Load Balancing Hashing in Geographic Hash Tables

M. Elena Renda, Giovanni Resta, and Paolo Santi

Abstract—In this paper, we address the problem of balancing the network traffic load when the data generated in a wireless sensor network is stored on the sensor node themselves, and accessed through querying a geographic hash table. Existing approaches allow balancing network load by changing the geo-routing protocol used to forward queries in the geographic hash table. However, this comes at the expense of considerably complicating the routing process, which no longer occurs along (near) straight-line trajectories, but requires computing complex geometric transformations.

In this paper, we demonstrate that it is possible to balance network traffic load in a geographic hash table without changing the underlying geo-routing protocol. Instead of changing the (near) straight-line geo-routing protocol used to send a query from the node issuing the query (the source) to the node managing the queried key (the destination), we propose to "reverse engineer" the hash function used to store data in the network, implementing a sort of "load-aware" assignment of key ranges to wireless sensor nodes. This innovative methodology is instantiated into two specific approaches: an analytical one, in which the destination density function yielding quasi-perfect load balancing is analytically characterized under uniformity assumptions for what concerns location of nodes and query sources; and an iterative, heuristic approach that can be used whenever these uniformity assumptions are not fulfilled. In order to prove practicality of our load balancing methodology, we have performed extensive simulations resembling realistic wireless sensor network deployments showing the effectiveness of the two proposed approaches in considerably improving load balancing and extending network lifetime. Simulation results also show that our proposed technique achieves better load balancing than an existing approach based on modifying geo-routing.

Index Terms—Geographic hash tables; load balancing; wireless sensor networks; in-network data storage; network lifetime.

1 Introduction

Geographic hash tables [15] have been recently proposed as an approach for effectively retrieving data from a wireless sensor network when sensed data is stored on the sensor nodes themselves (in-network data storage). More specifically, in ght approaches[1], each sensor node is assigned a value in a certain range (for instance, the two-dimensional real interval $[0, 1]^2$) by hashing its coordinates in the deployment region; also, each sensed data is tagged with a meta-data used for retrieval purposes, and meta-data is hashed to a key value within the same range used for mapping node coordinates through a properly designed hash function. Keys are then stored at the sensor nodes – in general, at a sensor node different from the one which generated the data – based on a geographic proximity criterion, and overlay links in the distributed hash table are actually collapsed to the underlying physical, wireless links thanks to the use of geo-routing. In geo-routing (see, e.g., [2], [11]), a message is routed towards a specific geographic location (x, y) instead of a specific destination node, and it is typically delivered to the node whose key is closest to destination point (x, y).[2] Thus, routing a query for a specific data in a ght is a very simple task: if query for a certain key $k = (x_k, y_k)$ is generated at a certain node $u = (x_u, y_u)$, node u simply generates a message setting (x_k, y_k) as the destination point, and the query will be delivered to the node whose ID is closest to (x_k, y_k), which is in charge of storing the data with key k. Geographic hash tables find application not only in wireless sensor networks [15], but also in P2P resource sharing in wireless mesh networks [3], [6].

Despite the fact that ghts have been proved effective in retrieving data in large scale wireless sensor networks, they suffer from problems affecting distributed hash tables in general, a major of which is imbalanced usage of storage and/or network resources. Balancing storage resources (number of stored keys per node) is important in sensor networks due to the limited memory size of sensor nodes. Imbalance in network traffic load instead has a negative effect on network lifetime, since energy-consuming transmit/receive operations are not evenly spread amongst network nodes. This explains why researchers have recently proposed techniques to improve storage and load balancing in ghts. In this paper, we are concerned with the load balancing problem in ghts, and we do not consider storage balancing – see [1] and references therein for ght storage balancing techniques.

In ghts, load imbalance can occur mainly due to the two following reasons. First, ght approaches are typically designed assuming that nodes are uniformly distributed in the deployment region, leading to severe imbalance in case nodes are concentrated around some locations and/or "coverage holes" in the deployment region occur

The authors are with Istituto di Informatica e Telematica del CNR, Pisa, Italy.

1. To avoid confusion, in the following we use *ght* to denote a generic geographic hash table approach, and *GHT* to denote the specific ght approach proposed in [15].

2. For simplicity, in this paper we consider a two-dimensional wireless sensor network deployment.

[16]. Even if nodes are uniformly distributed, load imbalance still occurs due to the well-known fact that geo-routing selects (near) straight-line trajectories, causing network traffic to concentrate around the center of the deployment region [4], [9].

A possible way of lessening the load imbalance problem in ghts is to modify the underlying geo-routing protocol, so that packets no longer travel along (near) straight-line trajectories. Approaches such as the ones proposed in [12], [14], [16] can be used to this purpose. However, changing the geo-routing protocol comes at a price. First, a customization of the routing protocol for the purpose of ght would be needed, which might entail considerable implementation efforts in those scenarios where other applications share the same routing protocol, and changes in the routing protocol might impact on upper layers and existing applications. Note that while implementing different routing protocols for different applications is in principle possible, this would negatively impact the memory footprint of the routing layer, and it is thus not advisable in wireless sensor networks. Second, a common feature of the load balancing geo-routing protocols mentioned above is that they require computation of complex geometric transformations, which are used to map the physical space into a virtual space in which the routing process takes place. Thus, current load balancing approaches can be impractical in application scenarios where the nodes composing the geographic hash table have limited computational power and memory resources, as it is typically the case in wireless sensor networks.

Given the above discussion, a question that arises is the following: *is it possible to achieve load balancing in a geographic hash table without changing the underlying straight-line geo-routing protocol?* In this paper, we give a positive answer to this question, presenting a novel methodology to address the load balancing problem in ghts: instead of changing the geo-routing protocol, we propose to "reverse engineer" the hash function so to reduce the load imbalance caused by straight-line geo-routing. The methodology is instantiated into two specific approaches. In a first, analytical approach the key idea is to characterize, for each point (x, y) in the deployment region, the desired *destination* probability density function (pdf), i.e., the probability that a node residing in (x, y) is the destination of a random query. By properly designing such destination density function, we formally prove that concentration of network traffic in the center of the deployment region can be avoided even in presence of straight-line geo-routing, and quasi-perfect load balancing can be achieved. Once the desired destination density has been characterized, the hash function can be "reverse engineered" so that the expected number of keys stored by a node (which, together with key popularity, determines its likelihood of being the destination of a query) results in the desired destination density.

The analytical approach is based, among others, on uniformity assumptions for what concerns node and source density[3]. When these assumptions are not met, we propose to use a different load balancing approach, based on an iterative heuristic that repeatedly changes key ranges assigned to nodes as long as a load balancing metric is improved.

Performance of our proposed approaches have been extensively evaluated through simulations resembling realistic wireless sensor network deployments. Simulation results show the effectiveness of our methodology in improving load balancing at the expense of only modestly increasing the overall network traffic (which is unavoidable when load balancing is sought [7]), even in presence of non-uniform node or source distribution. Despite increase in overall traffic, our technique is proved through simulation to substantially increase network lifetime, thanks to the more balanced spreading of network traffic among network nodes. Furthermore, simulation results show that our proposed technique provides better load balancing and longer network lifetime as compared to the load balancing approach of [12] based on modifying the geo-routing protocol.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces the network model and preliminary definitions. Section 4 describes our theoretical load balancing approach, while Section 5 introduces the HeLB heuristic to deal with those situations where uniformity assumptions on node and/or source distribution do not hold. Section 6 presents the results of our extensive simulation-based evaluation. Finally, Section 7 concludes the paper.

