Q-NiGHT: Adding QoS to Data Centric Storage in Non-Uniform Sensor Networks

Michele Albano[1] Stefano Chessa[1,2] Francesco Nidito[1] Susanna Pelagatti[1] *
[1]Dipartimento di Informatica, University of Pisa, Pisa Italy
[2]Istituto di Scienza e Tecnologie dell' Informazione, CNR, Pisa Italy

Abstract

Storage of sensed data in wireless sensor networks is essential when the sink node is unavailable due to failure and/or disconnections, but it can also provide efficient access to sensed data to multiple sink nodes. Recent approaches to data storage rely on Geographic Hash Tables for efficient data storage and retrieval. These approaches however do not support different QoS levels for different classes of data as the programmer has no control on the level of redundancy of data. They result in a great unbalance in the storage usage in each sensor, even when sensors are uniformly distributed. This may cause serious data losses, waste energy and shorten the overall lifetime of the sensornet. In this paper, we propose a novel protocol, Q-NiGHT, which (1) provides a direct control on the level of QoS in the data dependability, and (2) uses a strategy similar to the rejection method to build a hash function which scatters data approximately with the same distribution of sensors. The benefits of Q-NiGHT are assessed through a detailed simulation experiment, also discussed in the paper. Results show its good performance on different sensors distributions on terms of both protocol costs and load balance between sensors.

1 Introduction

A wireless sensor network is composed by a large number of low power, low cost sensors[1] which self organize into a (multi-hop) ad hoc network. Sensors are spread in an environment (*sensor field*) without any predetermined infrastructure and cooperate to execute common monitoring tasks which usually consist in sensing environmental data. The sensed data are collected by an external sink node, when it is connected to the network. The sink node, which could be either static or mobile, is in turn accessed by the external operators to retrieve the information gathered by the network. Storage of sensed data in wireless sensor networks is essential when the sink node is unavailable due to failure and/or disconnections, but it can also provide efficient access to sensed data to multiple sink nodes. Recent approaches to data storage rely on Data Centric Storage for efficient data storage and retrieval. The work that introduced Geographic Hash Tables [3] also depicted the scenarios where Data Centric Storage outperforms Local Storage and External Storage.

Q-NiGHT. In this paper we propose Q-NiGHT, a novel DCS protocol which moves from GHT incorporating QoS control and featuring good load balance among sensors. Q-NiGHT uses a strategy similar to the *rejection method* [7] to build a hash function biased with sensor distribution. This spreads data more evenly among nodes. In addition, Q-NiGHT can provide QoS with different redundancy techniques. We detail the protocol using pure replication, allowing the user to choose the number of replicas required for a given datum. We conduct detailed simulations of Q-NiGHT and GHT and compare the results obtained with respect to the load of each sensor (i.e. the number of data stored in each node) and the number of messages needed for data storage and retrieval. Results show the good performance of Q-NiGHT on different sensors distributions on terms of both protocol costs and load balance. In the first part of this paper we analyze the characteristics of GHT simulating its behavior with uniform and Gaussian distributions of sensors. Results show that the amount of data distributed to each sensor can have a large variance, causing hot spots on borders of the network even with uniform sensor distribution. This may cause data losses due to lack of storage in the boarder nodes and to battery exhaustion. Dead sensors on the borders may cause further unbalance due to ill shaped perimeters. As we may expect, the effect of load unbalance is more serious when sensors distribution is not uniform. This is due to the fact that locations are hashed by GHT uniformly on the sensornet regardless of the actual sensor density.

Paper organization. The paper is organized as follows. In Section 2 we present related works. Section 3 discusses

*Work funded in part by the European Commission in the framework of the "SMEPP" project (contract N. 033563), and "PERSONA" IP project (contract N. 045459)

the problems of GHT presenting the results of simulations carried on with uniform and Gaussian distributions. Section 4 discusses the "biased" hash function h used in our protocol. Then, Section 5 details Q-NiGHT and discusses how different redundancy techniques can be incorporated in the protocol. Finally, Section 6 reports on the simulation of our protocol and compares its performance with plain GHT. Conclusions are presented in Section 7.

Paper notation. Recurrent symbols used in the paper are summarized in Table 1.

$s\ s_i\ s_j$	generic sensors
n	number of sensors in the network
r	communication range of a sensor
A	deployment area in which sensors are located
f	geographical distribution of sensors in A
h	hash function used to locate data
D	datum to be stored/retrieved, in the system
M	name (meta-data) for D

Table 1: Recurrent symbols

We also planarize the input graphs using both RNG (*Relative Neighborhood Graph*) and GG (*Gabriel Graph*)[8], as requested by the GFG[4] routing protocol.

