An Algorithm for Dissemination and Retrieval of Information in Wireless Ad Hoc Networks*†

Hugo Miranda
University of Lisbon - Portugal
hmiranda@di.fc.ul.pt

Simone Leggio
University of Helsinki - Finland
simone.leggio@cs.helsinki.fi

Luís Rodrigues
University of Lisbon - Portugal
ler@di.fc.ul.pt

Kimmo Raatikainen
University of Helsinki - Finland
Kimmo.Raatikainen@cs.helsinki.fi

Abstract

Replication of data items among different nodes of a wireless infrastructureless network may be an efficient technique to increase data availability and improve data access latency. This paper proposes a novel algorithm to distribute data items among nodes in these networks. The goal of the algorithm is to deploy the replicas of the data items in such a way that they are sufficiently distant from each other to prevent excessive redundancy but, simultaneously, they remain close enough to each participant, such that data retrieval can be achieved using a small number of messages. The paper describes the algorithm and provides its performance evaluation for several different network configurations.

1 Introduction

Information management in wireless infrastructureless (ad-hoc) networks is not a straightforward task. The inherently distributed nature of the environment, and the dynamic characteristics of both network topology and medium

*This work was partially supported by the MiNEMA programme of the European Science Foundation and by FCT project MICAS, POSC/EIA/60692/2004 through FCT and FEDER.

†This paper was originally published in A.-M. Kermarrec, L. Boug, and T. Priol (Eds.): Euro-Par 2007, LNCS 4641, pp. 875884, 2007. Posted with permission of the publisher.

connectivity, are a challenge for the efficient handling of data. The limited resources and the frequent disconnection of the devices suggest that data should be replicated and distributed over multiple nodes. A data dissemination algorithm for such a decentralised approach should balance the need to provide data replication (to cope with failures) with the need to avoid excessive data redundancy (as nodes may have limited storage capability). Finally, since in wireless networks both bandwidth and battery power are precious resources, the algorithm should also minimise the amount of signalling data.

In this paper, we address the problem of finding adequate locations for the replicas of a data object using a distributed algorithm. The same problem has been addressed before (e.g. [2, 6, 9, 1, 3, 10]) although with a different set of assumptions.

System Model. We share most of the assumptions described in [3]. In brief, the ad hoc network is composed of cooperative nodes which are producers and consumers of uniquely identifiable data items, composed of a key and a value with application dependent semantics. Each node has storage space available for storing the items it produces. In addition, the nodes make available limited storage space for keeping replicas of a fraction of all the objects produced by other nodes. The system does not require the space at all nodes to be of the same size.

Replication is used to improve availability and reduce access latency. Also, like in [3], we assume for simplicity that all items are equally sized so that the space made available by each node can be referred in item units instead of bytes.

Contrary to [3], we assume that there is no predictable access pattern to the objects which is known in advance by the nodes and does not change. This access pattern may be used to bias the distribution of the replicas so that the most popular items have more replicas. We are interested in scenarios where these access patterns cannot be derived a priori or even during the lifetime of the system (for instance, short lived objects). Therefore, we aim at distributing data items as evenly as possible among all the nodes that form the network, avoiding clustering of information in sub-areas; an uniform dissemination of data items should leverage lower access latency to any item from any node in the network, i.e, whenever a data item is requested by a node S, the distance to the node that provides the reply should be approximately the same, regardless of the location of S. Naturally, the actual distance depends on multiple parameters, such as the number of nodes in the network, the amount of memory made available at each node, and the number of data items.

