EMPTY CELL MANAGEMENT FOR GRID BASED RESOURCE DISCOVERY PROTOCOLS IN AD HOC NETWORKS

SEBASTIEN HEUGUET

NATIONAL UNIVERSITY OF SINGAPORE

2006

EMPTY CELL MANAGEMENT FOR GRID BASED RESOURCE DISCOVERY PROTOCOLS IN AD HOC NETWORKS

SEBASTIEN HEUGUET

(B. E., Supelec, France)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2006

Acknowledgments

I consider myself extremely fortunate to have been given the opportunity and privilege of doing this research work by the National University of Singapore. I would like to thank all the people who have helped me during my Master's degree program at the National University of Singapore.

I would like to express my deepest gratitude for Professor Kee Chaing Chua and Professor Mehul Motani who accepted to be my supervisors and provided warm and constant guidance throughout progress of this work. Their rich experience in the field of communication networks has been extremely valuable for me and I have learned a lot from them during our frequent discussions. This experience has been a most valuable one.

My warmest thanks to the Computer Networks and Distributed Systems Laboratory officer Mr. Eric Poon and to the Open Source Software Laboratory officer Mr. David Koh, I appreciate their helpful nature and dedication in making both laboratories such nice places to work.

I am also thankful for the graduate research scholarship offered to me by the National University of Singapore without which this Master's degree programme would not have been possible.

Finally, I would like to express my deep affection for my parents and my sister who have supported and encouraged me throughout this work and for Fan Yi for giving me stability, support and happiness.

Contents

Summary

From geographic routing to data storage in sensor networks, the discovery of the resources shared is a vital feature of wireless ad hoc networks. The performances of the grid based resource discovery protocols rank them among the most efficient discovery protocols for ad hoc networks. In these location aware protocols, the network topology is divided into geographical regions, called *cells*, and pieces of information are stored and retrieved from a cell by using a given hashing function and the unique identifier of the piece of information. However, because of obstacles and nodes mobility, real mobility scenarios will create a heterogeneous density of nodes on the field and thus there will be empty cells in the grid. This phenomenon results in failures for most of the grid based protocols that have been proposed.

In this thesis, after an extensive review on the resource discovery protocols in ad hoc networks, the issue of empty cells in the grid of a resource discovery protocol is addressed. To solve this problem, we design Hidagrid, a fully distributed protocol that manages the empty cells of a grid. Hidagrid relocates the data items sent to empty cells and routes consistently the queries for such items. As a result, Hidagrid acts as a sub layer located between the routing layer and a grid based resource discovery protocol. It makes the actual grid appear homogeneous for the grid-based resource discovery protocol. Extensive simulations, using a variety of

mobility models and protocol parameters, show the severity of the empty cells on the resource discovery performance. Using Hidagrid with the resource discovery protocol then significantly increases the hit ratio and limits the communication overhead of the resource discovery by avoiding sending messages to empty areas, which creates useless traffic in the network.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The need for efficient resource discovery protocols in wireless ad hoc networks

Recent years have seen the explosion of wireless devices due to an increased need for connectivity: cell phones and laptop equipped with wireless connections are now all day life tools that were almost unknown 10 years ago. With this increased connectivity came also a need for bandwidth in order to provide more and faster services. For example, from the initial 2Mbps rate, the 802.11 protocol is now developed in the 802.11g version that provides a rate of 54Mbps. These new technologies opened a whole new market and created in return connectivity needs from the consumers who envision a fully connected world in the near future. Today's wireless technologies are mainly centralized architectures and they can be seen as extensions of the existing wired networks, i.e., the wireless devices communicate with a fixed station connected to the wired network. These stations act as gateways between the mobile device and the network, and they are responsible for managing

communications in their vicinity. Such architecture requires large investments in order to deploy these expensive infrastructures. Upgrading of such networks has also proven slow and extremely expensive. Even if the initial investments have been deployed, the need for alternate networks will be greater in order to provide wide and flexible connectivity coverage in any place and at any time. Centralized architectures cannot meet this objective because of the related costs. Mobile ad hoc networks (MANETs) represent the complementary solution to cell oriented architectures as they can be easily deployed without any infrastructure. This seamless approach could also be seen as the extension of the actual centralized architecture in order to provide wider coverage in remote areas or during emergency or disaster recovery scenarios.

In MANETs, all mobile nodes collaborate in order to transfer data between two points of the network without the help of any dedicated device. The nodes organize the network themselves and act as routers and servers for the whole network. Because of mobility and link weakness, the topology of the network continuously changes, raising many new challenges for researchers in an environment where bandwidth and energy are scarce resources. Because of these constraints, ad hoc networks perform poorly compared to centralized architectures. However their ability to cope with any environment and the lack of investment could allow the development of various new applications in the future for this kind of architecture.

As it is already the case today with Internet, a network is mainly used to find information, transmit files and communicate with other users. Therefore the first step for real deployment of MANETs is also to create an environment where the nodes can cooperate, work and share resources and information without any

infrastructure. All the applications that will be developed for MANETs will rely on the ability of locating a service, a file or a data item and exchange it. This collaborative environment requires an efficient platform responsible for this task, as it is already the case today for wired networks. Indeed, the internet would not be as popular as it is now without good search engines. However, the approaches used for fixed networks can not be derived for MANETs because of the new constraints imposed in these networks. The topology continually changes, nodes are unreliable and power limited, and they only have small storage and computation capacities. The initial centralized solutions that were developed in the past can therefore not be applied in MANETs. Decentralized (or peer-to-peer) protocols have also been proposed later for wired networks [1, 2, 3], but they require an abundant bandwidth that is not available in MANETs. Therefore new decentralized approaches must be further studied to comply with the constraints brought with these networks. Two obvious solutions to resource discovery in MANET could be proposed. First, a node looking for a service or data item could simply flood the whole network with a query (data pull), or on the opposite, a node sharing a service could flood the network with an advertisement (data push). These two simple approaches are not feasible in large ad hoc networks where flooding could be harmful for the performances of the network [4]. Therefore a trade off has to be found where nodes send some advertisements in specific places in the network that can be easily found by a query for the corresponding item, resource discovery protocols manage this mechanism.

Discovery protocols are not only useful at the application layer. Recently many routing protocols for MANETs propose to use geographic information [5]. Contrary to the graph model protocols where nodes forward packets based solely on

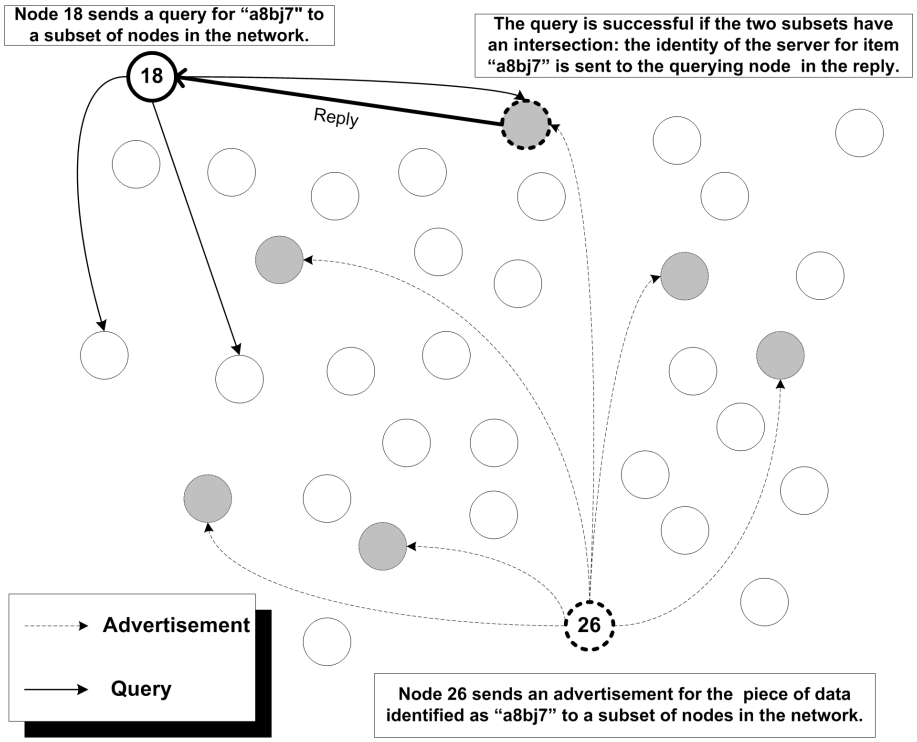


Figure 1.1: The resource discovery problem: advertisements and queries are spread over two sets of nodes such that both sets intersects.

their connectivity, geographic routing uses location information to forward packets toward the destination. Therefore, they scale very well with the network size to achieve good performance, but they require the implementation of a location service to efficiently distribute information on nodes positions in the network. This problem is very similar to the resource discovery mentioned earlier, as shown in Fig 1.1, i.e., given an identifier (here the destination node's identity), where can a node find the information related to that identifier (the destination node's location)?

Sensor networks represent another class of self organizing wireless networks where the constraints on throughput are relaxed. The applications implemented on top these networks use little communication compared to MANETs. The main goal of these networks is to improve the knowledge of an environment by gathering

information from many small sensors that are able to communicate with each other and to perform simple computation tasks. The data collected is then computed and transmitted for monitoring purpose. With sensor networks, many applications can be envisioned ranging from environment monitoring to health screening. Many useful applications could benefit from sensor networks, i.e., industrial control, home automation, security and military sensing, asset tracking and supply chain management are only a small sample of the commercial potential of these networks. Throughput is however not the only difference between MANETs and sensor networks. Indeed, energy consumption is the main constraint in these networks because it is proportional to the lifetime of the sensor network. Each node stores a very limited amount of energy that cannot be recharged. The protocols developed for these networks must therefore primarily focus on energy efficiency. Due to the high number of sensors that could be spread on a field, scalability is also a requirement for the sensor network protocols. These constraints associated with node and link unreliability make sensor network a challenging environment for network researchers. Even if research efforts have been put in this area, many issues have to be overcome in both hardware and software before effective deployment.

In sensor networks, the data processing mechanisms are very different depending on the application considered. In the case of monitoring, data items will be locally analyzed and transmitted under certain conditions to a sink that will process all the packets it receives, before taking a decision. Sensor networks could also be used in unattended mode where the data collected is stored inside the network before being extracted. This mode requires efficient mechanisms to retrieve the relevant information in the network when a query is sent from a user. It has been shown in [6] that data-centric storage is an efficient data dissemination scheme. In

this model, data item dissemination is solely based on the characteristics of sensed data items. Therefore, similar data items are stored together, requiring an efficient resource discovery protocol to retrieve information from the network.

The design of resource discovery protocols has been widely studied and many solutions have been proposed in the literature, each with their niche of applicability. However, the simulation results published so far show that the protocols using geographic location to perform their task have very good performance in terms of efficiency and scalability. This is true for routing protocols, as well as resource discovery protocols. And we believe that this approach should be promising in the future. Therefore, in this thesis, we focus on resource discovery protocols using geographic location information.

1.2 The empty cell problem for grid based resource discovery protocols

Under the assumption that nodes are able to locate themselves, grid based resource discovery protocols are used for a wide range of applications [7, 8, 9, 10, 11]. Service or file discovery protocols, as well as location service could rely on this flexible mechanism. In this kind of protocol, the network field is divided into geographical regions that we call *cells*. These cells are responsible for storing and managing a subset of the data items available in the network. The share of load among the cells is determined with a hashing function, i.e., when inserting data item, a node hashes the unique identifier of the data item to one of the cell in the network, called the *server cell* for that data item. The server cell will then be responsible for storing

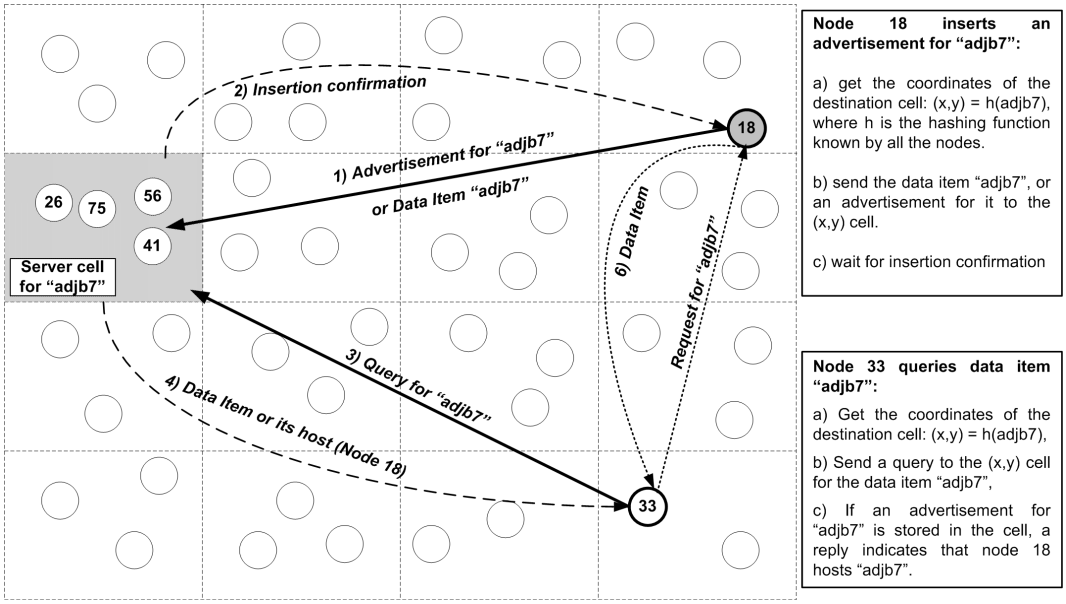


Figure 1.2: Insertions and queries in a grid based discovery protocol

the data item. A querying node carries out the same operation. Knowing the identifier of the queried item, the identifier is hashed to a position in the network, and a query is then sent to the cell. The mechanism is described in Fig. 1.2.

Das et al. [12] compare the performances of several categories of location services and state that the category the grid based protocols belong to have the overall best performances. However, the actual protocols assume that the node density is roughly uniform on the network and that the grid is homogeneously populated. In this case, all the cells in the grid are populated and no empty cell appears in the grid. This assumption in large real scale networks in unrealistic for two reasons. First, if the grid is not adapted to the geographic topology, obstacles like buildings or natural obstruction (lake, hill, park) naturally create empty cells. But even if the grid is customized, mobile nodes may also create empty cells when moving. Consider, for example, a university campus where each student carries a node. The node density varies according to the time schedule of the students, and

thus the canteen will usually be empty except at lunch time when the classrooms will not be occupied. As a result, real human activity creates great variations in the network densities. In this situation, the protocols mentioned above fail, with the only notable exception of the *Grids Location Service* (GLS) [7] that adapts by design to empty cells. However this protocol performs poorly compared to the other protocols [12]. The empty cells then results in denial of service because the queries sent to an empty server cell are not processed. Furthermore, if the nodes populating a cell leave it, all the data items that were stored are suddenly lost when the last node leaves the cell. But if a location service for geographic routing is considered, the location of some nodes cannot be known if their location server is empty. This means that these nodes cannot be contacted, which is unacceptable in a collaborative environment. Despite the good performance of grid based protocols, this significant weakness makes them unsuitable for real deployment at this stage. Some authors are aware of that problem and even mention some simple mechanisms to mitigate it. But none of them manages empty cells efficiently, and no performance results with empty cells have been reported so far.

1.3 Contributions

We believe that the resource discovery protocol must adapt to the shape and the state of the grid in order to solve the empty cell problem, i.e., the discovery protocol must avoid sending messages to empty cells, while data items are consistently located to populated cells. Another solution would be to create multiple hashing functions or grids, but this would be a waste of communication and it would require many security mechanisms to avoid failure of the protocol. In this thesis, we

therefore propose the *Hierarchical Dynamic Adaptation of a Grid*, or *Hidagrid*,a fully distributed protocol which achieves dynamic adaptation of the state of a grid. This adaptation is solely based on the node density in each cell. Hidagrid is implemented as a sub layer that transforms the heterogeneous actual grid into a homogeneous grid that can be efficiently used by any grid based resource discovery protocol, as shown in Fig. 1.3. As a result our protocol requires only minimum information before deployment and could be used quickly in a disaster recovery scenario without setting up a grid adapted to the geographic topology. As Hidagrid is used as a sub layer, several service location protocols could be used on top of it without generating any extra overhead. Extensive simulations show that Hidagrid significantly improves the performance of the service discovery protocol. The hit ratio of the discovery protocol highly increases as soon as some empty cells appear in the grid. But our study also shows that Hidagrid reduces the total communication overhead of the resource discovery protocol because it avoids sending messages to empty areas, limiting the number of packets dropped due to routing failures or TTL counter reaching 0.

1.4 Thesis organization

This thesis is organized in 5 chapters.

- This chapter introduces the reader to an overview of the resource discovery protocols for ad hoc networks. The principles of the grid based discovery protocols are explained and the need for an efficient empty cell management protocol is highlighted. The focus of the thesis and the main contributions

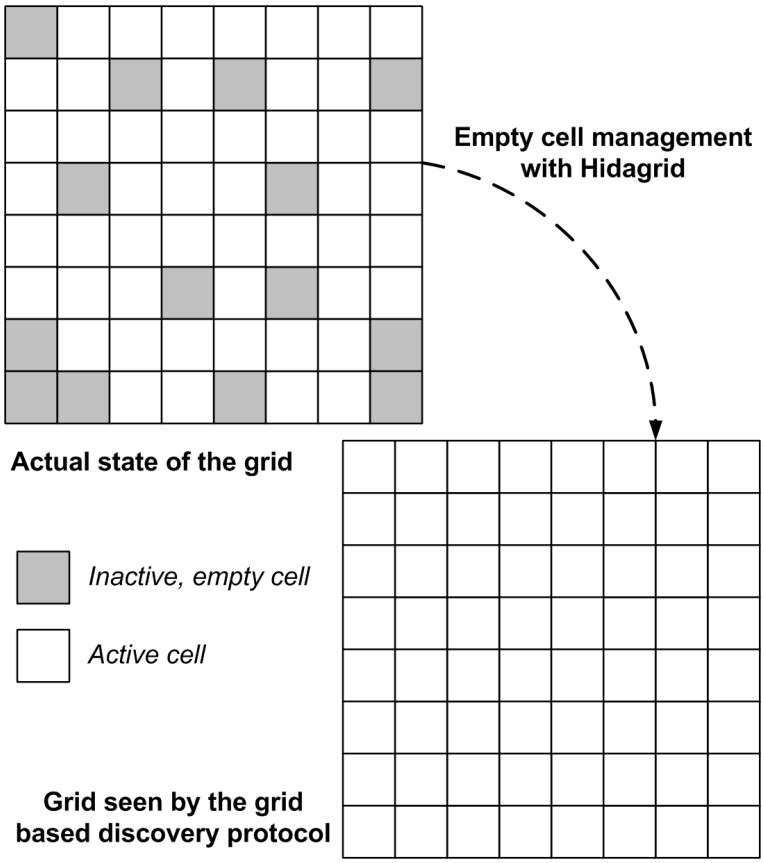


Figure 1.3: The adaptation role of Hidagrid: with Hidagrid, the actual heterogeneous grid appears homogeneous to the grid based discovery protocol.

are then summarized.

- Chapter 2 presents an extensive literature review on the resource discovery protocols in ad hoc networks. This literature review covers link based discovery protocols, as well as location aided protocols and introduces the reader to the problems encountered with empty cells.

- In Chapter 3, the Hierarchical Dynamic Adaptation of a Grid, or Hidagrid is described. This empty cell management protocol is designed to meet the basic requirements that are highlighted at the beginning of that chapter.

- Chapter 4 presents the results of extensive simulations for a simple resource discovery protocol with and without Hidagrid. The comparative simulations evaluate the severity of the empty cell problem in grid based resource discovery protocols and also measure the benefits of Hidagrid, for a wide range of scenarios.

- Chapter 5 concludes this thesis, highlighting the major contributions of this research. This chapter also discusses the limitation of Hidagrid and presents possible future research directions.

Chapter 2

Literature Review

2.1 Introduction

Service or data discovery is an important function in wireless ad hoc and sensor networks. For example, location service, service discovery or data storage, among others, strongly rely on the resource discovery layer. The nature of the resources managed by these protocols greatly depends on the application and will include the following items:

- **Files:** In a collaborative environment, users need to exchange documents and to find them easily on the network.
- **Services:** Services were initially provided for hardware devices like printers or cameras. However with the development of wireless devices, many new *soft* services are proposed over a network. The variety of such services should grow in the future in order to provide the mobile users with the ability to handle any operation from anywhere. Thus, the research of services should be a keystone for the commercial development of ubiquitous networks.

- **Small data items:** The query for small data items could include location information, measurements in sensor network or encryption keys for transactions.

This chapter presents the different solutions proposed in the literature to develop efficient resource discovery protocols in MANETs. The protocols must overcome the specific challenges of these unstable and resource limited networks. However, the review focuses only on the discovery mechanisms among the numerous issues that need to be addressed to deploy real databases over a wireless network. Such issues include: network and transaction security, the definition of an adapted description language for the data items and the definition of a query language, compatibility in a heterogeneous environment, and eventually QoS support.

This chapter is divided in four sections. The requirements for the design of a discovery protocol in wireless networks are first described in section 2.2 and explanations about the inadequacies of the solutions developed for wired networks will be given. In section 2.3, schemes using no location information will be introduced. We refer to such schemes as *link based protocols* in contrast to the *location aided protocols* that will be studied in section 2.4. Section 2.5 concludes this chapter with a comparison of the approaches analyzed in the chapter and an analysis of the strengths and weaknesses of the grid based protocols. In particular, the empty cell problem for this category of protocols is analyzed.

2.2 Design Requirements

Wireless networks are characterized by their unstable environment. Therefore, the first constraint for the protocols deployed over these networks is decentralization. Indeed, due to the lack of infrastructure, it can not be assumed that a node is reliable or powerful enough in terms of memory, computation capacities or even bandwidth to support the load of a protocol for a full network. Therefore several nodes or even all the nodes must share the workload and cooperate to provide services to each other. Decentralization is therefore the first requirement for a resource discovery protocol in wireless networks and the reason why approaches like Jini [13], Salutation [14], (service discovery), UDDI [15] (web services) or Napster [16] (file sharing) are not applicable here.

In a wireless communication environment, bandwidth is also a scarce resource. As a support function, resource discovery is therefore expected to consume as little bandwidth as possible. Bandwidth consumption is also related to the power consumption in the wireless devices. Therefore, limiting the overhead generated by the resource discovery protocol not only improves the network load and the quality of communications but it also increases the battery use time of the devices. For these reasons, network flooding should be limited as much as possible in wireless networks, as explained in [4], because of the congestion that it quickly generates. UPnP [17], SLP [18] for service discovery, Gnutella [1] for file sharing and DREAM [19] or RLS [20] in location service are examples of protocols relying on pure broadcasting to discover resources, they can therefore not be used in a wireless environment.

Due to mobility and wireless connections, the topology of the wireless network is also expected to continuously change. The resource discovery protocol should then be able to adapt quickly to the local changes without generating much overhead. File sharing systems like Pastry [2], Tapestry [3] or CAN [21] create a virtual organization of the network using hash functions. This mechanism requires periodic checks of the virtual structure's consistency in order to detect node failures. To route and recover from failures, nodes also need to maintain routes with other nodes. These two operations are extremely costly in a mobile and unstable network. Furthermore, the virtual structure usually routes messages from neighbor to neighbor in the virtual space. But, as the virtual space is built using a hashing function, two neighboring nodes in the virtual space are not likely to be close to each other in the actual network. Message routing then results in important inefficiencies. Protocols using this kind of mechanism are therefore not suitable for wireless networks.

As a result, an efficient resource discovery protocol for wireless network must possess at least the following characteristics: decentralization, adaptability to frequent topological changes and limited communication overhead.

2.3 Connectivity based protocols

This section introduces resource discovery protocols that do not make use of any location information. This category has been mainly studied for service discovery protocols. Two basic flooding solutions exist to find an item in the network: in *data pull* clients flood the network with a query message, while in *data push* the

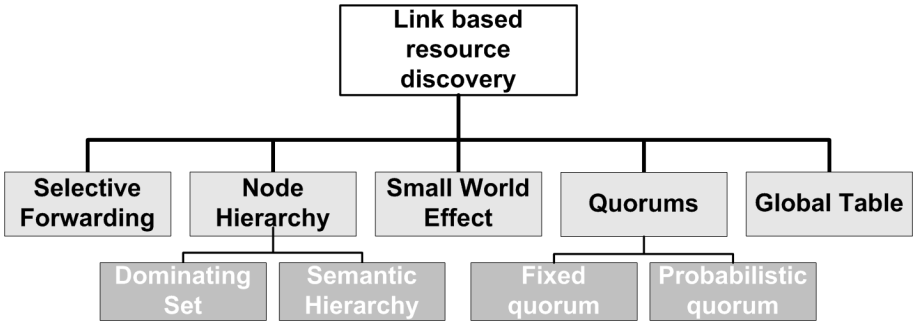


Figure 2.1: Classification of the link based resource discovery protocols

servers flood advertisement messages for the hosted resources. These solutions have a good hit ratio but they obviously do not scale in terms of number of nodes or query and advertisement frequency. The main goal of resource discovery is then to find a trade off between the spread of advertisements and queries such that a query matches an advertisement for the same resource with high probability. Basic flooding could also be improved using smart flooding or epidemic diffusion [4, 22], but these mechanisms are beyond the scope of this review. Due to the large number of schemes that have been proposed in the literature, a taxonomy depicted in Fig. 2.1 is defined to classify the proposals. Each branch of the tree will be introduced in the next sections.

