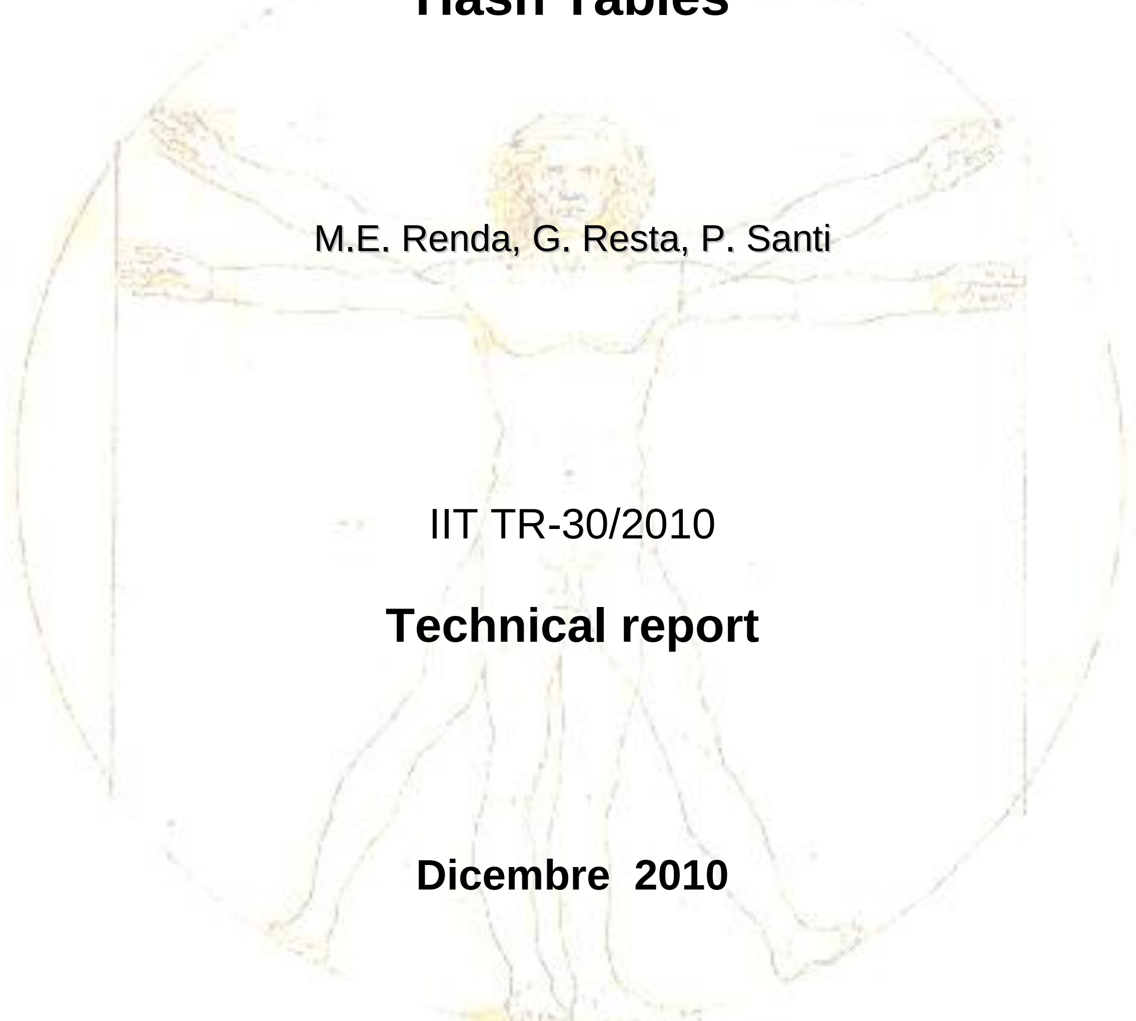
Consiglio Nazionale delle Ricerche

Load Balancing Hashing for Geographic Hash Tables



M.E. Renda, G. Resta, P. Santi

IIT TR-30/2010

Technical report

Dicembre 2010

Istituto di Informatica e Telematica

Load Balancing Hashing for 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 generated when querying a geographic hash table. State-of-the-art approaches can be used to improve load balancing by changing the underlying 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. Thus, current load balancing approaches are impractical in application scenarios where the nodes composing the geographic hash table have limited computational power, such as in most wireless sensor networks. In this paper, we propose a novel approach to solve the traffic load balancing problem in geographic hash tables: 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 so that the resulting destination density, when combined with a given source density, yields a perfectly balanced load distribution. We first formally characterize the desired destination density as a solution of a complex integral equation. We then present explicit destination density functions (taken from the family of Beta distributions) yielding quasi-perfect load balancing under the assumption of uniformly distributed sources. Our theoretical results are derived under an infinite node density model. In order to prove practicality of our approach, we have performed extensive simulations resembling realistic wireless sensor network deployments showing the effectiveness of our approach in considerably improving load balancing. Differently from previous work, the load balancing technique proposed in this paper can be readily applied in geographic hash tables composed of computationally constrained nodes, as it is typically the case in wireless sensor networks.**

***Index Terms*—Geographic hash tables; load balancing; sensor networks; hash function.**

I. Introduction

How to achieve load balancing in structured P2P systems is a challenging problem that has been extensively studied in recent years (see, e.g., [1], [3], [15], [22]). Challenges in addressing this problem are related to possible heterogeneity between peers, inhomogeneous key popularity, and to the fact that the mapping between overlay links – along which traffic could potentially be balanced – and physical links – along which data packets actually travel – is in general unknown.