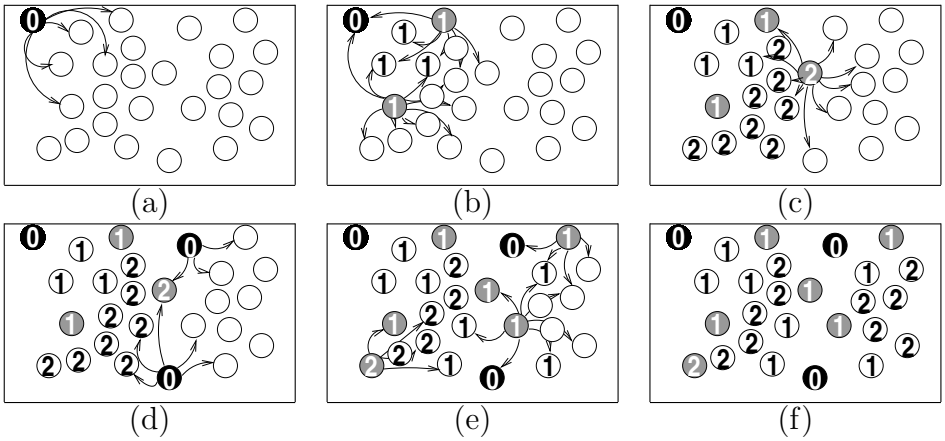


Figure 1: Example of dissemination of an item

There are multiple applications for a distributed storage with these characteristics. Cooperative teams may use it to share photographs, annotations or measurements while on the field [3]. Users on spontaneous networks can use it to advertise SIP records containing their interests to find other users willing to play distributed games or chat [5].

Scope and Contribution of the Paper. Implementing a full system with these characteristics is a complex task that must address multiple challenges and requires several algorithms. The contributions of this paper are the following. Firstly, it proposes an algorithm to perform an initial distribution of the data items that satisfies the requirements above. Additionally, it describes an algorithm for retrieving the information. Problems like updating the data items, shuffling the item distribution to address node movement or disconnection, or tolerating uncooperative nodes are out of the scope of this paper (the interested reader may consult [7]).

2 Overview

An example of the dissemination of an item is depicted in Fig. 1. The dissemination begins with the broadcast of a *registration* message. The item is stored at the producer and included in the message (Fig. 1(a)). The figure depicts in black the nodes that store a replica of the item. *Registration* messages carry a *Time From Storage* (TFS) field which records the distance (in number of hops) from the node sending the message to the known closest copy. The TFS for the message to be forwarded by each node is depicted in the centre of the node.

Figs. 1(b) and 1(c) show the progress of the dissemination. Nodes use a message propagation algorithm named Pampa [8] (to be discussed later) to reduce the number of transmissions. Nodes decide whether to forward the message after a small *hold period*, during which they monitor the network, listening for possible retransmissions of the same message. During the hold period, each node computes the lowest value of all the TFS fields it has received in a variable named *mTFS*. In the figure, nodes that forward a registration message but did not store the data item are depicted in gray. When forwarding a registration message, a node sets the TFS field to *mTFS*+1, accounting for the additional hop needed to reach the closest copy of the item.

Central to our algorithm is a constant *Distance Between Copies* (DbC). The DbC dictates the maximum value of the TFS field and, implicitly, the degree of replication of the items. DbC is expected to be small. In this example, we use DbC=2. Fig. 1(d) shows that a node with *mTFS*=DbC at the end of the hold period stores a copy of the item and retransmits the message. The TFS of the message is reset to 0 to let other nodes learn about the newly stored copy and update their *mTFS* variables accordingly (see for example Fig. 1(e)).

The final state of the system after the dissemination of the item is depicted in Fig. 1(f). Although only a small number of nodes have stored the item, a replica is stored at no more than DbC hops away from any of the nodes.

Broadcast Algorithm. We use the Pampa [8] broadcast algorithm to propagate dissemination and query messages. In comparison with a conventional flooding algorithm, Pampa reduces the number of nodes required to transmit a message by having nodes more distant to the previous forwarder to broadcast the message earlier. Nodes closer to the source (i.e., those whose expected additional coverage would be smaller) do not retransmit. Pampa does not require devices to be aware of their location or of the location of their neighbours. Instead, each node uses the Received Signal Strength Indicator (RSSI) of the first retransmission listened to set the hold period. The hold period is set such that nodes with a lower RSSI expire their timers first. During the hold period, nodes count the number of retransmissions listened and, at the end of the hold period, they do not retransmit the message if a predefined threshold was reached. Based on evaluation results presented in [8], in this paper we use a threshold value of 2.

Due to the store-and-forward nature of the algorithm, Pampa is not used as a black-box. Next, we discuss how Pampa was adapted for our purposes.

3 Dissemination

A global overview of the dissemination algorithm was presented in Sec. 2. This section provides additional details on the steps executed by each node.

Forwarding registration messages and storing items. A node only decides whether to forward (or drop) a *registration* message and whether to store or not the corresponding data item at the end of the hold period. The decision takes as input the following parameters: *i*) the output of the Pampa's algorithm, that accounts only with the number of retransmissions listened; and *ii*) the value of *mTFS*. The data item is stored if, at the end of the hold period, *mTFS*=*DbC*. Note that if some other node in the vicinity previously decided to store the data item, it had retransmitted the message with TFS set to zero and, therefore, *mTFS* would have been reset accordingly. The message is forwarded if the data item was stored or if the output of Pampa's algorithm suggests it.