2.3.1 Selective Forwarding

In selective forwarding protocols, the advertisements are propagated in a gossip fashion without flooding and are cached by the nodes. Nodes then use their cache to forward queries to areas where the queried item is likely to be hosted. Usually, these schemes rely on a language that describes data items using a tree structure in order to evaluate the forwarding opportunities. Even if the approach might prove

efficient, it can only suit the needs of small or medium networks.

2.3.1.1 Allia

In *Allia* [23], servers periodically broadcast advertisement beacons to their H hops neighbors. Nodes providing the same kind of services cache each others' advertisements to create an alliance. The advertisement beacons are forwarded depending on local policy. When a node receives a query, it replies if the data item queried is hosted or cached. If not, it selectively forwards it to other members of its alliances or to very active neighbors. The strength of Allia relies in the flexibility given by the agents ruling the caching, forwarding, and advertising policies. The protocol easily adapts to local conditions and user's characteristics. For example, unlike other protocols, the scope of messages can be easily adapted. In order to detect environment changes and facilitate policies adaptations, Allia is also more push-oriented than pull-oriented and tends to forward more the advertisements. It is however not clear how these policies are implemented as the paper mainly focuses on describing the framework. The agent approach could also be difficult to use for devices with small computational capacities.

2.3.1.2 Group-based Service Discovery

The *Group-based Service Discovery* (GSD) [24] is based on an XML description language that classifies each service in a tree. Services can then be characterized by the groups they belong to. Every node advertises the *description* of its services every T seconds to its H hops neighbors. The message also includes the list of the

groups the node has heard of in its vicinity, in order to create a gossip environment. These items are all cached by the neighbors. The cached information is then used when a query is created to select the nodes that have heard of similar services in the vicinity. The query will be sent to these nodes. Each query contains the *description* of the service as well as the *group* it belongs to. The nodes receiving the query use the *group* field to duplicate the message and forward it to other nodes hosting similar services.

GSD reduces efficiently the number of messages to discover a service as it targets the potential hosts. However, this scheme can only be efficient in small and medium networks whose diameter (in hops) is rather limited. Indeed, a service will be discovered if it is less than 2H hops away from the querying node. Above this limit, network flooding may be necessary. With this limit in mind, GSD could be used over a network whose diameter reaches 2H+3 or 2H+4 with good performances compared to pure flooding. Protocols mixing proactive and reactive mechanisms, like GSD, recommend spreading the beacons for only 2 or 3 hops in mobile networks. As a result, it can be estimated that GSD could perform quite well on networks of diameter 8 or 10 hops maximum depending on the mobility and application considered. Finally, the size of the advertisement beacons should be limited in order to restrict the bandwidth consumed.

2.3.2 Creating a node hierarchy

Some protocols create a hierarchy of nodes to handle the resource discovery. Usually, this hierarchy consists of only two levels: the normal nodes and the backbone nodes. Resource providers then register their services to the backbone nodes

which also receive the queries submitted. The messages are then propagated and processed in the backbone using the links that are maintained between backbone nodes. This approach generates high overhead in a mobile environment because the backbone nodes have to maintain the backbone structure.

[25] is one of the first proposals suggesting an alternative to the client-server paradigm in service discovery for wireless networks. It uses a third type of nodes called *mediator* which form the backbone. In this scheme, the mediators are exclusively elected among the service providers. Thus, the clients reach the available services without querying zones that do not provide any service. In this case, the behavior of the service discovery is unpredictable as it depends only on the repartition of the servers in the network and if a client does not have any mediator in its vicinity, it has to fall back to flooding. It is also inappropriate to concentrate the mediators in areas that host the resources because the clients without any resource are more likely to issue queries than powerful server nodes. For these reasons, the mediators should be evenly spread in the network.

2.3.2.1 Creating a dominating set

In [26], a dominating set is created to handle the discovery of services. A dominating set is defined as a set that nodes belong to or are 1 hop away of, as shown in Fig. 2.2. This allows creating and maintaining the backbone structure, as well as routing messages, with only *hello* beacons sent 1 hop away. This message sent by each node contains: the status of the node, its ID, its virtual access point (VAP) to the backbone, some routing information and some other measures used to maintain the structure. This information is enough for the backbone nodes to know their

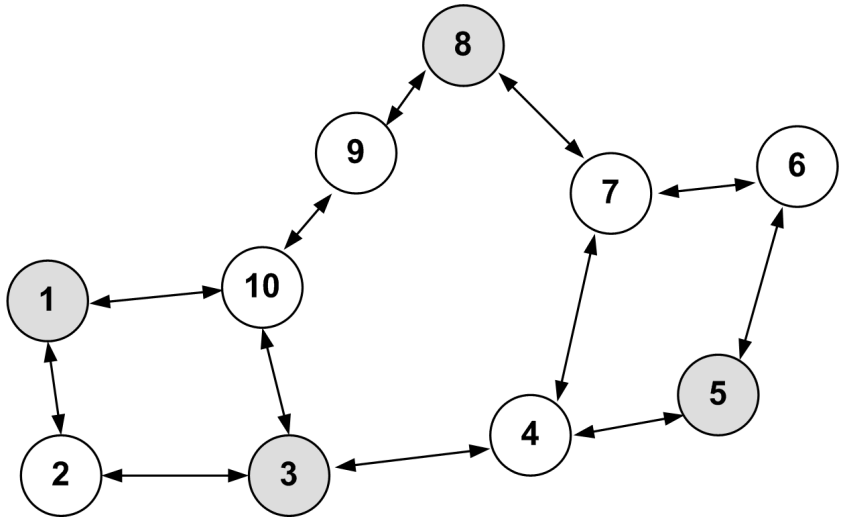


Figure 2.2: The grey nodes form a dominating set in the graph

backbone neighbors which are 2 or 3 hops away. For example, in the figure, let us assume that node 4 has node 5 as VAP. This information is included in the hello message and node 4 also adds that it sees node 3 as a backbone node. Then, node 3 knows the route to node 5 and reciprocally. For the three hops case, node 6 informs its VAP 5 that it sees node 7 with VAP 8 in its vicinity; node 5 knows the path to node 8. This structure is quite efficient for the discovery of services and it could be easily adapted for routing purpose for low or medium quality routes. Furthermore, only local communication is needed to achieve this double task, making this proposition quite appealing for low mobility scenarios. However, it is extremely costly to maintain the structure if it changes frequently as nodes need to register their services every time their VAP changes. Furthermore, the multicast algorithm used in the backbone to propagate messages requires some stability. Not only the overhead but also the hit ratio would be affected by mobility. This analysis is confirmed by the results displayed in the paper: the hit ratio and delay of the protocol is better compared to AODV or DSR based service discovery, but at the price of a huge overhead (at least 10 times higher). The hit ratio also decreases

with mobility. Thus this scheme could be only efficient in almost static networks, but it has the advantage of combining routing and resource discovery.

2.3.2.2 Backbone and selective forwarding

In order to make the service discovery backbone more robust to mobility, [27] widens the area covered by the backbone nodes and loosens at the same time the links between the backbone nodes, called *directories*. They are responsible for an area covering H hops around them. A directory caches proactively the services in its area and periodically broadcast its presence over H hops. No service description is sent. On top of this structure, directories also create summaries of the cached services using bloom filters [28]. This summary is sent periodically over 2H hops and flooded from time to time when the hit ratio of the discovery protocol degrades. This mechanism helps targeting specific directories when a global discovery is initiated. The links between the directories are not formally maintained but each directory knows the ID of all other directories in the network because each newly elected directory broadcasts its identity to the network. This scheme greatly improves the scalability of the backbone mechanism, but compared to the previous approach, this protocol cannot act as a routing protocol and it requires the help of a routing layer, increasing the costs. But in low mobility networks, this scheme should outperform the previous ones because of the selective forwarding achieved with the summaries.

2.3.2.3 Semantic Hierarchy

The previous protocols only build a flat hierarchy. In [29], a more complex hierarchy is built using the nodes' geographic and semantic proximity. In this scheme, nodes create local, or level 0, rings and elect one member of the ring as *ring head*. The ring heads can then create level 1 rings, and so on. Each ring head caches the summary of the services hosted in its ring. A query for a service then travels up and down in the hierarchy of appropriate rings. No simulation is reported in the paper and even if this solution is quite elegant, this scheme is extremely unstable and requires major reconfigurations and maintenance in mobile environments. However in almost static scenarios, the hierarchy should make the protocol scale very well with the size of the network as the rings and their summaries are geographically and semantically formed.

2.3.3 Quorums

A quorum system is created by forming sets of nodes (quorums) where the intersection between two quorums is not empty and no quorum includes another one. For example, the sets {1,2,3}, {1,4}, {2,4,6}, {3,4,5,6} form a quorum system. This approach is strongly related to the previous one as the proposals using quorums assume the creation of a backbone in the network. Therefore, all the following protocols will suffer from the high cost of backbone maintenance in mobile environment. However their appeal lies in the cost reduction of the registration and discovery for resources in the backbone. In the following, we will assume that n nodes called *servers* form the backbone. The resource discovery can then be

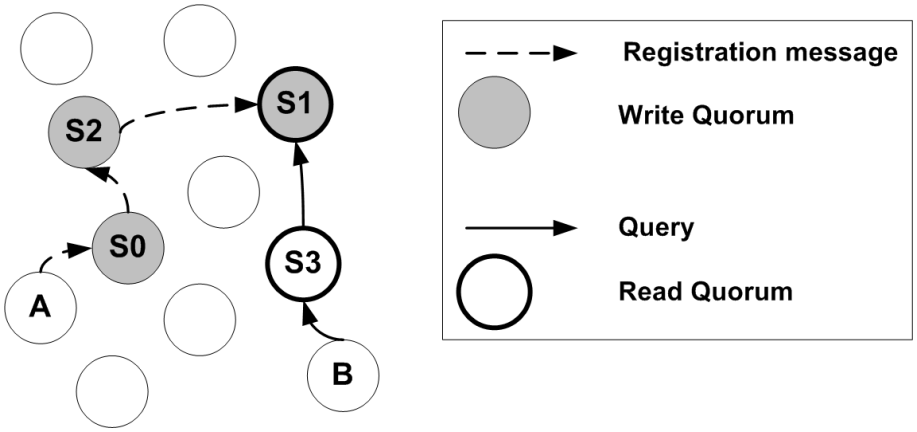


Figure 2.3: Resource discovery with quorums: Node A registers a resource in a write quorum, Node B queries that resource in a read quorum.

described as shown in Fig. 2.3:

- **Write:** Node A sends a registration message to its closest server (S0) that will propagate it in one of the quorums it belongs to: (S0, S1, S2).
- **Read:** Node B sends a location query for node A to its closest server (S3). The query will be propagated in one of the server's quorum (S1, S3). By construction, an intersection server resolves the query (S1).

2.3.3.1 Creating fixed quorums

[30] is the first proposal using quorums in wireless network. The authors build a quorum system of q quorums, each quorum of equal size k having r common servers with any other quorum, $r > 1$ to support server failures or disconnections. In order to spread the load among servers, each server should appear in the same number m of quorums. However the quorum system (n, q, k, m, r) has no simple solution and approximations of the parameters are used for cost analysis which proves that

the quorum system is optimized when the number of servers is small. Therefore several quorum systems with few servers are created in the network. Each node is attached to one quorum system only and the write/read operations are modified: client A sends a write/read message to its closest server (S0). Using A's ID, S0 derives A's quorum system and forwards the message to the closest server in A's quorum system (S1). S1 will then propagate the message in one of its quorum.

This complex system lacks flexibility: each quorum system has its (n, q, k, m, r) fixed. If servers appear or disappear, the system cannot be easily adapted, especially if several quorum systems coexist in the network: the quorum systems must be balanced and close to their optimal parameters (n, q, k, m, r) to avoid performance deterioration. Second, the system relies on a very strongly connected backbone as each server must not only be linked to its own quorum, but also to the other quorum systems. Finally, having several quorum systems increases the write/read costs as each message must be transferred to the good quorum system. All these issues, added to the costs of maintaining a backbone, make it likely that simple flooding could perform better, even in low mobility networks.

2.3.3.2 Probabilistic Quorums

The previous proposal ensures that the quorums intersect, but at the high cost of maintaining a complex structure and lacking flexibility. Another solution is to build a simpler quorum system where the intersection between quorums is not guaranteed but only highly probable, resulting in a *probabilistic quorum* [31]. Haas et al. [32] propose that each write/read operation is sent to k randomly chosen servers among the n available ones. The choice of k is a trade off between overhead and expected

intersection probability. Analysis shows that the system can optimally adapt to a wide range of situations as long as $k/n \approx 0.6$. However, this assumes that all the servers are able to distribute the write/read messages to any other server in the network, which requires costly maintenance. [33] relaxes this hypothesis using epidemic dissemination: when a server receives a write message it forwards it to F of its known server, where F is the fan-out of the epidemic algorithm [34] than can be locally adapted. The read operation is conducted as earlier: the query is send to a fixed number of servers ξr. The spread of the read query using epidemic propagation has not been studied although it could further decrease the number of links maintained by each server.

One of the main issues of these schemes is their reliance on backbones: their performances are strongly correlated with the shape of the backbone and its maintenance costs. Otherwise, the schemes are flexible and can adapt to the number of servers easily. If the network remains quite stable, the protocol could prove efficient. Because no flooding is used, this kind of protocol should also scale well for stable networks. Finally, if replicated items are present on the network, the query cost should be greatly decreased.

2.3.4 The small-world effect

The concept of small world, studied in the 60's in social networks showed that a letter could be delivered anywhere in the world in six steps in average, using only acquaintances. The authors of [35, 36, 37, 38] use this theory to create a scalable system for resource discovery in wireless networks. In this scheme, nodes proactively cache the resources present in their H hops vicinity when receiving

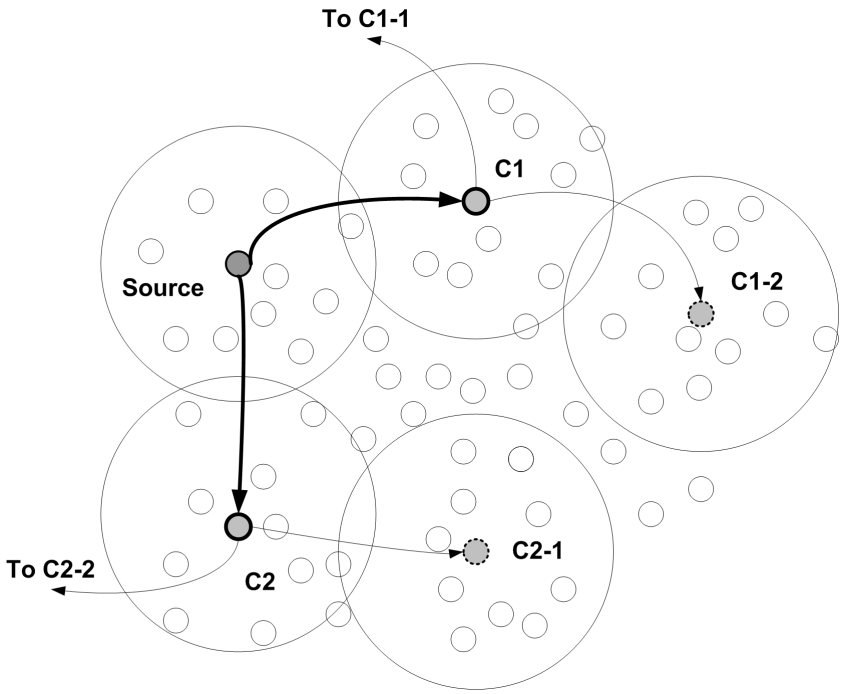


Figure 2.4: Resource discovery using two contacts: the source node sends the query to its contacts which forward it to their own contacts

beacons from their neighbors. The proactive protocol also allows each node to know the route to any neighbor in the H hops vicinity. When a node does not find a resource in its cache, it seeks the assistance of some nodes called *contacts* that are out the vicinity. Usually each node has around 4 contacts. The contact selection is fully distributed and decentralized. When the contacts receive a query, they start investigating their vicinity and if no match is found, they query their own contact and so on, as in Fig. 2.4. In order to limit the search, a depth of search is included in each query.

Several algorithms are proposed in [35, 36] and [37, 38] to select contacts whose vicinities do not overlap. The authors study the protocol with proactively maintained contacts in [35, 36], while [37, 38] propose to select the contacts on the fly when a query is issued. The authors also evaluate three different policies to

determine the depth of search. The reactive solution seems much more efficient than the proactive one which maintains contacts in a mobile environment, which results in high overhead. The reactive approach proves efficient for resource discovery in wireless networks because it combines a proactive and local monitoring with a reactive and delocalized query system. This combination proves scalable with the network size (up to 30000 nodes in [38]) and robust to mobility because no structure has to be maintained. The extensive simulations present a good hit ratio, while keeping the overhead very limited compared to other solutions. Another interesting feature of this solution relies in its sensitivity to resource replication: the simulations show great improvements when some replicas are scattered in the network.

The same proactive/reactive approach is also used in [39], where ZRP [40] is modified for resource discovery. However the performance is not as good, as shown in [38]. The difference lies in the number of nodes used as contacts in the protocol: ZRP "bordercasts" all the nodes at the border of the vicinity, while CAPTURE [38] contacts only a few nodes further away. Thus, the vicinities of the contacts in ZRP overlap a lot, resulting in inefficiencies.

2.3.5 The global table approach

In order to solve the location service problem for small to medium networks, LEAP [41, 42] uses a global table called *legend* that references the position of all the nodes in the network and travels from node to node.

In this protocol, nodes periodically send a beacon containing their position to

their neighbors that cache this information. A table containing an entry for each node in the network is also created by one elected node. The table contains three fields: *node ID*, *position*, *update_time*, *visited*. The legend then visits all the nodes. When receiving it, a node synchronizes its local cache and the legend so that both record the latest known information. It also sets the *visited* field corresponding to its address to 1 and then sends the legend to another unvisited node. When the legend has visited all the nodes, another one is created after a pause time.

Scalability is the main drawback of this protocol that can solve easily the location service problem for small and medium networks. As the network size grows, the size of the legend and the time for it to visit all the nodes increase. The use of several legends over a large network could solve this problem, but the coordination of these legends and the share of location information between legend systems are still open issues. The system has also no use for file sharing or service discovery applications as it is designed to share small data items that are frequently updated.

2.3.6 Conclusion

In this section, we reviewed the different approaches for link based resource discovery protocols and classified them in five categories. Most of the categories suffer from a lack of scalability and cannot adapt well to high mobility scenarios. The backbone/quorum approach suffers from the difficulties of maintaining a structure in unstable environments, while the legend approach does not scale well because of the legend's size. From this review, it seems, that the small world approach is the most promising technique to adapt to challenging scenarios. This class of protocol

Table 2.1: Comparison table of the link based protocols

Protocol Category	**Scalability**	**Mobility**	**Hit Ratio**
Selective Forwarding	Does not scale well	Limited effects	Good for small networks
Node Hierarchy	Good scalability for static networks	Not adapted to mobile networks because of high overhead	Good in static networks
Small World Effect	Good scalability	Mobility has a limited impact on performance	Good if the depth of search is adapted to the size of the network.
Quorums	Does not scale well	Very sensitive to mobility	Good in static networks only
Global Table	Limited scalability	Designed for mobile networks	Good if the network is not too large

is extremely adaptable because it can operate without maintaining any structure or link between the nodes. The strength of the protocol in large network lies in its ability to scan the network by sending messages to areas that do not overlap. This *geographic* forwarding is quite close to the *semantic* forwarding used by the selective forwarding protocols like GSD. But these protocols have their application limited to medium networks due to the predominance of the proactive component of the protocol. However, we believe that the two approaches could be complementary, as selective forwarding could be more efficient than small-world approaches in small and medium networks. Thus the semantic forwarding could be efficiently used for resources that are queried in the vicinity of the node, while a small world approach could be in charge of global queries. The legend approach seems also quite efficient for small networks but it does not scale well and is limited to location aware networks. Finally, the backbone and quorum protocols lack flexibility due to the structure they create. But in low mobility scenarios, these approaches could prove efficient, especially in large networks where the hierarchy of nodes could lower the costs of insertions and queries. This solution also suits to heterogeneous networks where powerful nodes and weak nodes coexist and collaborate. The most active and powerful nodes could then naturally play the role of backbone nodes. This would limit the election process of the nodes forming the backbone and stabilize the backbone.

2.4 Location aided protocols

In this section, we introduce the resource discovery protocols developed for location aware networks. Therefore, it is assumed that all the nodes know their position

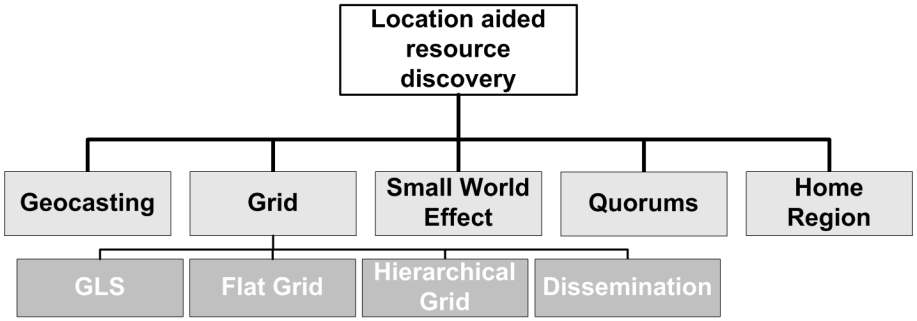


Figure 2.5: Classification of the link based resource discovery protocols

through any localization system such as GPS or one of the localization protocols proposed in the literature. Location service is the main application for resource discovery in this kind of network. So most of the proposals presented in this section aim at solving this problem. We group these protocols in five categories, as shown in Fig. 2.5.

2.4.1 Geocasting and epidemic dissemination

Geocasting consists in delivering a packet to all the nodes located in a geographic area. It can be seen as a geographically limited flooding. The most simple location services use this technique to locate a node. The *Distance Routing Effect Algorithm for Mobility* (DREAM) [19] is a proactive scheme in which each node broadcasts its position in the network. Each location packet (LP) contains the node's ID, position, and speed. When a node looks for another node D, it evaluates the *expected area* the target should reside in. This area is defined as the circle of radius $v(t_1 - t_0)$ centered at D's last known position, where v is D's speed at the update received at t_0, and t_1 the current time. The query is then geocasted in the *requested area* depicted in Fig. 2.6. If the query fails, it is flooded over the network. The

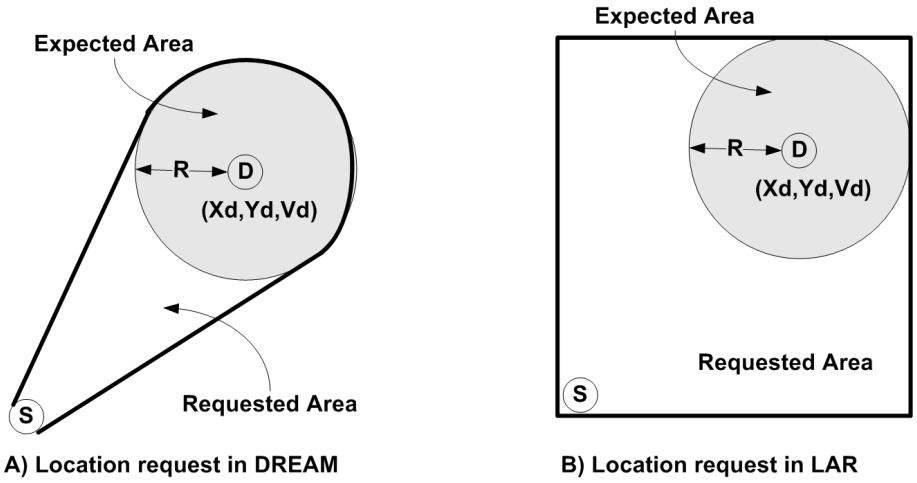


Figure 2.6: Location Query in DREAM and LAR: the query is geocasted in the requested area.

reactive protocol Location Aided Routing (LAR) [43] uses the same mechanisms but modifies the shape of the request zone to improve the hit ratio for remote nodes, as shown in Fig. 2.6. Finally, [44] proposes to expand gradually the request zone of LAR instead of flooding when the first query fails, because the target node should be in the vicinity of the expected zone. These basic mechanisms obviously generate too much overhead for the localization performances provided.

A comparison [20] of reactive flooding, proactive flooding and epidemic dissemination shows that the epidemic dissemination of the location table has the best performance for a 100 nodes network. But only the accuracy of the location information is measured: the cost of finding a destination is not evaluated, thus the results are not fair with the reactive flooding algorithm that usually finds a route in the process. Epidemic dissemination only consists of periodically broadcasting the node's current position and the cached location information of E other nodes. This scheme is enhanced in [45]. The node broadcasting a location packet (LP) includes its velocity vector and the E entries are chosen among the entries updated

since the last LP sent. Active caching is also introduced and the protocol uses linear movement prediction to evaluate a destination's position. Simulations show a good accuracy of the cached positions compared to other protocols like GLS [7] or LEAP [42] for a lower cost. However, the simulations are not fair because the authors use the Random Waypoint mobility model where the nodes have a uniform and linear movement that matches the prediction algorithm. It is doubtful that the performance of the algorithm is as good in real scenarios. Epidemic mechanism proves also inefficient when the network size increases, in which case flooding would be more frequent. This approach remains close to LEAP and should prove efficient for small and medium networks.