2 Related work

The problem of achieving load balancing in geographic hash tables in presence of non-uniform node distribution has been recently addressed in [16]. In case of inhomogeneous node distribution, some of the nodes (those close to the boundary of scarcely populated regions) tend to be overloaded in terms of the number of keys they are requested to store, resulting in highly imbalanced (storage) load. The authors of [16] propose to use complex geometric transformations to map the physical network deployment space into a virtual space in which node distribution is shown to be near-uniform. The ght abstraction (both key assignment and routing) is then realized on the virtual, instead of physical, space, thus considerably improving load balancing at the expense of increasing the overall network load. Increase of overall network load is due to the fact that geo-routing in the virtual space results in a trajectory in the physical space which is longer than a straight-line trajectory.

Other approaches [12], [14], originally proposed for geo-routing, can be exploited to improve load balancing in ght under the assumption of uniform node distribution in the physical space. The idea is to avoid the

3. Source density defines the probability, for each point (x, y) in the deployment region, that a node located in (x, y) is the source of a query.

concentration of network traffic in the center of the deployment region by performing geo-routing on a virtual space instead of on the physical space. In other words, a point (x, y) in the physical space is univocally mapped to a point (x_v, y_v) in the virtual space, and geo-routing is performed using virtual, instead of physical, coordinates. This way, straight-line trajectories in the virtual space are turned into non-linear trajectories in the physical space, thus improving overall load balancing. The difference between [12] and [14] is on the choice of the virtual space, which is a suitably defined distance-preserving symmetric space in [12], and a sphere in [14]. Similarly to [16], the price to pay is an increase of the overall network load, due to the fact that non-linear trajectories in the physical space are necessarily longer than straight-line ones. Indeed, it has been proven in [7] that this load balancing vs. total network traffic tradeoff cannot be avoided in geometric graphs.

Differently from existing approaches, our load balancing technique does not rely on a notion of virtual space, so no changes to the geo-routing protocol are required. Instead, we propose to modify the hash function design, so that the probability mass of the destination *pdf* is concentrated towards the border of the deployment region, assigning relatively more keys to manage to border nodes than to central ones. This probability mass concentration on the border has the effect of slightly increasing the average trajectory length, which in this case is not due to the fact that trajectories are not straight-lines as in [12], [14], [16], but to the fact that the expected distance between a randomly chosen query source and destination is relatively longer. Thus, our results confirm that the load balancing vs. total network traffic tradeoff described in [7] cannot be avoided in geometric graphs.

Another major advantage of our load balancing methodology with respect to existing techniques is versatility: throughout this paper, we describe how our ideas can be extended to deal with inhomogeneous query source density and arbitrary key popularity distribution. Furthermore, in the supplemental material we also describe other possible applications of our load balancing methodology in the fields of mobility modeling and security.

A further positive aspect of our approach is that, differently from existing proposals [12], [14], [16], our analytical approach allows a *theoretical characterization* of the expected overall network traffic increase due to load balancing. This theoretical characterization, which is proved to be very accurate based on extensive simulation results, can be used to properly tune the load balancing vs. total network traffic tradeoff at design stage. The possibility of properly tuning this tradeoff is very important, e.g., to extend wireless sensor network operational lifetime, which is determined by both the average node energy consumption (related to total network traffic) and the imbalance of node energy consumption (related to load balancing).

3 Network model and preliminaries

We consider a (infinitely dense) wireless sensor network whose nodes are located in an arbitrary two-dimensional convex region A. Network nodes implement a geographic hash table, which is used to perform in-network data storage as in [15]. A query in the geographic hash table abstraction is understood as the process of retrieving the data associated with a certain key possibly stored in the ght. The node which initiates the query process for a certain key k is called the *source node* in the following, denoted s; similarly, the node responsible for key k is called the *destination node*, denoted d. Similarly to [15], in the following we assume that a geographic routing protocol such as GPSR [11] is used to route the query from s to d. Given our working assumption of infinite node density, this is equivalent to assume that the query travels from s to d along a straight line. This is a quite standard assumption in the analysis of geographic routing protocols and hash tables [7], [9], [12], [14].

We define the following probability density functions on A:

- the *source density* $\mathbf{s}(x, y)$, denoting the probability density of having the source node s of a random query located at (x, y);
- the *destination density* $\mathbf{d}(x, y)$, denoting the probability density of having the destination node d of a random query located at (x, y);
- the *traffic density* $\mathbf{t}(x, y)$, denoting the probability density that a random query traverses location (x, y) on its route from node s to node d.

Based on functions $\mathbf{s}, \mathbf{d}$ and $\mathbf{t}$, we can define the *load density* $\mathbf{l}(x, y)$, denoting the total load density at location (x, y), as follows:

$$\mathbf{l}(x,y) = \frac{1}{a+b+c} \cdot (a \cdot \mathbf{s}(x,y) + b \cdot \mathbf{d}(x,y) + c \cdot \mathbf{t}(x,y)) \ ,$$

where a, b, c are constants representing the relative impact of the various density functions when computing the load at (x, y), and $1/(a + b + c)$ is the normalization constant of the density function. Informally, the load density can be understood as the density of transmitted messages for a node located at (x, y), which depends on the probability of being either the source or the destination of a query, or of being in the path from the source to the destination. It is important to observe that the traffic density $\mathbf{t}$, under our working assumption of straight-line routing, is indeed a functional $\mathbf{t}(x, y, \mathbf{s}, \mathbf{d})$, i.e., given densities $\mathbf{s}$ and $\mathbf{d}$, the traffic density $\mathbf{t}$ can be computed, e.g., using the approach of [9] (see next section).

Note that, while the source density $\mathbf{s}$ depends on parameters such as node locations and data query patterns and can be considered as an input to the load balancing problem, the destination density $\mathbf{d}$ depends on factors such as the number of keys managed by a node located at (x, y), and/or their popularity. The key observation to our approach is that, while relative key popularity

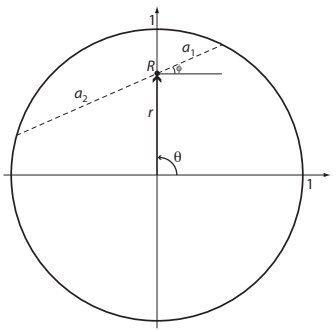

Fig. 1. Polar coordinate system on the unit disk, and definition of segments a_1, a_2.

is beyond control of the network designer, *the expected number of keys managed by a node located at (x, y) can actually be arbitrarily chosen in a ght design*. Our goal in the following is to "reverse engineer" the hash table design in such a way that the resulting destination density $\mathbf{d}$ is such that, when combined with the given source density $\mathbf{s}$ and the traffic density $\mathbf{t}$ resulting from $\mathbf{s}$ and $\mathbf{d}$, it produces a spatially uniform load density. Observe that implicit in our approach is the assumption that network designer is able to estimate the source density $\mathbf{s}$, i.e., the expected number of queries generated by a region of the deployment area A. Thus, our approach can be reasonably applied in situations where node positions are mostly fixed, and traffic patterns predictable, as it is the case in many wireless sensor network applications.

4 Load-balancing hash table design

4.1 Implicit destination density characterization

We start presenting a formal, implicit characterization of the density function $\mathbf{d}$ yielding uniform load. To start with, we need to compute the traffic distribution $\mathbf{t}$ for given source and destination densities $\mathbf{s}$ and $\mathbf{d}$. This can be done using a result in [9], which has been originally derived to characterize the stationary node spatial distribution of the RWP mobility model [10] with arbitrary waypoint distribution[4].

To keep the presentation simple, in the following we assume that the deployment region A is the unit disk[5]. Furthermore, we make the assumption that both distributions $\mathbf{s}$ and $\mathbf{d}$ are rotationally symmetric, i.e., the value of the density function at point (x, y) depends only on the distance of (x, y) from the origin. Given these assumptions, in the following we will make use of polar coordinates. In other words, we shall write

$$\mathbf{s}(r, \theta) = \mathbf{s}(r)$$

to denote the value of density function $\mathbf{s}$ (similarly, of density function $\mathbf{d}$) at the point located at distance r from the origin, and making an angle of θ with respect to the x-axis (see Figure 1).