2 Related works

Data centric storage. Our work originates from the GHT [3], we discuss this protocol in depth in Section 3. Cell Hash Routing (CHR)[6] is another DCS routing based on hash tables. CHR first clusters nodes in cells of predefined and globally known shape using a distributed protocol (e.g. dividing the sensor field in a mesh of squares). Then it uses the cells, instead of individual nodes to hold the values. CRT uses a variant of GFG working on cells. Data are hashed into geographic coordinates as in GHT[3] and routed to the cell that includes that location. As soon as the data reaches the destination cell it is stored in all the nodes in the cell. If the cell is empty (e.g. it has no sensor) CHR uses an approach similar to GHT to replicate data *around* the empty cell. Problems related with using geographic face routing in practical settings have been discussed in [13]. To solve this problems new solutions were proposed. For instance, GEM [14] proposes a graph embedding for sensor networks which defines a set of virtual coordinates that can be used for routing and DCS in place of true geographic coordinates. Another approach similar to GEM is Hierarchical Location Identifiers (HLI)[15]. The system uses a hierarchical location system to identify nodes by some characteristics. The system uses DSDV[16] routing protocol and aggregate routes. This provides a good base to perform unicast, multicast and anycast. The system, authors claim, can be used as a base for other solutions as TinyDB[2] and GHT[3]. An alternative approach for routing is Information Directed Routing (IDR)[19]. To route a message, IDR finds *a path with maximum information gain at moderate communication cost*. The network routes the message from the source to the destination finding a path that is not minimal in terms of energy but this path pass through *high interest areas* to collect as much information as possible to reply to the query.

Non uniformity. Non uniformity is a brand new topic in wireless sensor networks and it is little explored. The same is for non uniformity and data management. Two works that consider non uniformity are [17] and [18]. The first considers the problem of connectivity in sensor networks and covers also non uniformly deployed networks. The third study the problem of broadcast using also non uniformity (introducing the "Hill distribution").

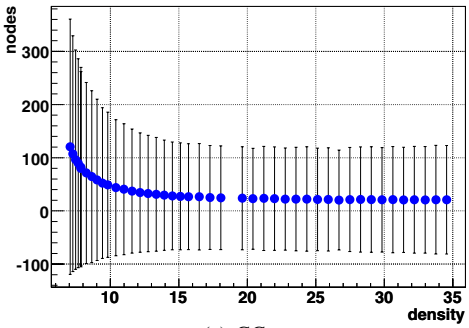
3 Load unbalance in GHT

GHT[3] implements Data Centric Storage using Geographic Hash Tables. Each datum has a unique *meta-datum* (or name) which is hashed uniformly as a coordinate in the sensing area, represented as a two-dimensional plane. GHT implements two operations: `put`, which stores data, and `get`, which retrieves them. In the `put` operation, the name of data to be stored is first hashed into a location (x, y) in the sensing field. Then, GHT selects the sensor closest to (x, y), which becomes the *home node* for that data. The home node is selected using GFG[4]. GFG uses two operation modes: greedy and perimeter. Each packet starts in the *greedy mode*, in which it is routed progressively closer to its destination at each hop. When a packet reaches a node s_i whose neighbors are all farther than s_i to the destination, GFG switches to the *perimeter mode* and the packet is forwarded using the *right hand rule*, that is the packet is forwarded on the *next* edge clockwise from the edge from which the packet has been received. As soon as the packet reaches a node closer to destination than the previous ones, it returns to the greedy mode. If the destination (x, y) does not correspond to any sensor, GHT uses the perimeter mode of GFG to locate all the sensors surrounding (x, y) (called the *perimeter* of (x, y)). The closest sensor in the perimeter becomes the *home node* for (x, y). GHT stores a copy of the data in the home node as well as in all the sensors belonging to the perimeter. Storing on all the perimeter is essential to guarantee data persistence also in presence of node faults. Data retrieval uses a `get` operation. The name is first hashed into the destination (x, y), then GFG is used to route the request to (x, y). When the request reach a node in the perimeter of (x, y), the data is returned back to the sender. Replicating all data on the perimeter of (x, y) is a simple choice, which allows to use GFG with almost no changes and which can work quite well on very large sens-