With respect to this last challenge, geographic hash tables [19] present unique load balancing opportunities, since the overlay/physical link mapping in this case is known. More specifically, geographic hash tables are a special class of distributed hash tables in which peers are location-aware, and both peer IDs (their geographical coordinates) and keys are mapped in the same space, which typically coincides with the two- (or three-) dimensional domain on which peers are deployed[1]. In ght approaches[2], keys are assigned to peers based on a geographic proximity criterion, and overlay links are actually collapsed to the underlying physical links thanks to the use of geo-routing for querying the hash table. In geo-routing (see, e.g., [4], [14]), a message is routed towards a certain geographic location (x, y), and it is typically delivered to the node whose ID is closer to destination point (x, y). Thus, routing a query in a ght is a very simple task: if query for a certain key $k = (x_k, y_k)$ is generated at a certain (peer) 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 closer to (x_k, y_k), which is in charge of managing key k. Geographic hash tables find application mostly in wireless networks, e.g., for in-network data storage in large-scale wireless sensor networks [19], and for P2P resource sharing in wireless mesh networks [6], [9].

Despite usage of geo-routing, which gives unique load-balancing opportunities, severe load unbalancing can occur also in ghts, 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 unbalancing in case nodes are concentrated around some locations and/or "coverage holes" occur in the deployment region [20]. Even if nodes are uniformly distributed, load unbalancing 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 [7], [12]. We stress that load unbalancing causes well-known problems in wireless networks, such as reduction of network operational lifetime in energy-constrained environments (e.g., wireless sensor networks), and QoS degradation.

A possible way of lessening the load unbalancing 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 [16], [18], [20] can be applied to this purpose. However, changing the underlying routing protocol comes at a price. First, a customization of the routing protocol for the purpose of ght is needed, which might entail considerable implementation efforts especially considering that several applications (in principle all sharing the same routing protocol) often co-exist in a network. Second, a common feature of the load

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

[1]For simplicity, in this paper we consider the case of two-dimensional deployment region.

