CHR: a Distributed Hash Table for Wireless *Ad Hoc* Networks*

Filipe Araújo
Universidade de Lisboa
filipius@di.fc.ul.pt

Luís Rodrigues
University of Lisboa
ler@di.fc.ul.pt

Jörg Kaiser
University of Ulm
kaiser@informatik.uni-ulm.de

Changling Liu
University of Ulm
changling.liu@informatik.uni-ulm.de

Carlos Mitidieri
University of Ulm
carlos.mitidieri@informatik.uni-ulm.de

Abstract

This paper focuses on the problem of implementing a distributed hash table (DHT) in wireless *ad hoc* networks. Scarceness of resources and node mobility turn routing into a challenging problem and therefore, we claim that building a DHT as an overlay network (like in wired environments) is not the best option. Hence, we present a proof-of-concept DHT, called *Cell Hash Routing* (*CHR*), designed from scratch to cope with problems like limited available energy, communication range or node mobility. *CHR* overcomes these problems, by using position information to organize a DHT of clusters instead of individual nodes. By using position-based routing on top of these clusters, *CHR* is very efficient. Furthermore, its localized routing and its load sharing schemes, make *CHR* very scalable in respect to network size and density. For these reasons, we believe that *CHR* is a simple and yet powerful adaptation of the DHT concept for wireless *ad hoc* environments.

*Selected sections of this report were published in the Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS'05), in conjuction with the 25th International Conference on Distributed Computing Systems (ICDCS-25), Columbus, Ohio, USA, June 2005. This work was partially supported by LaSIGE and by the FCT project INDIQoS POSI/CHS/41473/2001 via POSI and FEDER funds and by the ESF MiNEMA Research Network.

1 Introduction

In the recent past, researchers have given a lot of attention to distributed hash tables (DHTs) as these turned out to be a simple, elegant and powerful building block for distributed systems. In particular implementations of DHTs for wired scenarios have the following interesting properties: good scalability, tolerance to node failures and low congestion. Furthermore, they are efficient in terms of node degree/path length characteristics. In wired scenarios, routing can be taken for granted and DHTs are implemented as overlay networks on top of IP. In these overlays, two nodes can always use an IP tunnel to communicate with each other, no matter how far apart they are.

Although we can also find some attempts to build DHTs in wireless networks, namely [4, 15, 16], this exercise is much harder than building DHTs for wired networks [5, 8, 14, 17–19, 22]. The main challenge resides in the efficiency of routing. In wireless *ad hoc* networks routing of messages between two arbitrary nodes is an extremely expensive operation, that may involve flooding of the network to find a path or global state updates when the topology changes due to node mobility. Furthermore, each additional hop consumes significant resources at intermediate nodes. Therefore, any DHT for wireless networks must be aware and adapted to the underlying routing strategy.

In this paper, we present *Cell Hash Routing* (*CHR*), which is a proof-of-concept DHT designed from scratch for wireless *ad hoc* environments. *CHR* uses an inexpensive localized cell-based clustering method that groups nodes according to their location. This clustering method groups nodes inside cells of predefined and globally known shape. As a consequence, nodes are not individually addressable, because routing works at the cell level. Such an approach is particularly well-suited to the world of small and simple wireless devices or embedded systems, where nodes may look for specific contents and not for peers. Consider the case of sensor networks. In these networks, it is often irrelevant to know the output of the individual sensors; it may suffice to compute some function like an average or a maximum temperature at some particular zone. Hence, globally known individual addresses are not necessary as sensor nodes are not individually queried.