2.4.2 The small-world effect

Using the small-world concept presented in the previous section, the *Acquaintance Based Soft Location Management* (ABSLM) [46] modifies the selection of the contacts using location information. As explained earlier, nodes select C contacts in the network and the resource discovery is achieved with the help of these contacts. In the proposal, the contacts are proactively created and maintained as shown in Fig. 2.7. Using periodic beacons, nodes are aware of their neighbors' position. The symmetrical contact selection then takes place in the vicinity of the node: two neighbors first agree being each others' contact and send their positions to each other when moving apart. However, if two contacts remain neighbors, the link is broken and a new contact selection starts over.

As we have explained earlier the resource discovery is propagated between contacts, with a depth of search specified by the requesting node. Simulations show

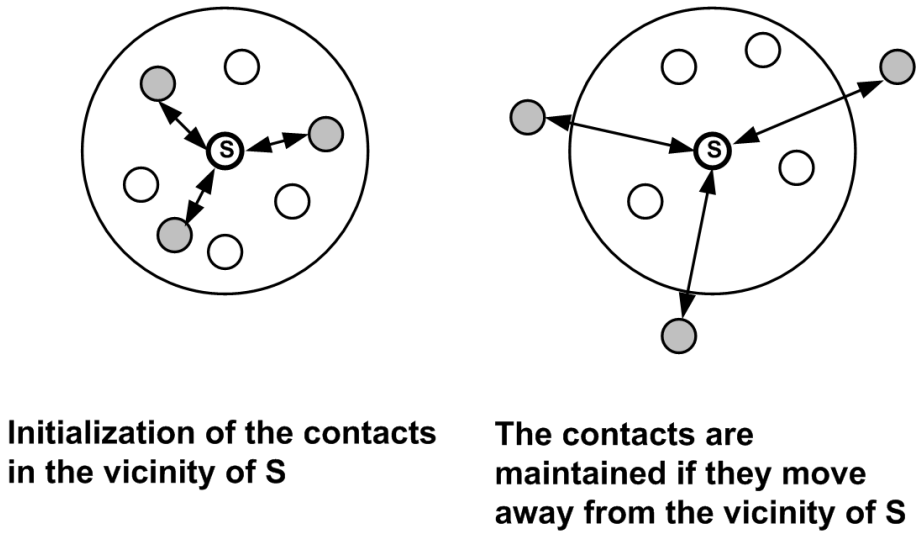


Figure 2.7: Contact selection in ABSLM

that ABSLM performs well with medium networks. A good hit ratio is reached with a low cost compared to LAR and SLALoM [10] (a grid based protocol studied in subsection 2.4.6.1). However, we believe that the difference in overhead performances between SLALoM and ABSLM would drop if the radius transmission (350 meters in the simulations) decreases. Indeed, a smaller radius requires more contacts, a deeper search among the contacts and more frequent updates. Scalability is not studied here, but SLALoM should scale better when the network size grows, i.e., the update and query costs should only linearly increase, while the query cost exponentially increases in ABSLM. A reactive selection of the contacts as in [37] should also improve the performances, especially in real scenarios where nodes are likely to stay grouped, making the proactive selection less efficient when nodes do not move apart.

2.4.3 Geographic quorums

In the geographic quorums approach, nodes share the network in geographic shapes where any *write* shape intersects any *read* shape.

2.4.3.1 Straight lines quorums

[47] proposes to store the advertisements in a straight north-south line, while queries are sent over a west-east line, as shown in Fig. 2.8. Both lines are supposed to cross at a node. In order to increase the intersection probability in mobile environments, the column is widened over two or three hops using overhearing of the messages. An important issue is the update frequency of the advertisements. As a basic update, a node broadcasts its location to its neighbors when a broken or a new link is detected. After E basic updates, a main update along the north-south column is initiated. To increase the intersection rate, [48] sends the messages along a cross shape and [49] improves that scheme by limiting the propagation of duplicate services in the network. These propagation schemes are shown in Fig. 2.8. The main drawback of this solution lies in the high update/query costs in large networks because the number of nodes implied in each operation is $O(\sqrt{n})$. The difficulty of routing around local obstacles or holes raises also some issues for the messages to propagate in the whole network. Some simulations show that geographic routing algorithms still regularly fail [50, 51]. Therefore this kind of protocol could be quite efficient in low mobility, medium networks.

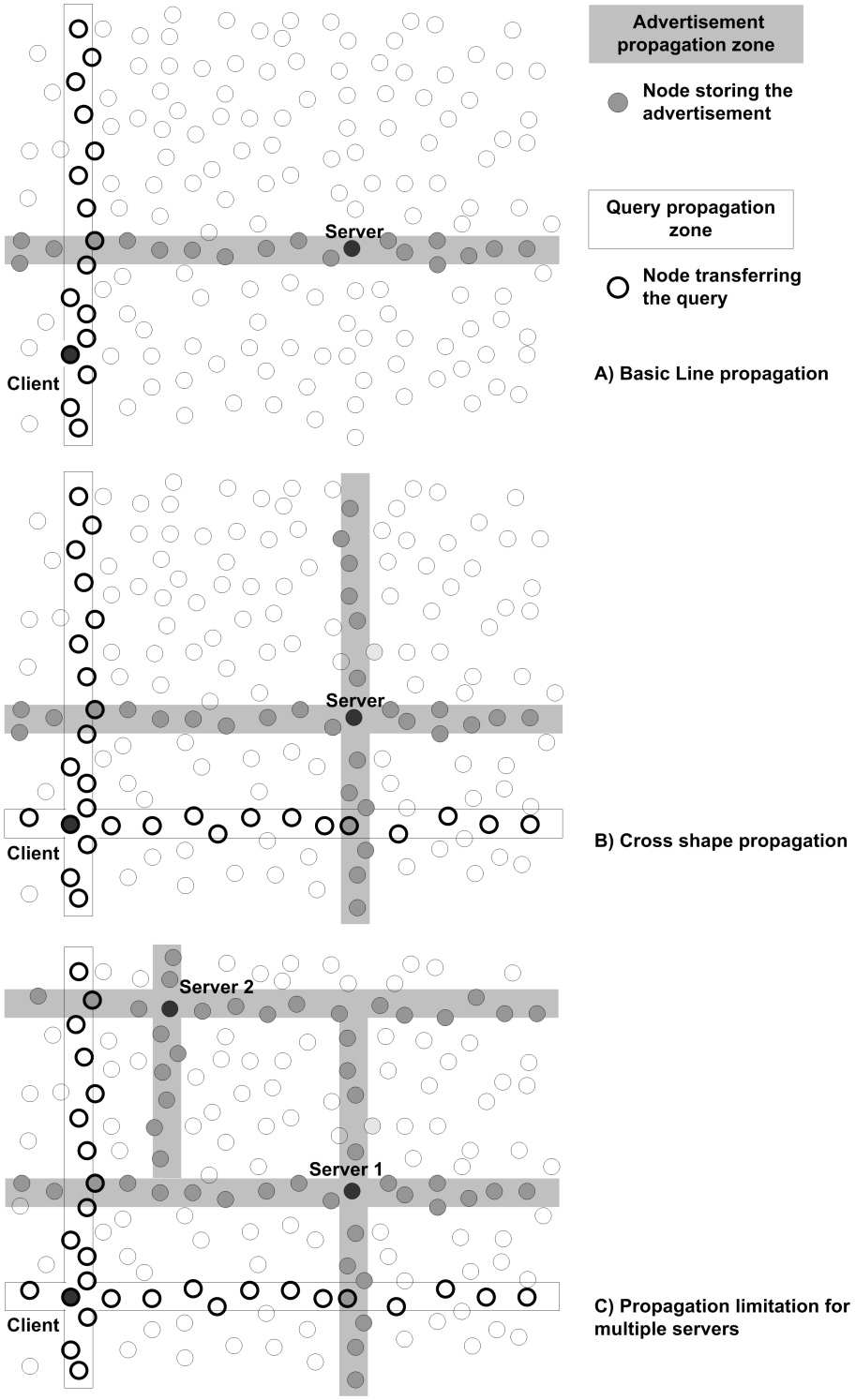


Figure 2.8: Crossing lines in geographic quorums

2.4.3.2 A spiral approach

The protocol in [52] is based on spirals that store the location of a node. The spiral, built on a hierarchical grid is displayed in Fig. 2.9. The nodes close to the corners of the rectangles are responsible for storing the advertisements. The origin of the grid is determined with a hashing function and the ID of the advertising node. Using the same algorithm, a query follows a spiral intersecting with the advertisement spiral. Some more mechanisms described in [52] improve the performances of the protocol. However, this scheme multiplies the number of storage points because for each corner of all the spirals, the closest node stores the advertisements associated. This results in numerous data transfers when nodes move. Instability and overhead should easily appear in mobile environments. This issue is studied in the next section. Finally, compared to a simple line propagation, the long length of the spiral increases the update costs and the risk of routing failures in environments with obstacles.

2.4.4 The Personal Home Region

In a personal home region protocol, each item is stored in its own and unique attributed geographic area that acts as a server. This concept is first developed in [53] [54] with the *Virtual Home Region* (VHR) that consists of selecting a circle to store a node's location. All the nodes know a hashing function H that is used to map a node's ID to a position (x,y). The VHR of a node is defined as the circle centered at H(ID) = (x,y) of radius R. When a node moves to a new position, the new location is then geocasted to its VHR and all the nodes in the area store the

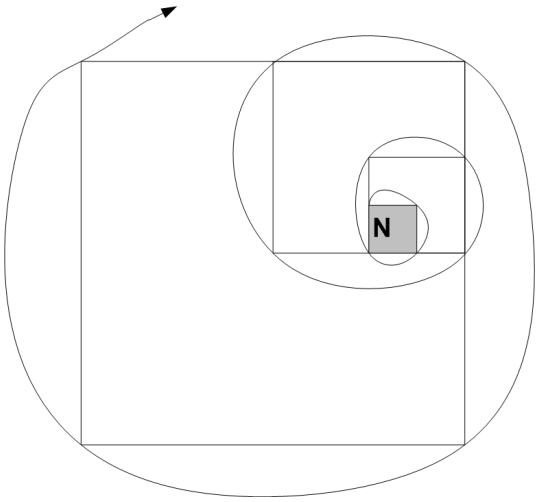


Figure 2.9: The spiral of node N is built on a hierarchy of rectangles. The nodes close to the intersections of the spiral with the hierarchical rectangles store the advertisements. A similar spiral is created for the queries.

update. The nodes also have to resolve the queries of the VHR they reside in. However, adjusting R to the local density of nodes is still an open issue. If it is too small, the VHR could be empty, but it should not be too large either in order to limit geocasting. Data management in mobile environment is also an issue. When a node enters a VHR, it should have its cache updated to be able to resolve queries. But there are as many VHRs as nodes, so the update operation would be extremely frequent. On the other side, if the nodes are not updated when arriving in a VHR, the performance of the protocol may degrade, unless frequent updates arrive in the VHRs. For these reasons, this mechanism should not outperform other solutions.

In order to solve the radius and mobility problems of the VHR approach, the *Geographic Hash Table* system (GHT) [55] for data centric storage sets R to 0, and a fixed point is used as server. The nodes around that point are elected as servers using the perimeter mode of the Greedy Perimeter Stateless Routing (GPSR) [56] protocol, i.e., GPSR recovers from greedy forwarding by routing around a hole

using the right hand rule. In GHT, when a data packet is sent to a destination D, no node will be exactly on that point. The packet will then travel around D in perimeter mode and the nodes receiving it store the data item. In order to ensure consistency in mobile environment, *refresh* packets are periodically sent around D. This approach has been adapted with minor modifications for location service in [12] for comparison purpose. GHT performs well in dense networks with medium mobility, and scales very well with the number of nodes. But this mechanism lacks the ability of adapting to its geographic area. Indeed, GHT fails if the hashing function points to an empty area created by a geographic obstacle or node movement patterns. Some more mechanisms should then be added to ensure that data items are effectively assigned a populated area.

2.4.5 The Grid Location Service (GLS)

GLS [7] is one of the first scalable location services using a hierarchical grid and it includes many innovative features. Therefore it is often used as a reference and we only briefly describe the protocol. More detailed information and proofs can be found in [7, 12, 50].

The geographic area is recursively divided in a hierarchy of squares from level *0* to λ where each square of level $i > 0$ is divided into four identical *sibling* squares of level $i - 1$. The smallest squares, of level 0, are called *cells*. With this structure, each node is located in exactly one square of each level and selects one node in each of its sibling square to act as a location server. $3(\lambda + 1)$ servers are then selected and they only record the node's cell because the nodes sharing the same cell know each other's coordinates using periodic beacons. The servers for a node A are the

nodes with the least ID greater than A's ID in their square. Fig. 2.10 displays an example of grid and the server repartition for a node A in the network.

If node B seeks node A, it sends a query to the node with the least ID greater than A's ID that it knows. This node will then forward the query to another node in the same way. This mechanism will find in a few steps one of A's server in the square common to A and B. When the server receives the query, it forwards it to A which replies to B. The update mechanism is quite similar. When A updates the servers in a square, the message is first sent to that square and then routed in the same way as a query. After a few steps, a node in the square won't have cached any other node with an ID between A's and its own. This node will act as the server. To limit overhead, a *distance effect* is implemented such that the closer servers receive more frequent updates than the remote ones, i.e., when a node travels the distance $2^{i-1}d$, it updates the servers in the square of level i, where d is a protocol parameter. The proofs for these mechanisms can be found in the original paper [7].

The mechanisms GLS rely on are extremely simple and easily implemented. The protocol does not use any form of flooding and is totally distributed and seamless as the nodes do not know the ID of their servers. However, mobility is an important factor in the performance of the protocol. [12] extensively compares GLS with a variant of GHT named GHLS and a line-quorum protocol called XYLS. Even in static networks, GLS creates much more overhead than GHLS due to the higher number of updates produced and longer query path length. The situation deteriorates with mobility. The overhead of GLS jumps far above that of the other protocols. The difference is deepened as the precision of the positions stored in GLS servers are often not good enough for routing. [45] and [50] observed the

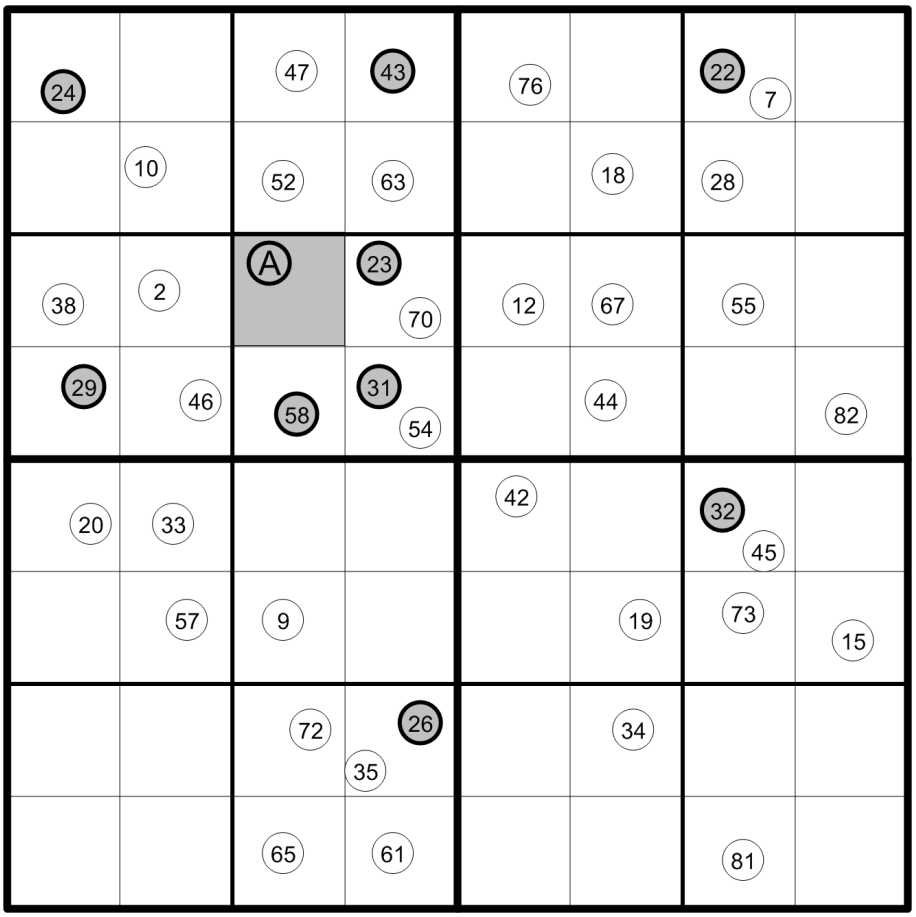


Figure 2.10: Repartition of the servers for node A (ID = 21) in GLS: the node with the least ID greater than 21 is elected server in each sibling square of node A.

same inaccuracy. This phenomenon is generated by the *distance effect*. If nodes are uniformly spread on the grid, 75% of them access the position of node A with the worst granularity because they stay in the level λ sibling squares of A. Therefore we believe that each level should be updated in a round robin fashion at the same frequency. This modification only improves the accuracy of GLS but not the overhead. Another solution would be to hierarchically route the queries from a server to node A, i.e., a level $i+1$ server would point to the level i square of A. Updates would then be initiated only when a node crosses region borders. But the query path length would then jump because the servers could be anywhere in their square. Further consistency problems would also raise when servers move out their region. Therefore, no simple modification improves the performances of GLS and based on the results of [12], we believe that GLS is not efficient enough compared to other protocols, even if it develops an elegant and innovative solution to the location service problem.

2.4.6 Grid based protocols

As in GLS, other resource discovery protocols divide the geographic area into *cells* that form a grid. In those grid based protocols, each cell acts as an independent server. We did not include GLS in the grid based protocols because this category groups the schemes that use the cells as servers. In GLS specific nodes are elected servers. For all the protocols in this section, it is assumed that all the nodes know the shape of the grid and periodically broadcast a periodic *hello* beacon with their position to their neighbors.

2.4.6.1 Flat Grid

The *Scalable Location Update-based Routing Protocol* (SLURP) [11] is one of the first location services using a grid. In SLURP, each node sends the ID of the cell it occupies (*host cell*) to another cell called *home cell*. This cell is determined with a hashing function H that hashes the ID of the node to the ID of a cell: H(node ID) = cell ID. Any node that wants to locate a target T only needs to hash T's ID and to send a query to the home cell using geographic routing. The knowledge of the host cell is enough to locate T, because intra cell routing is based on a link state protocol. Location updates are triggered when a node changes cell and geocasted in the home cell. All the nodes in the home cell store the location information and resolve the queries addressed to their host cell. Some updates must also be initiated when a node changes cell to maintain the consistency of the database, i.e., all the nodes in a cell maintain a list of the nodes present in their host cell, a node changing cell must then geocasts a message to erase or register its ID in the cells. This message is also used as a request to transfer the location information stored in the new cell to the node.

In this scheme, many messages are geocasted, which can be costly with high mobility. Compared to SLURP, *Rendezvous Regions* (RR) [9] modifies the internal management of the cell and creates a set of server nodes inside the cell. Only a few nodes inside a cell play the role of server, limiting the number of updates and geocast messages if the servers are known by the nodes inside a cell. Another solution to limit geocasting in stable networks is presented in [57]. Summaries of the nodes present in each cell are created and sent to the home regions.

2.4.6.2 Hierarchical Grid with uniform repartition of the servers

In a flat grid organization, all the location updates and queries travel to the unique home region of a node. This organization is quite efficient as long as the number of cells is limited. But in large networks, the path could be quite long resulting in high cost for both updates and queries. To solve this problem, nodes could choose several home cells to decrease the cost of the updates or the queries. Hierarchical grids are used for this purpose. For this kind of grid, the cells are grouped together to form regions of different hierarchical levels with the specific condition that regions of the same level do not overlap.

In the *Distributed Location Management* (DLM) [58], the hierarchical grid is organized in λ hierarchical levels. It is assumed that the diagonal of a cell is smaller than the transmission radius of a node, resulting in very small cells. This design aims at avoiding flooding, but it increases the update rate. It is also assumed that all the regions of same level have the same shape and the same number of cells. Each node then selects a position in a level m region, where m is a network parameter, with an hashing function and its ID. All the cells with this position in the level m regions will act as home cells, as shown in Fig. 2.11.

The location updates are triggered when a node changes cell and could be sent to all the servers. But this mechanism is expensive for mobile nodes because numerous servers are scattered in the grid. A second solution takes advantage of the hierarchical structure as shown in Fig. 2.11. If a node stays in the same level m region, it only updates the server in that region, while all the other servers point to the level m region. All the servers are updated only when a node moves to

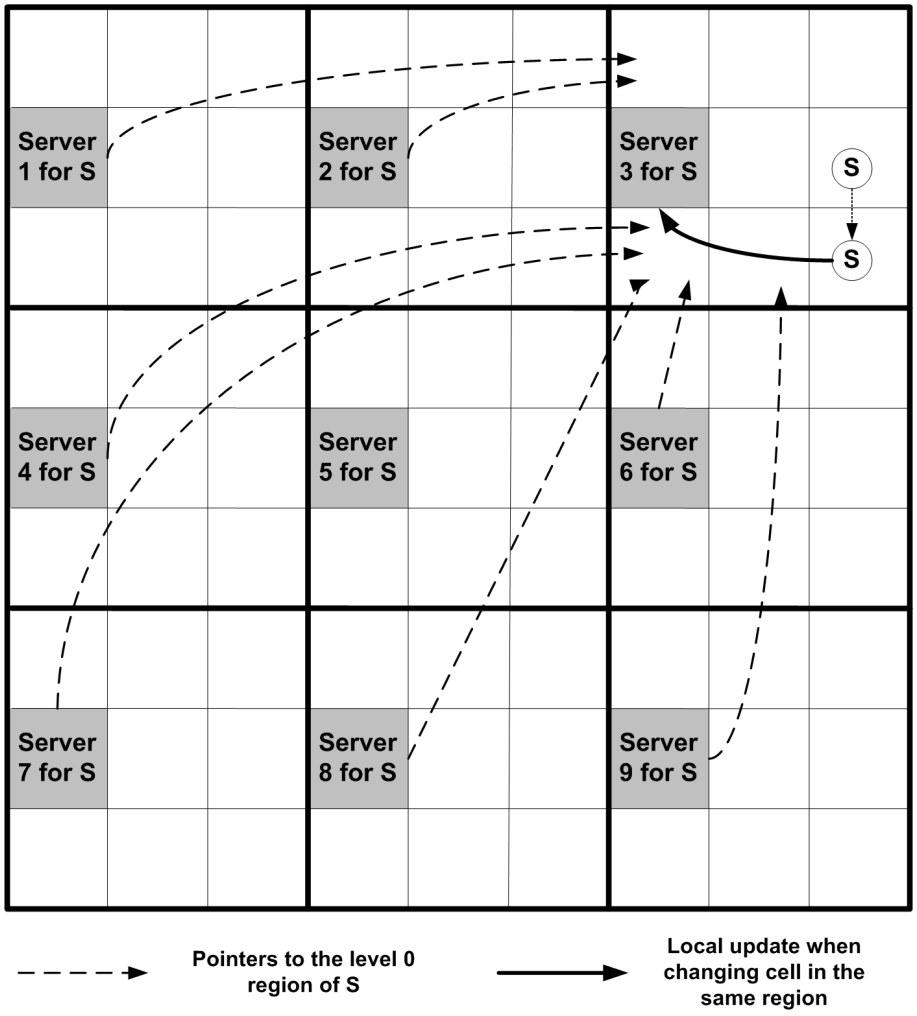


Figure 2.11: DLM with 3 levels, $m = 1$ and hierarchical discovery. Node S selects one server in each region of level m, but only the server in S's region knows its exact location. The other servers point to the level m region of S.

another level m region. When a node A initiates a query, it first evaluates the target's (T) server in its level m region and sends the query to that server. If the server knows T's host cell, it replies to A. Otherwise, the query is forwarded to the server cell in T's level m region, which will be able to reply to A. This second mechanism greatly decreases the update costs. This phenomenon is clearly shown in the simulations of [58]. The two update mechanisms bring flexibility because the nodes can adapt their update policy to their mobility, i.e., static nodes update all servers while mobile nodes use the hierarchical mechanism.

The *ScaLable Ad-hoc LOcation Management* (SLALoM) [10] is almost a specific case of DLM using two hierarchical levels (1 and 2) and setting $m = 2$. However the cells are supposed to be wider than in DLM and a state link routing is maintained inside the cells, as in SLURP. The position update uses a variant of the hierarchical mechanism of DLM. All the home cells surrounding a node (9 overall) store its exact host cell. Simulations show that SLALoM outperforms SLURP when the network size increases, demonstrating the benefits of hierarchical grids.

2.4.6.3 Hierarchical Grid with logarithmic repartition of the servers

It is envisioned that communication between nodes will mainly take place between close nodes. Thus it is unnecessary to update many remote server cells. The server density should then decrease with the distance to the node. Logarithmic repartition of servers in a hierarchical grid addresses this issue.

The *Hierarchical Grid Location Management* (HGRID) [59] relies on the same grid as GLS, i.e., a hierarchical construction sharing each region into 2x2 sub

regions for $(\lambda + 1)$ levels. In each region, a cell is selected as a leader, as displayed in Fig. 2.12. For each level i $(0 < i < \lambda)$ the top rightmost L_{i-1} leader is the i hierarchical leader of the bottom left L_i region, the top leftmost L_{i-1} leader is the hierarchical leader of the bottom right L_i grid, the bottom rightmost L_{i-1} leader is the hierarchical leader of the top left L_i grid and the bottom leftmost L_{i-1} leader is the hierarchical leader of the top right L_i grid.