[2]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 [19].

balancing 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, 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 under the assumption of uniformly distributed nodes, presenting a novel approach 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 lessen the load unbalancing caused by straight-line geo-routing. The key idea in our approach 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 managed 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 presented theoretical analysis is based on the assumptions of infinite node density and perfect straight-line geo-routing between source and destination of a query. In order to evaluate the impact of relaxing these assumptions on the load balancing achieved by our approach, we have performed extensive simulations resembling realistic wireless sensor network deployments. Simulation results show the effectiveness of our approach in improving load balancing at the expense of only modestly increasing the overall network traffic (which is unavoidable when load balancing is sought [10]). Thus, differently from previous work, our proposed load balancing approach can be readily and effectively applied in geographic hash tables composed of resource constrained nodes, e.g., in wireless sensor networks.

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

II. Related work

The problem of achieving load balancing in geographic hash tables in presence of non-uniform node distribution has been recently addressed in [20]. 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 unbalanced (storage) load. The authors of [20] 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 [16], [18], 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 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. 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 [16] and [18] is on the choice of the virtual space, which is a properly defined distance-preserving symmetric space in [16], and a sphere in [18]. Similarly to [20], 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 [10] 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 [16], [18], [20], 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 [10] cannot be avoided in geometric graphs.

As already discussed, our proposal has several advantages w.r.t. existing approaches, such as simplicity and versatility. Another positive aspect of our approach is that, differently from existing proposals [16], [18], [20], our analysis 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 unbalancing of node energy consumption (related to load balancing).

III. Network model and preliminaries

We consider a (infinitely dense) network whose nodes are located in an arbitrary two-dimensional convex region A. Network nodes implement a geographic hash table, which is used, e.g., to index shared resources as in [6], [9], or to perform in-network data storage as in [19]. 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 (peer) which initiates the query process for a certain key k is called the *source node* in the following, denoted s; similarly, the node (peer) responsible for key k is called the *destination node*, denoted d. Similarly to [19], in the following we assume that a geographic routing protocol such as GPSR [14] 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 [10], [12], [16], [18].

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 [12] (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 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.

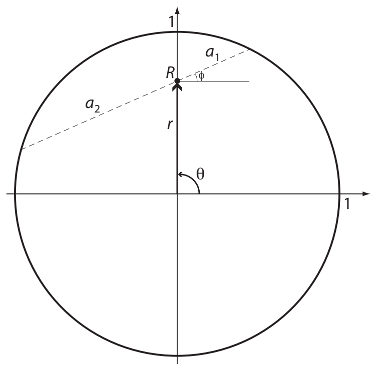


Fig. 1. Polar coordinate system on the unit disk, and definition of segments a_1, a_2.

IV. Load-balancing hash table design

A. 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 recent result [12], which has been originally derived to characterize the stationary node spatial distribution of the RWP mobility model [13] with arbitrary waypoint distribution.

To keep the presentation simple, in the following we assume that the deployment region A is the unit disk[3]. 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).

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

[3]Up to tedious technical details, using the techniques in [12] our design can be extended to the case where A is an arbitrary convex region.

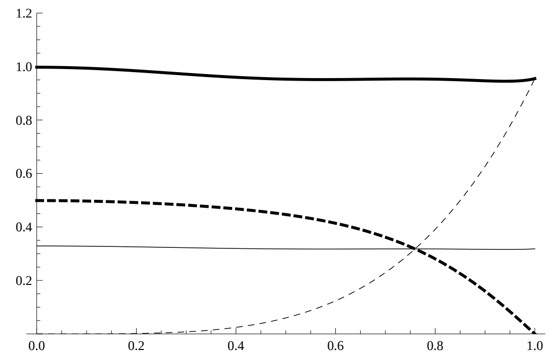


Fig. 2. Traffic and resulting load distribution with destination density function $\mathbf{d}_5$.

