \sum_{i \in Core(k)} n_{moves}(k,a,k,i) c_{intra-move-noncore-core}$$

$$+ \sum_{i \in RA(k)-Core(k)} n_{moves}(k,a,k,i) c_{intra-move-noncore-noncore}$$

$$+ \sum_{m \in \mathcal{R}, m \neq k} \sum_{i \in RA(m)} n_{moves}(k,a,m,i) c_{inter-outgoing-move-noncore}$$

The system model, as defined in Chapter 3, we have costs shown in Table 3.1. Using the values from the cost table, we compute a graph of the $Cost_{in} - Cost_{ex}$ for various values of the n_{calls} and n_{moves} (i.e. CMR), which is shown in Figure 3.8.

The graph shows that for high values of call and move rates the scheme will apply expanding to the RA. Although the domain of values of call and move rates that give negative $Cost_{in} - Cost_{ex}$ is rather small, the difference is referencing only one candidate cell. Usually we have multiple candidate cells whose the differences $Cost_{in} - Cost_{ex}$ are accumulating to give a good reduction of the Load with one reconfiguration. We also don't take into account the reduction of the update and search cost from remote requests (requests from other LRs) that happens due to the expansion of other RAs. In the next chapter, simulations show that at some cases we have significant reduction of the Load of an RA, as a result of the reduction of the update/search request rates that are served by each LR.

Table 3.1: **Costs for different hand-offs/call-deliveries.**

$c_{intra-move-core-core}$	$4\delta_{cv} + \alpha_c$
$c_{intra-move-core-noncore}$	$4\delta_{cv} + \alpha_c + 2\delta_{vv}$
$c_{intra-move-noncore-core}$	$4\delta_{cv} + \alpha_c + 2\delta_{vv}$
$c_{intra-move-noncore-noncore}$	$4\delta_{cv} + \alpha_c + 4\delta_{vv}$
$c_{intra-call-core-core}$	$4\delta_{cv} + \alpha_c$
$c_{intra-call-core-noncore}$	$4\delta_{cv} + \alpha_c + 2\delta_{vv}$
$c_{intra-call-noncore-core}$	$4\delta_{cv} + \alpha_c + 2\delta_{vv}$
$c_{intra-call-noncore-noncore}$	$4\delta_{cv} + \alpha_c + 4\delta_{vv}$
$c_{inter-outgoing-call-core}$	$2\delta_{cv} + \alpha_c + (2\gamma + \rho)\delta_{vv}$
$c_{inter-outgoing-call-noncore}$	$2\delta_{cv} + \alpha_c + 2\delta_{vv} + (2\gamma + \rho)\delta_{vv}$
$c_{inter-incoming-call-core}$	$2\delta_{cv} + \alpha_c + \rho\delta_{vv}$
$c_{inter-incoming-call-noncore}$	$2\delta_{cv} + \alpha_c + 2\delta_{vv} + \rho\delta_{vv}$
$c_{inter-outgoing-move-core}$	$2\delta_{cv} + \alpha_c + (2\gamma + 1)\delta_{vv}$
$c_{inter-outgoing-move-noncore}$	$2\delta_{cv} + \alpha_c + 2\delta_{vv} + (2\gamma + 1)\delta_{vv}$
$c_{inter-incoming-move-core}$	$2\delta_{cv} + \alpha_c + \delta_{vv}$