On the other hand, the advantage of clustering is twofold. First, it creates a very structured and sparsely populated network of clusters, where we can apply a lightweight routing scheme. Second, the efficiency of the routing scheme is not affected by increasing node density. Furthermore, our routing scheme also scales with increasing network sizes, because it is localized. Hence, by using a location-based clustered approach, routing in *CHR* is scalable with respect to both, network size and node density. Although simple, this scheme is powerful and enables us to implement a DHT in a straightforward and efficient way. In this paper we present *CHR* and claim that the DHT implemented by *CHR* perfectly fits wireless *ad hoc* environments and can be used as a component of more complex architectures, like a publish/subscribe system. When compared to other solutions, uniqueness of *CHR* comes from the use of routing *on top* of a clustered network, which is simultaneously the basis for the DHT.

The remainder of this paper is organized as follows. Section 2 presents the architecture of *CHR* and Section 3 presents early experimental results that illustrate the efficiency of the architecture. Related work is briefly surveyed in Section 4. Section 5 concludes the papers and presents directions for future work.

2 Architecture of *CHR*

2.1 Overview

In *CHR* we divide the space into equally-sized cells that have the shape of squares, as the grid depicted in Figure 1. Division of the space into cells allows a simple definition of the network of clusters: all nodes inside the same cell belong to the same cluster. In a sense we will use the notion of cluster as a kind of "supernode", where interactions occur at the level of clusters. The result is a much sparser network, where the routing scheme consumes fewer network and node resources. Since cells are immutable, clusters can receive an identification that does not change with time. Hence, nodes can always determine the identification of their cluster, as well as identify other nodes in the same cluster, even in dynamic scenarios, where they move around and change from cell to cell [1]. A crucial aspect of this division is that there must be a mapping between the identification of a cluster and its physical location, to enable the use of a geographical routing scheme.

Any routing scheme is comprised of two parts: *i*) a pre-processing algorithm that sets all data structures for *ii*) the distributed routing algorithm. For geographical routing schemes, the most popular routing algorithm is perhaps the *Greedy Perimeter Stateless Routing* [2,10] (GPSR). When possible, GPSR uses the greedy strategy of forwarding messages to the neighbor closest to destination. When it finds a local minimum, GPSR switches to perimeter mode and routes around faces. As soon as it finds a node closest to destination than the previous local minimum, GPSR goes back to greedy mode. We adopt this algorithm because it is simple and can achieve good routing performance in sparse networks. The most important restriction of this algorithm, however, is that it only ensures routing convergence in planar graphs. To overcome this problem, in Section 2.3, we will show how to create a non-planar graph for *CHR*, where GPSR is still able to converge. This simplifies the pre-processing algorithm and allows GPSR to perform better.

A DHT stores (key, value) pairs in its nodes. The node that stores a given pair (keyA, valueA) depends deterministically on the result of applying the hash function on keyA. One of the fundamental aspects of our architecture is that we use clusters instead of nodes to hold the values. We do not require each wireless node to participate in the DHT, because the relevant network entity is

[1] Usually, we use the term "cell" to mean the geographical square and the term "cluster" to mean a group of nodes. However, in some contexts, both concepts apply and therefore, we use the terms interchangeably.

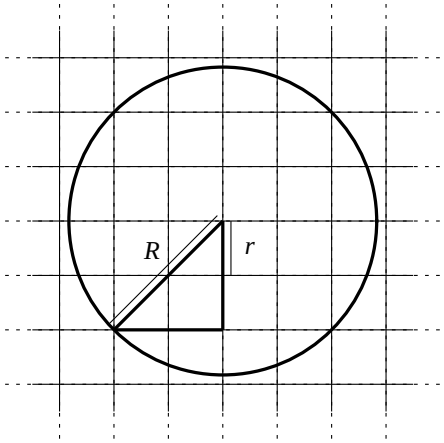


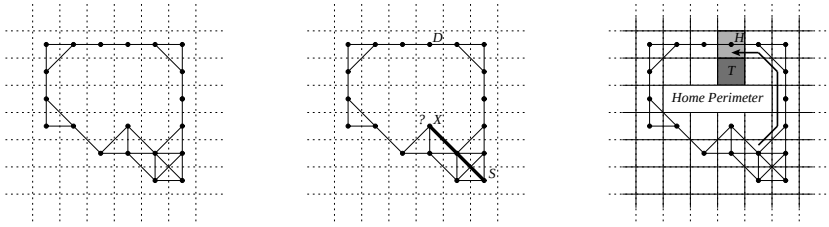
Figure 1: Division of the space into cells of fixed size