1). The values of $a_1(r,\phi)$ and $a_2(r,\phi)$ in the unit disk are as follows (see [12]):

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

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 [12], and is given by:

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

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 very difficult, since (2) is a very complex integral equation 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:

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

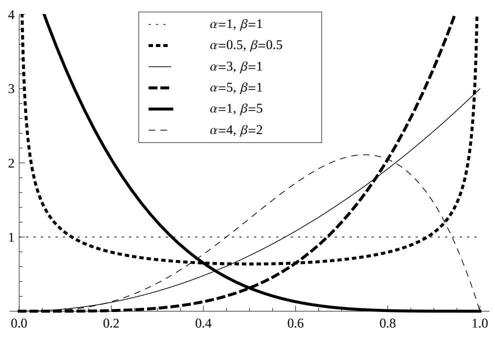


Fig. 3. Beta distribution for different values of the shape parameters.

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.

B. Explicit destination density characterization with uniform sources

Since attempting to directly solve integral equation (2) to derive $\mathbf{d}_u$ is very difficult, an alternative approach is trying to "guess" a close approximation of $\mathbf{d}_u$ driven by the observation at the end of Section IV-A, 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. 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:

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

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 [12]. 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 [6], [9], or to retrieve small data from a certain location using, e.g., GHT [19]. 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 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). Indeed, 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 I 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

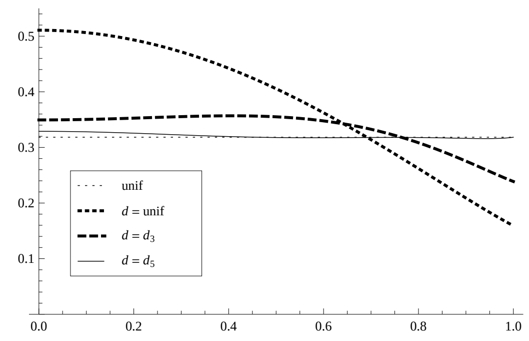


Fig. 4. Load density cross-section with different destination densities in the small data case.

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 I
LOAD BALANCING METRICS IN THE SMALL DATA CASE.

with respect to the uniform distribution. More specifically, the MSE for density $\mathbf{l}$ is computed as

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

As seen from Table I, 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 II. 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 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 II).

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

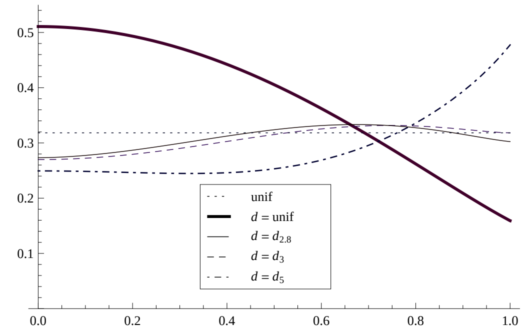


Fig. 5. Load density with different destination densities in the large data case.

Dest. density	M_l	Mm_l	MSE	exp. lengh
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 II
LOAD BALANCING METRICS AND EXPECTED PATH LENGTH IN THE LARGE DATA CASE.

with the well-known tradeoff between load balancing and overall network load [10]). 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 II 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.

C. 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 III, 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, data description in an WSN in-network storage system, etc.) into a real number in the $[0, 1]$ interval, i.e.:

$$\mathcal{H} : \mathbb{D} \to [0, 1] \ ,$$

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 [6], [19] 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 [19], 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 $0 \leq k \leq 1$, 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[4] of node 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

[4]Let N be a set of points in a bounded region A. The Voronoi cell $V(u)$ of a point $u \in N$ is the locus of all points in A closer to u than to any other point in N. The Voronoi diagram of N is the tessellation of A composed of the Voronoi cells of all points in N.

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 [17]:

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

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. 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. Thus, 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.

V. Simulations

The load balancing hash function design described in the previous section 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.