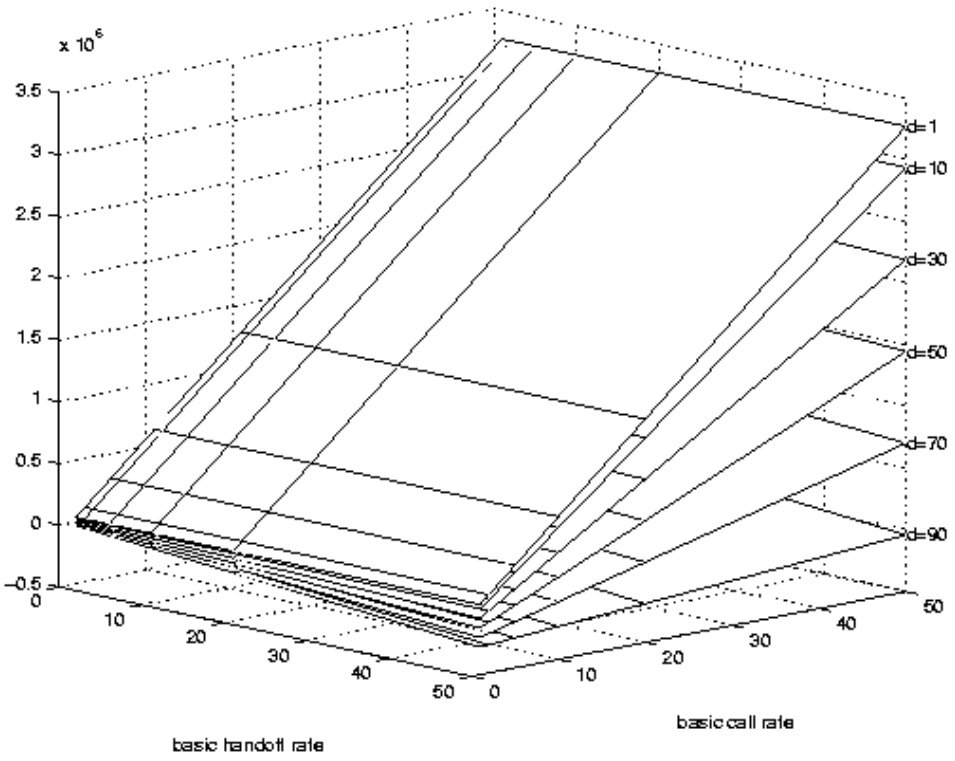

Figure 3.8: **The $Cost_{in} - Cost_{ex}$ difference as computed analytically for various values of call and move rates**

Chapter 4

Simulation

The first set of simulations were to demonstrate the behavior of overlapping in highway systems. As it is shown in Figure 4.1, the overlapping does not reduce the inter-RA hand-offs. The is true when the overlapping degree is small enough so that we don't have RAs completely covered by other RAs, or when the registration follows a "default" LR strategy, that is, when there is a inter-RA hand-off to a cell, the registration goes to a predefined LR, usually the closest one (i.e. the MSC).

In the previous analysis, we are neglecting the fact that, when the RA expands, the inter-Ra hand-offs are reduced, since the probability of a MH getting out of the RA boundaries is reduced. The simulations we have run show that that factor is helping a lot in the reduction of a LR's load.

We have used a 30x30 grid of cells, organized in 25 RAs, initialy each RA having 36 cells. We scattered 20000 MHs across the 900 cells and gave each MH a random move pattern with pole of attraction the MH's initial position. The pole of attraction makes their moves look localized. Each MH's move is interrupted in random times when the MH is placed randomly across the network and is let to 'return' to the home location. The pole of attraction is implemented with the help of a Markov matrix that holds the transition probabilities for a cell moving from distance d to distance d+1 or d-1.

As concluded from the analysis in Chapter 4, the domain of CMR that reduces the cost is confined. Therefore, the simulations have focused on reducing the hand-off rates. So we used a threshold of 1.5 and a limit of load twice of the initial load observed in the simulation series.

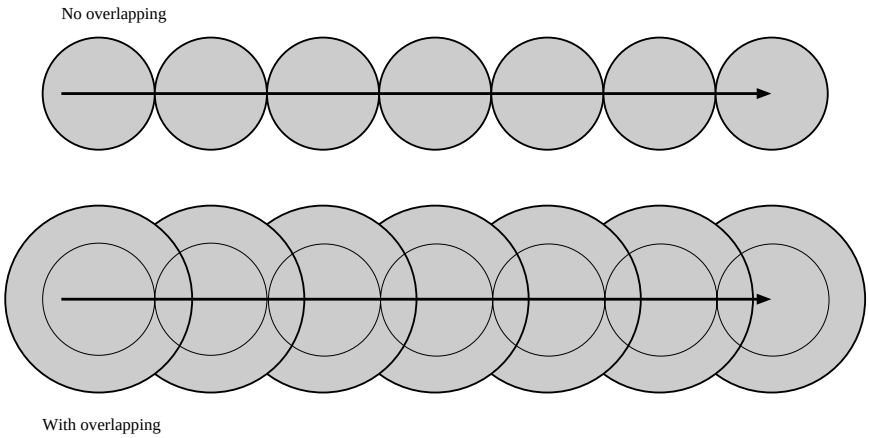

Figure 4.1: **When the mobile moves in a rectilinear fashion, as in highway systems, the overlapping doesn't reduce the hand-offs.**