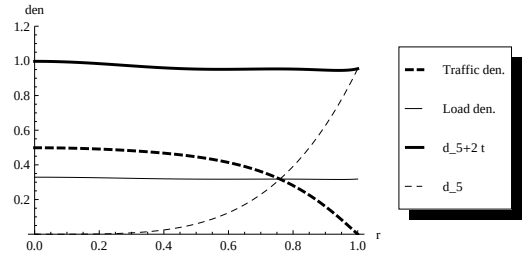

Fig. 2. Traffic and resulting load distribution with destination density function $\mathbf{d}_5$.

Let us fix a point R on the unit disk, and assume without loss of generality that R is located at $(0, r)$ ($(r, \pi/2)$ in polar coordinates). Denote by $a_1(r, \phi)$ the distance from R to the boundary of the disk along direction ϕ, and let $a_2(r, \phi)$ be the same distance along the opposite direction $\pi + \phi$ (see Figure 1). The values of $a_1(r, \phi)$ and $a_2(r, \phi)$ in the unit disk are as follows (see [9]):

$$\begin{aligned} a_1(r, \phi) &= \sqrt{1 - r^2 \cos^2 \phi} - r \sin \phi \\ a_2(r, \phi) &= \sqrt{1 - r^2 \cos^2 \phi} + r \sin \phi \ . \end{aligned}$$

For given source and destination densities $\mathbf{s}(r)$ and $\mathbf{d}(r)$, the resulting traffic density $\mathbf{t}(r)$ is equivalent to the density of random segments crossing point R, where the endpoints of the segments are randomly chosen according to densities $\mathbf{s}(r)$ and $\mathbf{d}(r)$, respectively. This latter density corresponds to the node spatial density of the nonuniform random waypoint process as defined in [9], and is given by:

$$\begin{aligned} \mathbf{t}(r, \mathbf{s}, \mathbf{d}) &= \frac{1}{E[\ell]} \int_0^{2\pi} d\phi \int_0^{a_2(r,\phi)} dr_2 \\ &\int_0^{a_1(r,\phi)} dr_1 (r_1 + r_2) \cdot \mathbf{s}(r_1, \phi) \cdot \mathbf{d}(r_2, \pi + \phi) \ , \end{aligned} \quad (1)$$

where $E[\ell]$ is the expected length of a random segment with endpoints chosen according to densities $\mathbf{s}$ and $\mathbf{d}$, and $\mathbf{s}(r_1, \phi)$ (respectively, $\mathbf{d}(r_2, \pi + \phi)$) is the source density (respectively, destination density) computed at point $R + r_1 \cdot (\cos \phi, \sin \phi)$ (respectively, at point $R + r_2 \cdot (\cos(\pi + \phi), \sin(\pi + \phi))$).

We are now ready to provide the implicit characterization of the destination density function $\mathbf{d}_u$ yielding uniform load:

Theorem 1: For a given rotational symmetric source density $\mathbf{s}$, the destination density function $\mathbf{d}_u$ yielding uniform load is a solution to the following integral equation:

$$\frac{1}{a + b + c} \cdot (a \cdot \mathbf{s}(r) + b \cdot \mathbf{d}_u(r) + c \cdot \mathbf{t}(r, \mathbf{s}, \mathbf{d}_u)) = \frac{1}{\pi} \ , \quad (2)$$

where $\mathbf{t}(r, \mathbf{s}, \mathbf{d}_u)$ is defined in (1).

Unfortunately, deriving a closed-formula expression for $\mathbf{d}_u$ is difficult, since (2) is a complex integral equation

4. In the following, equation (1) and the preceding definitions are taken from [9].

5. Up to tedious technical details, using the techniques in [9] our design can be extended to the case where A is an arbitrary convex region. Extension to non-convex shapes is highly non-trivial, since in this case linear routing paths cannot be used to connect source with destination nodes.

in non-standard form. However, and under the assumption of uniform source density $\mathbf{s}$, a hint on the shape of the desired destination density $\mathbf{d}$ can be derived as follows. If $\mathbf{s}$ is the uniform distribution, the density function on the left hand side of equation (2) becomes the sum of three components, one of which is uniform. Then, in order for the l.h.s. of equation (2) to become uniform distribution, we must have:

$$b \cdot \mathbf{d}(r) + c \cdot \mathbf{t}(r, \mathbf{s}, \mathbf{d}) = k \ \forall r \in [0,1] \ ,$$

for some constant $k > 0$. This situation is well explained in Figure 2 for the case of $a = b = 1$ and $c = 2$, corresponding to the small data case (see next section). The figure reports the $\mathbf{d}_5$ distribution – defined in the next section – for destinations (dashed plot), and the resulting traffic density $\mathbf{t}$ (thick dashed plot). The key observation is that, while density $\mathbf{t}$ *is not* uniform, function $\mathbf{d}_5(r) + 2 \cdot \mathbf{t}(r)$ is almost constant in the range $r \in [0,1]$ (thick black plot), yielding a near uniform load distribution $\mathbf{l}$ (black plot).

In the next section, we will show that, given its versatility in terms of possible shapes of the distribution, the family of Beta distributions can be used to closely approximate $\mathbf{d}_u$ under the assumption that $\mathbf{s}$ is the uniform density.

4.2 Explicit destination density characterization with uniform sources

Since attempting to directly solve integral equation (2) to derive $\mathbf{d}_u$ is difficult, an alternative approach is trying to "guess" a close approximation of $\mathbf{d}_u$ driven by the observation at the end of Section 4.1, using a suitably chosen family of candidate density functions. A suitable family of candidate functions are the Beta distributions. A member of this family is a probability density function in the $[0,1]$ interval defined based on two parameters $\alpha, \beta > 0$, called the *shape parameters*, as follows:

$$\mathbf{B}(x, \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1} \ ,$$

where $B(\alpha, \beta)$ is the Beta function, used as a normalization constant, and is defined as follows

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt \ .$$

The family of Beta distributions is very appealing since, by varying the shape parameters, the probability mass can be shifted almost arbitrarily from the left to the right of the $[0,1]$ interval (see Figure 3).

In order to have some hints on which Beta distributions are good candidates for approximating $\mathbf{d}_u$, we first compute the load density $\mathbf{l}$ under the assumption that both source and destination density are uniform. Under this assumption, the expression for the traffic density $\mathbf{t}$ can be simplified as follows:

$$\mathbf{t}(r, \mathbf{s}, \mathbf{d}) = \frac{1}{\pi^2 E[\ell]} \int_0^{2\pi} d\phi \int_0^{a_2(r,\phi)} dr_2 \int_0^{a_1(r,\phi)} dr_1 (r_1 + r_2) \ ,$$

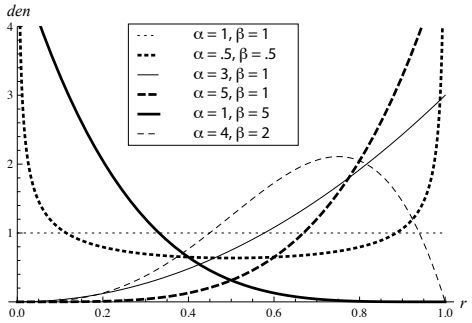

Fig. 3. Beta distribution for different values of the shape parameters.