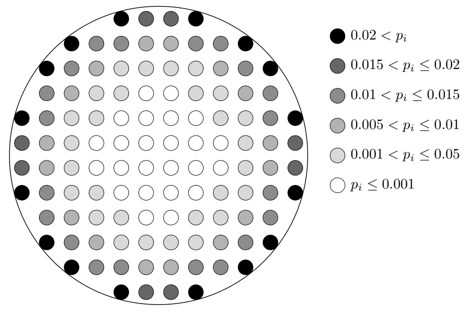


Fig. 6. Node deployment and discretized destination distribution $\mathbf{d}_5$ used in the simulations.

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 [14]: 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[5].

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 IV-B, and are computed according to the procedure described in Section IV-C: we first

[5]Indeed, we have implemented only the greedy forwarding step of [14], 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 [21].

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.

A. 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. 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 results of this first set of simulations are reported in figures 7 and 8 for the small and large data cases, respectively. In both cases, 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 (respectively, about 3) in the small (respectively, large) data case. This comes at the expense of increasing the average hop count (and, hence, the overall network load) of about 13% for both small and large data case. 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 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, 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. We believe 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 the second 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. As seen from Figure 9, 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 $\mathbf{d}_5$. 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%. The results for the small data case, which are not reported due to lack of space, confirmed that $\mathbf{d}_5$ is the best performing destination distribution in this scenario, 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%.

B. 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 [11], 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), which are not reported for lack of space, 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. We believe 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.

C. Discussion

Simulation results have shown that our proposed load balancing approach can be successfully used also in practical situations, where the assumptions of i) infinite node density and ii) perfect straight-line trajectory on which the analysis is based do not hold. Clearly, the achieved load balancing is not quasi-perfect as the one achievable if assumptions i) and ii) would hold. Yet, load balancing improvements are clearly visible also when node density is near the minimal needed to achieve connectivity.

Another interesting observation is that, from a quantitative point of view, our approach in practical scenarios achieves similar load balancing to existing approaches: the existing approaches based on uniform node distribution achieve a max load reduction in the order of about 25-40% [18] and about 30% [16], at the expense of an average path length increase of about 8% [18] and 30% [16], respectively. The approach of [20] cannot be directly compared to ours, since it addresses load balancing induced by non-uniform node distribution, hence the term of comparison for estimating benefits of the load balancing approach is significantly different from ours. The notable feature of our approach w.r.t. existing work is that *it improves load balancing while preserving simplicity of geo-routing, as well as of the hash function design*. Thus, we believe the approach presented in this paper has the potential of being readily applicable in a wireless sensor network scenario.

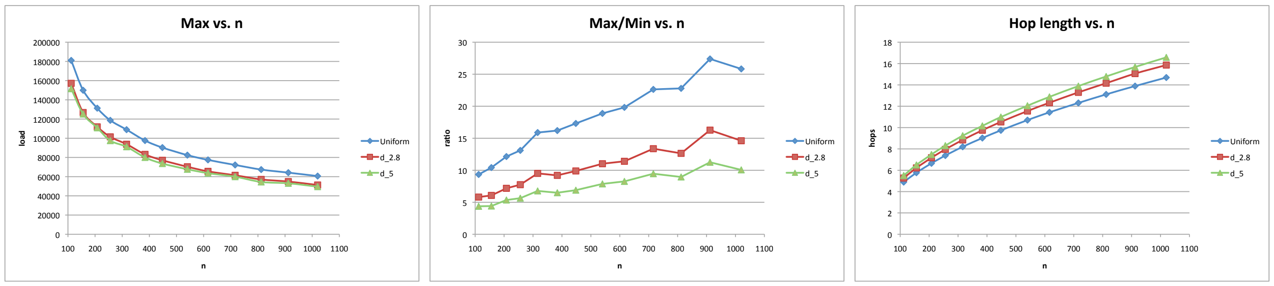


Fig. 7. Grid-based topology with fixed density: Maximum node load (left), max to min node load ratio (center), and average hop count (right) in the small data case.