$$P = \begin{bmatrix} \cdots & & & & & & & & \\ & \frac{1}{4} & 0 & \frac{3}{4} & & & & & \\ & & \frac{1}{3} & 0 & \frac{2}{3} & & & & \\ & & & \frac{1}{2} & 0 & \frac{1}{2} & & & \\ & & & & \frac{2}{3} & 0 & \frac{1}{3} & & \\ & & & & & \frac{3}{4} & 0 & \frac{1}{4} & \\ & & & & & & & & \cdots \end{bmatrix},$$

The results show a dramatic decrease of inter-RA hand-offs Figure 4.2, which results in a reduction of the LR's load Figure 4.3. The simulation has been run under various values of CMR, as shown in the figures. Although the costs n some cases exceed the initial load, it still complies with the values of threshold and do not

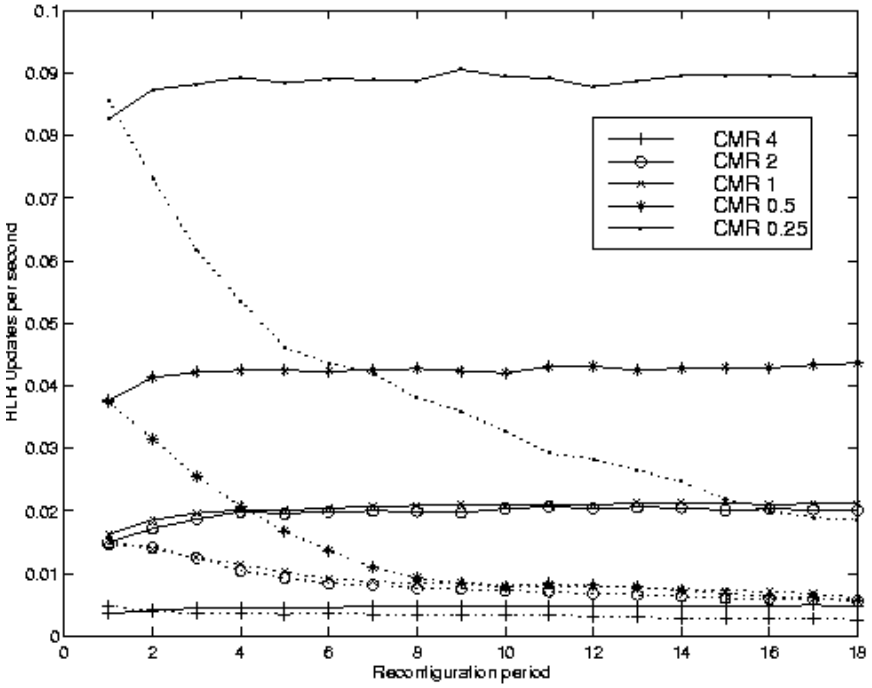

Figure 4.2: **Simulation results for the hand-off rates, for various values of CMR. Solid lines denote the standard non-overlapping scheme, and dotted lines denote the dynamically overlapping scheme.**