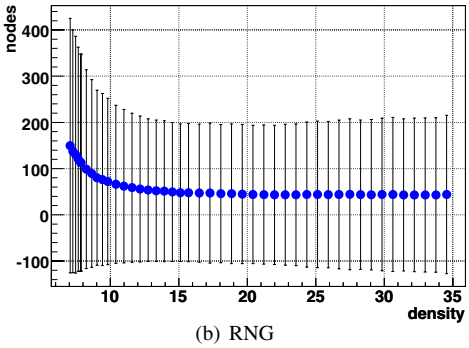
ing fields with uniformly distributed sensors. In order to measure the degree of unbalance of GHT on more realistic scenarios, we simulated a flat square sensing field, with a $400m$ side. Each node has circular transmission range with $10m$ radius. In this area, we simulated several sensornets ranging from 3600 to 18000 sensors, which correspond to a mean network density ranging in $[8, 35]$. For each density, we randomly generated 100 networks with uniform distribution. For each network, we compute the mean and the variance of the number of nodes found in a GFG perimeter as follows. For each sensor network the simulator randomly selects 1000 points and, for each point, it computes the number of nodes in the perimeter surrounding the point, in the case that GG or RNG are used in the protocol. Figure 1, summarizes the mean and standard deviation of the perimeters. The figure shows that, as the network density increases, the average number of nodes in a perimeter decreases. However, the actual number of nodes remains highly variable, and the standard deviation is higher with RNG than with GG. This variance is partly due to the behavior of nodes in the outer part of the sensing area (the grey part in Figure 2), since in that area the probability of having very long perimeters (i.e. following the whole boarder) is high. With low densities, the probability that a random point belongs to the exterior of the network (and thus it is associated to the external perimeter) is not negligible.

Perimeters on the border. In order to understand this *border effect*, we performed another set of simulations in which the sensing nets are generated in the same way as above but the boarder area is not used to store data. We "cut away" the 5% of the area from each border (the grey area in Figure 2.a), for a total of 19% of the total area. We randomly generate 1000 points in the white area and again measure the length of each perimeter and compute the mean and the variance. Figure 2.b shows the results obtained using GG. The mean and the variance improve if the border nodes are let out but standard deviation remains high, leading to a high unbalance in the node load. Results with RNG are much worse, as with the whole sensing area.

Non uniform sensor distribution. In order to understand the behavior of GHT with non-uniform sensor distribution, we repeated our experiments using a Gaussian function ($\sigma = 1$ with maximum on the center of the area) for distributing sensors. The function is normalized to have the 99 percentile matching the area. The results are shown in Figure 3. The behavior is much worse than with uniform distribution because GHT, uses a uniform hash function independently of the real distribution of sensors. This bring to a pathological state of *load unbalance* that is due to the different quantity of data that must be managed by an equal number of sensors: A sensor on the border of the deployment area must manage a quantity of data that is larger than the quantity managed by a sensor in the center of the network.



(a) GG



(b) RNG

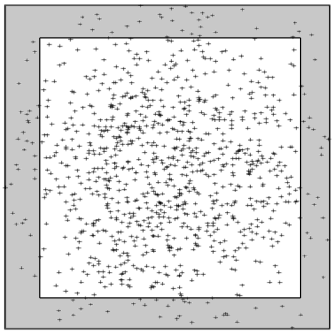
Figure 1: Mean and variance of perimeters (number of nodes) measured for different densities with GG and RNG planarization.

work.

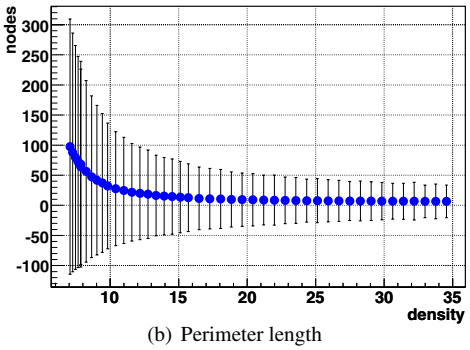
Load unbalance and QoS. Another issue with GHT is that there is no way to control the QoS provided for each datum. Since the point (x, y) is obtained computing an hash function h on its associated meta-data M, the selection of the sensors candidate for storage is in practice independent from the "value" of the data. In principle this ensures the same treatment for each stored datum. However, if the meta-datum M is particularly popular and many sensors generate data described by M, the sensors located in the perimeter around $(x, y) = h(M)$ would be burdened with an high storage and communication load. For this reason the authors of [3] introduce the technique of *structured replication*. However nor GHT, neither the structured replication ensure that the level of redundancy associated to a data is related to the importance of the data itself: GHT assure only the same average treatment of each stored data. Another aspect is that, although the average level of redundancy of the meta-datum is constant, in practice it can vary significantly (due to the fact that each geographic points is surrounded by a different perimeter), even in case of uniform distribution of the sensors.