the cluster. Compared with a network of nodes, the simplicity of the network of clusters brings benefits to both, the routing scheme and the DHT operation. For this idea to work, the space of outputs of our hash function is the address space of the clusters. Hence, operation of the DHT becomes simple, when the key hashes to a cluster that really exists, i.e., that is populated by at least one node. Unfortunately, it is impossible to prevent that, in some cases, the key hashes to an empty cluster. In this case, *CHR* resorts to a technique first proposed in [16] that forces queries (or store operations) to take an entire loop around the empty cluster. In the following sections we will detail our architecture.

2.2 Division into cells

The size of the cells is limited by the communication range of the nodes, because we require that a node in a cell can always listen to any other node either in its own cell or in any adjacent cell. This restriction ensures that in most circumstances, the clustered network stays connected, as long as the initial network is also connected, even if only one node is active per cell. If we assume that nodes have a communication range of R, the resulting square side is at most $R/\sqrt{8}$. This can be seen in Figure 1. By adjacent cell, we are always referring to one of the 8 cells that surround the cell of a node. Note that we do not strictly require a Unit Disk Graph (UDG) model [2]. We only require that nodes

[2]In the UDG model two nodes A and B are neighbors if and only if $|AB| \leq 1$, i.e., if their distance is at most 1.



(a) Definition of the graph of clusters

(b) Routing may fail

(c) Home cell and home perimeter

Figure 2: Network of cells

can communicate with all the other nodes in their own cell and in the adjacent cells. We provide a short discussion of more general models in Section 5.

Nodes need to be aware of other nodes in the neighboring cells, mainly for two reasons: *i*) the routing scheme requires nodes to know whether or not the adjacent cells are populated and *ii*) we do not require nodes to perform some kind of leader election algorithm; on the contrary by making nodes know all (or at least part of) their neighbors, we can use randomization to share the routing load among all the nodes. For instance, when a node is routing a message that needs to go through some neighboring cell, it can arbitrarily select any node of that cell as the next hop. Otherwise, the nodes of a given cell would still need to decide which of them would forward the message. More formally, we state in Assumptions 2.1 and 2.2, the conditions that we need to build our architecture.

Assumption 2.1 *All the nodes know all the other nodes in their own and in any of the adjacent cells.*

Assumption 2.2 *A broadcast message inside a cell is received by all nodes of that cell.*

The purpose of Assumption 2.1 is to ensure that a node has sufficient knowledge to route messages and to support the DHT in the clustered hierarchy. However, as we describe in Sections 2.3, 2.4 and 5 this assumption can be relaxed, for the sake of scalability. For routing to work, a node only needs to know a single neighbor in each of the adjacent cells (nevertheless knowing more than one neighbor will increase robustness). On the other hand, knowledge of the nodes of the own cell is not needed for routing, but it is useful to support the DHT. But again, partial knowledge of the own cell will be enough in most circumstances. The purpose of Assumption 2.2 is to ensure a simple means of communication among nodes of the same cell, to enable operation of the DHT. For instance, in the UDG model this is easy to ensure, because all nodes of the

cell are within range and the reach of the radios between nodes in adjacent cells is very predictable.

Finally, Definition 2.1 defines a graph comprised of our cell-based clusters. This is represented in Figure 2a).