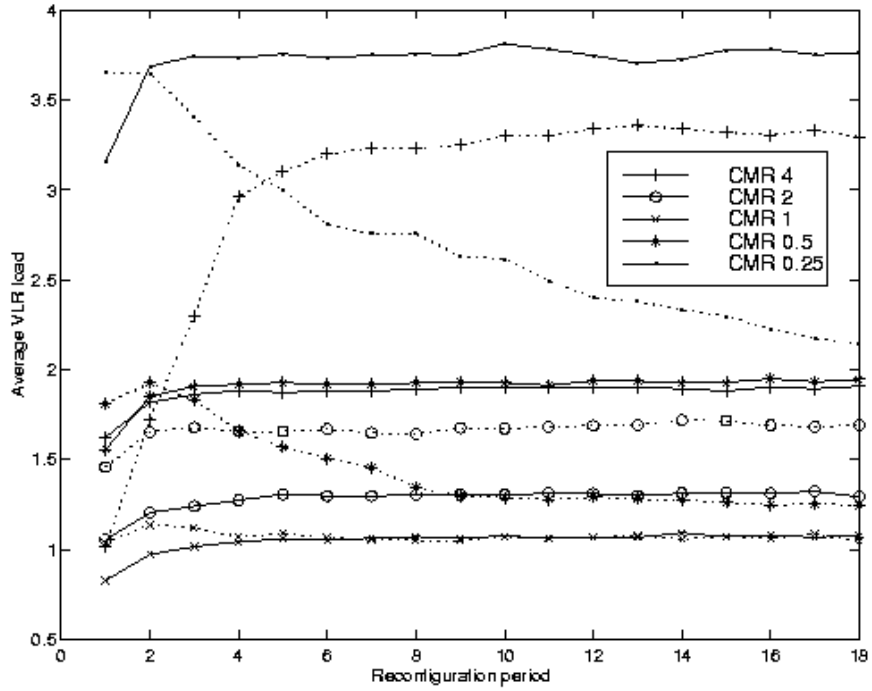

Figure 4.3: **Simulation results for the LR costs, for various values of CMR. Solid lines denote the standard non-overlapping scheme, and dotted lines denote the dynamically overlapping scheme**

Chapter 5

Conclusions and future work

We have introduced a new concept, the idea of dynamically overlapping registration areas, an idea that will help with regions that have strong variations in the user mobility and call patterns, or are difficult to be measured and modeled at the design phase. Furthermore, it will be more efficient with the introduction of multimedia and location-dependent applications, which are expected to produce low CMRs, as they are expected to have constant connections under motion.

As a further step, the profit of caching schemes from the proposed scheme should be evaluated. Since the scheme tries to reduce the inter-RA hand-offs, caching strategies are expected to benefit much from it. Another property that has to be explored is the behavior of the scheme when applied on multi-layered location management schemes, like the one proposed in [1].

Some other things that could be explored is the behaviour of an unconstrictioned version of the scheme, where any RA may expand unlimetedly. Lastly, there is always research to be done on how to apply the idea in a real-world cellular network.

The proposed scheme was designed as an enhancement to already proposed location management strategies, not as a complete location management service. Since it is the first work that gives a solution to the dynamically resizing registration areas, we hope it will receive some attention and contribute into assisting applications of PCS networks where mobility patterns are hard to analyze or change their statistics over time.

REFERENCES

[1] Y. Bejerano and I. Cidon. An Efficient Mobility Management Strategy for Personal Communication Systems. pages 215–222, April 1998.

[2] S. K. Das and S. K. Sen. Adaptive Location Prediction Strategies Based on a Hierarchical Network Model in Cellular Mobile Environment. Proceedings of Second International Mobile Computing Conference (IMCC), March 1996.

[3] EIA/TIA. Cellular radio-telecommunications intersystem operations, is-41 revision b. Technical report, EIA/TIA, 1991.

[4] J. S. M. Ho and I. F. Akyildiz. Dynamic Hierarchical Location Management in PCS Networks. *IEEE/ACM Transactions on Networking*, 5(5):646–660, October 1997.

[5] S. Kryukova, B. Massingill, and B. Sanders. An Algorithm for Distributed Location Management in Networks of Mobile Computers. Technical report, California Institute of Technology, Computer Science department, 1996.

[6] M. Mouly and M. B. Pautet. The GSM system for mobile communications. Technical report, 49 rue Louise Bruneau, Palaiseau, France, 1992.

[7] R. Prakash and M. Singhal. Dynamic Hashing + Quorum = Efficient Location Management for Mobile Computing Systems. In *ACM Symp. Principles Distributed Computing*, page 291, August 1997.

[8] S. Rajagopalan and B. R. Badrinath. An Adaptive Location Management Strategy for Mobile IP. MobiCom, 1995.