4 Non-uniform hashing

As we have seen, a serious problem with GHT is due to the fact that it uses a uniform hash function independently

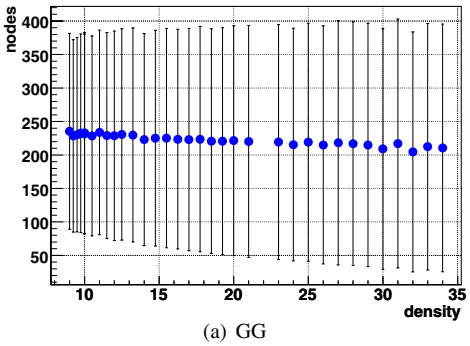


(a) The border

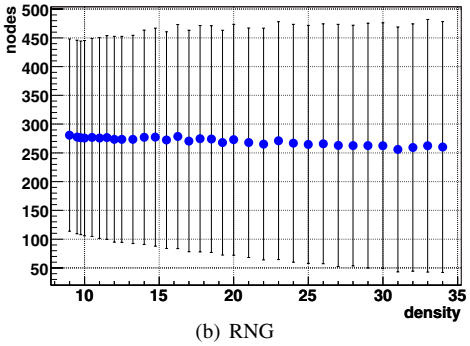


(b) Perimeter length

Figure 2: The border area (gray) and the storing area (white). Mean and variance of perimeters measured for different sensornet densities in case of GG using only the white area.



(a) GG



(b) RNG

Figure 3: Mean and variance of GHT perimeters for different sensornets densities, Gaussian sensor distribution.

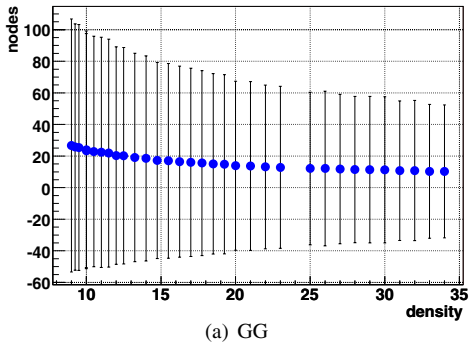
of the real distribution of sensors. This leads to the pathological load unbalancing shown in Figure 3. We propose to use hash functions which scatter data approximately with the same distribution of the sensors. We can observe that an hash function $h(k)$ is a kind of pseudo-random number generator: Starting from a seed (in our case the key k) it produces an output (in our case in a value in $\mathbb{R}^2$) such that for *near* values of key the hashed values must be *distant*. With this consideration in mind we define a new hash function, whose pseudo-code is shown in Algorithm 1. This function uses a strategy similar to the one used in the *rejection method*[7], but with some differences. Rejection method is a technique used in random number generation to produce random numbers following any probability distribution, with limited dominion. The basic idea is the following. The probability function is boxed and we generate uniform random values in the box. If the value generated is below the distribution function the value is accepted and returned. Otherwise we randomly generate new points in the box until values are below the function. Notice that in principle there is a non-null probability of non termination because the values can be generated all above the function. In practice a good uniform hash grants to generate values in all the box. The function RejectionHash returns a pair

Algorithm 1 REJECTIONHASH(k, f)

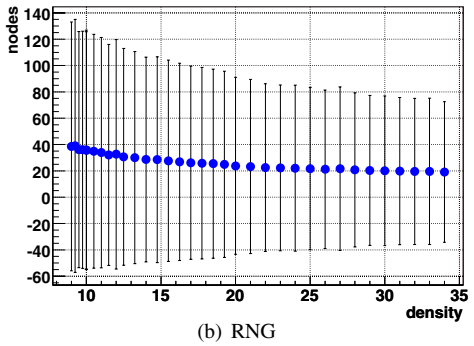
Require: A key k and a function f.
Ensure: A coordinate pair (x, y).

```
i ← 0
loop
  (x, y, z) ← Hash(k + i)
  i ← i + 1
  if z < f(x, y) then
    return (x, y)
  end if
end loop
```

(x, y) of coordinates where to place data from its key `k`, belonging to distribution `f`. Instead of using uniform randomization uses random hashing on the key. At each iteration, if necessary, changes lightly the key in a deterministic way. Figure 4 shows a good behavior of the non-uniform hash function. RejectionHash fits well the the sensor distribution in the data dissemination strategy with a good global load balancing. These results are better than the results provided with uniform distribution and uniform hashing (Figure 1 and Figure 2.b). This is due to the Gaussian distribution of the nodes that does not have a border effect as evident as in the uniform distribution.



(a) GG



(b) RNG

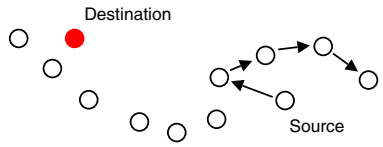
Figure 4: Mean and variance of GHT perimeters for different sensornets densities, Gaussian sensor distribution and non uniform hashing.