Definition 2.1 *Consider the graph where nodes represent clusters and edges exist between adjacent clusters if and only if their corresponding cells are both populated. To the embedding of the graph where nodes are placed in the center of the cells they represent, we call Geographically Clustered Graph (represented as $\mathcal{G}$).*

2.3 Routing scheme

CHR uses a routing scheme based on GPSR combined with a pre-processing algorithm that creates $\mathcal{G}$. This combination allows to address scenarios such as the one depicted in Figure 2b), where empty cells create voids in the cluster network. In this case, node X could be a local minimum in the path from S to D. There is plenty of evidence (see for instance [1, 11]) that i) routing performance of GPSR in planar graphs is better if node density (vs. communication range) is sparse (because longer edges imply fewer hops) and furthermore, ii) for a given density of nodes, denser graphs, i.e., with more edges also allow better performance. The reader should notice that there is no contradiction between i and ii: i states that network should have few nodes, while ii states that network should have many edges. Condition i is already fulfilled by our clustering approach. Condition ii is met because we do not need to planarize our graph, i.e., we do not remove intersections. The rationale for this is that the only intersections that may occur in $\mathcal{G}$ are in the diagonals (e.g., in the lower corner of Figure 2a)). However, as we prove in Theorem 2.1, GPSR always converges in $\mathcal{G}$, despite the existence of these intersections. As a consequence, the pre-processing algorithm of *CHR* only requires nodes to beacon the number of their cells (to fulfill Assumption 2.1). Besides this, nodes do not need further communication to define their local view of $\mathcal{G}$.

Theorem 2.1 *GPSR converges in $\mathcal{G}$.*

Proof 2.1 *GPSR converges in planar graphs. The only intersections that exist in $\mathcal{G}$ are in the diagonals of 4 nodes defining a square, say A, B, C and D, with edges AB, BC, CD and DA. Only one of these nodes, say A, could ever send a message in perimeter mode in a face inside the square. However, this could only happen for a destination in the direction of the 90^o angle $\angle DAB$ defined at A. But this is a contradiction, because, in this case A would not use perimeter, but greedy mode. Hence, GPSR converges, because it never uses intersecting edges while in perimeter mode.*

2.4 DHT implementation

Basic Mechanism

In the most basic setting, the hash function determines the single cluster that will hold the (key, value) pair. In the case of a given pair (keyA, valueA), the cluster whose identification equals hash(keyA) will be responsible for storing valueA. For instance, consider the ("Bob", 18) pair, where the key "Bob" hashes to 144. In this case, the value 18 should be stored at the cell 144. Therefore, if we could ensure that at least one node is kept active in each cell, implementing the DHT would be straightforward. However, some cells may be empty and therefore, we need some mechanism that can also deal with this case.

Addressing of the Cells

Although the routing scheme does not require cells to have specific addresses (position of the destination node would be enough), the DHT requires that cells have globally-known logical addresses. The restriction here is that the space of outputs of the hash function must have a direct mapping with the address space of the cells. Multiple mappings could be defined. We illustrate one possible mapping with a very simple example that assumes a bounded geographical space whose bounds are known by all the nodes. This scheme is equivalent to addressing the elements of a matrix in row-major order. Equation 1 shows how to determine the address of a cell in this scheme. D_x and D_y are the size of the space in the two dimensions, d_x and d_y are the sizes of each cell and L_x and L_y are the coordinates of the center point of the cell (it can also be any other point inside the cell). For instance, this equation is useful to let a node determine the number of its own cell.

$$A = \lceil D_x/d_x \rceil \times \lfloor L_y/d_y \rfloor + \lfloor L_x/d_x \rfloor \tag{1}$$

The reverse correspondence is also useful to allow nodes to perform geographical routing in $\mathcal{G}$. Equations 2 and 3 determine the center point (L_x, L_y) of the cell. c represents the number of columns and is computed as $c = \lfloor L_x/d_x \rfloor$, while % is the remainder of the division. To route to a given cell A, nodes need to determine the center point (L_x, L_y) of the destination cell, before they apply the GPSR routing algorithm. These equations are needed because geographical routing takes place using the center points of the cells (graph $\mathcal{G}$), while the DHT addresses of the cells are only logical. Consider again the ("Bob", 18) pair, where the key hashes to 144. To compute the center (L_x, L_y) of the target cell, a given node would have to replace A by 144 in the Equations 2 and 3.