Observe that an explicit formula for $\mathbf{t}(r)$ cannot be obtained even for this relatively simple case, since the integral in $\mathbf{t}(r)$ is an elliptic integral of the second kind, which cannot be expressed in elementary functions [9]. Hence, we have to resort to numerical integration to compute $\mathbf{t}(r)$. For definiteness, in computing density $\mathbf{l}$ we will consider two different settings for parameters a, b, c: $i)$ $a = b = 1$ and $c = 2$; and $ii)$ $a = 0$, and $b = c = 1$. Case $i)$ corresponds to a situation in which the ght is used to retrieve the address of the data holder as in [3], [6], or to retrieve small data from a certain location using, e.g., GHT [15]. In fact, in this situation the load at a certain location can be computed considering that a node transmits once if it is the source or the destination of a query, and it transmits two messages if it lies on the route between source and destination (one message for forwarding the query to d, and one message for returning the response to s). In the following, we will call case $i)$ the *small data* case. Case $ii)$ models situations where large amounts of data must be transmitted back from d to s in response to a query, in which case the load induced by being the source of a query is negligible compared to the load generated by transmitting a large amount of data from d back to s. In the following, we will call case $ii)$ the *large data* case.

The cross section of the load density resulting in case of uniform source and destination distribution for the small and large data case is reported, e.g., in Figure 4 ($d = unif$ plot). For comparison, the uniform load distribution is also reported. It is easy to see that, when both $\mathbf{s}$ and $\mathbf{d}$ are the uniform distribution, the load density for small and large data is the same, since in both situations the load density is composed of a uniform and a non-uniform component with the same relative weight. From the figure, it is seen that, as expected, when source and destination density are uniform, nodes in the center of the region observe a much higher load than those near the border. In order to compensate for this load concentration near the center, it is reasonable to change the destination distribution $\mathbf{d}$ in such a way that nodes relatively closer to the border are selected relatively more often as destinations of a query.

Driven by this observation, we have computed the load distribution resulting when source density $\mathbf{s}$ is uniform, and destination density is one of the two following

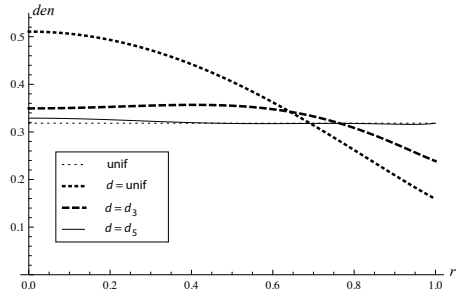

Fig. 4. Load density cross-section with different destination densities in the small data case.

Dest. density	$M_{\mathbf{l}}$	$Mm_{\mathbf{l}}$	MSE
unif.	0.51	3.21	$1.05 \cdot 10^{-2}$
$\mathbf{d}_3$	0.36	1.49	$7.07 \cdot 10^{-4}$
$\mathbf{d}_5$	0.33	1.04	$6.62 \cdot 10^{-6}$

TABLE 1
Load balancing metrics in the small data case.

Beta distributions $\mathbf{d}_3 = \mathbf{B}(3,1)$ and $\mathbf{d}_5 = \mathbf{B}(5,1)$ (the thin solid and thick dashed lines in Figure 3), which have been chosen since they tend to concentrate the probability mass towards the border of the disk. Note that the Beta distributions must be suitably normalized in order to ensure that their integral on the unit disk is 1. Hence, $\mathbf{d}_3$ and $\mathbf{d}_5$ are defined as follows:

$$\mathbf{d}_3(r) = \frac{2}{\pi} r^2 \ , \ \mathbf{d}_5(r) = \frac{3}{\pi} r^4 \ .$$

The load density cross-section obtained with destination densities $\mathbf{d}_3$ and $\mathbf{d}_5$ in the small data case are reported in Figure 4. As seen from the figure, setting the destination density to $\mathbf{d}_5$ yields a load distribution which is virtually indistinguishable from uniform. To assess uniformity of the various load densities, we have used different metrics, which are summarized in Table 1 for the small data case: the maximum load $M_{\mathbf{l}}$, the ratio $Mm_{\mathbf{l}}$ of the maximum to the minimum load, and the Mean Square Error (MSE) computed with respect to the uniform distribution. More specifically, the MSE for density $\mathbf{l}$ is computed as

$$MSE = \frac{1}{\pi} \int_A \left(\mathbf{l}(r) - \frac{1}{\pi} \right)^2 d^2 r \ .$$

As seen from Table 1, a proper design of the destination density (i.e., of the hashing function) has the potential to yield quasi-perfect load balancing: with respect to the case of uniformly distributed destinations, the maximum load is reduced of about 35%, the ratio $Mm_{\mathbf{l}}$ is reduced of a factor 3, and the MSE with respect to uniform load is improved of 4 orders of magnitude.

The load density cross-section obtained with destination densities $\mathbf{d}_3$ and $\mathbf{d}_5$ in the large data case are reported in Figure 5, with corresponding uniformity metrics reported in Table 2. As seen from the figure and the table, in this case destination density $\mathbf{d}_5$ turns out to be too concentrated on the border, leading only to a minor load balancing improvement over the case of

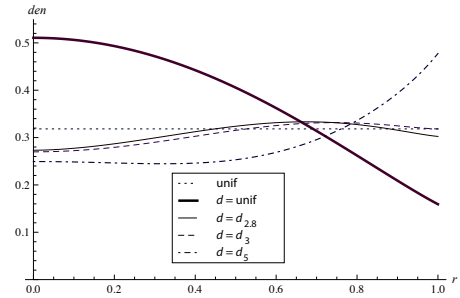

Fig. 5. Load density with different destination densities in the large data case.

Dest. density	$M_{\mathbf{l}}$	$Mm_{\mathbf{l}}$	MSE	exp. length
unif.	0.51	3.21	$1.05 \cdot 10^{-2}$	0.90541
$\mathbf{d}_{2.8}$	0.33	1.22	$9.48 \cdot 10^{-5}$	0.97718
$\mathbf{d}_3$	0.33	1.23	$1.17 \cdot 10^{-4}$	0.98302
$\mathbf{d}_5$	0.48	1.91	$4.85 \cdot 10^{-3}$	1.02121

TABLE 2
Load balancing metrics and expected path length in the large data case.

uniformly distributed destinations. On the other hand, destination density $\mathbf{d}_3$ yields a considerable load balancing improvement (slightly better than that obtained in the small data case), with a 35% reduction of $M_{\mathbf{l}}$ with respect to the case of uniform destination density, and a near three-fold reduction in $Mm_{\mathbf{l}}$. However, the MSE with respect to uniform load, although reduced with respect to the small data case, remains in the order of 10^{-4}, i.e., two orders of magnitude worse than the best performing destination density in case of small data. Indeed, a more accurate fine tuning of the α parameter of the Beta distribution leads to a slightly more uniform load distribution. This is obtained using the following destination density: $\mathbf{d}_{2.8} = 0.6048 \cdot x^{1.8}$. With this destination density, the MSE with respect to uniform load density is reduced to $9.48 \cdot 10^{-5}$ (see Table 2).

Although in our approach load balancing is achieved without changing geo-routing straight-line trajectories, a certain increase in overall network load can be expected (in accordance with the well-known tradeoff between load balancing and overall network load [7]). This is due to the fact that, although queries are still routed along the shortest path from source to destination, the *average length* $E[\ell]$ of a path (i.e., the normalization constant in the traffic distribution (1)) increases as the load distribution becomes more balanced. The last column of Table 2 reports the expected path length for the various destination distributions. Note that the expected path length is determined by the source and destination densities $\mathbf{s}$ and $\mathbf{d}$, and is not influenced by the choice of the weights a, b, c used in computing the load density. Hence, the average path length obtained with a certain destination density function is the same in both small and large data cases. It is interesting to observe that the expected path length increase (i.e., overall network load increase) for the most uniform load density with respect to the case of uniformly distributed sources and

destinations is very limited, and amounts to less than 13% in the small data case, and to less than 8% in the large data case.