HGRID is based on a hierarchical update and query process as explained in the previous section. Each L_λ leader stores the level λ region of all the nodes. For $i \leq \lambda$, the L_i region leaders point to the level L_{i-1} regions of the nodes occupying the same level L_i region. When a node changes its home cell, it sends a location update message with the movement to the L_1 leader managing the new cell. The first node in the leader receiving the update geocasts the message in the cell and evaluates if the movement requires a higher level update. If needed, the message is forwarded to the L_2 leader and the same mechanism is recursively repeated. Another update is eventually sent to the leaders which should erase the node from their database. A location discovery message is then sent to the local L_1 leader of the querying node. If the leader cannot resolve the query, it is forwarded to the L_2 leader and so on, until the query reaches the leader of the region common to the querying node and its target, as shown in Fig. 2.12. This server replies with the location of the next location server towards the target. The querying node will then contact that server and the message will be routed hierarchically to the destination.

With this structure, each node only has one server in each hierarchical region and the update costs are further reduced, but the query cost increases a lot with the

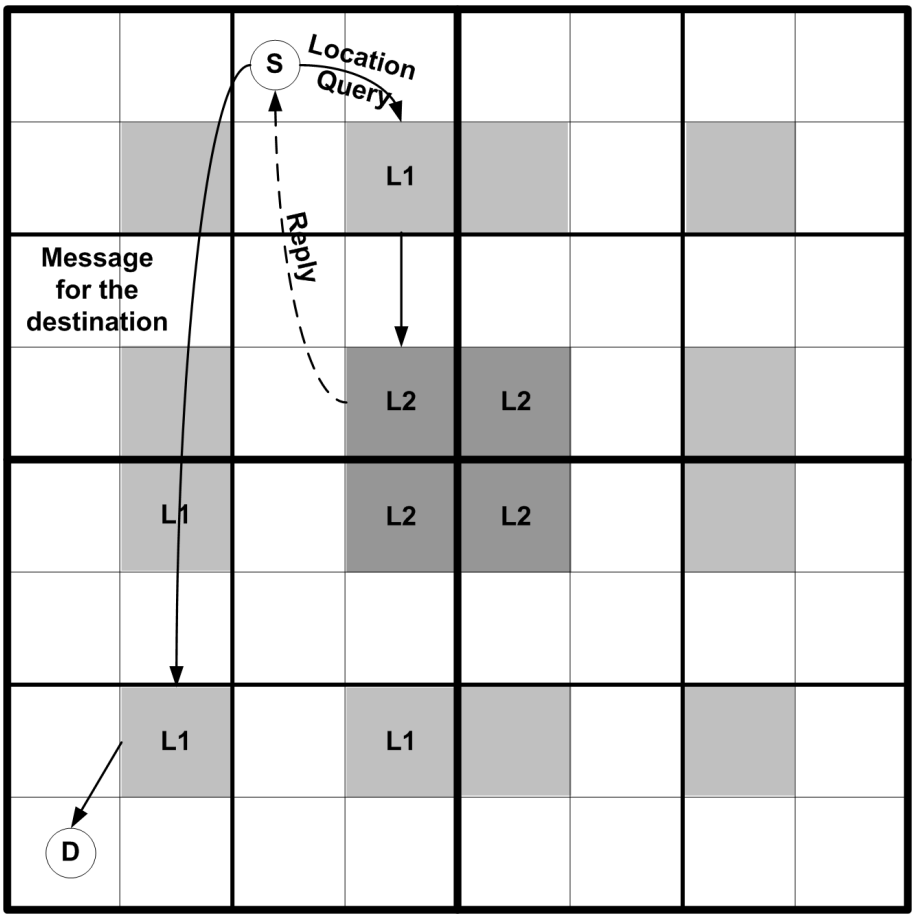


Figure 2.12: Structure of HGRID for 3 levels. Node S sends a message to D in three steps: 1) S issues a Location Query to find one of S's leader; 2) The leader L_i replies with a pointer to the leader L_{i-1} of D; 3) S sends the message that is hierarchically routed from L_{i-1} to D

distance to the target. Compared to SLURP and SLALoM, HGRID performs best in high mobility scenarios taking advantage of the lower update costs. However, this protocol relies only on the region leaders. Only 25% of the cells are in charge of maintaining the structure and the L_λ leaders are even keystones in the scheme because any query between two nodes residing in two different level λ regions passes through one the four L_λ leaders. HGRID is then extremely weak to empty cells and the lack of workload sharing could create bottlenecks in the network.

To solve these problems, the *Hierarchical Location Service* (HLS) [60] takes the advantages of SLURP and HGRID. HLS can be implemented on any hierarchical structure where the regions of same level do not overlap. Each node chooses a home cell in each hierarchical level it resides in using an hashing function adapted to each region, like: H_{region}(node ID) = node ID mod (number of cells in the region). Then the update mechanism is the same as in HGRID, and queries travel up the hierarchy of the *potential* home cells of their target until reaching the home cell of the region common to the querying node and its target. HLS should have almost the same efficiency as HGRID even if the delay and the communication overhead of the queries should be higher because the path in the hierarchy is longer than in HGRID, in which the hierarchy of servers forms a straight line. But the workload is well spread among the nodes, limiting any bottleneck effect and sensitivity to empty cells in the grid.

2.4.6.4 Location dissemination in hierarchical grid

The *Geographical Region Summary Service* (GRSS) [61] relies on the same grid as GLS or HGRID. But the location service approach is totally different, as it is based

on information dissemination and not a server structure. As in many schemes, intra cell routing is performed by a link state protocol and inter cell routing by a geographic routing.

The boundary nodes of each region, for each hierarchical level, create summaries of the nodes inside the region and geocast them to their sibling regions. The summaries of level 0 regions are created using bloom filters on the ID of the nodes hosted by the cell. Whenever a node leaves or arrives in a cell, a new summary is created and geocasted. Summaries for level i regions are created with a simple bitwise OR operation on the summaries of the level i-1 sub regions and they are geocasted only when modified. The bloom filter creates small summaries that can be quickly spread with one small packet. The nodes then use the summaries to find a region that a target T occupies. The query is then forwarded to that region. Once in that region, a node using a more precise summary forwards the query to a smaller region where T is located, until reaching the right cell.

The extensive simulations of GRSS exhibit a good hit ratio and a good scalability with the size of the network and the speed of the nodes. The bandwidth consumed by the protocol is quite small, as long as the summary size is below a few hundred bytes. These results could seem surprising considering the extensive use of geocasting. Indeed, the location update cost is quite high, but the query cost is almost null as the queries are almost optimally delivered to the target node with the direct routing, i.e., the queries do not have to travel to servers before being forwarded to the target like in SLURP or HGRID. GRSS appears to be an original and efficient solution for the location service problem that would deserve comparison simulations with other protocols. But contrary to the server approaches, GRSS can

only be applied to service discovery because of the small summary size required.

2.4.7 Conclusion

In this section, we have grouped the location aided resource discovery protocols in 5 categories. Compared to section 2.3, the quorum and the small world approaches have been augmented with location awareness to improve their efficiency. However, for a location service application, the grid protocols should perform better in large networks. Indeed these protocols send only one message for updates or queries to a server cell while the small world approach requires sending several messages for each query, and the quorum approach suffers from mobility weakness. The grid based protocols could also benefit from large cells. With large cells, the number of location updates decrease because border crossing would become less frequent. But this requires an efficient data management inside the cells limiting geocasting, as introduced in RR. However research is still needed in that area, in order to design adaptive policies that could be efficient for any cell size and any number of nodes inside the cell.

Grid based protocols can also adapt to a wide range of mobility scenarios and network sizes using hierarchical grids and different repartition of servers in the network. However, the grid based protocols fail with varying effects when empty cells appear in the grid. This weakness is common to the home region approaches which store data items at geographic points. If wide areas are empty, items sent there would experience problems to be stored and queried. For these protocols however, adapting the common hashing function to the nodes' position seems a very difficult task and would require a high amount of communications. There-

fore, we believe that this approach would be unsuitable in mobile scenarios where the node distribution could vary greatly over time. The performance comparison works [12] [9] show that this approach, close to the grid mechanism, outperforms easily the quorum protocols and GLS. This protocol, even if elegant, seems to generate too much overhead and performs poorly in large, mobile networks. The quorum approach could prove quite efficient in medium networks with low mobility. It also has the advantage of not relying any kind of hashing functions, which make it easily adaptable for service discovery.

2.5 Conclusion

2.5.1 Strengths and weaknesses of grid based protocols

Even if some link based protocols achieve successful resource discovery, they still suffer from a lack of scalability and their efficiency decreases with the diameter of the network, except for the small world approach that performs best in large networks. However, location information greatly improves the performances of resource discovery protocols. Comparison works like [12] or [9] also show that the grid based and GHT-like protocols have the best performances among the location aware protocols. These protocols use only few messages to maintain the structure of the resource discovery, and only need to send one message for each insertion and query, resulting in good efficiency. But both approaches cannot dynamically adapt to topology changes that create empty spaces on the geographic area. We believe that adapting the hashing function of GHT to the topology is not possible or would require too much communication between the nodes. Thus this approach

should be used in scenarios where the density of nodes would remain quite high everywhere.

Grid based protocols not only perform well, they also offer a wide adaptability in their implementation. The shape of the grid, its management, as well as the intra cell mechanisms offer a wide choice of designs that could be separately studied and improved. Thus they could be used and adapted to many different scenarios using large networks. We could also imagine a resource discovery protocol mixing a grid with a link based resource discovery inside the cells in order to improve the research of close services, while taking advantage of the grid structure for global queries.

For these reasons, we believe that the grid based protocols are a promising approach to resource discovery in wireless networks, even if many improvements have to be made to use them in real scenarios.

2.5.2 The effect of empty cells on grid based protocols

The main weakness of the grid based protocol is their sensitivity on empty cells. The empty cells in the grid have severe effects for two reasons. First, if a cell is empty, all the queries that will arrive there won't be replied. The hit ratio of the protocol then quickly decreases. If the application using the grid based protocol is a service or file sharing application, then the resources are lost and not accessible anymore. But the phenomenon is worse for a location service supporting a geographic routing protocol because all the nodes which use the empty cell as home cell cannot have their position queried. This means that many nodes in

the network are not contactable anymore because their position is unknown. This situation is unacceptable in a collaborative environment. Second, all the messages that are sent to an empty cell don't actually have any destination, so the packets just wander around the empty cell exhausting their TTL and creating useless traffic in this area. Therefore, empty cells, not only result in denial of service, but they also generate overhead and maybe congestion if the utilization of the protocol is high.

The severity of holes on the hit ratio also depends on the type of grid based protocol. If a flat grid is considered, like in SLURP, all the nodes having an empty home cell are affected. In a hierarchical grid, the situation is more complex and the effects vary between two extreme cases. In the best case, the servers are uniformly distributed and they all know the host cell of the node they work for. As a result, the information is replicated in the network, and the effect of the empty cell could be easily limited. For example, a node receiving no reply for one of its query could query another server cell containing the information. The worst case scenario occurs with a hierarchical grid like in HGRID where the protocol relies on only 25% of the cells, the leaders. If one of those cells is empty, all the queries arriving in that cell are lost. The area covered by the leader is then virtually isolated.

Adapting the grid is therefore a necessary task to maintain the good performance of the grid based protocols. We describe our solution, Hidagrid, in the next chapter.

Chapter 3

Description of the Protocol

Grid based protocols are powerful schemes but they lack the ability to dynamically adapt to the density of nodes. Some basic solutions have been proposed, and we will analyze them in the first section of this chapter to identify the challenges Hidagrid must face. Then, based on these requirements, we describe our solution, Hidagrid in section 3.2. Additional features and improvements are discussed in Section 3.3, in particular we will see how to stabilize the state of the cells, and will generalize some mechanisms for any scenario. Section 3.4 concludes this chapter with a summary of Hidagrid's distinct features.

3.1 The design requirements of Hidagrid

3.1.1 Basic strategies for empty cells management

All the grid based protocols reviewed in the previous chapter fail with varying effects when a cell is empty, except GLS [7] and GRSS [61]. However their database

management differs from the other approaches that use cells as servers: in GLS, nodes are selected as servers, while GRSS is based on gossiping and not storage. Thus these two grid protocols do not suffer from the *hole* problem by design. However, some proposals are aware of the empty cell problem and mention simple mechanisms to deal with them. But none of them strives looking for an efficient empty cell management, and no simulation with empty cells has been reported in those works.

SLURP [11] deals with the empty cells by locally maintaining a list of the nodes present in each cell. When the last nodes in a cell leave, they replicate the locally stored items in the eight cells surrounding the empty cell which form the new server for the messages targeted to the empty cell. As a result, any update or insertion targeted to the empty cell must be geocasted in the eight cells, generating high update and insertion costs. This mechanism also increases the size of the storage space in the network because the data items are then replicated in eight cells instead of being stored in only one cell.

HLS [60] slightly improves this scheme by storing the items hosted in an empty cell in only one cell: a node detecting an advertisement message that cannot reach a target will temporarily save the item in its own cell and try to insert it regularly in the original host cell. This mechanism results in vain reinsertions attempts for each item as long as the cell remains empty. Queries looking for such data items also have to wander around the empty cell to find the temporary host cell, because the relocation is not determined by a policy. This increases highly the query cost and this system easily fails as messages do not have any assigned target and just visit all the cells to try to find the queried item. Even if the relocation in only one

cell is an improvement, this approach lacks a common relocation policy. It is also not explained how a node differentiates an empty cell from a routing failure.

DLM [62] follows the same lead as HLS but the backup server is designed with the help of a common policy. Nodes can detect empty cells because of the very small size of the cells: the area covered by a node's transmission radius is supposed to cover a full cell. This small size increases highly the risk of empty cells. However, more importantly here, nodes are not aware of an empty cell being populated again. As a result, an item could be stored in a backup server but it will not be reinserted in the main server cell when it becomes active.

Finally, RR [9] also mentions the empty cell problem and proposes to use a backup hashing function when an insertion acknowledgement or a reply is not received. This means that in the case of an empty cell, at least two messages have to be sent for each insertion and query. Furthermore, this mechanism cannot make the difference between empty cells and routing failures that are quite common in wireless networks. Thus after a simple routing problem, a node would change the target cell instead of trying to contact one more time a cell that may be active.

3.1.2 Requirements for empty cells management

This short review shows that all the simple mechanisms proposed do not manage to bring an effective solution to the empty cells. But these proposals highlight the problems encountered when empty cells appear in the grid. A protocol dealing with that issue must face the following challenges:

- When holes are detected, the items must be relocated in one or several other servers according to a common policy that can be used to route queries to the right cell.
- The mechanism should not increase the update or query cost in order to keep the bandwidth consumed by the discovery service as low as possible.
- Consistency must also be maintained when empty cells get populated: the data items that were stored away must be relocated quickly in order to avoid denial of service.
- Finally, the protocol is only a background protocol that should generate minimum overhead when the grid is stable. However, the changes in the grid depend only on the local variation of node density, which is a slow process. As a result, to ensure the protocol's consistency, higher communication overhead is acceptable when a cell gets deserted or populated, as these events should have a low frequency.

Based on these requirements, we describe Hidagrid in the next section.

3.2 Description of Hidagrid

3.2.1 Assumptions

A mobile wireless data network is considered. We assume that the network is well connected but that the node density can strongly vary, resulting in empty areas as well as dense clusters, as shown in Fig. 3.1. The nodes know their location through

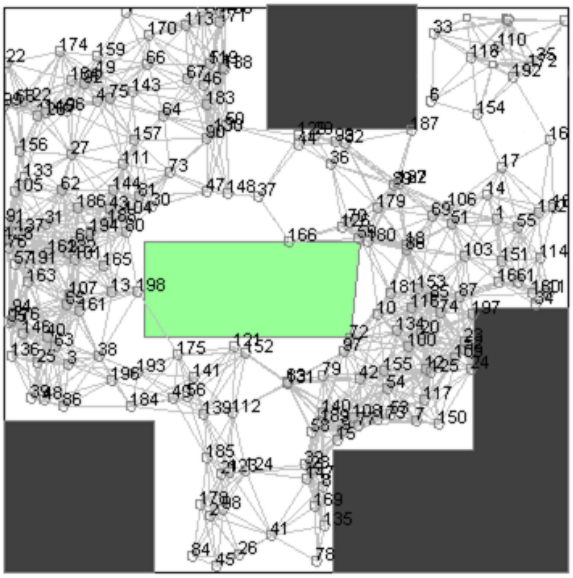


Figure 3.1: Example of network supported by Hidagrid: the movement of the nodes (numbers) is restricted by the obstacles (dark shapes), creating empty areas.

Global Positioning System (GPS) or any other localization algorithm proposed in the literature [63] [64] and this ability supports a geographic routing protocol such as the Greedy Perimeter Stateless Routing (GPSR) [56].

The network space is divided into a *grid* formed of basic rectangular zones that we call *cells*. To divide the network area into cells, all the nodes must know the extent of the area as well as the dimensions of the cells. They can then compute the borders of the cells.

Finally, we assume that the discovery protocol supported by Hidagrid uses the messages displayed in Table 3.1 and that it implements an acknowledgement mechanism after the successful insertion of a data item in a cell.

Table 3.1: Messages used by the discovery protocol and Hidagrid

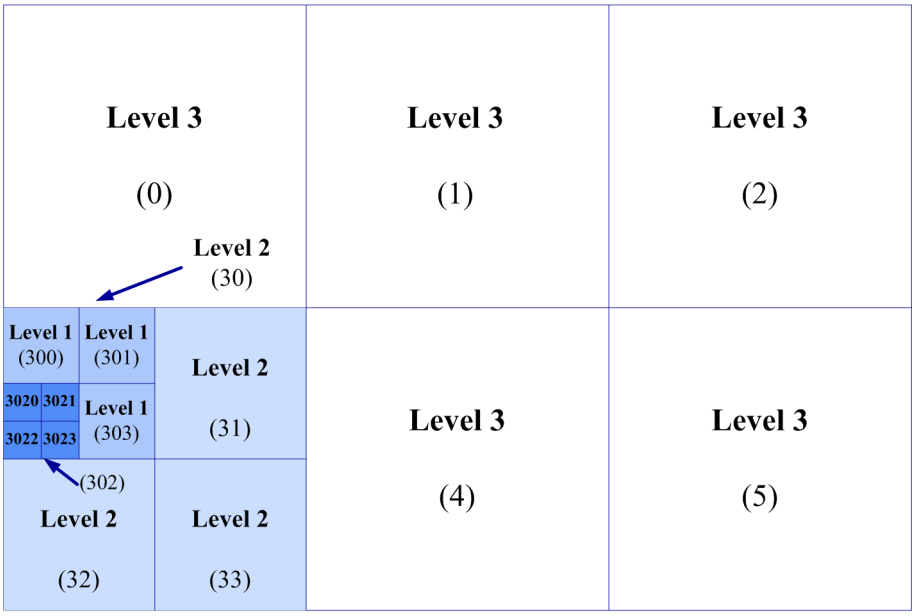
Discovery Protocol	
INSERT	Insertion of a data item in a server cell.
INSERT_ACK	After a successful insertion the server cell sends an acknowledgement.
QUERY	Query for a data item.
REPLY	The reply for a query can be positive if the queried item is stored in the cell, or negative otherwise.
Hidagrid	
GRID_UPDATE	Advertisement for the state change of a cell.
UPDATE_RQT	Request for updating the internal table of a node.
UPDATE_RPL	Reply to UPDATE_RQT containing an internal table.

3.2.2 Overview of Hidagrid

Hidagrid is located between the routing layer and the service discovery protocol layer. It is not responsible for the data items management inside the cells (this is the role of the discovery protocol), but it distributes items of empty cells among the populated ones. Its role is then to consistently route the QUERY and INSERT messages of the discovery protocol to a populated cell, based on the original server cell decided by the discovery protocol, the data item identifier and the state of the grid. To support this task, Hidagrid implements five main functions that are described in the next sections: (i) advertising cell state changes, (ii) updating the node's internal table, (iii) routing messages hierarchically, (iv) relocating data items, and (v) detecting the state changes of the cells.

3.2.3 Advertising cell state changes

Each Hidagrid cell is characterized by a state: active or inactive. When a cell is inactive, the nodes inside can still use the discovery service but the cell will not be in charge of storing any information, i.e., all the workload is transferred to active cells with higher node densities. When the state of a cell changes, as explained in Section 3.2.7, the event is advertised in the network with a GRID_UPDATE message. A simple solution would be to flood that message, but in large networks with many cells, the frequency of state changes and then the overhead could be high. To limit this, Hidagrid is built on a hierarchical grid that defines the spread of these advertisements. A GRID_UPDATE message is then geocasted in one of the regions of the hierarchical grid, as described in this section.



Level	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	3	3
Addr.	3020	3021	3022	3023	300	301	302	303	30	31	32	33	0	1	2	3	4	5
State	Act	Act	Act	In.	In.	In.	Act	In.	In.	Act	Act	Act	In.	Act	Act	Act	In.	Act

Figure 3.2: A 4 levels hierarchical grid with 6 level-3 regions, and the internal table of the nodes located inside region 302. The hierarchy is depicted in the lower left corner and each hierarchical region is created with four regions of lower level. The internal table lists the state of the regions depicted on the figure.

3.2.3.1 The hierarchical grid structure

The hierarchical grid is built by grouping cells into *regions* of different levels. Four neighboring cells form a region of level 1 and the hierarchy is built by grouping four neighboring regions of level i into a region of level $i+1$. Regions of the same level do not overlap and as a result each cell belongs to only one unique region for each level of the hierarchy. Four regions of level i sharing the same region of level $i+1$ are called *sibling regions*, while the region of level $i+1$ is called the *parent region* of its *child regions* of level i. We will consider a grid of $\lambda+1$ hierarchical levels with a cell being a level 0 region. The N number of regions of level λ can vary in order to

adapt to the shape of the network area. Fig. 3.2 shows an example of a hierarchical grid with 4 levels and 6 level-3 regions.

Each region of the grid is also characterized by a state, active or inactive. The state of a region is based on the state of its child regions and is evaluated according to the State Policy. Data items hosted by an inactive region are relocated to active regions so that no information is stored in any cell of a region with low node density.

3.2.3.2 The State Policy

Each time the state of a cell changes, the state of the regions containing that cell is evaluated by the State Policy when the node advertising the change evaluates the scope of the GRID_UPDATE message. The State Policy is displayed in the pseudo code of Fig. 3.3. It first evaluates the state of the lower hierarchical levels, and the algorithm stops when the state of a region does not change. Let us consider a region R of hierarchical level i. If R is inactive, it is activated when $a_R > A(i)$, where a_R is the number of active child regions and $A(i)$ is the *activation level* for the hierarchical level i. Similarly, R is deactivated when $D(i) \geq a_R$ where $D(i)$ is the *deactivation level* for the hierarchical level i, with $A(i) \geq D(i)$. As a result, a region is deactivated when few active cells populate it so that the workload of the discovery protocol is shared between the most populated areas.

3.2.3.3 The spread of GRID_UPDATE messages

Using the hierarchical grid and the State Policy, a node changing the state of a cell first evaluates the impact of the change on the hierarchy of regions. It determines

```
for(i = 1; i ≤ λ; i++;){
  R = Region_Of_Level(i) ;
  old_state = R.state_ ;
  T_R = R.total_child_regions_ ;
  a_R = R.active_child_regions_ ;
  if(!Is_Active(R) && a_R > A(i)){
    R.state = active ;}
  if(Is_Active(R) && D(i) ≥ a_R){
    R.state = inactive ;}
  if (R.state == old_state) {return i;}}
```

Figure 3.3: Pseudo code of the State Policy.

the first region in the hierarchy whose state remains unchanged after updating. A GRID_UPDATE message containing the identity of the cell and the new state of the cell will then be geocasted in that region only. For example, in Fig. 3.2, the table lists the state of the regions displayed on the graph. It is assumed that a region is declared inactive when at least two of its child regions are inactive. As a result, region 302 is active because three of its child regions (3020, 3021 and 3022) are active, as shown in the table. If we assume that cell 3022 turns inactive, region 302 then also turns inactive because only two of its child regions are active. However, region 30, the parent region of region 302 remains inactive after the state change. As a result, the GRID_UPDATE message is geocasted in region 30.

3.2.4 Updating the node's internal table

Each Hidagrid node only needs to know the state of the regions it resides in (λ), the state of its sibling regions (3λ), and the state of all the level λ regions (N), i.e., the highest level in the grid. As a result, the nodes only store the state of 4λ+N regions in order to have an accurate knowledge of the grid in their vicinity while remote regions' states are not precisely known. For example, Fig. 3.2 displays the internal table of the nodes inside region 302. The internal table of the nodes depends only on the cell occupied by the nodes and it must be updated when a node changes cell or after receiving a GRID_UPDATE message.

- When a GRID_UPDATE is received by the node, it first checks if it is located inside the geocasted area, otherwise it discards the packet. Then it updates its table according to the information carried by the message and broadcasts it to its neighbors.
- When moving to a new cell, a node sends an UPDATE_RQT message to the center of the new cell. The first node in the cell receiving the message replies with an UPDATE_RPL message containing its internal table. Upon receiving that reply, the node only updates the fields that it needs in its internal table. This mechanism is illustrated in Fig. 3.4.