Computing the hold period. The base value for the hold period is given by the underlying Pampa broadcast algorithm. Pampa computes the delay based on the signal strength which, in turn, depends on the relative location of nodes. We have also seen that if a node is the first node in its own vicinity to decide to forward a message and mTFS=*DbC*, then it stores a copy of the data item. Therefore, depending on the deployment of the nodes, and of the location of the sources of the registration messages, some nodes may end up storing much more items than others. To promote a balanced distribution of items, regardless of the physical location of nodes, our algorithm applies a bias to the base value of the hold period derived by Pampa. The bias is a function of the number of items already stored by the node.

When a node whose storage occupancy ratio is above some threshold receives a *registration* message with TFS=*DbC*, it multiplies Pampa's hold period by a factor proportional to the occupancy ratio of its storage space. Precisely, the delay is determined by the function $holdPeriod = \mathrm{hP} \times \left(1 + \frac{\mathrm{occup-thresh}}{\mathrm{1-thresh}} \times \mathrm{bias}\right)$, where *hP* represents the hold period computed by Pampa, *occup* is the current occupancy ratio of the storage space, and *thresh* and *bias* are configuration parameters indicating respectively the minimal threshold for triggering this function and a weight of this component on the final value of the hold period. In the simulations presented in Sec. 5 *thresh*=0.7 and *bias*=2.0.

Memory management. We assume that each node has some memory region reserved for storing data items. Nodes keep on adding items to this

region until it is completely filled. Only then, nodes are required to drop stored items to make room for new items. Note that our scheme to compute the hold period already attempts to balance the memory occupation among nodes. If a nodes needs to make room for a new item, it will randomly selected one of the previous entries. We note that policies like Least Recently Used (LRU) or other deterministic policies should be avoided in our algorithm. This is because a deterministic criteria applied to different nodes would likely select the same entry for replacement. This undesirable behaviour would eliminate a large number of replicas of the same item resulting in an uneven distribution.

Analytical Properties of the Algorithm. At the end of the dissemination, and assuming a perfect networking environment without message losses and if nodes did not discard any item from their local storage, the following properties can be derived concerning the distance of the nodes to some data item.

All nodes, with the exception of those at the margins of the networked region, should be able to find a copy of the data item at a distance not higher than $\frac{\mathrm{DbC}+1}{2}r$ (where r is the transmission range of the devices) or $\left\lceil \frac{\mathrm{DbC}+1}{2} \right\rceil$ hops. This results from the fact that each node storing a copy of the item will become the "closest copy" to nodes that have served either as predecessors or successors in the dissemination of the registration message. An interesting case happens when DbC is even what may leave some nodes equidistant (in hops) of at least two copies of the item. Fig. 1(f) shows such a node at the centre of the network.

From the previous result it is possible to derive the expected average distance from any node to a data item, assuming an uniform deployment of the nodes. Function τ is given by $\tau\left(\mathrm{DbC}\right) = \sum_{i=0}^{\mathrm{DbC}} \left(\left\lceil \frac{i+1}{2} \right\rceil \frac{\pi\left(\frac{i+1}{2}r\right)^2 - \pi\left(\frac{i}{2}r\right)^2}{\pi\left(\frac{\mathrm{DbC}+1}{2}r\right)^2} \right) = \frac{\sum_{i=0}^{\mathrm{DbC}} \left(\left\lceil \frac{i+1}{2} \right\rceil (2i+1) \right)}{(DbC+1)^2}$ and successively partitions a circle with radius corresponding to the DbC in semi-circles centred at the node storing the copy of the item. The function accounts with the proportion of the area contributed by each semi-circle and with the distance in hops of the nodes located in that semi-circle to the node storing the copy. Function τ has the following values for small DbCs: $\tau(2) = 1.55(5), \tau(3) = 1.75$ and $\tau(4) = 2.2$.

4 Data Retrieval

To retrieve some item, a node begins by looking for it in its local storage. If the item is not found locally, the node initiates a search in its vicinity. This is implemented by broadcasting a *query* message with a limited range, given by a variable $qTTL$. This variable is initialised with a small value and is successively adjusted, in order to adapted to the network conditions. Function τ, introduced in the analysis above, is used to set the initial value of $qTTL$. If no answer to the query message is received from the vicinity within a predefined amount of time, the query will be broadcast with a TTL large enough to deliver the message to every node in the network. This broadcast should be avoided as it requires the transmission of as many messages as the data dissemination algorithm.