4.3 Hash function design

In the previous section, we have characterized destination densities yielding close to uniform load density in case of both small and large data. As outlined in Section 3, a further step is needed in order to define the hash function for a given desired destination density.

Traditionally, hash functions are designed to map a certain domain $\mathbb{D}$ (e.g., file names in a P2P file sharing application, meta-data description in an wireless sensor network in-network storage system, etc.) into a real number in the $[0,1)$ interval, i.e.:

$$\mathcal{H} : \mathbb{D} \rightarrow [0,1) \ ,$$

where $k = \mathcal{H}(D)$ is the *key* associated with element $D \in \mathbb{D}$. Similar to other distributed hash table approaches, in geographic hash table designs [3], [15] the nodes realizing the ght are assigned a subset of the $[0,1)$ interval to manage, called their *key range*. Note that in some ght designs such as GHT [15], the hash function indeed maps the domain $\mathbb{D}$ into the $[0,1)^2$ interval; however, through straightforward geometric transformations, the $[0,1)^2$ interval can be mapped into the $[0,1)$ interval preserving the hash function properties. Thus, to keep presentation simple, in the following we assume the co-domain of the hash function is the $[0,1)$ interval, and we assume the key range $kr(u)$ of a specific node u is a sub-interval of $[0,1)$. Denoting with N the set of network nodes, we have:

$$\bigcup_{u \in N} kr(u) = [0,1) \text{ and } \forall u \neq v \in N, kr(u) \cap kr(v) = \emptyset \ .$$

In ght designs, the key range $kr(u)$ of a specific node u depends on its geographic coordinates, and those of its geographical neighbors. We now present a method for assigning key ranges to a set of nodes N of known coordinates such that, when coupled with an arbitrary hash function $\mathcal{H}$ with co-domain in $[0,1)$, the resulting destination density equals a certain given function $\mathbf{d}(x,y)$. For simplicity, we first present the method under the assumption that the probability density function $\mathbf{q}(k)$, with $0 \leq k < 1$, corresponding to the probability that a specific key is the argument of a random query, is the uniform distribution over $[0,1)$ (uniformly popular keys). We then describe how to generalize our method to non-uniform key popularity distributions.

Given the set of nodes N deployed in a bounded region A, we first compute the Voronoi diagram on the nodes, and let $V(u)$ be the Voronoi cell of node u – i.e., the locus of points in A closer to u than to any other node in $N - \{u\}$. For each $u \in N$, the width $w(u)$ of the key range $kr(u)$ is given by

$$w(u) = \int_{V(u)} \mathbf{d}(x,y) dxdy \ , \tag{3}$$

where $\mathbf{d}(x,y)$ is the desired destination density. We now order the nodes in N according to their geographical locations, e.g., starting from the Southern-Western node (the specific ordering used is not relevant). Let $u_1, \ldots, u_n$ be the resulting ordered list of nodes, where $n = |N|$. We then set the key range $kr(u_i)$ for node u_i, with $i = 1, \ldots, n$, as follows:

$$kr(u_i) = [a_i, b_i) \ , \text{ where } a_i = \sum_{j=1}^{i-1} w(u_j) \text{ and } b_i = a_i + w(u_i) \ .$$

Fact 1: Let p_i denote the probability that the key requested in a random query belongs to key range $kr(u_i)$, and assume $\mathbf{q}(k)$ is the uniform distribution over $[0,1)$. Then,

$$p_i = \int_{a_i}^{b_i} \mathbf{q}(k) dk = w(u_i) = \int_{V(u_i)} \mathbf{d}(x,y) dxdy \ ,$$

where $V(u_i)$ is the Voronoi cell of node u_i and $\mathbf{d}$ is the desired destination density.

By Fact 1 and by observing that $\sum_{u_i \in N} area(V(u_i)) = area(A)$, we have that the destination density resulting from the above described key range assignment is the discrete counterpart of the desired density function $\mathbf{d}$.

If $\mathbf{q}(k)$ is a generic probability density function, deriving the key range assignment is more complex, as it involves solving the following Volterra integral equation [13]:

$$\int_{a_i}^{y} \mathbf{q}(k) dk = \int_{V(u_i)} \mathbf{d}(x,y) dxdy \ . \tag{4}$$

Given the starting point $a_i = \sum_{j=1,\ldots,i-1} w(u_j)$ of key range $kr(u_i)$, the end point y of $kr(u_i)$ (and, consequently, $w(u_i)$) can be computed solving the above Volterra integral equation. By iteratively solving equation (4) starting from $i = 1$, we can compute all the key ranges $kr(u_i)$, and generalize Fact 1 to hold for arbitrary key popularity distributions.

An important feature of our proposed hash function design lies in its practicality. In particular, we want to stress that the hash function can be computed in a fully distributed and localized way, thus according to typical wireless sensor network design guidelines. This is because, if node density is high enough – a prerequisite for our approach to be effective, each node can compute its Voronoi cell by exchanging location information with its immediate neighbors. For this to happen, it is sufficient to have the nodes' transmission range larger than the maximum distance between two Voronoi neighbors. In case of uniformly distributed nodes, it is easy to see that the minimal density required for connectivity w.h.p. [8] is sufficient to ensure that the Voronoi graph can be locally computed. Furthermore, re-computation of a Voronoi cell in response to changes in network topology (for instance, because a sensor node runs out of energy) can also be easily done in a fully distributed, localized way. As for computing the key range of a node, say u, the integral in equation (3) can be numerically approximated by, e.g., pre-computing and storing at u a lookup table

containing the value of the integral in a sufficiently fine lattice around u (say, a lattice covering u's transmission range); the value of the integral in equation (3) can then simply be reconstructed on-the-fly by adding the values stored in the lookup table corresponding to lattice elements at least partially included in the Voronoi cell $V(u)$. Considering that, e.g., a 256 elements lattice is used to compute the key range, and assuming the value of the integral on a lattice element is represented with 8 bytes, the total size of the lookup table would be 1.25KB, which represents only a small portion of the memory size typically available on wireless sensor nodes.

It should be observed, though, that the hash function design described herein relies upon some global knowledge, namely a total ordering of nodes which is assumed to be known to all nodes and is used to (locally) compute a node's key range. Changes in node ordering due to a topology change can be initiated by the base station once the topology change has been detected. The amount of messages to be sent in order to distribute the new ordering of nodes is comparable to the cost of a single broadcast message, i.e. it is $O(n)$ in a network with n nodes. In general, our proposed approach is suitable to those scenarios where node locations are mostly fixed and node population changes at a relatively slow rate, as it is the case in many wireless sensor network applications.

5 Non-uniform distributions

The analytical approach presented in Section 4 is based on three assumptions, namely that: $i)$ nodes are uniformly distributed, $ii)$ source nodes are uniformly selected among the network nodes, and $iii)$ node density is very high. In practice, though, these assumptions might not be met, especially the uniformity assumptions on node and source distribution. In this section, we release these two uniformity assumptions, and present an iterative heuristic, named HeLB (Heuristic Load Balancing), aimed at modifying each node's key range in order to balance network load even when assumptions $i)$ and $ii)$ do not hold. The basic idea of HeLB is to start by observing the network load resulting with uniform key ranges, and to iteratively re-arrange key assignment in order to reduce the range of keys managed by overloaded nodes, while increasing the number of keys managed by nodes with a lower load.

With a slight abuse of notation, let $kr(i)$ denote the length of the key range of node i. Under the assumption of uniform key popularity, that we retain in this section, the value of $kr(i)$ determines the average number of queries managed by node i[6]. Initially, nodes are assigned with key ranges of the same length, i.e. initially $kr_i = 1/n,\ \forall i \in N$, where $n = |N|$ is the number of nodes in the network. At each iteration, the procedure consists of the following phases:

1) a large number of queries is spread out in the network;
2) for each node $i \in N$, the average load per-query $c(i)$ is computed;
3) key range $kr(i)$ is modified according to the heuristic method specified below, in order to reduce the load on overloaded nodes and increase the load on under-loaded nodes;
4) compute the max-min load ratio Mm_{l}.