However, it is possible that a node arrives in a totally empty cell. In that case, Hidagrid tries to send an UPDATE_RQT message several times. If this operation fails, the node widens the range of the request by sending it to the center of its higher level regions, starting with level 1. When receiving an UPDATE_RPL, the node updates its internal table and if the state of some low level regions cannot

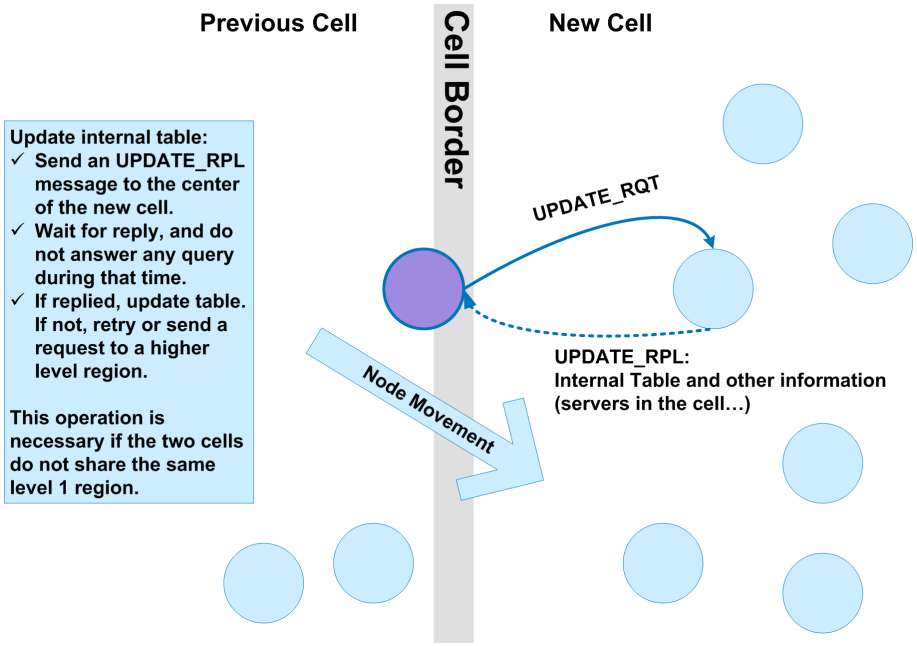


Figure 3.4: The internal table update mechanism: the dark node crossing the cell border must update its internal table and sends an UPDATE_RQT message in the new cell.

be known, Hidagrid assumes that they are inactive. Indeed, if those regions had been populated and active, the perimeter mode of the routing protocol would have found nodes in these regions and they would have replied to the UPDATE_RQT message.

3.2.5 Routing messages hierarchically

When the discovery protocol of a node sends a query or an insertion message, the packet is transferred to Hidagrid, which interacts with the routing layer to determine the destination cell of the message. Each packet contains the unique identifier (*ID*) of the data item, the type of operation (*insertion or query*), the coordinates of a destination cell (*destination*), a hierarchy level (*level*) and some

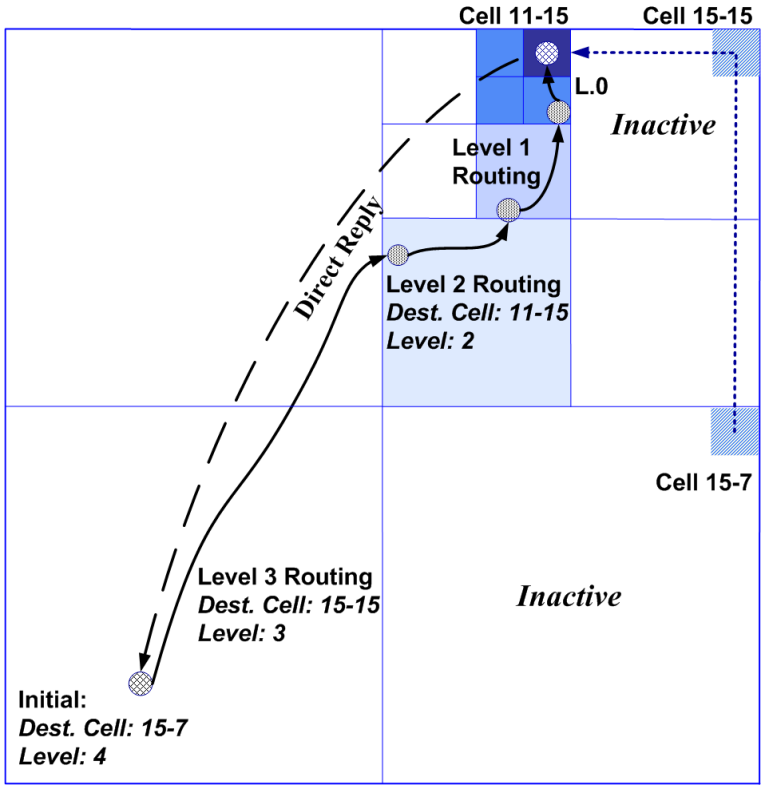


Figure 3.5: Hierarchical routing of insertions and queries in the grid: Hidagrid reevaluates the destination of the packet when it enters the region of level *level* containing the *destination* cell of the packet.

information related to the resource discovery protocol. The replies or insertion acknowledgements are not processed by Hidagrid as they target a specific node, and not a region. Hidagrid then uses the internal table to route the packet hierarchically in the grid, from level λ to level 0, as shown in Fig. 3.5.

3.2.5.1 Changing the destination cell of a message

When receiving a packet from the discovery protocol, the *destination* field is already set by the discovery protocol, and Hidagrid initializes the *level* field to $\lambda + 1$. Then the hierarchical routing algorithm depicted in the pseudo code of Fig. 3.6

is executed each time the packet arrives in the region of level *level* containing the *destination* cell stored in the packet. The node checks if the level *(level-1)* region of the *destination* cell is active. If it is, the packet is sent to the center of that region after decrementing by one the *level* field. If the region is not active, Hidagrid determines a new host region with the Relocation Policy described in the next section. In that case, the *destination* cell of the message has to be modified. This is simply realized by translating the position of the destination cell inside the original region to the new region. This mechanism is illustrated in Fig. 3.5. In this example, the Relocation Policy has to be used for the routing steps of level 3 and 2 because the destination cell lies in an inactive region. The packet initially has the cell 15-7 as the destination cell, but its region of level 3 is inactive. The Relocation Policy then decides that the region of level 3 in the right top corner should host the packet. The destination cell is then modified to be 15-15 because 15-15 and 15-7 have the same position in their respective level 3 regions. The same mechanism is used at the level 2 routing.

```
i = packet->level;
R = region of level i-1 containing packet->destination;
if (i==0) then {
  send packet to discovery protocol;}
else if (R is inactive) then {
  R = Relocation_Policy(packet);}
packet->level--;
packet->destination = update_destination(R, packet);
forward packet to R;
```

Figure 3.6: Pseudo code of the hierarchical routing at each level.

3.2.5.2 The Relocation Policy

The previous section shows the importance of the Relocation Policy as it is responsible for assigning a destination to packets that are sent to inactive regions. Therefore, it should evenly share the workload of the inactive regions between active regions.

The Relocation Policy is slightly different for the level λ routing and the other levels.When the level λ region of a packet is inactive, we simply use a hashing function H on the data item identifier that returns the identity of one of the active region. A simple solution is to choose a function returning a value between 1 and the number of active regions of level λ like: H(item) = item ID mod(number of active regions) and to attribute one value to each active region in the increasing order of their identity.

A more advanced policy is implemented for the lower levels if a region remains activated when at least two of its child regions are still active. We use a hashing function F on the identifier of the data item returning an integer between 0 and 5 to share the workload. When one region is inactive each active region hosts the items whose hashed value, or *index* corresponds to {0,1} or {2,3} or {4,5}: the region with the smallest identity takes care of the smallest indexes and the region with the highest identity takes care of the highest indexes. When another region is deactivated, the active regions host the same indexes and one of the indexes of the deactivated region, as shown in Fig. 3.7. In this example, region 3 and then region 0 turn inactive. In this scenario, region 1 and region 2 are first responsible for the indexes {2,3} and {4,5} respectively, and then for indexes {0,2,3} and {1,4,5}. The

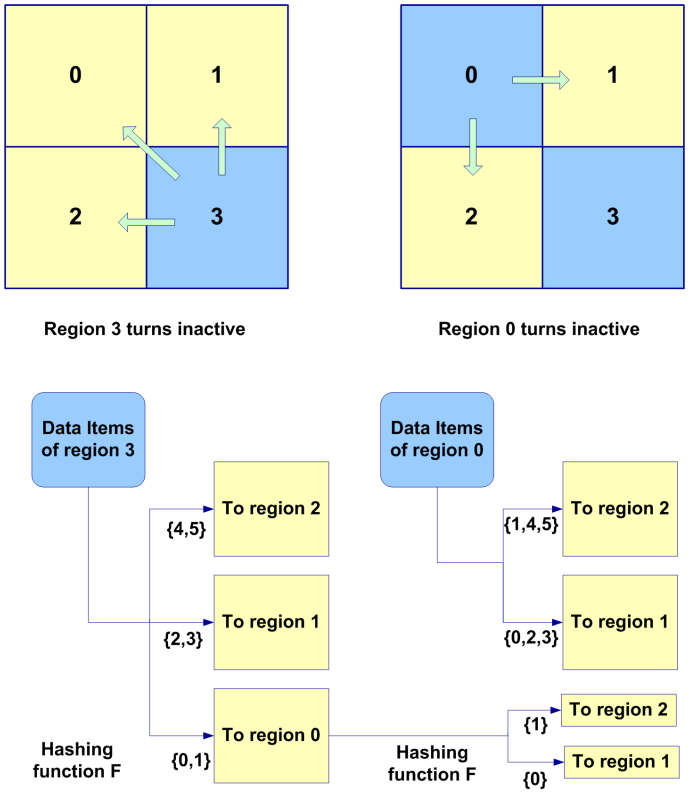


Figure 3.7: Share of workload with the Relocation Policy when region 3 and then region 0 close: the items stored in those regions are shared between the active regions.

indexes region 0 was responsible for are then implicitly shared.

3.2.6 Relocating data items

When the state of a cell changes, the hierarchical routing of some data items stored in the network may change. Therefore, when nodes receive a GRID_UPDATE message, they check the consistency of the data item distribution after updating their internal map. This test is carried out by simulating internally the hierarchical routing of the items stored in each node with the pseudo code of Fig. 3.6. If one of the routing steps returns a region the node does not reside in, the data item

must be reinserted somewhere else. With this mechanism, a cell that has just been activated only waits for data items to be inserted, while inactive regions automatically transfer all their items to active regions.

3.2.7 Detecting state changes

3.2.7.1 Deactivation detection

We have assumed that the discovery protocol is responsible for the data item management inside the cell and builds a server structure inside it. It is then easy to implement a function evaluating a lack of nodes in the cell at that level, as in SLURP and RR. Therefore, the deactivation detection ability is delegated to the discovery protocol and when this event is detected, the information is passed to Hidagrid which starts deactivating the cell.

3.2.7.2 Activation detection

The activation of a cell is managed by the Hidagrid layer only. Each time a node crosses a cell border, it first updates its internal table and if it arrives in an inactive cell, it checks the number of its 1-hop neighbors that belong to the new cell. If that number is above the *activation threshold*, Hidagrid activates the cell.

The activation and deactivation processes then consist only in launching and stopping the server structure of the discovery protocol inside the cell and then advertising in the network the state change of the cell with a GRID_UPDATE message.

3.3 Improvements on the basic scheme

As Hidagrid relocates data items to adapt to the condition of the grid, the frequency and the impact of the state changes on the hierarchy have an important influence on the performances of our protocol. Indeed, frequent and large state changes generate numerous relocations which highly increase the overhead and the risk of losing data items. It is therefore important to stabilize the state of the grid using hysteresis. We address this issue in this section. Another main drawback of the protocol is its reliance on a specific grid. We show how this assumption could be removed by generalizing the State Policy and the Relocation policy for any hierarchical or even flat grid.

3.3.1 Hysteresis mechanisms

3.3.1.1 Hysteresis at the cell level

Instability is extremely harmful to our scheme as it generates numerous data item transfers, resulting in high overhead and risk of losses. Therefore, we have to implement some hysteresis at the cell level to avoid oscillations of the cell state. The goal is then to activate a cell with a good confidence that it will not be deactivated in the near future. As density hysteresis is already implemented to activate and deactivate cells, some more types of hysteresis could be used to improve the stability of the cells:

- **Geographic hysteresis:** The cells overlap a little, creating wide borders

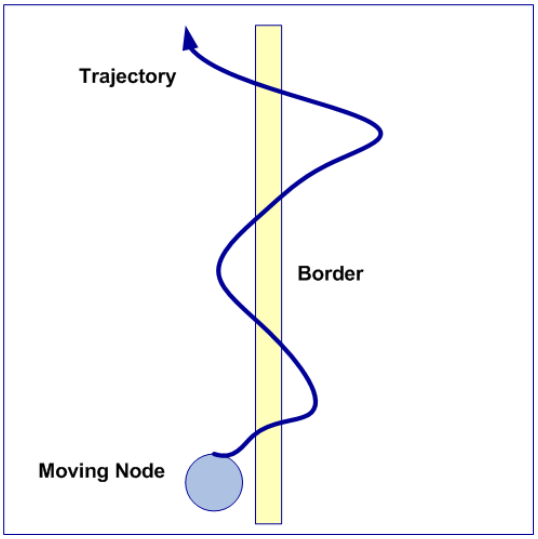


Figure 3.8: Unnecessary cell changes could be avoided with wide borders.

where nodes belong to only one cell. As long as the nodes stay in the border, they belong to their previous cell. They must reach the heart of another cell to be attached to that new cell. This technique prevents the *ping-pong* phenomenon, as shown in Figure 3.8 and decreases the number of internal table updates when nodes change cells.

- **Time hysteresis:** An arriving node measures for a period of time T the number of neighbors that are present in an inactive cell. If during T, the mean number of nodes is above a threshold, the node assumes there are enough nodes in the cell to declare it active. This mechanism could be also associated with a mobility indicator in order to make sure that the nodes present in the cell are likely to stay there for a long time.

3.3.1.2 Hysteresis at the region level

In the State Policy, the parameters A(i) and D(i) are used to control hysteresis at the region level. If $A(i) = D(i)$ the policy does not benefit from any hysteresis. In this situation state oscillations for high level regions could occur. In the example of Figure 3.9, $A(i) = 2$ and $D(i) = 2$ for all the levels. Let us consider for example the cells in region 2. If one of its inactive cells turns active, then the whole region would be declared active. But if that same cell turns back to inactive, without any other state change in the region, the whole region 2 would also be deactivated. The same situation arises in the active region 1, where any cell deactivation turns the status of the whole region to inactive. If that same cell returns to active, the state of region 1 would also change back to active. Finally, if we assume that the regions in Figure 3.9 form a parent region of level 3, then the state of that region would suffer from the same oscillations. As a result, for some configurations and without hysteresis, oscillations at the cell level could result in oscillations of high level regions. To avoid this kind of situation, the State Policy creates hysteresis by setting $A(i) > D(i)$ for some hierarchical levels.

3.3.2 Generalizing the State Policy for any grid

To adapt Hidagrid to a wider set of grids, we only need to modify the State and the Relocation Policies because the other mechanisms used in Hidagrid are totally grid independent. Therefore we slightly improve these policies for a general case. Here, we consider a region R, of level i, with c_R child regions (previously $c_R = 4$), and a_R active child regions. The State Policy can then be easily generalized by

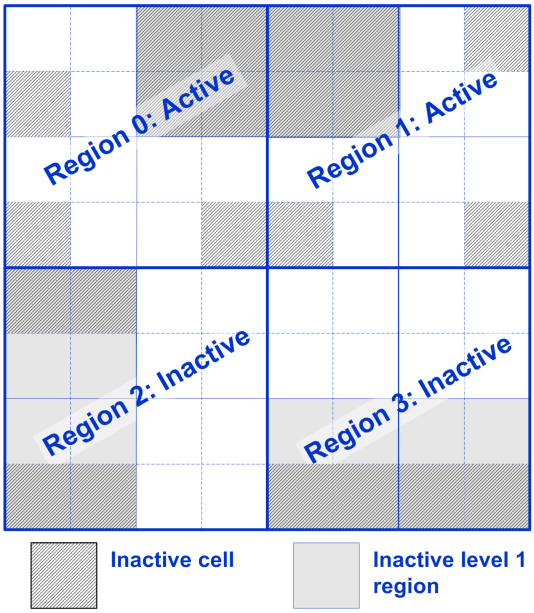


Figure 3.9: Example of unstable grid if $A(i) = 2$ and $D(i) = 2$ for all levels.

changing the definitions of the *activation and deactivation levels* $A(i)$ and $D(i)$. For the general case, $A(i)$ and $D(i)$ should be defined as *activation and deactivation ratios*. R is then activated when $\frac{a_R}{c_R} > A(i)$ and deactivated when $D(i) \geq \frac{a_R}{c_R}$, with $A(i)$ and $D(i) \in [0, 1]$

3.3.3 Generalizing the Relocation Policy for any grid

In this section we first justify the design of the Relocation Policy described previously in Section 3.2.5.2, in order to highlight the design goals and to introduce the generalized Relocation Policy.

3.3.3.1 Justification of the Relocation Policy

The goal of the Relocation Policy is to relocate data items in active regions, but for efficiency purpose it must minimize the number of transfers when the state of the grid changes. This means that data items must move from inactive regions to active regions, or the opposite, without transferring data items between active regions. In our basic grid where each region is constituted of 4 child regions, the efficiency problem is raised when a region remains active with 2 active child regions. When three child regions are active, the workload of the inactive region can be easily shared. But if another child region is deactivated, the Relocation Policy must avoid transfers between the two active sibling regions. Let us consider the scenario in Figure 3.10 where region 3 is first deactivated, followed by region 0. We assume that their parent region remains active. In this example, two different hashing functions (*f1* and *f2*) are used to relocate the items of regions 3 and 0 to the active regions. First, region 3 turns inactive and its workload is shared by hashing with *f1* the identifiers of the data items in region 3 to one of the active regions 0,1 or 2. When region 0 also turns inactive, *f2* is used to relocate *all* the data items of the inactive regions. Then, not only the data items of region 0, but also those of region 3 will be relocated, independently of their current location. As a result, some items of region 3 will be transferred between region 1 and 2. To solve this problem, we must use a hashing function F returning an integer between 0 and 5 and share the responsibility of those indexes between active regions as described earlier in this chapter.

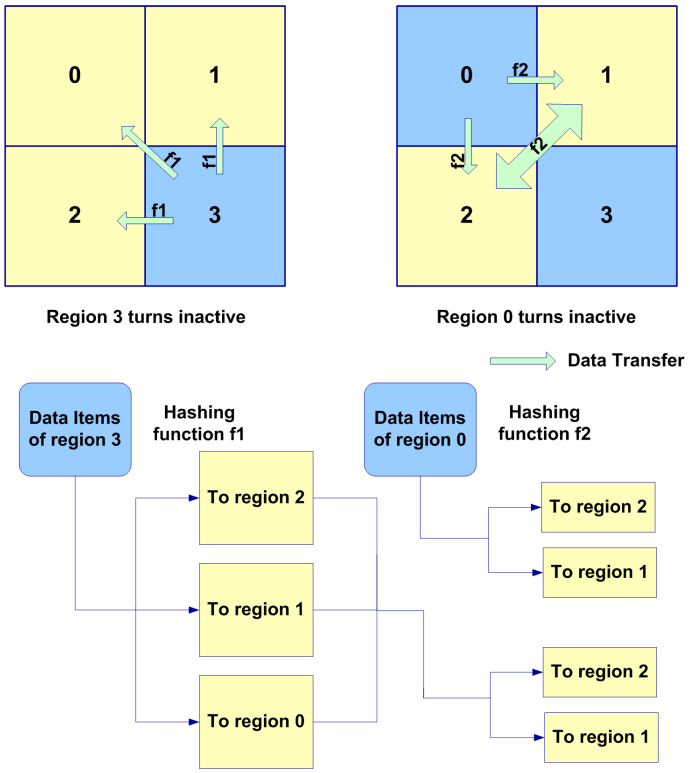


Figure 3.10: Share of data items with two hashing functions: data items are exchanged between two active regions, resulting in inefficiencies.

3.3.3.2 Generalization

We now need to generalize the basic Relocation Policy for any kind of grid. For this purpose, we use the indexes returned by a hashing function F to share the responsibility for data items equally between active regions. Statistically, each index corresponds to the same number of data items. Therefore to share evenly the workload, each active region must be responsible for the same number of indexes, when sibling regions are deactivated.

In this section, we consider a region R of level i with c_R child regions, a_R active child regions, and a deactivation ratio $D(i)$. Therefore, R remains active as long as $a_R > D(i) \times c_R$.

If the child region R_1 is deactivated, its workload must be shared among $c_R - 1$ regions. Then, if R_2 is deactivated, its own workload must be shared among $c_R - 2$ regions, as well as the workload it received from R_1. This sharing mechanism must be repeated until the parent region R is deactivated. As a result, to enable fair sharing, the number of the indexes given to a region can be divided by j for any j $\in [c_R - 1, D(i) \times c_R + 1]$. To create sets of indexes satisfying this condition, F is chosen such that the indexes are integers between 0 and $(\frac{(c_R-1)!}{(c_R-c_R\times D(i))!} - 1)$.

The indexes have then to be shared between the active child regions in a distributed way. For this purpose, the index space is shared into *slices* that are attributed to active regions. We now assume that l child regions are active, with $l > D(i) \times c_R + 1$ (if a region is deactivated, the parent region remains active). We first consider the case of a child region being deactivated. When a region turns inactive, all the slices it hosts are *cut* into $l - 1$ sub slices that will be attributed to

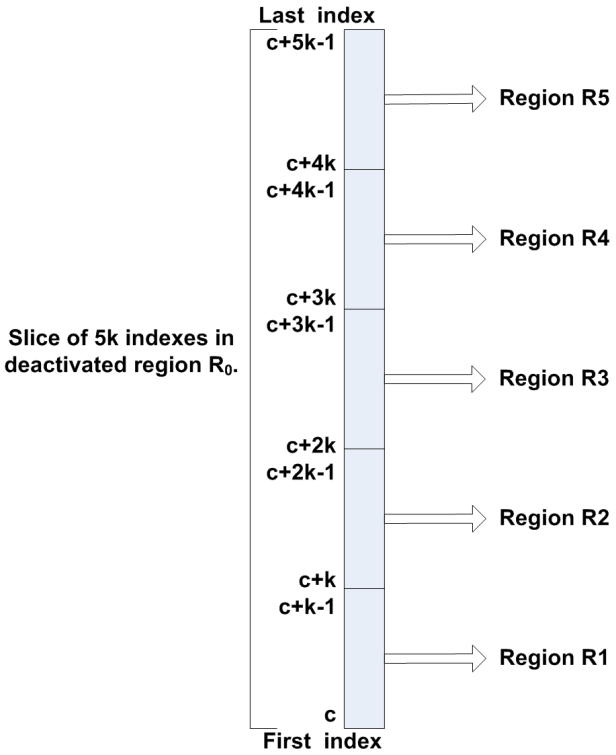


Figure 3.11: Sharing the indexes: Region R_0 is deactivated and its index slice $[c, c + 5k - 1]$ is split into 5 sub slices that are attributed to each active sibling region.

the $l - 1$ remaining active regions, as illustrated in Figure 3.11. This mechanism will always create sub slices of equal sizes because the size of any slice is in the form $\frac{(c_R - j)!}{(c_R - d_R)!}$ where $j \geq l$.

An example of that general Relocation Policy is given in Figure 3.12 with c_R=6, and $D(i) \times c_R$=2. In this example, the indexes have their value between 0 and 5x4x3-1=59, and slices of 12 items are created when Region 5 is deactivated. When Region 0 also turns inactive, its slice is cut into 4 sub slices of 3 items. Finally, when Region 2 is deactivated, the two slices it is responsible for are cut into 3 slices of 4 items or 1 item. This mechanism can be derived in a fully decentralized way based only on the state changes history of the parent region R.