5 Adding QoS to GHT

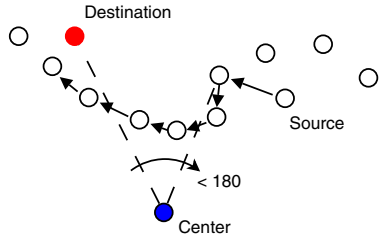
As well as GHT [3], Q-NiGHT is built atop the GFG routing protocol[4][1]. Q-NiGHT provides data insertion and data retrieval on the sensor network. To our purposes the interface of the `put` includes, along with the meta-data M and the data D, also a parameter Q expressing the desired QoS. Q gives a measure of the dependability required for the data, and may be expressed using different metrics and ranges according to the particular redundancy technique used. For instance, if Q-NiGHT adopts pure replication then Q can express the number of sensors on which the data should be replicated, or, if Q-NiGHT adopts n out of m redundant encodings[9, 10], then Q can express the number of fragments in which partition the data (each fragment to be stored in a different sensor) and the number of redundant fragments. In the following, we describe Q-NiGHT assuming pure replication of the data. Data insertion is involved with `put(M,D,Q)`. We assume Q ranges in $[1, Q_{max}]$ and gives the number of sensors on which the data should be replicated. Let s be the source node of a `put(M,D,Q)` operation. s firstly computes $h(M)$, where h is the hash function conditioned with the sensor distribution function, f, in the sensing field, as discussed in Section 4. $h(M)$ returns a pair of geographic coordinates (x, y) as the destination of the packet `P`$_p$`=<(x,y),<M,D,Q>>`. The packet in turn is sent to the destination using the GFG protocol. As in GHT we call *home node* the sensor s_d (of coordinates (x', y')) geographically nearest to the destination coordinates. The home node naturally receives the packet as a consequence of applying GFG. Upon the reception of packet `P`$_p$, sensor s_d begins a *dispersal protocol* which selects Q sensors to store a copy of `<M,D>`. The dispersal protocol is iterative and uses the concept of *ball*. Given a sensor s_d of coordinates (x, y), we denote with $B_{(x,y)}(\overline{r})$ the *ball* centered in (x, y) of radius $\overline{r}$, that is the set of sensors that are within a Euclidean distance $\overline{r}$ from (x, y). In the first iteration s_d broadcasts a replica of `D` to all the sensors included in the ball $B_{(x',y')}(\overline{r})$. $\overline{r}$ is chosen in order to reach the Q sensors nearest to (x, y) with high probability. A complete proof of our statement can be found in [12]. Each sensor receiving a replica responds with an acknowledgment to s_d. Sensor s_d confirms the $Q - 1$ acknowledgments received from the sensors geographically nearest to (x, y) and disregards the others. The confirmation requires an extra packet sent by s_d. Sensors which receive the confirmation keep the data while the other sensors will disregard the data after a timeout. If s_d receives $Q' < Q$ acknowledgments, then it executes another iteration of the dispersal protocol with $\overline{r} = 2\overline{r}$ in which it considers only the sensors in $B_{(x',y')}(2\overline{r}) - B_{(x',y')}(\overline{r})$. The dispersal protocol stops as soon as Q sensors have been hired or the outermost perimeter has been reached. The dispersal protocol is a simple implementation a of geo-casting protocol[11]. When a node s_g of coordinates (r, z) executes `get(M)` it firstly computes $(x, y) = h(M)$, and sends a query packet `P`$_g$`=<(x,y),<(r,z),M>>` using the GFG protocol. In turn, packet `P`$_g$ will reach the perimeter surrounding (x, y) and it will start turning around the perimeter. Eventually, the packet will reach either the home node or another node containing a replica of the data D associated to M. This node will stop packet `P`$_g$ and will send the required data back to s_g. The complexity of the `put` protocol clearly depends upon the choice of $\overline{r}$ as this determines the number of iterations made to successfully place the Q replicas. However, if we know the distribution of sensors f, for any given (x', y'), coordinates of s_d, and Q it is possible to fix $\overline{r}$ in such a way that, with high probability, at least Q sensors belong to the ball $B_{(x',y')}(\overline{r})$. In simulations, to compute $\overline{r}$, we use a simple strategy based on the approximation of the ball with its inscribed square. Using this approximation, we are have experimented that the method converges in no more than 2 iterations and the average number of messages generated is minimal [12].

Enhanced GFG. Q-NiGHT uses a slightly modified version of GFG. Usually, when GFG is in the *perimeter mode*, it always adopts clockwise turn to reach the destination coordinates. This behavior leads to pathological situations as

[1] In [3] the authors call it GPSR[5]



(a) Standard perimeter mode



(b) Enhanced GFG strategy

Figure 5: GFG routing perimeter mode

the one shown in Figure 5.a. The right hand rule make the packet traverse all the perimeter before reaching the destination node. This is not a problem for GHT as the data are replicated on all the perimeter, but may be very inefficient for Q-NiGHT, which replicates only on a ball surrounding the destination. In our GFG version, we turn clockwise or counterclockwise depending on the destination, as shown in Figure 5.b. Let s_a be the position of the sender node, c the position of the center of the deployment area and s_d be the position of the destination. We turn clockwise if $0 < \widehat{s_a c s_d} < 180$[2], and counterclockwise otherwise.