The procedure stops when the max-min load ratio Mm_{l}, which is initialized at an artificially high value, stops decreasing.

The heuristic method adopted at step 3) of HeLB modifies the key range of each node as follows:

$$kr(i) = \max\{0, kr(i) + ((\bar{c} - c(i)) * k)\}\ \forall i \in N, \qquad (5)$$

where $\bar{c}$ is the average of the $c(i)$ values, i.e. $\bar{c} = \sum_{i \in N} c(i)/N$, and k is a small constant, used to control the sensitivity of the load balancing procedure. The $kr(i)$ values are then normalized in order to obtain $\sum_{i \in N} kr(i) = 1$. Note that with the above rule the key range length of overloaded nodes (those with $c(i) > \bar{c}$) is *decreased*, while that of underloaded nodes (those with $c(i) < \bar{c}$) is *increased*.

In the next Section and supplementary material, we will show through extensive simulations that the proposed heuristic performs well in case of both non-uniform node and non-uniform query source distribution.

Before ending this section, we want to comment about feasibility of the proposed HeLB heuristic. In principle, two approaches can be used to implement HeLB: a simulation-based approach, as reported in the next section; and a fully distributed, adaptive approach. In the former approach, appropriate key range values that balance the load are estimated through simulations: a prerequisite for this approach is that information about expected node spatial distribution, source distribution, and key popularity are available before actual network deployment. This is indeed quite often the case in wireless sensor networks, where node locations and data transfer patterns are quite predictable at the design stage. In case the above information is not available, a more cumbersome, yet feasible approach consists in initially setting uniform key ranges and start operating the network; as more and more queries travels within the network, load statistics are communicated to a centralization point (e.g., the base station), which is in charge of re-computing the key ranges according to the rule defined above. New key range values are then broadcast to all nodes in the network. Notice that, once key range values are changed, transfer of data items between nodes is needed in order to let each node store all data items in its key range. This is an additional burden on the network; however, if the hash function

6. More formally, this fact follows from the Law of Large Numbers, after that a sufficiently high number of queries circulates in the network.

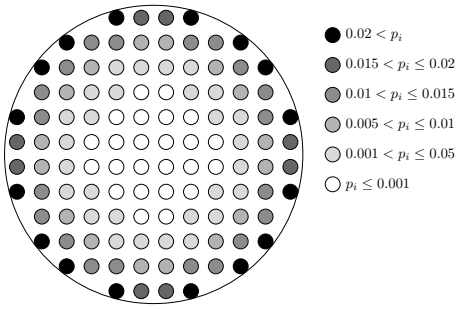

Fig. 6. Node deployment and discretized destination distribution $\mathbf{d}_5$ used in the simulations.

is suitably defined (see, e.g., [3]), data transfer occurs only between nearby nodes in the network.

6 Simulations

In this section, we evaluate the two approaches for load balancing hash function design described in sections 4 and 5 through simulation. In particular, in subsection 6.1 we evaluate performance of the analytical approach to hash function design described in Section 4, while in subsection 6.2 we evaluate performance of the HeLB heuristic.

6.1 Analytical hash function design

The analytical approach to load balancing hash function design described in Section 4 is based, among others, on the two following assumptions: i) infinite node density, and ii) straight line trajectory between source and destination nodes. In practical scenarios, node density is finite – although it might be very high in, say, dense wireless sensor network deployments, and the trajectory followed by a message on its route from source to destination is typically a piecewise linear trajectory approximating the straight line connecting source with destination.

In order to evaluate the impact of relaxing assumptions i) and ii) on the load balancing achieved by our hash design approach, we have performed extensive simulations resembling realistic wireless sensor network deployments. More specifically, we have considered two deployment scenarios: a) grid-based; and b) random.

In the grid-based scenario, sensor nodes are arranged in a grid-like fashion covering a disk of a certain radius. For convenience, we set to 1 the step of the grid, i.e., we normalize distances with respect to the grid step. A varying number of nodes (ranging from 112 to 1020) is deployed in the disk (the 112 nodes deployment is reported in Figure 6). In the random scenario, a certain number of nodes is randomly distributed in a disk, whose radius is set to 1 for convenience.

Communication links between nodes are established based on their distance, as follows: there is a wireless link connecting nodes u and v if and only if their distance is at most r, where r is the radio range and is a simulation parameter. In the following, when referring to node density, we mean the (average) number of neighbors of a node in the network.

Messages are routed from source to destination according to geo-routing, and more specifically using the GPSR protocol [11]: a message M with final destination d currently processed at node u is forwarded to u's one-hop neighbor whose distance to d is smaller[7].

Both the small data and the large data case are considered in our simulations. In the former case, once source and destination nodes s, d are selected, a (query) message is sent from s to d, and a (reply) message is sent back from d to s. In the latter case, we ignore the single (query) message sent from s to d, and we simply send 100 (data packet) messages from d back to s. In both the small and large data case, the source node is chosen uniformly at random among the network nodes, while the destination node is chosen according to one of the following distributions: 1) uniform; 2) $\mathbf{d}_{2.8}$; and 3) $\mathbf{d}_5$. Indeed, both $\mathbf{d}_{2.8}$ and $\mathbf{d}_5$ are the discretized versions of the distributions described in Section 4.2, and are computed according to the procedure described in Section 4.3: we first compute the Voronoi diagram on the nodes, and then, for each node, compute its key range width integrating the destination distribution in the respective Voronoi cell. As an example, the discretized version of the $\mathbf{d}_5$ distribution for the 112 nodes grid deployment is reported in Figure 6.

Grid deployment

In a first set of simulations, we have fixed the radio range to 1.5 (corresponding to connecting each node to its vertical, horizontal, and diagonal neighbors in the grid), and varied the number n of nodes from 112 to 1020. This is the minimal density scenario considered in our experiments, corresponding to each node having approximately 8 neighbors. For each considered topology, we have randomly generated 10^6 (query) messages, and computed the following metrics: i) maximum node load (number of transmitted messages); ii) maximum to minimum node load ratio; and iii) average hop count of the source/destination paths.

The small and large data cases returned quite similar results. In the interest of space, in the following we report only the results for the small data case, while those for the large data case are reported in the supplementary material.

As shown in Figure 7, distribution $\mathbf{d}_5$ is very effective in improving load balancing with respect to uniform destination distribution, reducing the maximum load of about 20%, and reducing the max/min ratio of a factor about 2.5. This comes at the expense of increasing the average hop count (and, hence, the overall network load) of about 13%. It is interesting to observe that the simulation results matches very well with theoretical analysis, which predicted a 35% max load reduction, a 3-fold max/min load ratio reduction, and a 13% average

7. Indeed, we have implemented only the greedy forwarding step of [11], since local minima – leading to deadlock in message forwarding – cannot occur in the considered grid-like deployments, and occurs with negligible probability in case of random deployments of the density considered herein [17].

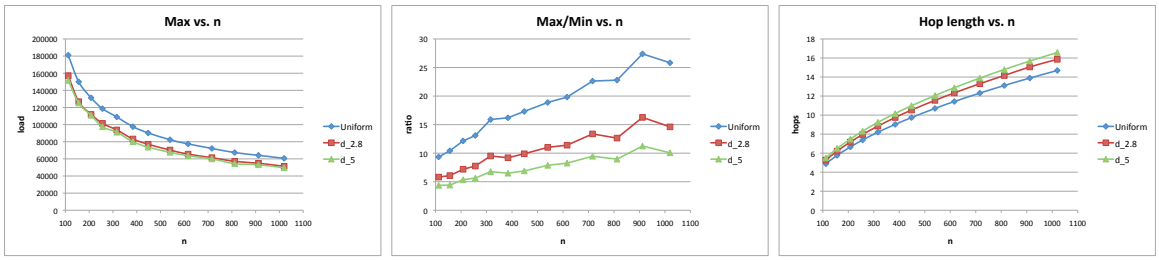

Fig. 7. Grid-based topology with fixed density: maximum node load (left), max to min node load ratio (center), and average hop count (right) in the small data case.