$$L_y = d_y \left(\lfloor A/c \rfloor + 0.5 \right) \tag{2}$$

$$L_x = d_x \left(A\%c + 0.5 \right) \tag{3}$$

Division of the Keys Among the Nodes in a Cell

The best way of dividing the keys among the nodes inside each cell may depend on the global number of keys to store and on the number of nodes inside a given cell. If the total number of keys to store is fairly small, the best policy may be to store all the keys in all the nodes of the cell. This is simple and tolerant to individual node failures. We believe that, despite simple, this scenario may have wide application. Consider that the average number of nodes per cell is n_c and that each node stores an average of s_n bits in the DHT. If the distribution of the nodes and the keys by the cells is even, the total size of items of the DHT to store in each node is approximately $s_n \times n_c$. This number is reasonable, for moderate node density and if memory of nodes is not too small. It is easy to derive alternative schemes, where the load is balanced among all nodes of a cell, but these are omitted here due to lack of space. The important point to note here is that these schemes may not require full knowledge of the neighbors that populate the node's cell. This is important in densely populated cells, because otherwise we could not relax Assumption 2.1.

Resolving Empty Cells

One of the difficulties with our DHT architecture is that it is impossible to ensure that there are no empty cells. The problem with empty cells is that some keys may be left without nodes to store them. Since we use GPSR to route messages, we can follow an approach similar to GHT [16] to tackle this problem. Similarly, we define the concepts of *home cell* and *home perimeter*. Home cell is either the destination cell of a packet, if destination cell is populated, or the cell closest to the destination, in the other case (this requires a tie breaking rule, because many cells may be at the same distance). The home perimeter is the set of edges defining a face that encloses an empty destination cell (more precisely, the destination may be inside a face or outside the exterior face). These concepts are depicted in Figure 2c), where T represents the empty destination cell and H the home cell.

Like in GHT, *CHR* can take advantage of the standard behavior of the GPSR routing algorithm to ensure that a packet always reaches the home cell. This is easy to do if the home cell is the intended destination cell. If, on the contrary, the destination cell is empty, GPSR will also route the packet to the home cell. In the first time the packet reaches the home cell, any node in this cell will recognize that *i*) this cell is not the destination cell and *ii*) there is no edge connecting to the destination cell. Furthermore, the home cell is a local minimum in the path to destination. This forces the packet to enter perimeter mode (it might be in perimeter mode already). Standard behavior of GPSR forces the packet to loop in the home perimeter if destination does not exist. However, GPSR drops the packet as soon as it discovers a cycle in the path. Like in GHT, we need to change this behavior to ensure that the packet is not discarded at the end of the cycle. At this point, the nodes in the home cell know that the destination cell is empty and assume that their cell will be the

destination instead (refer to Figure 2c)). In fact, situation in our architecture is even simpler in some cases, because the home cell may already know that the destination cell is empty. Hence, it can avoid the loop around it (this may happen if the empty destination cell is adjacent).

3 Evaluation

To compare *CHR* with GHT [16], we tested the average path lengths in store/lookup operations. To do this, we routed messages from arbitrary existing nodes to arbitrary points in space. Hence, in general, these points did not correspond to any node and both, *CHR* and GHT had to route to the home cell/home node. We used a square of size 300×300 and a communication range of approximately 106 to have an 8×8 grid. Distribution of nodes in the square was uniform in all our experiments. Since node density is a key aspect to performance, we varied the number of nodes between 80 and 600.