Behavior in case of faults. In case some of the nodes holding the replicas of `<M,D>` fault our protocol continues to operate correctly. Due to GFG protocol, any `get` with key `M` is routed to the node geographically nearest to $(x, y) = h(M)$ (home node of `M`). If the faulty node is not the home node, the protocol implicitly discards it. If this is not the case, the second nearest node in the perimeter is always included in the ball built by our protocol [12].

6 Simulations and results

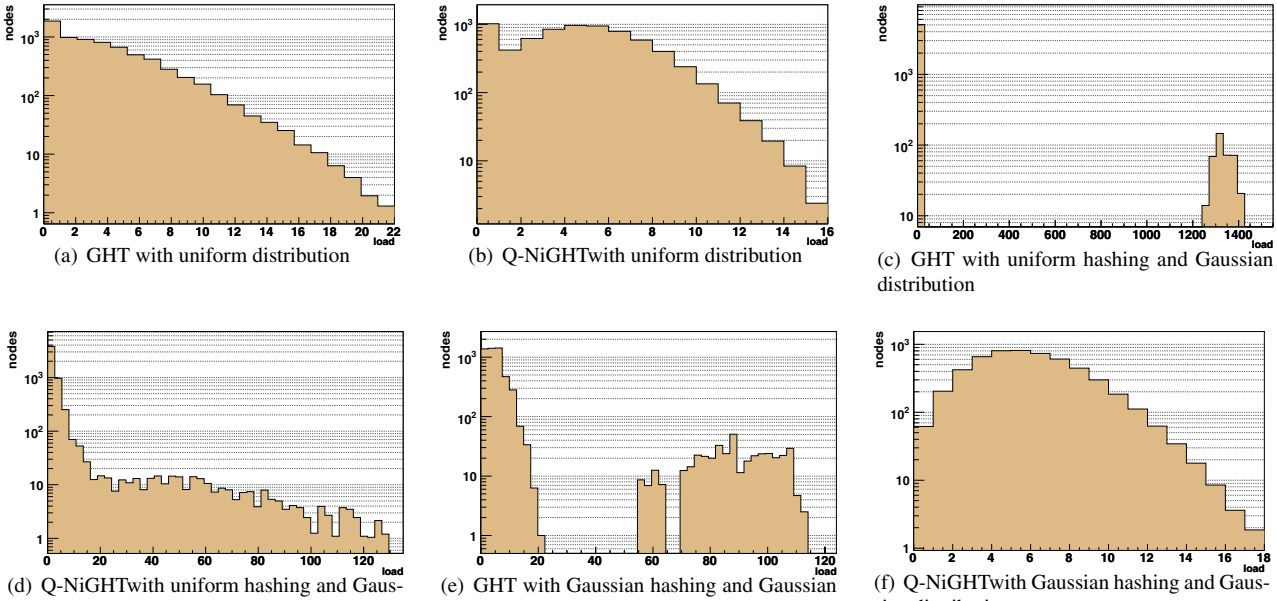
In this section, we discuss the results of our simulation. We simulated a square with a $400m$ side, with sensor transmission range of a perfect $10m$ radius circle. We assumed a density of 14 and performed 2000 `put` operations with randomly generated meta-data using both GHT and Q-NiGHT . In these trials, Q-NiGHT uses a pure replication QoS with 15 replicas for each datum. Figure 6 compares the behavior of GHT (graphics a,c,e) and Q-NiGHT (graphics b,d,f). All graphics show the average load of sensors using RNG. Graphs (a,b) consider uniform sensor distribution and uniform hashing, (c,d) Gaussian sensor distribution and uniform hashing and finally (e,f) Gaussian sensor distribution and Gaussian hashing. show the average load of sensors in the In all graphs, the x axis shows the different load (e.g. Number of data) on a node and the y axis shows the number of nodes storing exactly this number of data. Values on the y axis follow a logarithmic scale for better comprehension. We can see that Q-NiGHT reaches better load balance even in the uniform case (graphs a,b), while from graphs (c,d) it is evident that Q-NiGHT reaches better load balance even in the uniform case. Figure 6.(c,d) shows the average load of sensors in case of Gaussian sensor distribution and uniform hashing. GHT shows its usual unbalance problems, while Q-NiGHT manages to balance the load is able to balance the load (despite uniform hashing) because it keeps replica distribution localized and avoids replication on long perimeters (which happens with GHT in low density areas). This behavior is even more evident in Figure 6.(e,f) in which we compare the load of GHT and Q-NiGHT in case of Gaussian distribution of sensors and Gaussian hashing.