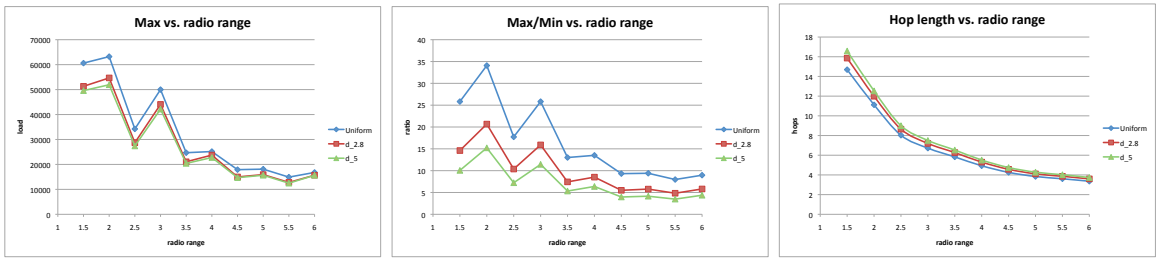

Fig. 8. Grid-based topology with $n = 1020$ and varying density: maximum node load (left), max to min node load ratio (center), and average hop count (right) in the small data case.

trajectory length increase when distribution $\mathbf{d}_5$ is used instead of the uniform distribution as the destination density. Distribution $\mathbf{d}_{2.8}$ turns out to be less effective in balancing load in both the small and large data cases (see supplementary material), partially contradicting our analysis that indicates that distribution $\mathbf{d}_{2.8}$ should indeed yield better load balancing than $\mathbf{d}_5$ in the large data scenario. This is due to the fact that the node density considered in this first set of experiments is quite low (close to the minimum density needed for connectivity), while the analysis is based on the infinite node density assumption.

This observation is validated by the results of another set of simulations, in which we have kept the number of nodes fixed to 1020, and varied the radio range (hence, node density) from 1.5 to 6 in steps of 0.5. When the transmission range is set to 6, nodes in the center of the grid have 60 neighbors. Also in this case we have generated 10^6 queries for each setting of the transmission range. As seen from Figure 8, the results confirmed that $\mathbf{d}_5$ is the best performing destination distribution in the small data case, yielding a max load reduction of about 7-20%, and a max/min load ratio reduction of about 50-60%, at the expense of a hop count increase of about 11-13%. However, in the large data case results have shown that, as the node density increases, distribution $\mathbf{d}_{2.8}$ achieves a better load balancing as compared to $\mathbf{d}_5$. In particular, when node density is maximal, $\mathbf{d}_{2.8}$ yields a lower maximum load and a slightly smaller max/min load ratio than $\mathbf{d}_5$, as predicted by the analysis (see supplementary material). When compared to uniform destination distribution, $\mathbf{d}_{2.8}$ reduces maximum load of 5-15%, and the max/min load ratio of about 50%.

Random deployment

In the second set of simulations, we have randomly deployed 1020 nodes as follows: we have first divided the unit disk in 1020 square cells of nearly the same size (due to border effects), and then deployed one node uniformly at random in each of these cells. This Poisson-like node distribution is used to generate quite homogeneous, yet random, node spatial distributions. We have then empirically computed the critical transmission range for connectivity [8], i.e., the minimum value of the radio range yielding connected topologies with high probability, which turns out to be $r = 0.06$ in our experiments. We then generated 100 random node deployments, and for each of them computed different network topologies varying the radio range from r to $5r$. For each generated topology, we generate 10000 queries for either small and large data case.

The simulation results (averaged over the 100 deployments), reported in the supplementary material, outlined that the relative advantage of our load balancing approach over uniform destination density are less significant than in the grid deployment scenario. In particular, with the small data case (similar results with large data) and $\mathbf{d}_5$ destination distribution, we have a max load reduction of as much as 12%, and a reduction of the max/min load ratio of a factor as large as 2. However, with the largest node densities (radio range larger than $3r$), the benefits of our approach become less apparent. This is due to the fact that, under higher densities, the average hop count of source/destination paths becomes too low (below 3, which is lower than in case of grid-like deployments), implying that the impact of relaying traffic becomes relatively less significant than the load induced by being either the source or the destination of

a query.

6.2 HeLB performance

In this section, we evaluate HeLB performance under the assumption of uniformly distributed source and node density. Uniformity assumptions are retained in this section to allow comparison of our heuristic with the Outer Space load balancing approach proposed in [12], which requires uniformity of node deployment to properly operate. HeLB performance evaluation when uniformity assumptions are released is reported in the supplementary material.

In order to be able to directly compare HeLB with Outer Space, we have deployed 1000 nodes in the unit square uniformly at random: this is because the virtual space used in [12] to define Outer Space is defined starting from a square deployment region. We have then considered different values of the transmission range, ranging from 0.15 (minimal density for connectivity w.h.p.) to 0.45. For each value of the transmission range, 150 sample topologies were generated to produce statistically significant results. In the interest of space, we report results only for the small data case; very similar results were obtained in the large data case.

Performance of HeLB is compared both with Outer Space and with the standard ght approach in which key ranges are evenly distributed among nodes (called *uniform* in the following). Besides the three load balancing metrics considered in the previous set of experiments, we have also evaluated network lifetime under the three approaches.

Network lifetime is computed as follows. Initially, all nodes are assigned the same amount of energy units, set to 5000 in our experiments. As nodes take part in the query process, their energy level is decremented as follow: a unit of energy for each received message, and 1.4 units of energy for each transmitted message. Note that these values of energy consumption are in line with typical values of IEEE 802.15.4 radio transceivers [5]. When the first node in the network exhausts its battery, network operation is terminated, and the total number of queries served by the network is reported as the network lifetime – which, then, is expressed in terms of number of successfully served queries.

Notice that HeLB requires running preliminary simulations to estimate the correct values of the key ranges. In order to evaluate HeLB performance in a situation closer to reality, where real-world conditions are different from those found in simulations, the experiments performed to evaluate HeLB load balancing performance have been done using a slightly different network topology than the one used to compute the key ranges. In particular, the position of nodes in the network has been changed, re-locating each node uniformly at random in a circle of radius 0.025 centered at its previous position.

The load balancing metrics obtained using the various approaches (averaged over 150 experiments) are reported in Figure 9. As seen from the figure, HeLB resulted superior to Outer Space under all respects: the maximum load, max/min load ratio, and hop length with HeLB are always lower than with Outer Space. It is worth observing that the relative advantage of HeLB over Outer Space is more noticeable at lower node densities, indicating that our approach is very effective in improving load balancing also when the node density is relatively low. On the contrary, Outer Space requires higher node density to display some load balancing advantage over uniform key range assignment.

A similar trend is observed for what concerns network lifetime – see Figure 10: HeLB increases the number of served queries of as much as 28% as compared to uniform key range assignment, and it provides noticeable lifetime benefits also at the minimal node density. On the contrary, at low densities Outer Space provides *lower* lifetime than uniform key range assignment, and lifetime improvements (as high as 23% at maximal density) can be noticed only when the transmission range is larger than 2 times the minimal value required for connectivity w.h.p.