The first thing we evaluate is the probability of having empty cells for each one of these node densities. This is depicted in Figure 3. In our settings, this probability is quite high when node density is low and rapidly decreases, to become nearly 0 as node density approaches 7 nodes per cell. To see the impact of node density on routing performance we evaluate the average path lengths for each node density. In Figure 4 we show the average path lengths. The first observation that we can do from inspection of the figure is that, as expected, clustering really benefits *CHR*. Furthermore as node density increases path lengths stabilize in *CHR*, while they grow in a non-clustered approach like GHT. This has to do with the length of the edges of the planar graph needed by GHT. As node density increases, the average edge length decreases and therefore, packets need more hops to reach destination. Finally, we can also observe that the impact of even a large number of empty cells is very small in the path lengths achieved by *CHR*.

Another advantage of *CHR* is that a moving node only needs to rebuild its database of keys, when it crosses a cell boundary. Finally, *CHR* is more robust, because a single key may be stored at many nodes, allowing the DHT to resist better to abrupt departure of nodes. In fact, this is a trade-off, where the down sides are that nodes have to know more neighbors (as required in Definition 2.1) and need to store more keys (all the keys in their cell). However, as we stated before, both of these problems can be mitigated at the cost of decreasing the robustness of the DHT (because a node can know fewer neighbors and store only a subset of keys in its cell). Nevertheless routing performance is not impaired by these techniques.

4 Related Work

The idea of routing on top of logical cells can be found in the Content and Cell based Predictive Routing Protocol [9] (CCPR), which uses a publish/subscribe

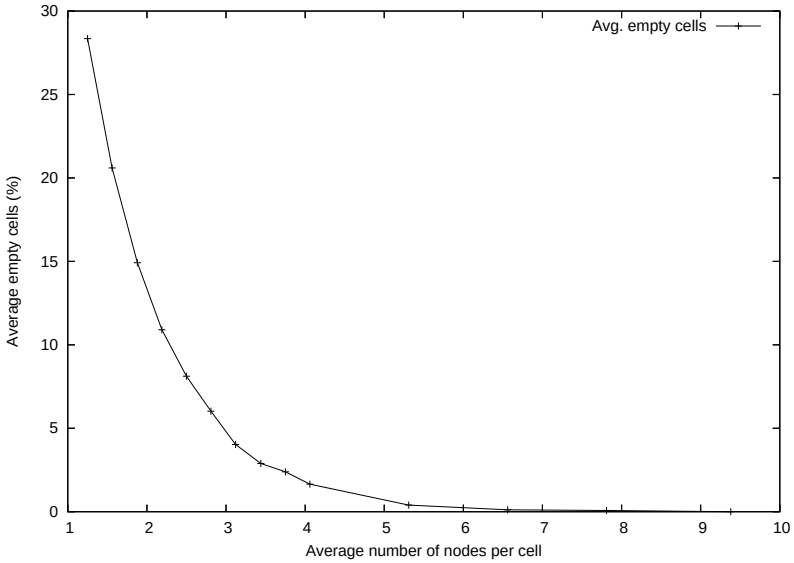


Figure 3: Average percentage of empty cells in CHR

interaction model, to distribute contents. Differently, many authors have focused on creating DHTs, specially for wired environments, where they operate as overlay networks on top of IP [5, 8, 14, 17–19, 22]. However, it is not trivial to operate these DHTs under wireless environments. On the other hand, the issues specific to wireless *ad hoc* networks turn routing into a difficult problem that has deserved a lot of attention. In this paper we are particularly interested in geographical routing schemes that ensure convergence, like GPSR [2, 10]. Hence building DHTs in wireless *ad hoc* networks has deserved fewer attention, but there are, nevertheless, some examples, e.g., [4, 15, 16].