When a node receives a query message and does not have the item in its local storage, it will have to decide whether to forward or drop the query message. Again this decision is made after an hold period, according to the criteria defined by the underlying Pampa broadcast algorithm. Note that if the TTL field of the message has reached the value of 0 the query is simply dropped. If the query messages is retransmitted, the forwarding node pushes its own address to a *route stack* field of the message, in a route construction process similar to the route discovery algorithm in some source routing protocols for MANETs (e.g. [4]).

If the key is found, the node sends a point to point reply to the source of the query without waiting the delay suggested by Pampa. The reply message follows the path constructed in the *routeStack* field of the query. The *TFS* field of the reply is set to 0 at the origin of the reply and incremented at every intermediate hop to capture the distance at which the data item was found.

The reply message is unreliably forwarded by the intermediate hops, thus there is some probability that the reply is lost. On the other hand, no provision is taken to limit the number of replies sent to the node. Therefore, there is a reasonable probability that at least one of the routes constructed during the query propagation remains valid until the reply is delivered.

When the node that issued the query receives the first reply, it performs corrective measures over the data distribution and the $qTTL$ value. A reply found far away from the source of the query signals an uneven distribution of the item. Therefore, the node that issued the query stores the item if the reply was received from a node located more than DbC hops away. Given that the dissemination algorithm aims at achieving an adequate distribution of the items, the distance (in number of hops) from the source of the query to any item should be approximately the same and will depend mostly of the

number of neighbours of the node and their storage space. After each query, $qTTL$ is tuned by weighting its previous value with the distance at which the reply was found (available in the *TFS* field of the reply). The goal is to reduce the number of queries requiring a second broadcast while keeping $qTTL$ as small as possible.

5 Evaluation

We have implemented a prototype of our algorithm in the *ns-2* network simulator v. 2.28. The simulated network is composed of 100 nodes uniformly disposed over a region with 1500mx500m. The simulated network is an IEEE 802.11 at 2Mb/s.

Runs are executed for 900s of simulated time. Each run consisted of 400 queries over a variable number of disseminated data items, as described below. Data items have 300 bytes and are disseminated in time instants selected uniformly between 0 and 400s. Note that the size of the data items is only relevant for estimating the traffic generated at the network; when considering memory availability at each node we have simply taken into account the number of data items stored at each node. Queries start at 200s and are uniformly distributed until the 890s. The nodes performing the queries and the queried items are selected using an uniform distribution. The simulation ensures that only advertised records can be queried.

No warm-up period is defined. All values presented below average 100 independent runs, combining different node deployments, query and dissemination times. The evaluation uses two metrics. The "average distance of the replies" measures the distance (in number of hops) from the querying node to the source of the first reply received. The distance of a reply is 0 if the value is stored in the querying node. The "average number of transmissions per query" measures the total number of query and reply messages (initial transmissions and forwarding) performed by all nodes and divides it by the number of queries.

Theoretic Idealised Model and Saturation Point. The simulation results are compared with an execution of the algorithm, analytically computed for an idealised network where nodes are uniformly distributed and the space made available at the nodes within each circle with radius $\frac{\mathrm{DbC}_{+1}}{2}r$ is sufficient to store all the disseminated items.

The storage capacity in a network region containing n nodes is given by $n \times \left(s + \frac{i}{N}\right)$ where i is the number of items advertised, N the number of nodes in the system (recall that nodes keep the items they advertise in

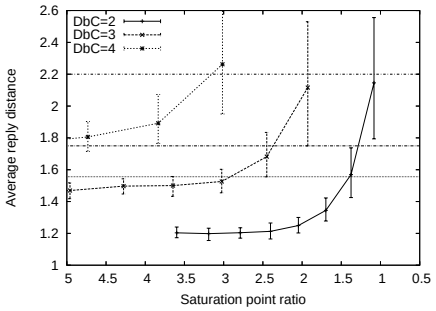
a separate region of the storage space), and s is the storage space made available at each node for data items advertised by other nodes. We define the "saturation points" (SP) of our algorithm as the multiple solutions of the equation $n \times \left(s + \frac{i}{N}\right) = i$. Each solution will correspond to a different configuration that is capable of storing all the data items being advertised and, therefore, that should be able to provide all the replies in the target average distance given by function τ.