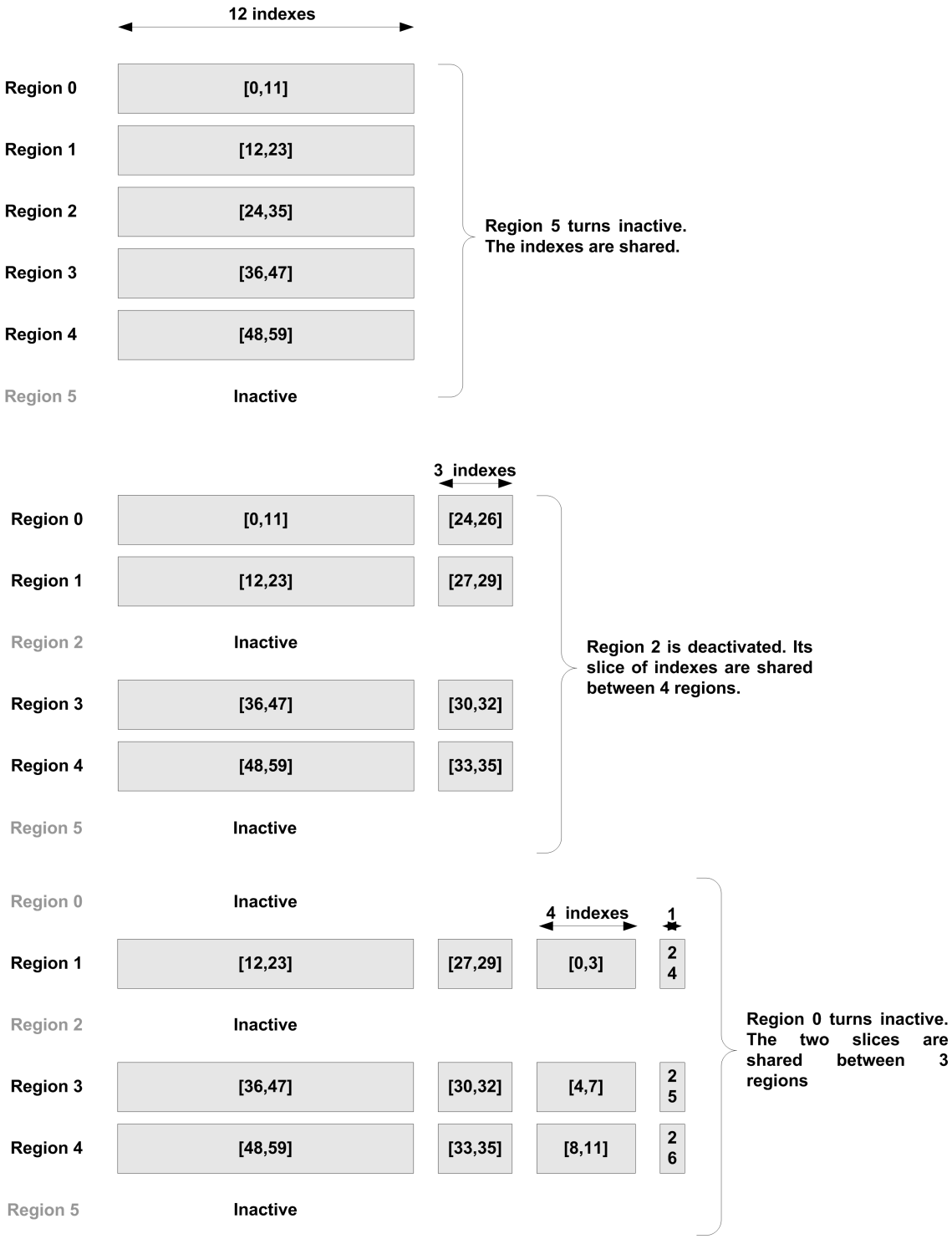


Figure 3.12: Generalized Relocation Policy: indexes are equally shared between the active regions, spreading evenly the data items among the regions

However, when a child region gets active, the consistency of the slices has to be maintained and the activated region will then be responsible for the slices of the last region that was deactivated. This mechanism implies that a short history of the state changes in R is maintained in order to rebuild the slices. As a result, when a region is activated, it receives its data items together with this history. A last issue is then raised, when region R is activated, because several of its child regions are inactive, without any history. This history is then implicitly created by assuming that the inactive regions with the smaller identity turned inactive first. The slices are then computed by all the nodes.

These mechanisms evenly spread the workload between active regions and ensure that no data item transfer takes place between two active regions. However, maintaining this structure is quite complex when $(c_R - D(i) \times c_R)$ increases, and we believe that using the simpler mechanisms described earlier should be enough to achieve good performances.

3.4 Conclusion

This chapter presented the mechanisms used by Hidagrid to address the various constraints of empty cells management. Based on a hierarchical grid, Hidagrid locally achieves the grid adaptation tasks. This local adaptation restricts the effect of cell state changes to small areas. Hidagrid then uses the State Policy to define the spread of the state changes advertisements, and relies on the Relocation Policy and a hierarchical routing mechanism to consistently route data queries and insertions. As a result, Hidagrid manages to host data items in active regions when the original

targeted region suffers from a lack of nodes. We now investigate the performances of Hidagrid to check whether it manages to fulfill all the requirements listed in Section 3.1.2.

Chapter 4

Simulation Results

4.1 Implementation of a Simple Discovery Protocol (SDP)

For our simulations, we implement a simplified and general discovery protocol so that the evaluation of Hidagrid is independent of the discovery protocol characteristics. The Simple Discovery Protocol (SDP) that we use for the simulations is based on a flat grid and exhibits the following features:

- The nodes use a unique hashing function to determine the server cell of the data items that they insert (INSERT) or query (QUERY) in the network.
- The data item management inside the cells relies on a server structure that is able to detect a lack of nodes in the cell, as in RR or SLURP.
- When changing cells, the nodes need to update some information for the data management inside the new cell (like a list of servers or data items). Therefore they send an UPDATE_RQT message to the center of the new cell

and receive an UPDATE_RPL message from one of the nodes inside the new cell. These messages are shared with Hidagrid when it is used.

- An insertion acknowledgement (INSERT_ACK) is sent when an item has been correctly inserted inside a cell.

Designing a server structure is out of the scope of this thesis so we only simulate the above characteristics. The database of the cell is actually a part of the simulating computer memory. A node accesses it with a pointer that is used to insert data items inside the cell when receiving an insertion, or to check the content of the cell database when replying a query. When changing cell, the nodes erase their current pointer when sending an UPDATE_RQT message and the new one is passed with the UPDATE_RPL message.

The number of nodes in the cells is also globally monitored. When the number of nodes inside a cell is below the *deactivation threshold*, the discovery protocol detects a lack of nodes:

- **SDP alone:** A lack of nodes characterizes a database failure, so we simply erase the cell memory in that case. This mechanism also simulates the phenomenon of temporary empty cells, as it is difficult to create temporary empty cells with the classical mobility models.
- **SPD and Hidagrid:** When SDP detects a lack of nodes, Hidagrid initiates a deactivation at the node closest to the center of the cell. It sends the MAP_UPDATE message and transfers the data items out of the cell. However, when using Hidagrid, we also need to activate the cells when they get

populated. This operation is still distributed, and initiated by a node entering an inactive cell and detecting at least *activation threshold* nodes in that cell, as described in section 3.2.7.

4.2 Simulation scenario

Hidagrid has been implemented on the ns2 simulator [65]. The geographic routing protocol GPSR [56] has been modified and linked with Hidagrid in order to share location information and to route packets hierarchically. The GPSR beacon interval is set to 1.5 seconds, the beacon expiration to 4.5 seconds and perimeter mode is activated to route packets around obstacles. We use the 802.11 MAC provided with ns2 and the two-ray ground model for signal propagation.

The simulations assess the benefits of Hidagrid in a pedestrian scenario whose parameters are summarized in Table 4.1. Therefore, the speed of the nodes is chosen between 1 and 2 m/s. We choose to set the transmission range of each node to 50m based on the measurements found in [66]. Many grid based discovery protocols recommend to choose the cells as squares whose side is two times the transmission range. Therefore, each cell is a 100x100m square. The simulations also use different mobility models as advised in [67]. In particular, it has been shown that the distribution of the nodes in the Random Waypoint (RW) model is far from uniform and that it creates periodic oscillatory movements [67, 68, 69]. As the node distribution is an important factor in Hidagrid, we also use the Random Direction (RD) and the Gauss-Markov (GM) mobility models implemented in ANSiM [70] for our simulations. The detailed parameters of the mobility models are displayed

in Table 4.1.

Finally, we simulate a network of 4x4 cells, with 250 nodes and 100 data items for 1000 seconds. As a result, each cell is populated with 15.6 nodes on average. The *activation threshold* of the nodes is heuristically set to 6 in order to activate a cell when 10-11 nodes populate the cell, while the *deactivation threshold* is set to 4. We implement Hidagrid without hysteresis and use the basic State Policy and Relocation Policy, as described in Section 3.2. For the State Policy, we decide to deactivate a region when two of its child regions are inactive. As a result, we choose the activation and deactivation levels such that $A(i) = 2$ and $D(i) = 2$ for all the hierarchical levels. The reason behind this simple choice is to create instability for the level 1 regions to test the robustness of Hidagrid. The other parameters of SDP and Hidagrid are summarized in Table 4.2. In the simulations, the insertions of data items start after 10 seconds: a node is randomly chosen to insert a random data item. The mean delay between two insertions is 1.5 seconds. After 100 seconds, the queries start by randomly choosing a node and an inserted item, with a mean delay of 0.5 seconds between two queries. We evaluate the benefits of Hidagrid by comparing the performance of SDP alone and the performance of SDP with Hidagrid.

4.3 Impact of Mobility and deactivation threshold

We first study the effect of the mobility model (GM, RD, RW) and the *deactivation threshold* (4, 5, 6, 7) on SDP and Hidagrid, and on SDP alone. We do not display

Table 4.1: Simulation Parameters

Number of nodes	250
Number of cells	4x4
Transmission range	50 m
Dimension of cells	100x100 m
Number of data items	100
Simulation time	1000 sec.
Mobility Models	GM, RD, RW
GM, RD, RW Velocity	[1.0 ; 2.0] m/s
GM Angle Variation	5.0
RD Pause Time	[20.0 ; 50.0] sec.
RD Move Time	[30.0 ; 120.0] sec.
RW Pause Time	[20.0 ; 50.0] sec.

Table 4.2: Protocols Parameters

SDP	
Insertion time out	3 sec.
Number of insertion attempts	10
Deactivation Threshold (for cells)	4 nodes
Hidagrid	
UPDATE_RQT time out	2 sec.
Number of UPDATE_RQT attempts	5
Activation level, A(i)(for regions)	2
Deactivation level, D(i) (for regions)	2
Activation Threshold (for cells)	6 nodes

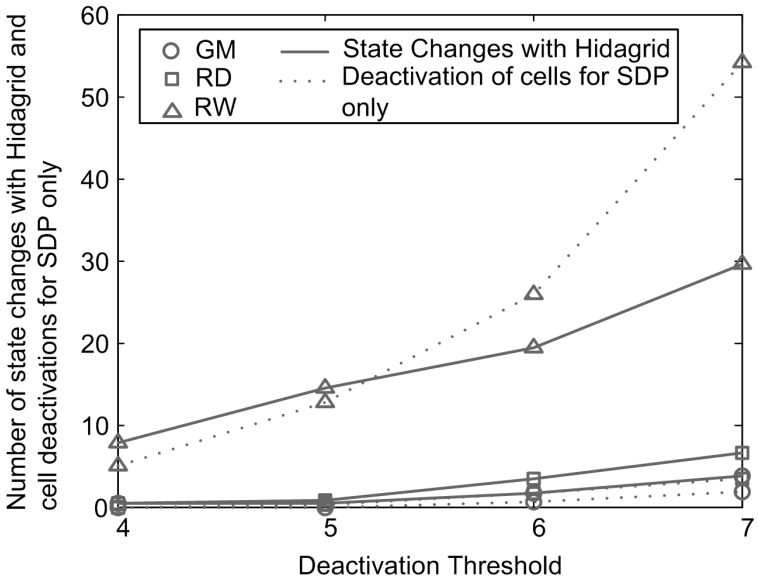


Figure 4.1: Number of state changes vs *deactivation threshold* in a 4x4 grid.

performance results for other deactivation thresholds because values smaller than 4 result in a stable environment while values greater than 8 create extremely unstable scenarios in which cells are deactivated and reactivated almost simultaneously. Hidagrid fails in the latter situation.

As shown in Figure 4.1, the two parameters affect greatly the stability of the grid and test the robustness of Hidagrid in a dynamic environment. The figure displays in plain lines the number of state changes, activation and deactivation are counted, during the 1000 simulated seconds. While the grid remains stable with RD and GM with a maximum of 5 state changes, RW generates a growing instability with up to 29 state changes on average with a deactivation threshold of 7. A state change takes place on average every 35 seconds, but the simulations oscillate between long stable periods and short grid adaptations, because of the oscillations occurring with RW. This means that Hidagrid has to adapt to high frequency updates. The hit ratio displayed in Figure 4.2 proves that Hidagrid

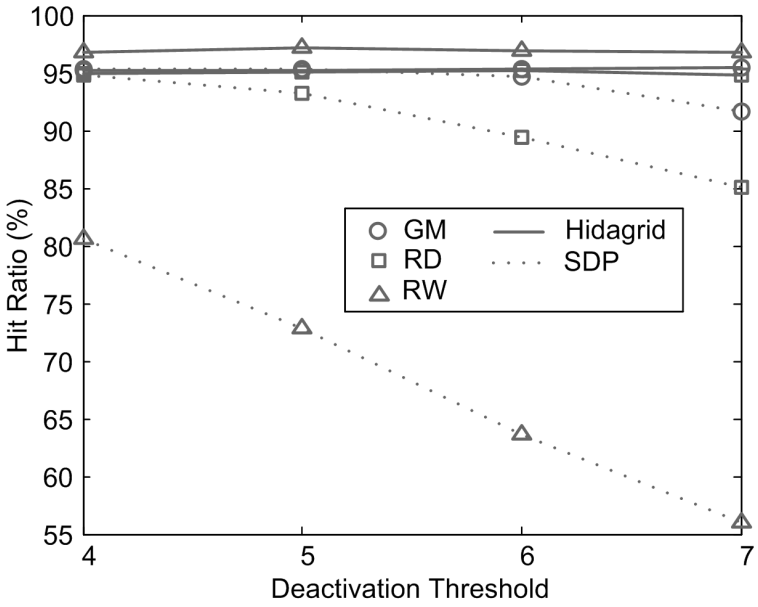


Figure 4.2: Hit ratio vs *deactivation threshold* in a 4x4 grid.

(plain lines) copes with this unstable environment as the percentage of successful queries remains high and stable with the mobility model: around 97% of successful queries for RW and 95% for RD and GM.

The dotted lines in Figure 4.1 show the number of cell deactivations when SDP alone is used. Only the RW mobility model generates instability with a number of deactivations between 7 and 55 when the deactivation threshold increases. As we explained in section 4.1, when Hidagrid is not used, a deactivated cell has its database fully erased. However, the deactivations mainly hit the cells in the corners, and the center cells are almost never deactivated. As a result, the hit ratio of SDP, in dotted lines in Figure 4.2, drops when the grid becomes unstable and reaches only 55% with RW and a deactivation threshold of 7, although Hidagrid manages a 97% hit ratio in the same conditions by relocating the items.

Figures 4.3 and 4.4 respectively show the proportion of negative replies and

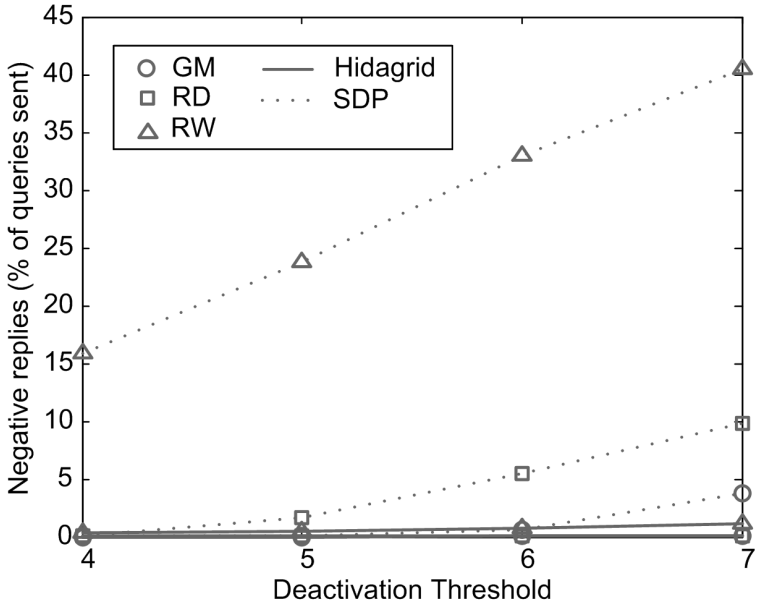


Figure 4.3: Negative replies vs *deactivation threshold* in a 4x4 grid.

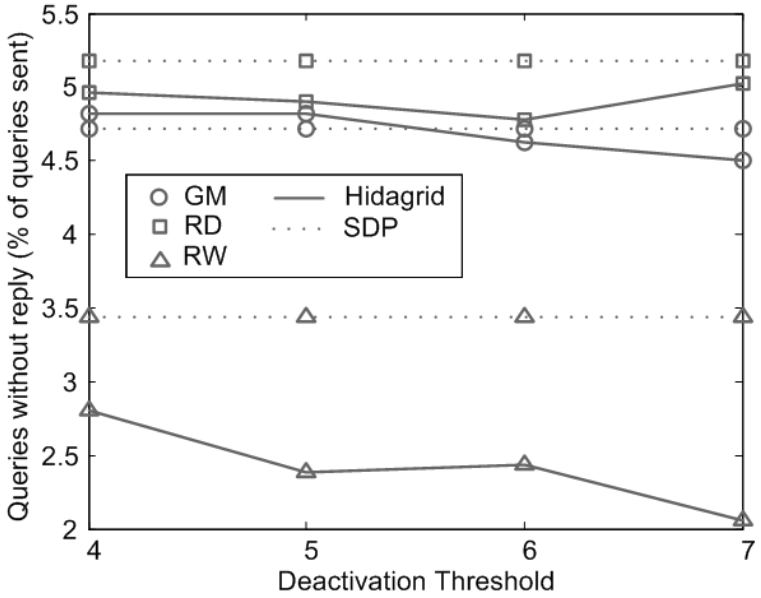


Figure 4.4: Queries which were not replied vs *deactivation threshold* in a 4x4 grid.

queries that were not replied, for Hidagrid (plain lines) and SDP alone (dotted lines). The query failures of Hidagrid are almost all due to loss of messages occurring with routing failures.These routing failures could have two sources. First, the short transmission range of the nodes may generate a small lack of connectivity, as calculated in [71]. Second, in perimeter mode, GPSR sometimes fails, as explained in [51]. But due to the higher concentration of nodes in the center of the network, RW loses fewer messages than the other mobility models, resulting in a better hit ratio. However, a slight increase of negative replies is experienced in unstable scenarios with RW, but this phenomenon is limited to 1% of the queries sent and Hidagrid still manages a 97% hit ratio with RW. Indeed, these negative replies result from the data item relocations when the state of a cell changes. The items have to be transferred according to the Relocation Policy. Thus, between the initiation of a grid update process and the end of the relocations, a small inconsistency period takes place, resulting in query failures. The two figures also show in dotted lines the effect of instability on SDP alone. The number of negative replies sharply increases with instability because more data items get erased when cells are deactivated.

The good performance of Hidagrid however incur some communication overheads. The stacked bars in Figure 4.5 display the average number of messages sent by each node. Each layer represents a type of message and the thickness of the layers indicates the number of messages sent. The overhead of RW jumps from 120 to 160 messages when the *deactivation threshold* varies from 4 to 7. The extra insertions (INSERT and CONFIRM_INSERT) generated by the state changes make up the majority of the overhead increase while the geocasted MAP_UPDATE messages represent only 25% of the extra overhead. These MAP_UPDATE mes-

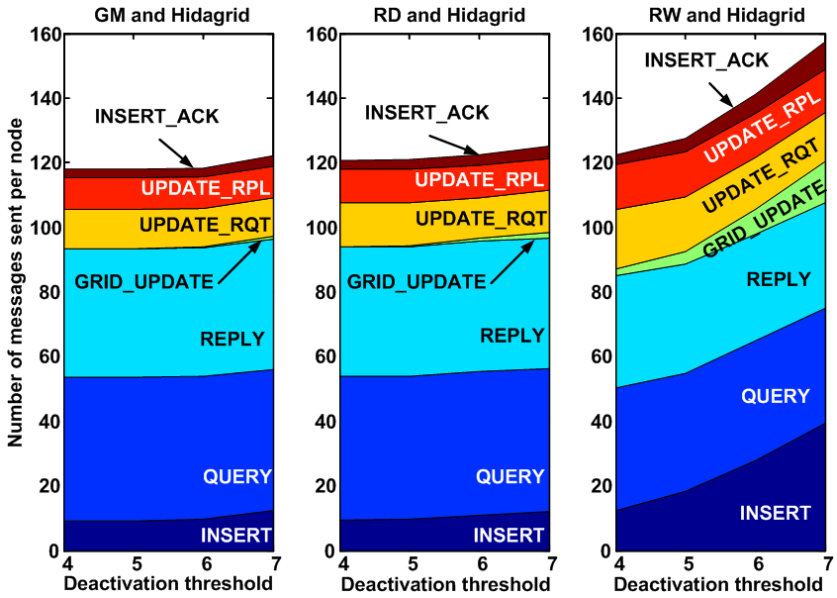


Figure 4.5: Number of messages sent by each node with Hidagrid for each mobility model (GM, RD, RW) and varying *deactivation threshold*.

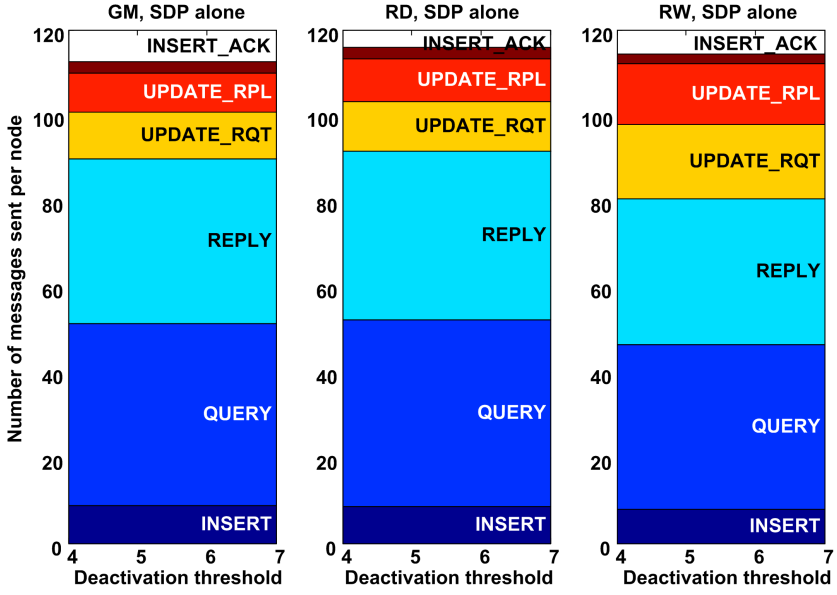


Figure 4.6: Number of messages sent by each node with SDP alone for each mobility model (GM, RD, RW) and varying *deactivation threshold*.

sages only appear in unstable environment with RW, none of them is generated in the stable scenarios with RD and GM. The UPDATE_RQT and UPDATE_RPL messages also have an important impact on the overhead of Hidagrid and SDP as they make up for around 20% of the total overhead. This source of overhead is quite important and is a major weakness of Hidagrid. Each time a node crosses a border it must generate 2 messages, which makes Hidagrid unsuitable for high mobility environments. However, no special mechanism has yet been implemented to limit the number of UPDATE_RQT and UPDATE_RPL messages transmitted, like piggybacking with the GPSR beacon or on another data packet. This solution may prove efficient. Finally, Figure 4.5 also confirms the higher concentration of nodes in the center of the map with RW as the number of queries and replies forwarded by RW are lower than that of RD and GM.

The detailed communication of SDP alone is displayed in Figure 4.6. For each mobility model, the number of messages sent remains stable with the *deactivation threshold*. Indeed with SDP alone, the *deactivation threshold* only erases the memory of the cells, without initiating any other mechanism, so the *deactivation threshold* only has an impact on the hit ratio and the number of negative replies. The figure shows that SDP alone generates less overhead than with Hidagrid. Less than 115 messages are sent by each node for SDP alone, while the minimum is 118 with Hidagrid. In stable situation (GM, RD), the overhead difference between Hidagrid and SDP alone represents on average 4.5% because of the hierarchical routing of the insertions and the queries. But as the grid becomes unstable, even if the total overhead of Hidagrid increases due to the relocations, it is interesting to notice that the number of QUERY and REPLY messages sent by Hidagrid decreases under that of SDP alone. This phenomenon is observed in the RW simu-

lations with a *deactivation threshold* of 7: nodes forward 35 QUERY messages with Hidagrid and 39 with SDP only. Indeed, Hidagrid concentrates the data items in the populated areas, shortening the average path between a querying node and the location of the item. Therefore, in a stable clustered network, the overhead of Hidagrid should be smaller than that of the simple protocol. We investigate this hypothesis in the next section.

4.4 Static empty cells

For this set of experiments, we create cells that are empty of nodes during the whole simulation due to the mobility patterns of the nodes or geographic obstacles. The mobile nodes have then to go around these empty areas during their movement. These experiments measure the effect of Hidagrid in an environment with *static* and *dynamic* inactive cells. In the previous section, inactive cells only appear when the number of nodes is below the deactivation threshold. However the cells are never empty and queries sent to inactive cells with SDP alone always receive replies. Here, if Hidagrid is not used, SDP sends messages to fully empty cells. In this situation, the insertions and queries should wander around the empty server cell without being able to reach their destination, generating useless overhead. This phenomenon is also one of the motivation behind Hidagrid, i.e., the protocol is not only supposed to improve the hit ratio, but it should also generate less overhead than a resource discovery protocol alone by directing messages to populated areas.