Clustering is a well-known technique used to reduce complexity of the routing schemes and to reduce path lengths in geographical algorithms [3, 11]. Clustering the space in squares of equal size can be found in [20] with the goal of saving energy. [7, 13] also create clusters that can be used to create DHTs. For instance, [13] also creates a DHT on top a logical structure of cells. Another work that also has some similarities with *CHR* is [21], which presents a Two-tier Data Dissemination (TTDD). TTDD creates a grid of dissemination for each type of data. This grid also follows a geometrical arrangement. Then, TTDD uses flooding within the local area, while global dissemination is achieved through the grid, for the sake of efficiency. When compared to these previous approaches, uniqueness of *CHR* comes from the simplicity of routing *on top* of the clusters, while no routing is needed underneath. I.e., in *CHR* the size of the cells is carefully selected to allow direct communication of nodes inside neighboring cells. In this way, routing really takes place in the logical graph $\mathcal{G}$. Even the problem of circulating around voids is done in $\mathcal{G}$. As we have shown, one

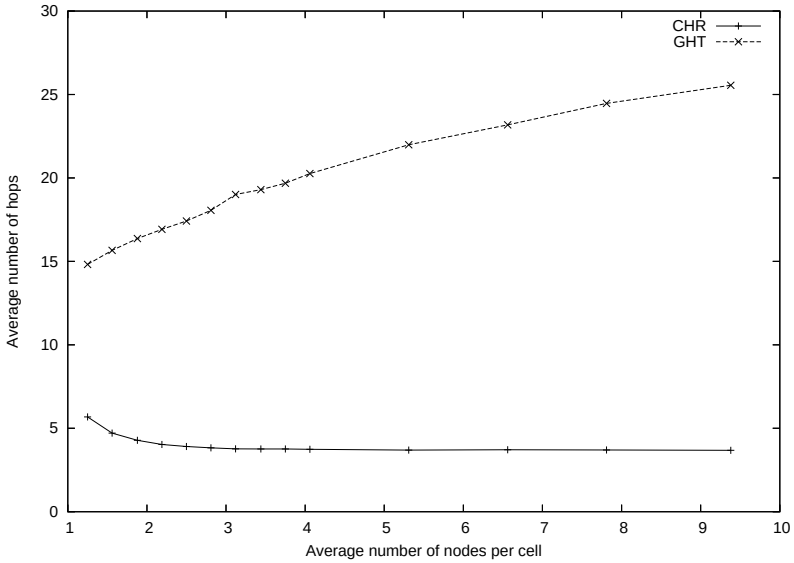


Figure 4: Performance of CHR vs. GHT

of the advantages of doing this is that there is no need to remove intersections of the graph. Unlike *CHR*, in the previous approaches, routing from one cell to the next requires the normal node-to-node routing. As a consequence, voids in the network are much more difficult to handle.

One aspect that we do not explore here and that we can find in previous work like [6, 12, 16] is the replication of data in different areas (in these cases to reduce distance to the stored items). Nevertheless as we point out in Section 5, many of the solutions employed in these systems can also be adopted in *CHR* not only to improve performance, but also to increase tolerance to node failures.

5 Conclusions and Future Work

The results presented here show that *CHR* is a viable alternative to build a DHT for *ad hoc* wireless networks. Therefore we plan to pursuit this work, by designing a complementary set of mechanisms that address the limitations of this current implementation. Namely:

• **Dynamic Cell Structure**: when the number of nodes in a cell drops below l, the cell is considered empty. On the contrary, the cell needs to acquire h nodes before it is considered populated ($h > l$). Note that the value h should be fairly small. As a consequence, knowing h does not require much memory, because, in general, a node will not need to know all the neighbors in its own cell (i.e., Assumption 2.1 can be relaxed). A cell leaving the network delivers its keys to its home cell. An entering cell needs to query its home perimeter to receive its keys. Additionally, it will also receive keys of empty cells for which

it becomes the home cell.

• **General non-UDG model**: in more general models, it is possible for a node not to see some of its neighbors in its cell or in some adjacent cell. One possible approach to overcome this problem is to create routing tables to reach only the invisible nodes inside the cell or adjacent cells.