In this evaluation we are interested in comparing the implementation of the algorithm with this ideal model, given that it characterises the best results that can be achieved. In particular, to evaluate the performance of the algorithm close to the SP and to compare the performance for different values of DbC.

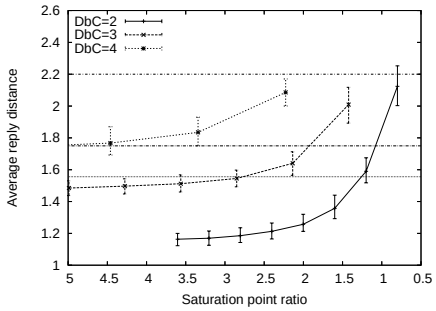
Sensitivity to Different Network Configurations. The performance of the algorithm is affected by the number of nodes in the neighbourhood of each node, the storage size at every node and the number of items advertised in the network. To evaluate the effect of the variation of each of these parameters individually, we fixed a value for each in a baseline configuration. Each parameter was then individually varied keeping the remaining consistent with the baseline configuration. The number of neighbours was varied by configuring the nodes with transmission ranges between 150 and 325 meters. The number of neighbours was estimated by counting the number of nodes that received each broadcast message on each simulation with the same transmission range. A transmission range of 250m was settled for the baseline configuration. The storage size was varied between 2 and 16 items. In the baseline configuration, each node makes available storage for 10 items. The number of items advertised was varied between 50 and 800. Advertisements were uniformly distributed by the nodes. In the baseline configuration, 200 data items are advertised.

Note that the baseline configuration is below the SP for all values of DbC. Figure 2 shows the average distance of the replies in the simulations. The x axis harmonises the results by presenting them according to a ratio to the SP given by $\frac{n \times \left(s + \frac{i}{N}\right)}{i}$. Error bars show the highest and lowest average distance of the replies of a subset of the simulations that excluded the 10% with higher and lower values. To facilitate the comparison with the theoretical model, the figures show the values of function τ for every DbC tested.

When the system is below the SP, our algorithm exhibits a smaller average reply distance than the computed for the idealised model. Our approach is creating more replicas than estimated by the idealised model, resulting in an increased proximity of the nodes to the data items. When the system



(a) Variation of number of neighbours



(b) Variation of storage size

Figure 2: Average distance of the replies

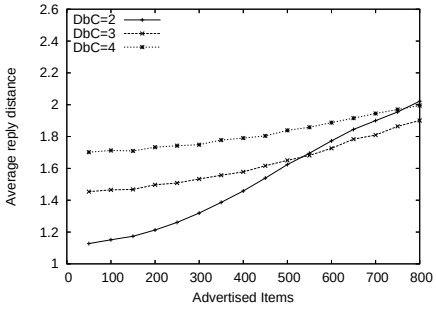


Figure 3: Variation of number of items

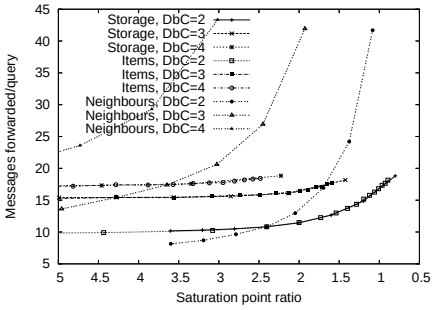


Figure 4: Frames per query

approaches the SP, as expected, the average reply distance increases (as it becomes impossible to store all the items in the target DbC). The target distance is reached with SP > 1. Still, the system continues to provide acceptable results, and in the majority of the cases copies are found within only a few hops in excess of the optimum limit.

We identify two differences between the idealised and the experimental models to justify the discrepancies: *i*) since nodes are randomly deployed, it is unlikely that at every retransmission, there exists one node located precisely over the limit of the transmission radius of the previous source. Smaller distances result in additional hops travelled by the messages, and reduce the effective area (and nodes) that should be accounted in the estimation of the SP; *ii*) concurrent decisions, amplified by the delay in the propagation of the messages may permit to nodes in proximity to simultaneously decide to store the items.

Figure 3 depicts results for different numbers of advertised items (in the x axis). It shows that DbC plays an important role in the performance of the algorithm. It can be seen that the average distance of the replies

for the different values of DbC tend to approximate as the storage capacity of the region is reduced. In particular, the lines for DbC=2 and DbC=3 intersect around the SP for DbC=2. This is the expected behaviour of the algorithm, given that, by definition, above the SP, it is not possible to store all information at the target range.