Evaluating `put`s and `get`s. Figure 7 shows the mean and standard deviation of the cost of the basic `put` and `get` operations (number of hops needed to store a data). We performed 2000 `puts` and 2000 `gets` with randomly chosen meta-data. The QoS for Q-NiGHT is again pure replication with 15 replicas for each datum. We always consider RNG networks. In all graphs, the x axis shows the sensor density in the network and the y axis the operation cost measured. Graphs in the first row compare the cost of a `put` operation in GHT (a) and Q-NiGHT (b). The `put` is much more efficient in Q-NiGHT as it keeps the replicas localized in a ball without following long perimeters across the network. On the other hand, Figure 7.(c,d) shows the mean and standard deviation of the cost of a `get` operation in case of RNG networks with GHT (c) and Q-NiGHT (d). The cost of Q-NiGHT is greater than GHT. This is due to the fact that in GHT, as soon as a `get` request hits a node in the perimeter it immediately finds the data, on the other hand, using Q-NiGHT that request must travel until it reaches the replication ball, which may need a few more hops. This behavior was much worse using standard (not enhanced) GFG as travelling along the perimeter could mean traverse the entire network before hitting the ball. However, in our opinion the cost of `get` is still rather low as a price to be payed in order to get load balance on the network and QoS. We can expect that the `put` operations would be much more common during the network lifetime.

7 Conclusions and future work

In this paper, we discussed the limitations of the GHT in practical sensor networks and we have proposed a new protocol (Q-NiGHT) which overcomes these limitations and allows a fine QoS control by the user. Q-NiGHT corrects the ill behavior of GHT in three ways. (1) It uses non uniform hash tables, which allow a much balanced distribution of

[2] Computed in clockwise way



(a) GHT with uniform distribution

(b) Q-NiGHTwith uniform distribution

(c) GHT with uniform hashing and Gaussian distribution

(d) Q-NiGHTwith uniform hashing and Gaussian distribution

(e) GHT with Gaussian hashing and Gaussian distribution

(f) Q-NiGHTwith Gaussian hashing and Gaussian distribution

Figure 6: Average load of sensors

data across the sensor network when sensors are distributed in a non uniform way. (2) The *dispersal protocol* used in Q-NiGHT allows the user to control the number of replicas of a given datum (using the quality of service parameters in the `put`). (3) Data replicas are placed in sensors which are "as close as possible" to the home node, which result in much balanced load on all the sensor network. The merits of Q-NiGHT have been evaluated through simulation in uniform and Gaussian distributed sensor networks. The results show that the protocol performs a better load balancing with respect to GHT, and has a smaller cost for `put` operations. We also proposed an enhanced version of GFG protocol to correct the ill behavior in the case of perimeter mode.

Future work will include the study of protocols to estimate the sensor distribution on the deployment area. We would work on new routing protocols that work better in non-uniformly distributed networks. Another issue we will address is the latency of `get`/`put` operations. We would compare standard GHT with Q-NiGHT to evaluate their performance. We also want to use, in our approach, the modified GFG protocol proposed in [20]. We will study, also, the opportunity to use virtual coordinates systems [21] instead of GPS-based positioning system.

References

[1] Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communications Magazine **40**(8) (2002) 102–114

[2] Maddenand, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: The design of an acquisitional query processor for sensor networks. In: Proc. of the 2003 SIGMOD Conference, San Diego, (2003) 491–502

[3] Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., Yu, F.: Data-centric storage in sensornets with GHT, a geographic hash table. Mob. Netw. Appl. (MONET) **8**(4) (2003) 427–442

[4] Bose, P., Morin, P., Stoimenovìc, I., Urrutia, J.: Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. Wireless Networks, **7**(6) (2001) 609–616 Also in *DialM'99*, Seattle, August 1999, 48–55.

[5] Karp, B., Kung, H.T.: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In: Proc. of MobiCom 2000, Boston, (2000) 243–254