The superior performance of HeLB with respect to Outer Space is due to the fact that, similarly to our analytical approach, Outer Space is based on the (implicit) assumption that geo-routing results in near straight-line trajectories. This assumption well approximates reality only when the node density is relatively high. In fact, Outer Space performance in [12] is evaluated under node density ranges higher than those considered in our study. Differently from Outer Space, HeLB is a heuristic which is not based on any specific assumption about the underlying routing protocol, but rather tries to "reasonably" re-arrange key distribution based on the currently observed load.

Summarizing, simulation results clearly show the advantage of load balancing in terms of more uniform energy consumption and, hence, longer lifetime: with HeLB, although the average hop count (hence, energy consumption) to serve a single query is *higher* than with uniform key range assignment, we do have a lifetime increase, since the rate at which energy is consumed at the most loaded node in the network is significantly reduced. This can be clearly seen from the load distribution reported in the Supplementary material, showing that the number of "highly loaded" nodes with HeLB is considerably reduced with respect to the case of uniform key distribution and Outer Space routing.

6.3 Discussion

Simulation results have shown that our proposed load balancing approaches can be successfully used also in practical situations, where the assumptions of infinite node density, perfect straight-line trajectory, as well as uniform node and source distribution do not hold. Clearly, the achieved load balancing is not quasi-perfect as the one achievable if assumptions on which the analytical technique presented in Section 4 would hold.

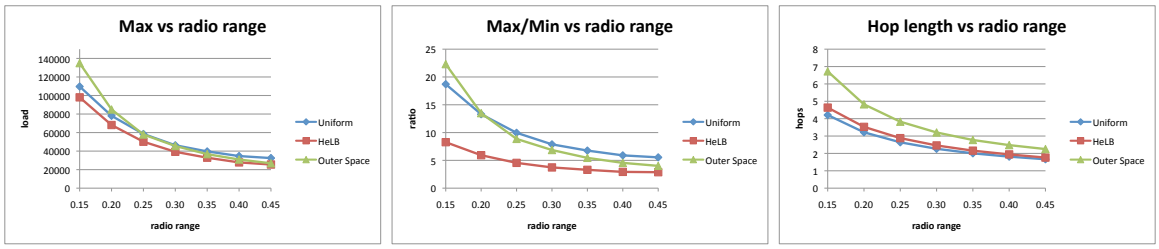

Fig. 9. Square topology with $n = 1000$ and varying density: maximum node load (left), max to min node load ratio (center), and average hop count (right) in the small data case.

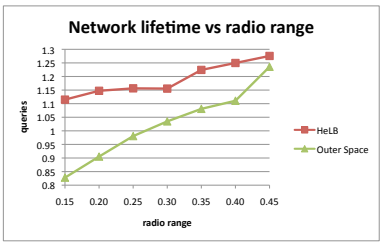

Fig. 10. Square topology with $n = 1000$ and varying density: network lifetime in the small data case, normalized w.r.t. lifetime with uniform key range.

Yet, load balancing and network lifetime improvements are clearly visible also when node density is near the minimal needed to achieve connectivity.

It is interesting to compare the relative performance achieved by the analytical and heuristic load balancing approaches. The two approaches provide similar performance, with somewhat better performance figures provided by the HeLB heuristic. Notably, HeLB heuristic provides these interesting performance figures even in presence of non-uniformity in node deployment or source distribution (see supplementary material).

Another interesting observation is that, from a quantitative point of view, our approach in practical scenarios achieves better load balancing and network lifetime than an existing approach [12]. Even more notably, our approach *improves load balancing while preserving simplicity of geo-routing, as well as of the hash function design*. Thus, the load balancing methodology presented in this paper has the potential of being readily applicable in a wireless sensor network scenario.

7 Conclusions

In this paper, we have presented a novel way of approaching the load balancing problem in ght design. The proposed methodology, instantiated into an analytical and a heuristic approach, has been shown to provide very good load balancing in ideal conditions, and to provide load balancing improvements comparable or even superior to those provided by existing schemes in practical scenarios, even when uniformity assumptions are not valid. The major advantage of the presented load balancing methodology over existing ones lies in its practicality and versatility. Possible extension of our ideas to other fields such as mobility modeling and security have been discussed.

For future work, we plan to investigate possible ways of extending and integrating our approach with existing ideas. In particular, one interesting direction for research is integrating our approach with the ideas proposed in [16] to address the case of concave shapes of the node deployment region.

References

[1] M. Albano, S. Chessa, F. Nidito, S. Pelagatti, "Q-NiGHT: Adding QoS to Data Centric Storage in Non-Uniform Sensor Networks", Proc. IEEE Conf. on Mobile Data Management, pp. 166–173, 2007.

[2] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks", *Wireless Networks*, Vol. 7, n. 6, pp. 609–616, 2001.

[3] C. Canali, M.E. Renda, P. Santi, S. Burresi, "Enabling Efficient Peer-to-Peer Resource Sharing in Wireless Mesh Networks, *IEEE Trans. on Mobile Computing*, Vol. 9, n. 3, pp. 333-347, March 2010.

[4] C. Canali, M.E. Renda, P. Santi, "Evaluating Load Balancing in Peer-to-Peer Resource Sharing Algorithms for Wireless Mesh Networks", *Proc. IEEE MeshTech*, pp. 603–609, 2008.

[5] *Texas Instruments CC2530 Data Sheets*, available online at `http://www.ti.com/ww/en/analog/cc2530/index.shtml?`.

[6] L. Galluccio, G. Morabito, S. Palazzo, M. Pellegrini, M.E. Renda, P. Santi, "Georoy: A Location-Aware Enhancement to Viceroy Peer-to-Peer Algorithm", *Computer Networks*, Vol. 51, n. 8, pp. 379–398, June 2007.

[7] J. Gao, L. Zhang, "Tradeoffs between Stretch Factor and Load Balancing Ratio in Routing on Growth Restricted Graphs", *Proc. ACM PODC*, pp. 189–196, 2004.

[8] P. Gupta, P.R. Kumar, "Critical Power for Asymptotic Connectivity in Wireless Networks", *Stochastic Analysis, Control, Optimization and Applications*, Birkhauser, Boston, pp. 547–566, 1998.

[9] E. Hyytiä, P. Lassila, J. Virtamo, "Spatial Node Distribution of the Random Waypoint Mobility Model with Applications", *IEEE Trans. on Mobile Computing*, Vol. 5, n. 6, pp. 680–694, 2006.

[10] D.B. Johnson, D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", *Mobile Computing*, Kluwer Academic Publishers, pp. 153–181, 1996.

[11] B. Karp, H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks", *Proc. ACM Mobicom*, pp. 243–254, 2000.

[12] A. Mei, J. Stefa, "Routing in Outer Space: Fair Traffic Load in Multi-hop Wireless Networks", *Proc. ACM MobiHoc*, pp. 23–31, 2008.

[13] A.D. Polyanin, A.V. Manzhirov, *Handbook of Integral Equations*, CRC Press, Boca Raton, 1998.

[14] L. Popa, A. Rostamizadeh, R.M. Karp, C. Papadimitriou, I. Stoica, "Balancing Traffic Load in Wireless Networks with Curveball Routing", *Proc. ACM MobiHoc*, pp. 170–179, 2007.

[15] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, F. Yu, "Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table", *Mobile Networks and Applications*, Vol. 8, No. 4 pp. 427–442, 2003.

[16] R. Sarkar, W. Zeng, J. Gao, X.D. Gu, "Covering Space for In-Network Sensor Data Storage", *Proc. ACM/IEEE IPSN*, pp. 232–243, 2010.

[17] L. Wang, F. Yao, C.-W. Yi, "Improved Asymptotic Bounds on Critical Transmission Radius for Greedy Forward Routing in Wireless Ad Hoc Networks", *Proc. ACM MobiHoc*, pp. 131–138, 2008.