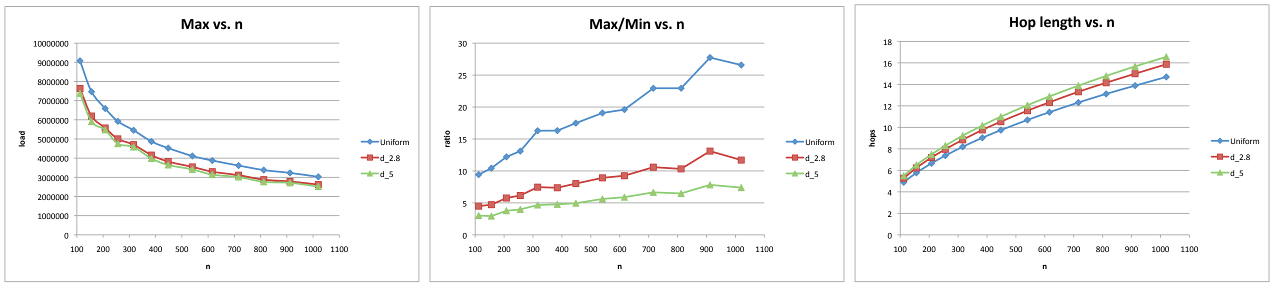


Fig. 8. Grid-based topology with fixed density: Maximum node load (left), max to min node load ratio (center), and average hop count (right) in the large data case.

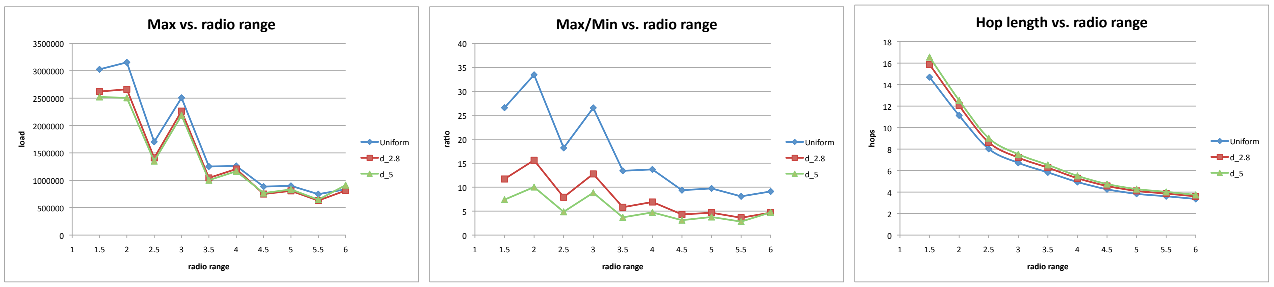


Fig. 9. Grid-based topology with $n = 1020$ and varying density: Maximum node load (left), max to min node load ratio (center), and average hop count (right) in the large data case.

VI. Other applications

The ideas presented in this paper can be used in other application scenarios where density distribution $\mathbf{d}$ can be modified in order to obtain the desired balancing property.

A first example of possible use of our ideas is in mobility modeling. A well-known problem of widely used mobility models such as RWP [13] is that the stationary node spatial distribution resulting when nodes move in a bounded area is not uniform, but it tends to be concentrated in the center of the movement region [2]. This so called *border effect* might cause serious inaccuracies in wireless network simulation, due to the fact that the initial node distribution, which is typically uniform, is different from the stationary one. Techniques have been proposed to improve wireless simulation accuracy, such as $i)$ considering nodes as moving on a torus instead of a bounded region, or $ii)$ gathering simulation results only after a sufficiently long "warm up" period [5]. However, $i)$ reduces the realism of the simulated scenario, while $ii)$ wastes considerable computational power – this is especially true considering that wireless network simulation is a computational intensive task. The ideas presented in this paper can be readily applied to completely remove the border effect in RWP mobile networks without adding unrealistic toroidal movement or wasting computational power, by simply modifying the default destination density function $\mathbf{d}$ of the RWP model, which is the uniform distribution. In particular, it is sufficient to solve (or approximate the solution, as done in this paper) the integral equation (2) with $a = b = 0$ and $c = 1$ to obtain a destination density $\mathbf{d}$ such that the resulting node spatial distribution (corresponding to density $\mathbf{t}$) is uniform. This way, uniform stationary node spatial distribution can be achieved (thus eliminating the need of a warm-up period) without changing the mobility rules with toroidal wrap-around.