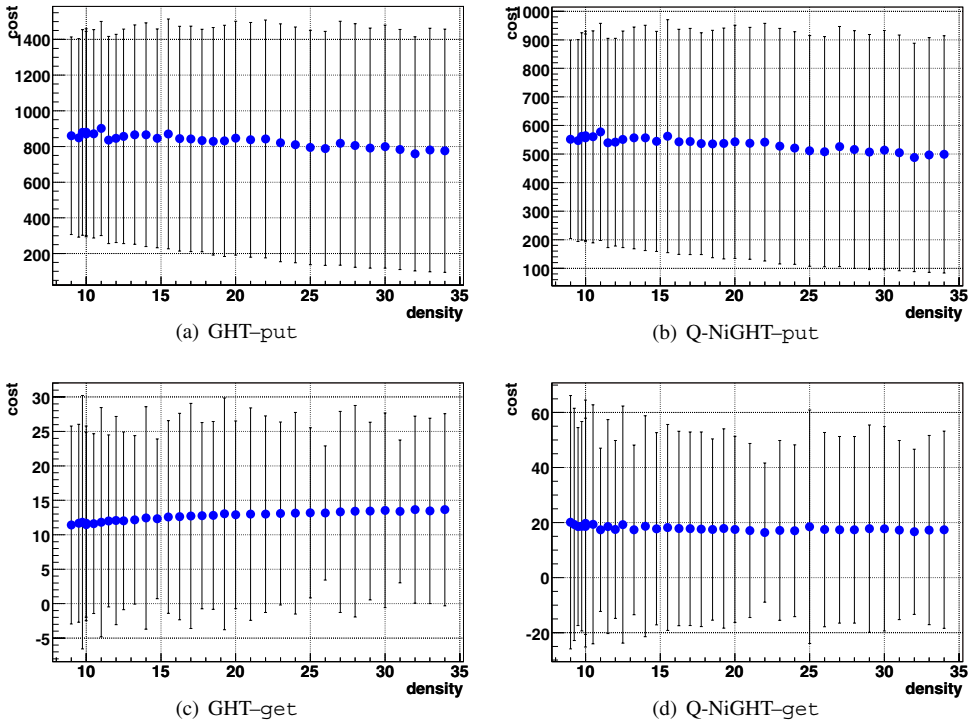
[6] Araujo, F., Rodrigues, L., Kaiser, J., Liu, C., Mitidieri, C.: CHR: a Distributed Hash Table for Wireless Ad Hoc Networks. In: Proc. of the 25th IEEE ICDCSW'05. (2005)

[7] Neumann, J.V.: Various techniques used in connection with random digits. In Taub, A.H., ed.: John von Neumann, Collected Works. Volume 5. Pergamon Press, Oxford (1951) 768–770

[8] Jaromczyk, J., Toussaint, G.: Relative neighborhood graphs and their relatives. Proc. of IEEE **80**(9) (1992) 1502–1517

[9] Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of the ACM **36**(2) (1989) 335–348

[10] Rizzo, L.: Effective erasure codes for reliable computer communication protocols. ACM Computer Communication Review **27**(2) (1997) 24–36



(a) GHT–put (b) Q-NiGHT–put

(c) GHT–get (d) Q-NiGHT–get

Figure 7: Mean and standard deviation of the costs (number of hops) of put and get with Gaussian distribution (Q-NiGHT uses the enhanced GFG).

[11] Seada, K., Helmy, A.: Efficient and robust geocasting protocols for sensor networks. Computer Communications **29**(2) (2006) 151–161

[12] Albano, M., Chessa, S., Nidito, F., S.Pelagatti: Q-night: Adding QoS to data centric storage in non-uniform sensor networks. Technical report, Dipartimento di Informatica, Università di Pisa (2006)

[13] Kim, Y., Govindan, R., Karp, B., Shenker, S.: On the pitfalls of geographic face routing. In: Proc. of ACM/SIGMOBILE DIAL-M-POMC 2005, Köln, Germany. (2005)

[14] Newsome, J., Song, D.: GEM: Graph EMbedding for Routing and Data-Centric Storage in Sensor Networks Without Geographic Information. In: Proc. of the First International Conference on Embedded Networked Sensor Systems, Los Angeles, (2003) 76–88

[15] Bian, F., Govindan, R., Schenker, S., Li, X.: Using hierarchical location names for scalable routing and rendezvous in wireless sensor networks. In: SenSys '04, New York, (2004) 305–306

[16] Perkins, C.E., Bhagwat, P.: Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: SIGCOMM '94, New York, (1994) 234–244

[17] C. Bettstetter. On the connectivity of ad hoc networks. The Computer Journal **47**(4), (2004).

[18] L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti. Localized techniques for broadcasting in wireless sensor networks. In Proc. of DIALM-POMC '04, New York, (2004).

[19] Liu, J., Zhao, F., Petrovic, D.: Information-directed routing in ad hoc sensor networks. In: WSNA '03: Proceedings of the 2nd ACM Int. Conf. on Wireless Sensor Setworks and Applications, New York, NY, ACM Press (2003) 88–97

[20] Arad, N., Shavitt, Y.: Minimizing recovery state in geographic ad-hoc routing. In: Proc. of MobiHoc 2006, Florence, Italy (2006) 13–24

[21] Caruso, A., Chessa, S., De, S., Urpi, A.: GPS free coordinate assignment and routing in wireless sensor networks. In: Proc. IEEE Infocom'05, Miami, (2005)