Based on the previous results, we only simulate RW and RD because RD and GM have similar impacts on the simulations. We vary the number of empty

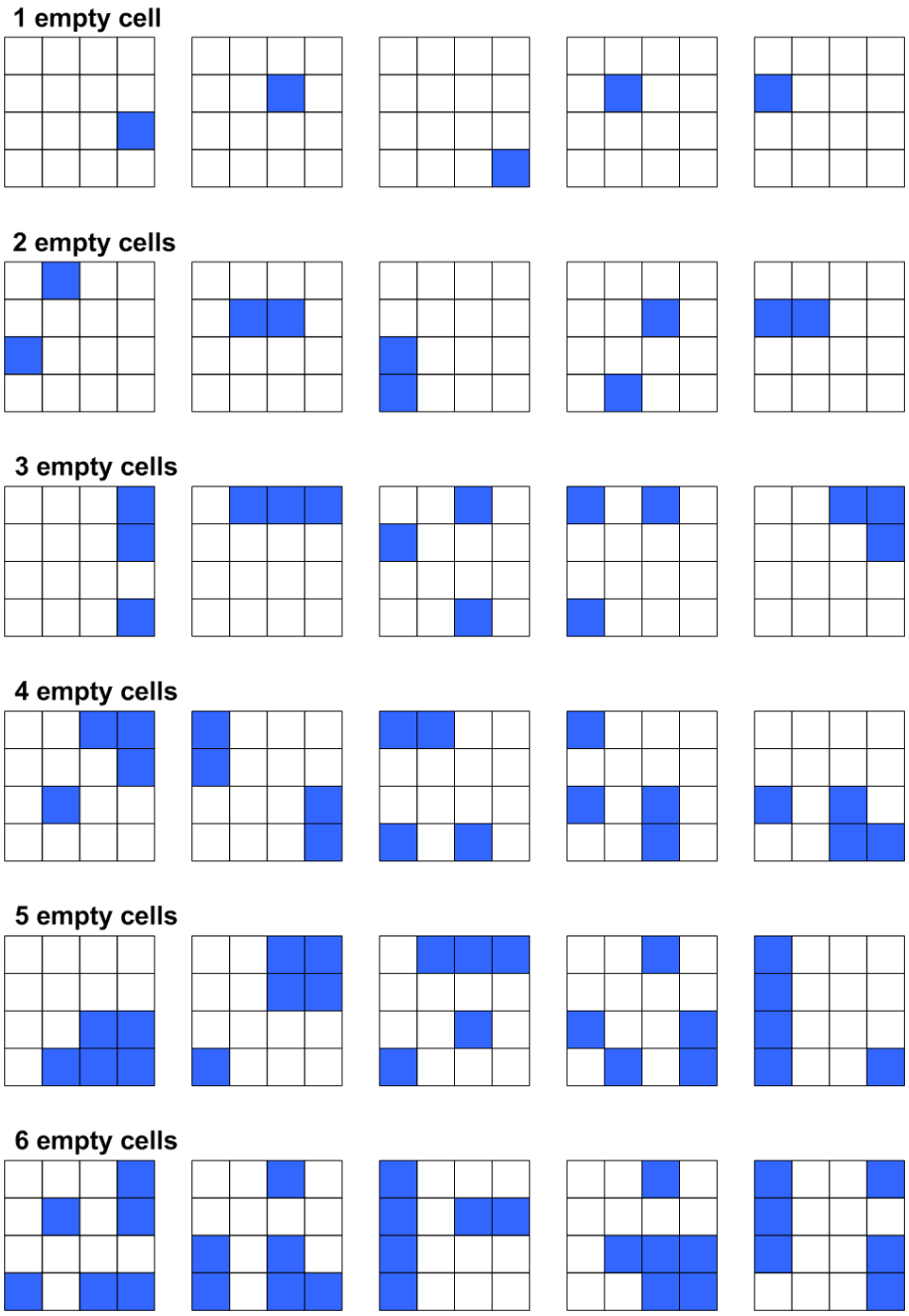


Figure 4.7: The maps used for the empty cells scenarios: each dark cell remains empty during the whole simulation.

cells from 1 to 6 in a 4x4 grid, create 5 different maps for each case and carry out 4 simulations on each map for each mobility model, with the parameters of Table 4.1. The maps used for this set of simulations are displayed in Figure 4.7. ANSiM is used to create the maps and generate the movement patterns of the nodes. The deactivation threshold of the cells is set to 4 in order to simulate a stable environment with RD and a relatively unstable one with RW. We do not set the threshold higher in order to limit the number of inactive cells in the network, because all the empty cells are already inactive. We also limit the TTL of the messages to 32 hops instead of the standard 124 TTL, in order to limit the congestion generated by the wandering packets around empty areas. Finally, for fair comparison, the density of nodes should remain roughly stable, but we also want to increase it slightly to improve the routing performances around the empty cells. Therefore, we heuristically set the number of nodes to $250 - 10 \times$ *number of empty cells* so that the density slightly grows from 16 to 19 nodes per cell.

Figure 4.8 compares the resource discovery performances of SDP with and without Hidagrid, for the two mobility models. Each bar displays the *hit ratio*, the percentage of *negative replies*, and the percentage of *unreplied queries* for each number of empty cells in the maps.

4.4.1 Resource discovery performance of Hidagrid

As we noticed earlier, the results vary a lot between RD and RW, although the simulations ran on the same maps. First, the number of unreplied queries is usually much higher for RD because of routing difficulties around the empty cells, which

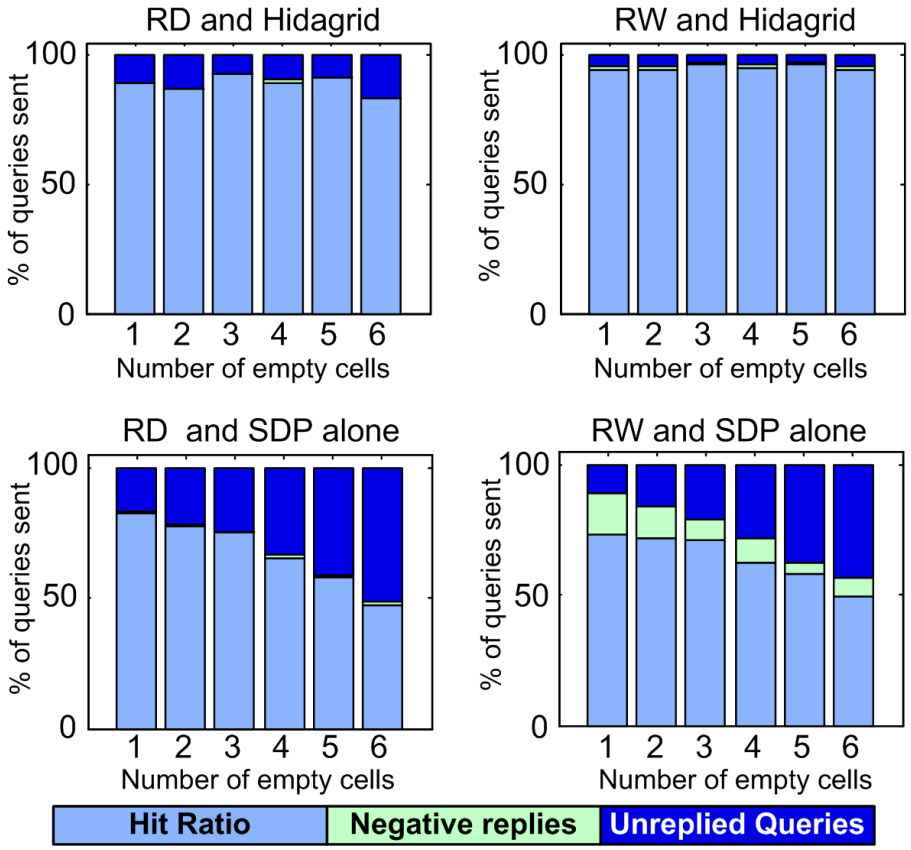


Figure 4.8: Performances of the resource discovery with and without Hidagrid in an environment with static empty cells.

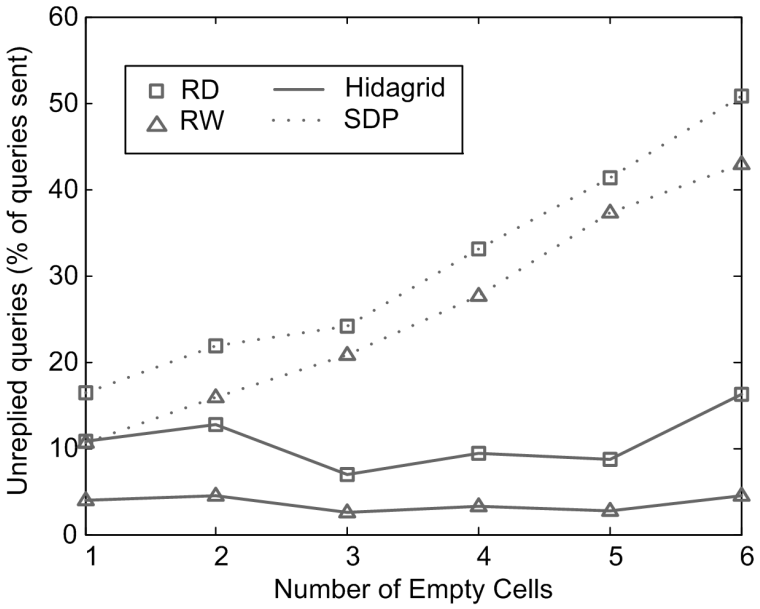


Figure 4.9: Percentage of queries which were not replied with static empty cells.

behave as obstacles for the routing protocol. While RD evenly spreads nodes everywhere, RW usually concentrates the nodes around the center of the map making it less sensitive to empty cells. However RD does not exhibit any negative replies due to the stable repartition of the nodes. The instability generated by RW triggers the relocation of the data items from time to time, which results in temporary unavailability of some items. This phenomenon is amplified by the inactive empty cells that facilitate state changes of level 1 regions. The State Policy closes a level 1 region as soon as two cells are inactive, generating many relocations.

4.4.2 Comparison of SDP alone and with Hidagrid

Even if the performance of Hidagrid degrades in some maps because of routing failures, the hit ratio of SDP with Hidagrid always remains much higher than that of SDP alone. These benefits are more pronounced for the RW scenarios where

Hidagrid manages to keep the hit ratio around 95% for any number of empty cells, while the hit ratio of SDP alone degrades from 70% to 45% when the number of empty cells grows to 6. Two factors contribute to decreasing the hit ratio. First, it can be observed in Figure 4.9 that the number of unreplied queries is much higher for SDP alone because of the empty cells. The queries cannot enter the server cells and get lost in the network. The percentage of unreplied queries increases on average linearly with the number of empty cells.

Second, due to mobility, cells are also dynamically deactivated, like in the previous simulations. When SDP alone is used, data items are then lost resulting in negative replies when queried, as shown in Fig. 4.10. Thus SDP alone also suffers from negative replies, especially for RW. In our simulations, this phenomenon can still be severe and reach up to 16% of negative replies. However, it can be noticed that for this mobility model the number of negative replies decreases with the number of empty cells because of a better stability of the grid: the density of nodes slightly increases and a growing number of empty cells makes the other cells less likely to be deactivated.

4.4.3 Overhead comparison

Figures 4.11 and 4.12 compare the average overhead of SDP with and without Hidagrid for a varying number of empty cells. As expected, the overhead of Hidagrid is always much lower than that of SDP alone for all the scenarios, and that difference greatly increases with the number of empty cells because of the growing proportion of queries and insertions that wander around the empty cells. For example, with RD and SDP alone, the number of INSERT messages sent per node grows from 15

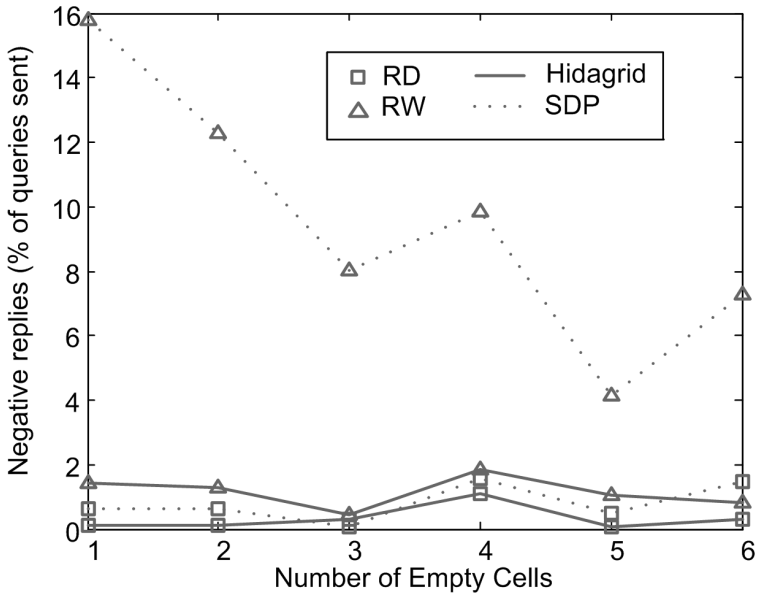


Figure 4.10: Percentage of negative replies with static empty cells.

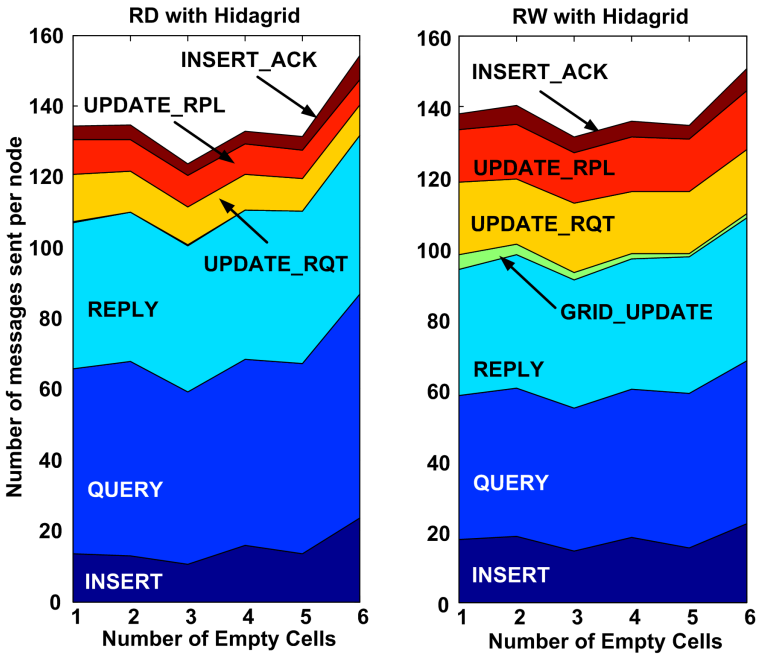


Figure 4.11: Number of messages sent by each node with Hidagrid for RD and RW in a static empty cells environment.

to 60, while the number of QUERY messages sent per node increases from 60 to 130. However, the overhead with Hidagrid increases also a little because of the longer

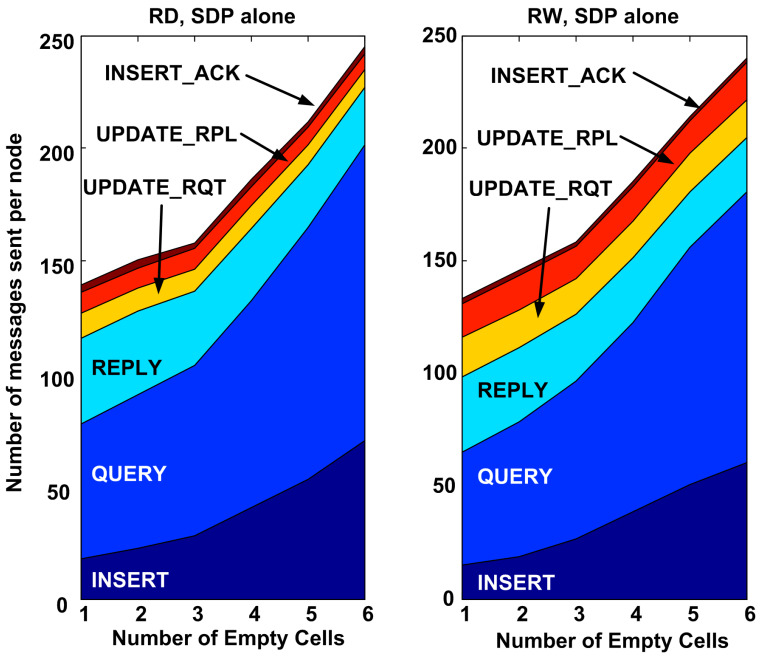


Figure 4.12: Number of messages sent by each node with SDP alone for RD and RW in a static empty cells environment.

paths caused by the obstacles. The number of messages sent per node varies from 124 to 168 with Hidagrid while SDP alone generates an overhead varying between 136 and 240 messages per node. With this set of simulations, Hidagrid proves its efficiency in both reducing the traffic of the resource discovery and improving its hit ratio, as soon as the network is slightly heterogeneously populated.

4.5 Sensitivity to mobility

Mobility is the main weakness of Hidagrid, and of the grid based protocols in general because communications is triggered when nodes cross cell borders. In the case of Hidagrid, an UPDATE_RQT and an UPDATE_RPL message will at least be sent by the nodes changing cells. We evaluate the behavior of Hidagrid in high

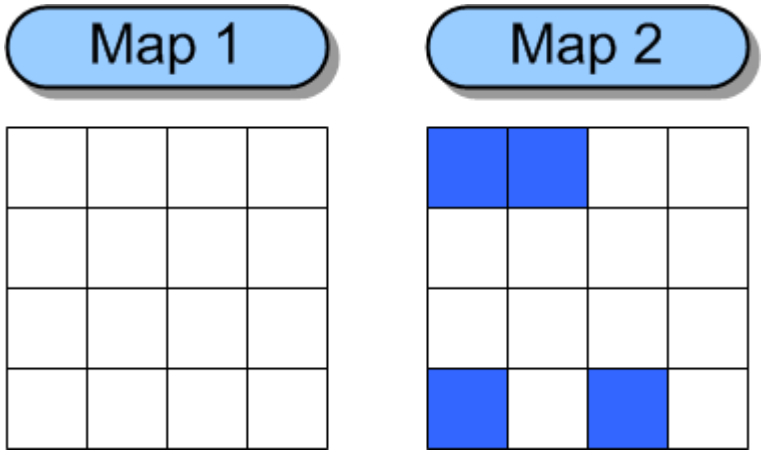


Figure 4.13: Maps used for the simulations with increasing query frequency. The dark cells are empty during the simulations.

mobility scenarios and study the effect of mobility on these messages. We choose to use RD for evaluation as the mobility parameters allow more freedom. In RW only the *maximum speed* and the *pause time range* can be chosen, while a mobile scenario with RD is described by a *speed range*, a *pause time range* and a *movement time range*. Therefore, we design three types of simulations by varying one of the parameters (speed, pause time, movement time) while the other two parameters keep the values in Table 4.1. First, the speed of the nodes increases from 1m/s to 10 m/s (the minimum speed and the maximum speed are the same). Then, with a speed range of [1.0 ; 2.0] m/s and a movement time range of [30.0 ; 120.0] seconds, the pause time varies from 0 to 100 seconds. Finally, with the same speed range and a pause time range of [20.0 ; 50.0] seconds, the movement time grows from 1 second to 120 seconds. We simulate these experiments on the two maps of Fig. 4.13.

Figures 4.14 and 4.15 present the hit ratio and the percentage of unreplied queries with Hidagrid for the three scenarios. As earlier, the main reason for failure

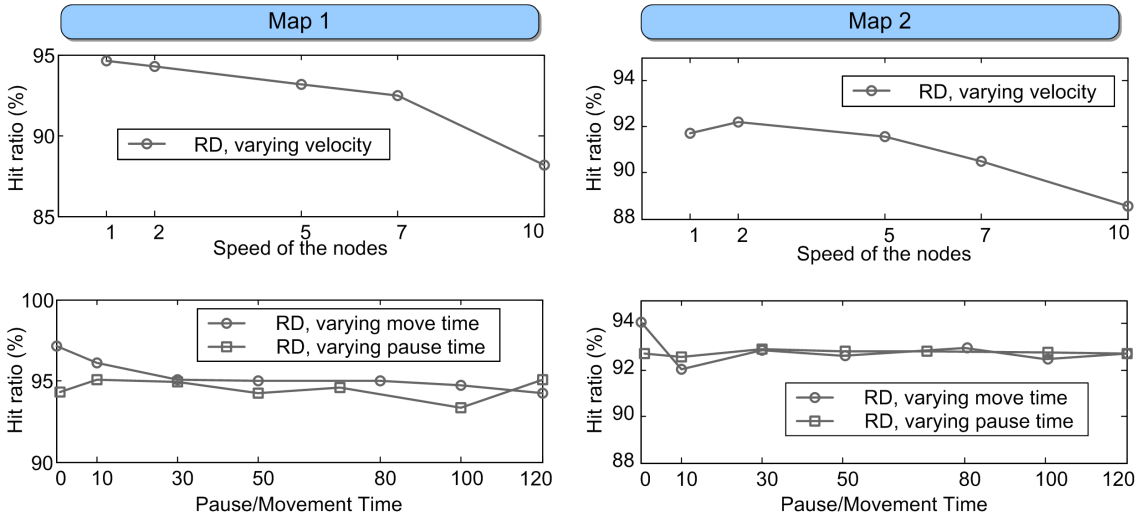


Figure 4.14: Hit ratio of Hidagrid with different settings of Random Direction.

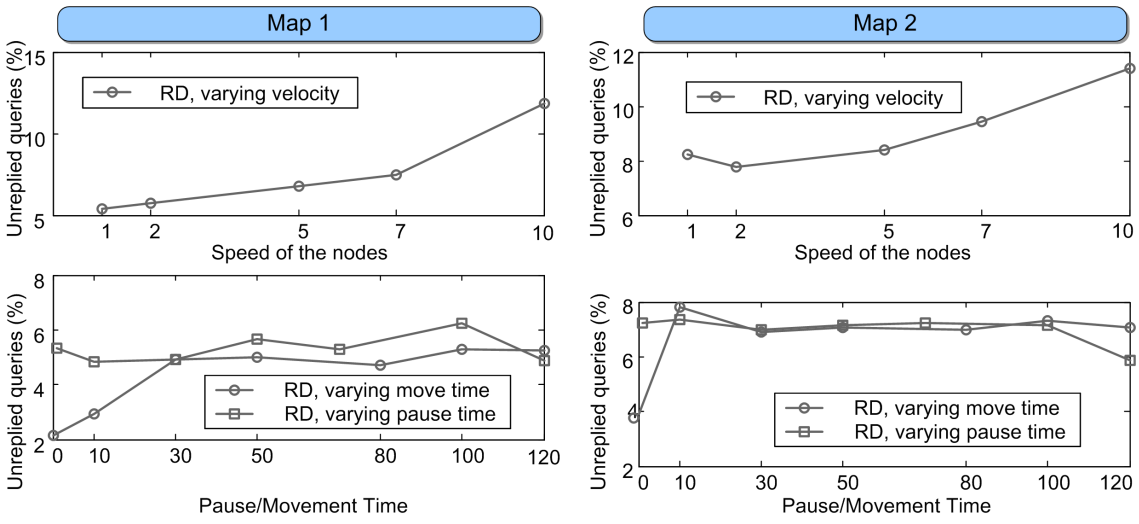


Figure 4.15: Percentage of queries which were not replied with Hidagrid for different settings of Random Direction.

is the loss of queries and replies. The grid remains stable even with high mobility, and the items are not widely transferred in the grid, therefore no negative reply is received (each simulation usually generates one or two state changes only). With high mobility, the routing layer does not adapt quickly enough to route safely all the messages, because of the short transmission range (50 meters). As a result, the hit ratio drops from 95% to 88% on Map 1 and from 92% to 88% on Map 2. Adapting the beacon frequency of GPSR to mobility could improve the situation but this issue is not in the scope of this thesis. If the speed of the nodes remains in the range 1m/s-2m/s, the movement time and the pause time have only little influence on the routing and the network remains stable enough for GPSR to efficiently adapt.

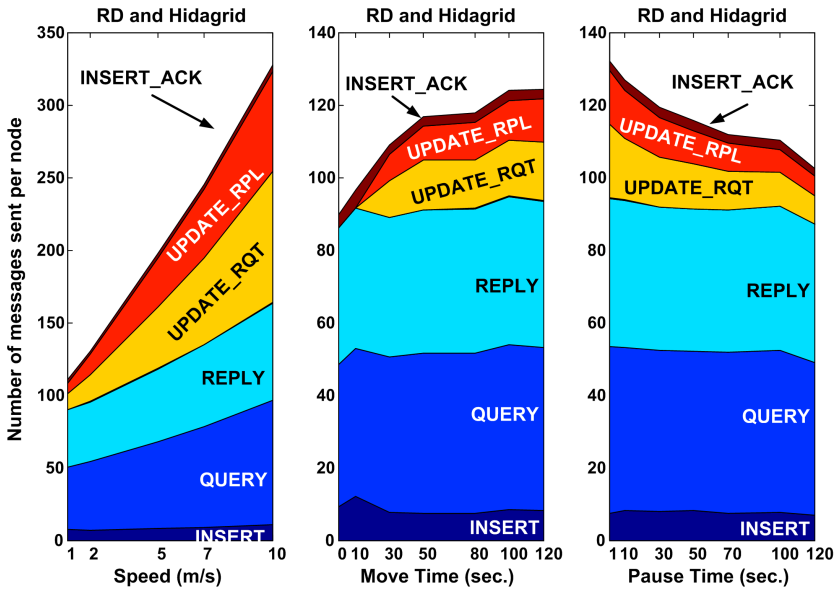


Figure 4.16: Number of messages sent by each node with Hidagrid for different settings of the Random Direction mobility model on Map 1. The scale for the varying speed is different from the two other graphs.