Another possible application of our ideas is in traffic anonymization. Traffic anonymization is a security approach aimed at countering traffic analysis attacks. In a wireless network, where traffic can be easily overheard, a common approach is trying to induce a uniform traffic load in the network, so that an external observer has no clue about the ongoing traffic pattern [8]. Anonymization is typically obtained by sending packets along a path that goes beyond the actual destination, so that an external observer cannot deduce the actual packet destination by simply observing where a packet stops. While several traffic anonymization approaches have been proposed, to the best of our knowledge the resulting "anonymization" (i.e., traffic load uniformity) has only been evaluated through simulation. In particular, only heuristics have been proposed to determine how further beyond the actual destination a packet should travel in order to achieve a good "anonymization". The ideas presented in this paper can be used as a starting point for formally characterizing a "virtual destination density" – obtained by suitably extending the trajectories induced by the given actual destination density – resulting in optimal traffic anonymization (i.e., uniform traffic load). We are currently actively working towards achieving such formal characterization.

VII. Conclusions

In this paper, we have presented a novel approach to the load balancing problem in ght design. The proposed approach has been shown to provide quasi-perfect load balancing in ideal conditions, and to provide load balancing improvements comparable to those provided by existing schemes in practical scenarios. The major advantage of the presented approach over existing ones lies in its practicality and versatility. Possible extension of our approach 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 [20] to address the case of non-uniform node distribution as well as concave shapes of the node deployment region.

References

[1] K. Aberer, A. Datta, M. Hauswirth, "Multifaceted Simultaneous Load Balancing in DHT-Based P2P Systems: A New Game with Old Balls and Bins", *LNCS 3460*, pp. 373–391, 2005.

[2] C. Bettstetter, G. Resta, P. Santi, "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks", *IEEE Trans. on Mobile Computing*, Vol. 2, n.3, pp. 257–269, July-Sept. 2003.

[3] S. Bianchi, S. Serbu, P. Felber, P. Kropf, "Adaptive Load Balancing for DHT Lookups", *Proc. Int. Conference on Computer Communications and Networks (ICCCN)*, 2006.

[4] 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.

[5] T. Camp, J. Boleng, V. Davies, "Mobility models for ad hoc network simulations", *Wireless Communication & Mobile Computing (WCMC)*, special issue on mobile ad hoc networking, Wiley, 2002.

[6] 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.

[7] 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.

[8] J. Deng, R. Han, S. Mishra, "Decorrelating Wireless Sensor Network Traffic to Inhibit Traffic Analysis Attacks", *Pervasive and Mobile Computing*, Vol. 2, n. 2, pp. 159–186, 2006

[9] 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.

[10] J. Gao, L. Zhang, "Tradeoffs between Stretch Factor and Load Balancing Ratio in Routing on Growth Restricted Graphs", *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 189–196, 2004.

[11] 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.

[12] 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.

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

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

[15] A.R. Karthik, K. Lakshminarayanan, S. Surana, R. Karp, I. Stoica, "Load Balancing in Dynamic Structured P2P Systems", *Proc. IPTPS*, pp. 68–79, 2003.

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

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

[18] 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.

[19] 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.

[20] R. Sarkar, W. Zeng, J. Gao, X.D. Gu, "Covering Space for In-Network Sensor Data Storage", *Proc. IPSN*, 2010.

[21] 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.

[22] Y. Zhu, Y. Hu, "Efficient, Proximity-Aware Load Balancing for DHT-based P2P Systems", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 16, n. 4, pp. 349–361, 2005.