• **Incorrect determination of position**: in this paper, we assumed that nodes can always determine their position exactly, which is actually not the case. The consequence of this is that a node may consider itself to be in the wrong cell. In a sense, this resumes to the non-UDG model.

• **Cluster induces disconnection**: occasionally, the clustering mechanism may disconnect the network. This event should be rare, especially in denser networks, where use of *CHR* is more worthwhile. A straightforward solution consists of creating the necessary links and then using a geographically-scoped broadcast to remove intersections.

• **Fault-tolerance requirements**: one of the occurrences that *CHR* should try to avoid as much as possible is the loss of stored (key,value) pairs. We already suggested the use of thresholds to ensure that keys are sufficiently spread among nodes of the same cell. This mechanims can be complemented with a technique already used in wired DHTs, e.g., [18], that consists in using k hash keys to replicate contents in different cells.

References

[1] Filipe Araújo and Luís Rodrigues. Fast localized delaunay triangulation. In *The 8th International Conference On Principles Of Distributed Systems (OPODIS 2004)*, Grenoble, France, december 2004. (to appear).

[2] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in *ad hoc* wireless networks. In *International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 48–55, 1999.

[3] G. Chen and I. Stojmenovic. Clustering and routing in wireless ad hoc networks. Technical Report TR-99-05, Department of Computer Science, SITE, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada, June 1999.

[4] Jakob Eriksson, Michalis Faloutsos, and Srikanth Krishnamurthy. Scalable ad hoc routing: The case for dynamic addressing. In *IEEE Infocom 2004*, 2004.

[5] P. Fraigniaud and P. Gauron. The content-addressable network D2B. Technical Report 1349, LRI, Univ. Paris-Sud, France, Jan 2003.

[6] Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless sensor networks. *IEEE Distributed Systems online*, 4(11), November 2003.

[7] Indranil Gupta, Robbert van Renesse, and Kenneth P. Birman. Scalable fault-tolerant aggregation in large process groups. In *DSN '01: Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS)*, pages 433–442. IEEE Computer Society, 2001.

[8] Frans Kaashoek and David R. Karger. Koorde: A simple degree-optimal distributed hash table, 2003.

[9] J"org Kaiser and Changling Liu. Content and cell based predictive routing (ccpr) protocol for mobile *ad hoc* networks. In *5th International Workshop for Advanced Parallel Processing Technologies (APPT'03)*, Lecture Notes in Computer Science 2834: Advanced Parallel Processing Technologies, Springer, Xiamen, China, September 2003.

[10] Brad Karp and H. T. Kung. GPRS: Greedy perimeter stateless routing for wireless networks. In *ACM/IEEE International Conference on Mobile Computing and Networking*, 2000.

[11] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *22nd ACM Symposium on the Principles of Distributed Computing (PODC 2003)*, Boston, Massachusetts, July 2003.

[12] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, August 2000.

[13] Mei Li, Wang-Chien Lee, and Anand Sivasubramaniam. Efficient peer-to-peer information sharing over mobile ad hoc networks. In *Second Workshop on Emerging Applications for Wireless and Mobile Access (MobEA II), in conjunction with the World Wide Web Conference (WWW)*, May 2004.

[14] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Twenty-First ACM Symposium on Principles of Distributed Computing (PODC 2002)*, Monterey, California, July 2002.

[15] Himabindu Pucha, Saumitra M. Das, and Y. Charlie Hu. How to implement dht in mobile ad hoc networks? Student poster, the 10th ACM International Conference on Mobile Computing and Network (MobiCom 2004), September-October 2004.

[16] 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 sensornets. In *First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, September 2002.

[17] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Conference on applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[18] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[19] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *ACM SIGCOMM*, San Diego, August 2001.

[20] Ya Xu, John S. Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Mobile Computing and Networking*, pages 70–84, 2001.

[21] Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of ACM MOBICOM*, 2002.

[22] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.