Traffic. The average number of messages per query is presented in Fig. 4. It is interesting to notice the overlap, for each value of DbC, of the lines that capture the behaviour of the system with the size of the storage space and with the number of items. This confirms that when the system is below the SP, none of these factors influences the number of messages transmitted per query. Additionally, we compared the growing ratios of the curves for the average distance of the replies (Fig. 2(b) and 3) and for the number of messages forwarded/query (Fig. 4) in both scenarios. The difference between these ratios is less than 2% when the storage space is changed and less than 7% when the number of items changes. These small values show that the growing of the average distance implies an almost linear grow of the number of messages. This confirms the efficiency of our adaptive mechanism for defining $qTTL$: it prevents the query algorithm from frequently resorting to a full broadcast, even in adverse conditions.

On the other hand, we expect the number of messages to drop significantly when the density increases, because we benefit from the properties of Pampa, which adapts the proportion of nodes retransmitting a message to the network density. Comparing results depicted in Figs. 2(a) and 4, it can be seen that although the distance of the replies tends to stabilise with the grow of the network density, the number of messages continues to diminish. Here, the difference between the ratios is higher than 36%.

6 Related Work

Several papers have addressed the problem of distributing copies of data items in MANETs. However, most of the previous work makes stronger assumptions about the network or the application scenario. Some assume that it is possible to collect statistics about data usage, such as which items are accessed more frequently [3] or obtain similar information from user profiles [1]. Others assume that there is a single data source [10] or that nodes are aware of their location [2, 6, 9]. In contrast with previous approaches, our work is targeted at spontaneous networks (such as rescue teams) where all nodes need to share many short lived data items. Our algorithm prevents the duplication of data items in neighbouring nodes by counting the number

of hops travelled by an item before being stored. Instead of using geographical information, we take advantage of the fine dissemination properties of Pampa to ensure the geographical distribution of the information.

7 Conclusions

This paper has presented an algorithm for retrieving and distributing information in ad-hoc networks. The algorithm is fully distributed. Its main goal is to ensure an even geographical distribution of the data items, so that requests for a given data item are satisfied by some nodes close to the source of the query.

This goal is obtained by combining different techniques. Data items are disseminated with a counter to provide a minimal distance between the copies; a broadcast protocol reduces the number of messages required for propagation and increases the geographical distance between the hops. Finally, an adaptive mechanism allows to limit the propagation of most queries. Simulation results show that the algorithm achieves a fair dissemination of items throughout the network and that a small number of messages is required to retrieve items.

References

[1] Anwitaman Datta, Silvia Quarteroni, and Karl Aberer. Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks. In *Proc. of the Conf. on Semantics of a Networked World, IC-SNW'04*, volume 3226 / 2004 of *LNCS*, 2004.

[2] Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless sensor networks. *IEEE Distributed Systems Online*, 2003.

[3] Takahiro Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proc. of the 20th Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2001)*, volume 3, pages 1568–1576, 2001.

[4] D. B. Johnson and D. A. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer Academic Publishers, 1996.

[5] Simone Leggio, Hugo Miranda, Kimmo Raatikainen, and Luís Rodrigues. SIPCache: A distributed SIP location service for mobile ad-hoc networks. In *Proc. of the 3rd Conf. on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS)*, 2006.

[6] Dandan Liu, Ivan Stojmenovic, and Xiaohua Jia. A scalable quorum based location service in ad hoc and sensor networks. In *Proc. of the 3rd IEEE Conf. on Mobile Ad-hoc and Sensor Systems*, 2006.

[7] Hugo Miranda, Simone Leggio, Luís Rodrigues, and Kimmo Raatikainen. A stateless neighbour-aware cooperative caching protocol for ad-hoc networks. DI/FCUL TR 05–23, Department of Informatics, University of Lisbon, 2005.

[8] Hugo Miranda, Simone Leggio, Luís Rodrigues, and Kimmo Raatikainen. A power-aware broadcasting algorithm. In *Proc. of The 17th IEEE Symp. on Personal, Indoor and Mobile Radio Communications (PIMRC'06)*, 2006.

[9] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile networks and applications*, 8(4):427–442, 2003.

[10] Liangzhong Yin and Guohong Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, 2006.