Figures 4.16 and 4.17 display the overhead of Hidagrid for the three cases on each map. As expected, the overhead increases a lot with the mobility as

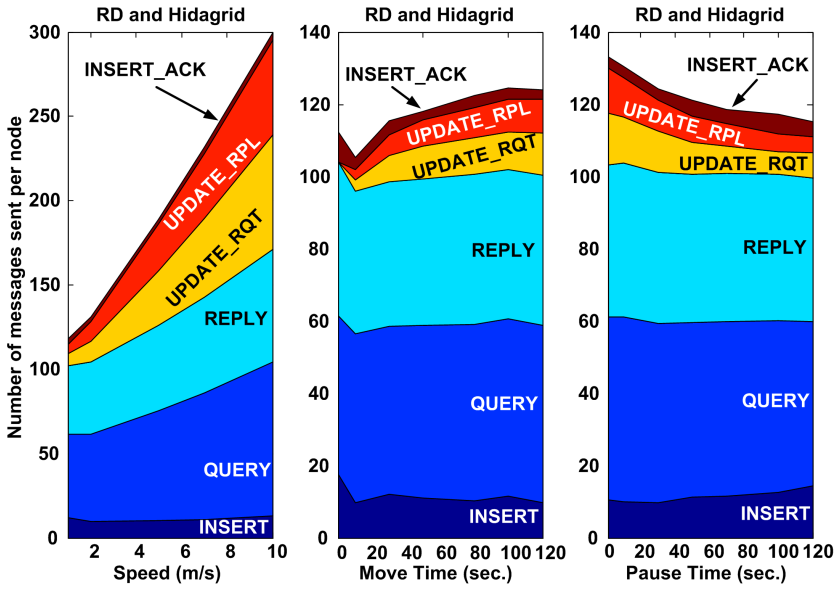


Figure 4.17: Number of messages sent by each node with Hidagrid for different settings of the Random Direction mobility model on Map 2. The scale for the varying speed is different from the two other graphs.

nodes cross more often the cell borders. For example, on Map 2, the number of UPDATE_REQUEST and UPDATE_REPLY messages jumps from 6 to 68 and from 5 to 56, respectively, when the speed of the nodes reaches 10 m/s. The routing problems are also illustrated in the number of queries and replies forwarded which significantly increases between the 1m/s and the 10 m/s simulations. On Map 2, the queries jump from 50 to 90 messages per node, and from 40 to 70 for the replies. The same trends appear with varying pause and movement times, but they are more limited. For example, on Map 2, when the movement time grows from 10 to 120 seconds, the total number of messages increases from 105 messages to 124, and when pause time grows, the number of messages decreases from 132 to 115 messages, because of the UPDATE_RQT and UPDATE_RPL messages only. As a result, the total overhead is almost multiplied by three between the 1 m/s and the 10 m/s scenarios, but it only varies of 30%-40% per node when the pause or the

movement time vary, with a speed limited to 2 m/s.

These results show Hidagrid's sensitivity to the nodes' speed. Hidagrid itself is sensitive to high mobility due to the frequent border crossings, but the geographic routing also experiences difficulties of routing in such environment. And even if the simulations do not take into account the data management inside the cells, this part of the resource discovery protocol should also face problems in very mobile environment. For these reasons, we think Hidagrid should be used in pedestrian scenarios only.

4.6 Varying the query rate

In this section, we evaluate the robustness of our protocol when the query rate increases. The goal is to compare the performances of the discovery protocol under heavy load and measure the impact of a growing number of wandering messages on the delay of the queries when Hidagrid is not used.

In the previous sections, the time between two queries was uniformly distributed between 0 and 1 second. As a result, a query was sent on the network every 0.5 second on average. In this section, the inter query time is uniformly distributed between 0 and $time_{max}$, with $time_{max} \in \{0.2, 0.4, 0.6, 1, 2\}$. The number of queries sent during the 1000 simulated seconds is displayed in Fig. 4.18 and varies between 900 and 9000. In order to generate wandering packets, we use the Map 1 from Fig. 4.13. The other simulation settings are listed in Table 4.1 and Table 4.2.

Even under heavy load, the discovery performances remain stable with and

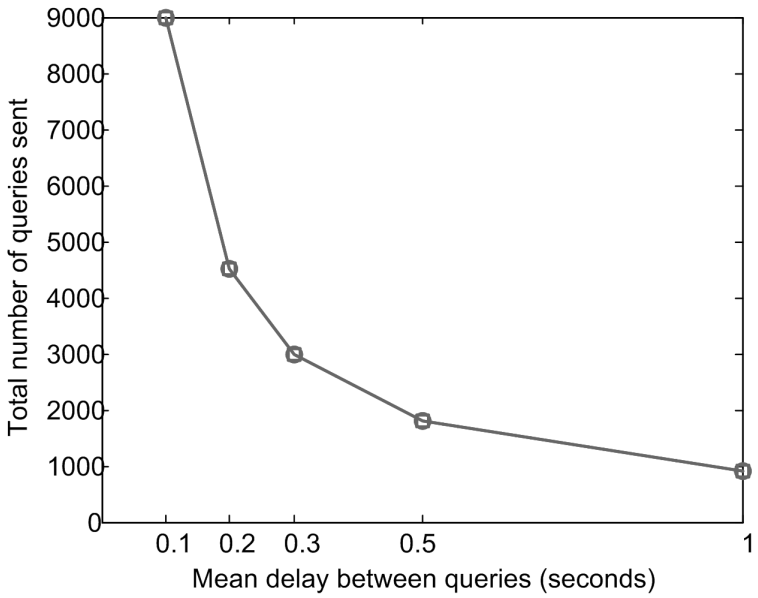


Figure 4.18: Number of queries sent when the query frequency increases.

without Hidagrid, as shown in Fig. 4.19. But the total overhead displayed in Fig. 4.20 increases a lot for SDP alone as well as for Hidagrid, due to the growing number of queries sent in the network. However, we can notice that the communication overhead difference between SDP alone and Hidagrid increases with the query frequency. With an average inter query time of 1 second, SDP alone generates around 100 messages per node more than Hidagrid. But with a 0.1 second inter query delay, the difference between Hidagrid and SDP alone reaches almost 300 messages. These results show that the wandering messages generate high overhead in the network.

Because of this overhead difference, we measure the delay between the departure of a query and the arrival of the corresponding reply. Both positive and negative replies are counted but the lost queries are not included in our results. Fig. 4.21 shows that the wandering messages created by SDP alone generates congestion in the network because the delay increases with the number of queries.

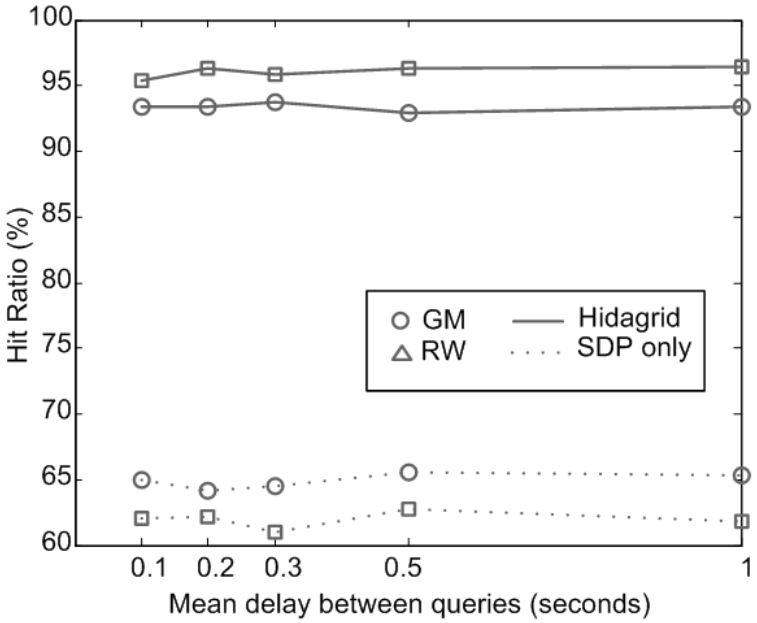


Figure 4.19: Hit ratio of SDP with and without Hidagrid vs inter query time.

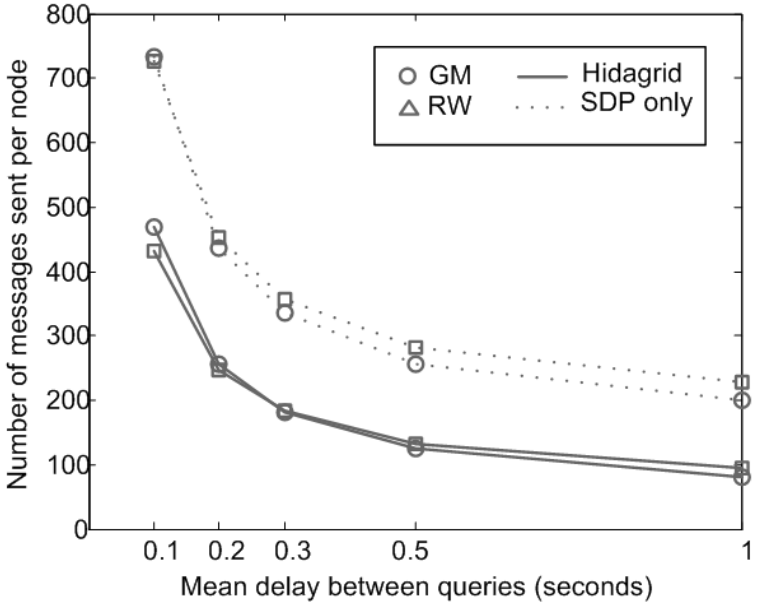


Figure 4.20: Overhead of SDP with and without Hidagrid vs inter query time.

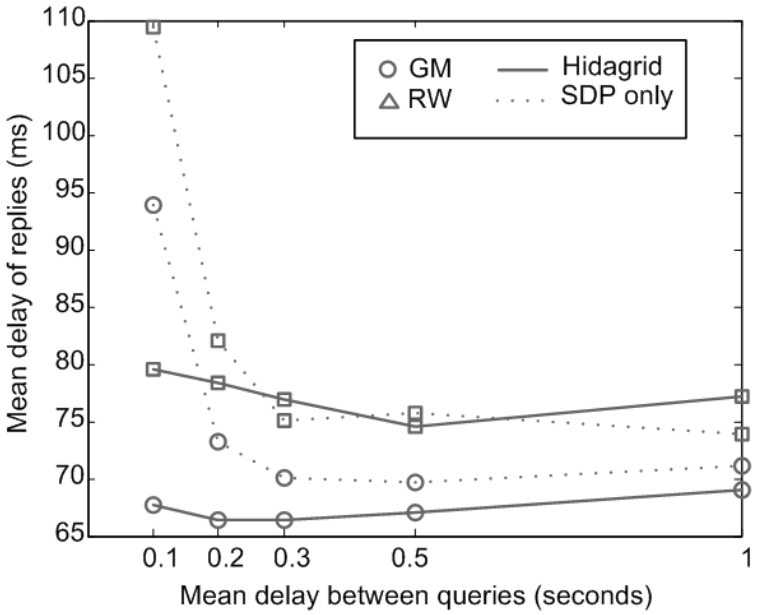


Figure 4.21: Delay between the departure of a query and the arrival of the corresponding reply vs inter query time.

The wandering messages saturate the areas around the empty cells, resulting in long delays when queries are targeted to active cells surrounding empty cells. As a result, Hidagrid also proves robust to heavy load by limiting the overhead and avoiding network congestion.

Chapter 5

Conclusion and Discussion

In this thesis, we investigated the management of empty cells for grid based discovery protocols in ad hoc networks. All the proposed protocols in the literature assumed that the network was homogeneously populated and fail if empty cells appeared in the grid. Even if some simple solutions have been proposed, they did not succeed in coping efficiently with empty cells. Furthermore, no resource discovery simulations with empty cells had been reported to evaluate the severity of this phenomenon. We addressed this issue and designed Hidagrid to efficiently manage the empty cells.

After an extended literature review on the resource discovery protocols for ad hoc networks, we highlighted the requirements for an efficient empty cell management protocol, i.e., consistency in relocating data items when their server cell is empty and low communication overhead. Based on these requirements, we designed the Hidagrid. Hidagrid acts as a sub-layer located between the routing layer and a grid based resource discovery protocol. As a result, it could be used with several discovery protocols. Hidagrid relies on a hierarchical grid that is used to locally manage the empty cells in order to limit the communication overhead and simplify

the different mechanisms. When the state of a cell changes, Hidagrid advertises that event in one of the hierarchical region of the grid, and data items relocation takes place in that region only. Based on the local knowledge of the grid, Hidagrid manages to consistently route queries and insertions to achieve good discovery performances.

Extensive simulations have proven the severity of empty cells for resource discovery protocols, i.e., loss of data items and denial of service. Hidagrid significantly improves the hit ratio of the discovery protocol as soon as the network is a little clustered. But it also improves the communication overhead of the discovery protocol by avoiding sending messages to empty areas, thus preventing congestion around those empty areas.

However, Hidagrid still suffers from some limitations that could be the subject of future research. First, Hidagrid cannot adapt to any grid, and a specific hierarchical grid must be designed. This results in limitations for the deployment of Hidagrid. We investigated this issue by generalizing the State Policy and the Relocation Policy for any grid. However, the generalized Relocation Policy could prove quickly complex, which could result in a lack of robustness. More research is therefore needed to develop an adaptable and efficient Relocation Policy, using different approaches.

Second, Hidagrid needs to stabilize the hierarchical grid as much as possible in order to avoid massive relocations. Some sophisticated hysteresis mechanisms must therefore be implemented at the cell level as well as at the State policy level. Even if we discussed the possible enhancements in that area, some investigations should be made to determine the optimal parameters, for Hidagrid to adapt efficiently to

a wide range of scenarios.

The hierarchical grid could also prove harmful when the size of the network is much smaller than the grid. If a small network is deployed in the center of the grid, the nodes could be located in several top level regions that are not populated enough to be activated. As a result, all the top level regions would be inactive, and nodes could only access the data items inserted in their region only because of the hierarchical structure. The hierarchical grid then virtually disconnects the network resulting in denial of service. The solution to that problem would be to use an adaptive State Policy when all the top level regions of the hierarchical grid are inactive. This area of future research could significantly improve the adaptability of Hidagrid so that Hidagrid could be deployed on any field. As a result, a grid based protocol could be used in any situation without setting up any grid.

Finally, as our simulations have shown, the communication overhead of Hidagrid increases significantly in high mobility scenarios because of its grid based structure. Each node crossing a cell border triggers communication. Therefore, it could be useful to limit the communication of the fast mobile nodes. For example, in Hidagrid, a fast mobile node should not update its internal table, and could delegate to one of its stable neighbor the task of inserting or querying an object if needed. This kind of mechanism could prove efficient if a majority of the nodes remain stable. Some studies [72] have shown the realism of such scenarios.

But even if Hidagrid suffers from some limitations, it proves to be an efficient protocol for a wide range of scenarios. The grid based discovery protocol and the whole network greatly benefit from Hidagrid by preventing the transmission of useless messages which can not be successfully processed. Even in the worst case

where the grid remains stable and homogeneously populated, the extra overhead generated by Hidagrid due to the hierarchical structure remains low, i.e., around 5% in our simulations. Hidagrid is also a step towards developing protocols which adapt to realistic scenarios. For this reason, we decided to use in our simulation a small transmission range (based on actual measurements), several mobility models, and obstacles in our maps, in order to catch as much as possible the features of a realistic network. We believe that research for ad hoc networks has reached a step where realistic considerations should be taken into account and therefore we developed Hidagrid with this idea in mind.

Bibliography

[1] "Gnutella." [Online]. Available: http://rfc-gnutella.sourceforge.net/

[2] "Pastry, a substrate for peer-to-peer applications." [Online]. Available: http://research.microsoft.com/~antr/Pastry/

[3] "Tapestry, infrastructure for fault-resilient, decentralized location and routing." [Online]. Available: http://www.cs.ucsb.edu/~ravenben/tapestry/

[4] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 1999, pp. 151–162.

[5] M. Mauve, A. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network*, vol. 15, no. 6, pp. 30–39, 2001.

[6] A. Khelil, C. Becker, J. Tian, and K. Rothermel, "An epidemic model for information diffusion in manets," in *MSWiM '02: Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*. New York, NY, USA: ACM Press, 2002, pp. 54–60.

[7] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2000, pp. 120–130.

[8] S. J. Philip and C. Qiao, "Hierarchical grid location management for large wireless ad hoc networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 33–34, 2003.

[9] K. Seada and A. Helmy, "Rendezvous regions: a scalable architecture for service location and data-centric storage in large-scale wireless networks," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004.

[10] C. Cheng, H. Lemberg, S. Philip, E. van den Berg, and T. Zhang, "Slalom: a ScaLAble LOcation Management scheme for large mobile ad-hoc networks," in *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, vol. 2, 2002, pp. 574–578.

[11] S.-C. M. Woo and S. Singh, "Scalable routing protocol for ad hoc networks," *Wirel. Netw.*, vol. 7, no. 5, pp. 513–529, 2001.

[12] S. Das, H. Pucha, and Y. Hu, "Performance comparison of scalable location services for geographic ad hoc routing," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2, 2005, pp. 1228 – 1239.

[13] "Jini." [Online]. Available: http://www.sun.com/software/jini

[14] "Salutation." [Online]. Available: http://www.salutation.org

[15] "Uddi." [Online]. Available: http://www.uddi.org

[16] "Napster." [Online]. Available: http://www.napster.com/

[17] "Upnp." [Online]. Available: http://www.upnp.org

[18] "Slp." [Online]. Available: http://www.openslp.org/

[19] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," in *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 1998, pp. 76–84.

[20] T. Camp, J. Boleng, and L. Wilcox, "Location information services in mobile ad hoc networks," in *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 5, 2002, pp. 3318–3324.

[21] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2001, pp. 161–172.

[22] C. Lindemann and O. P. Waldhorst, "Exploiting epidemic data dissemination for consistent lookup operations in mobile applications," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 3, pp. 44–56, 2004.

[23] O. Ratsimor, D. Chakraborty, A. Joshi, and T. Finin, "Allia: alliance-based service discovery for ad-hoc environments," in *WMC '02: Proceedings of the 2nd international workshop on Mobile commerce*. New York, NY, USA: ACM Press, 2002, pp. 1–9.

[24] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin, "GSD: a novel Group-based Service Discovery protocol for manets," in *2002. 4th International Workshop on Mobile and Wireless Communications Network*, 2002, pp. 140–144.

[25] H. Koubaa and E. Fleury, "A fully distributed mediator based service location protocol in ad hoc networks," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 5, San Antonio, TX, 2001, pp. 2949–2953.

[26] U. Kozat and L. Tassiulas, "Network layer support for service discovery in mobile ad hoc networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, vol. 3, 2003, pp. 1965–1975.

[27] F. Sailhan and V. Issarny, "Scalable service discovery for manet," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, 2005, pp. 235–244.

[28] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[29] M. Klein, B. Konig-Ries, and P. Obreiter, "Service rings - a semantic overlay for service discovery in ad hoc networks," in *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, 2003, pp. 180–185.

[30] Z. J. Haas and B. Liang, "Ad hoc mobility management with uniform quorum systems," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 228–240, 1999.

[31] D. Malkhi, M. K. Reiter, A. Wool, and R. N. Wright, "Probabilistic quorum systems," *Inf. Comput.*, vol. 170, no. 2, pp. 184–206, 2001.

[32] Z. Haas and B. Liang, "Ad-hoc mobility management with randomized database groups," in *Communications, 1999. ICC '99. 1999 IEEE International Conference on*, vol. 3, Vancouver, BC, 1999, pp. 1756–1762.

[33] J. Luo, J.-P. Hubaux, and P. T. Eugster, "Pan: providing reliable storage in mobile ad hoc networks with probabilistic quorum systems," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM Press, 2003, pp. 1–12.

[34] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "Epidemic information dissemination in distributed systems," *Computer*, vol. 37, no. 5, pp. 60–67, 2004.

[35] A. Helmy, "Small large-scale wireless networks: Mobility-assisted resource discovery," 2002. [Online]. Available: http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cs/0207069

[36] A. Helmy, S. Garg, P. Pamu, and N. Nahata, "Contact-based Architecture for Resource Discovery (CARD) in large scale manets," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, 2003.

[37] A. Helmy, "Capture: location-free contact-assisted power-efficient query resolution for sensor networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 1, pp. 27–47, 2004.

[38] A. Helmy, "Contact-extended zone-based transactions routing for energy-constrained wireless ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 1, pp. 307–319, 2005.

[39] C. Ververidis and G. Polyzos, "Routing layer support for service discovery in mobile ad hoc networks," in *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, 2005, pp. 258–262.

[40] Z. Haas, "A new routing protocol for the reconfigurable wireless networks," in *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, vol. 2, San Diego, CA, 1997, pp. 562–566.

[41] X. Jiang and T. Camp, "An efficient location server for an ad hoc network," The Colorado School of Mines, Tech. Rep. MCS-03-06, May 2003.

[42] X. Jiang and T. Camp, "An information dissemination protocol for an ad hoc network," in *Performance, Computing, and Communications, 2004 IEEE International Conference on*, 2004, pp. 337–345.

[43] Y.-B. Ko and N. H. Vaidya, "Location-Aided Routing (LAR) in mobile ad hoc networks," in *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 1998, pp. 66–75.

[44] F. De Rango, A. Iera, A. Molinaro, and S. Marano, "A modified location-aided routing protocol for the reduction of control overhead in ad-hoc wireless networks," in *Telecommunications, 2003. ICT 2003. 10th International Conference on*, vol. 2, 2003, pp. 1033–1037.

[45] X. Luo, T. Camp, and W. Navidi, "Predictive methods for location services in mobile ad hoc networks," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005.

[46] J. Ghosh, S. Philip, and C. Qiao, "Acquaintance Based Soft Location Management (ABSLM) in manet," in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 1, 2004, pp. 166–171.

[47] Y. Stojmenovic and P. E. V. Pea, "A scalable quorum based location update scheme for routing in ad hoc wireless networks," SITE, University of Ottawa, Tech. Rep. TR-99-09, September 1999.

[48] I. Aydin and C.-C. Shen, "Facilitating match-making service in ad hoc and sensor networks using pseudo quorum," in *Computer Communications and Networks, 2002. Proceedings. Eleventh International Conference on*, 2002, pp. 4–9.

[49] J. Tchakarov and N. Vaidya, "Efficient content location in wireless ad hoc networks," in *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, 2004, pp. 74–85.

[50] M. Ksemann, H. Hartenstein, H. Fler, and M. Mauve, " Analysis of a location service for position-based routing in mobile ad hoc networks ," in *Proc. of the 1st German Workshop on Mobile Ad-hoc Networking (WMAN 2002)*, Ulm, Germany, March 2002, pp. 121 – 133.

[51] K. Seada, A. Helmy, and R. Govindan, "On the effect of localization errors on geographic face routing in sensor networks," in *IPSN'04: Proceedings of the*

third international symposium on Information processing in sensor networks. New York, NY, USA: ACM Press, 2004, pp. 71–80.

[52] I. Abraham, D. Dolev, and D. Malkhi, "LLS: a Locality aware Location Service for mobile ad hoc networks," in *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing.* New York, NY, USA: ACM Press, 2004, pp. 75–84.

[53] S. Giordano and M. Hamdi, "Mobillity management: The virtual home region," EPFL-ICA, Switzerland, Tech. Rep., March 2000.

[54] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec, "Self organization in mobile ad hoc networks: the approach of terminodes," *IEEE Communications Magazine*, vol. 39, no. 6, pp. 166–174, 2001.

[55] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a Geographic Hash Table for data-centric storage." in *WSNA*, 2002, pp. 78–87.

[56] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for wireless networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking.* New York, NY, USA: ACM Press, 2000, pp. 243–254.

[57] S. Sivavakeesar and G. Pavlou, "Scalable location services for hierarchically organized mobile ad hoc networks," in *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing.* New York, NY, USA: ACM Press, 2005, pp. 217–228.

[58] Y. Xue, B. Li, and K. Nahrstedt, "A scalable location management scheme in mobile ad-hoc networks," in *Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on*, Tampa, FL, 2001, pp. 102–111.

[59] S. J. Philip, J. Ghosh, and C. Qiao, "Performance evaluation of a multilevel hierarchical location management protocol for ad hoc networks." *Computer Communications*, vol. 28, no. 10, pp. 1110–1122, 2005.

[60] W. Kiess, H. Fuessler, J. Widmer, and M. Mauve, "Hierarchical location service for mobile ad-hoc networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 4, pp. 47–58, 2004.

[61] P.-H. Hsiao, "Geographical region summary service for geographical routing," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 4, pp. 25–39, 2001.

[62] J. Tyan and Q. Mahmoud, "A network layer based architecture for service discovery in mobile ad hoc networks," in *Electrical and Computer Engineering, 2004. Canadian Conference on*, vol. 3, 2004, pp. 1379–1384.

[63] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking.* New York, NY, USA: ACM Press, 2003, pp. 96–108.

[64] S. Capkun, M. Hamdi, and J. Hubaux, "GPS-free positioning in mobile ad-hoc networks," in *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9.* Washington, DC, USA: IEEE Computer Society, 2001, p. 9008.

[65] "The network simulator - ns-2." [Online]. Available: http://www.isi.edu/nsnam/ns/

[66] G. Gaertner and V. Cahill, "Understanding link quality in 802.11 mobile ad hoc networks," *IEEE Internet Computing*, vol. 8, no. 1, pp. 55–60, 2004.

[67] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002. [Online]. Available: citeseer.ist.psu.edu/camp02survey.html

[68] C. Bettstetter, H. Hartenstein, and X. Perez-Costa, "Stochastic properties of the random waypoint mobility model," *Wirel. Netw.*, vol. 10, no. 5, pp. 555–567, 2004.

[69] C. Bettstetter, "Mobility modeling in wireless networks: categorization, smooth movement, and border effects," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 3, pp. 55–66, 2001.

[70] "Ansim." [Online]. Available: http://www.ansim.info

[71] C. Bettstetter and J. Zangl, "How to achieve a connected ad hoc network with homogeneous range assignment: an analytical study with consideration of border effects," in *2002. 4th International Workshop on Mobile and Wireless Communications Network*, 2002, pp. 125–129.

[72] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *MobiCom '04: Proceedings of the 10th an-*

nual international conference on Mobile computing and networking. New York, NY, USA: ACM Press, 2004, pp. 187–201.

List of Publications

Submitted for Review

S. Heuguet, K.C. Chua, M. Motani, "Empty cell management for grid based discovery protocols in ad hoc network,"submitted in *the IEEE Transactions on Mobile Computing*.