Dealing with Non Uniformity in Data Centric Storage for Wireless Sensor Networks: supplementary material

Michele Albano, *Member, IEEE,* Stefano Chessa, *Member, IEEE,* Francesco Nidito, and Susanna Pelagatti

I. Introduction

This document contains some supplementary material of the paper. In particular, Section II briefly introduces the routing protocol GPSR [1] and the data centric storage system DCS-GHT [2]. Section III presents the details about the load unbalance in DCS-GHT. Section IV gives details on the Periodical Refresh Protocol of LB-DCS. The error of the density sampling performed by LB-DCS and the costs of Broadcast, Stripes, and FatStripes protocols for the dissemination of network density are given in Section V. Finally, Section VI presents the analysis of data availability in LB-DCS.

II. A review of GPSR and DCS-GHT

A. The GPSR routing protocol

GPSR is a geographical routing protocol that uses two operation modes: *greedy* and *perimeter*. The greedy mode routes packets using every wireless link between the nodes of the WSN, while the perimeter mode operates on a planar subset of the network. For this purpose GPSR periodically computes a planarized topology of the WSN by using a distributed computation of the Gabriel Graph (GG) or of the Relative Neighborhood Graph (RNG).

Normally GPSR forwards a packet using the *greedy mode*, in which a packet is routed progressively closer to its destination at each hop. If the packet reaches a node whose neighbors are all farther than itself to the destination (i.e. it finds a void region), GPSR switches to the *perimeter mode*. In perimeter mode, a packet is forwarded in the planarized topology in order to turn around the void region. In this mode, the packet is forwarded clockwise along the nodes belonging to the perimeter surrounding the void. If the destination (x, y) does not correspond to any sensor, the perimeter mode of GPSR turns along the inner perimeter around (x, y), then it concludes that (x, y) is unreachable and it discards the packet. As a side effect, GPSR also identifies the inner perimeter of sensors around (x, y) and the sensor closest to (x, y) (that belongs to the home perimeter). In DCS-GHT the inner perimeter of sensors around (x, y) is called *home perimeter*, and the sensor closest to (x, y) is called *home node* for (x, y).

In the rest of this paper, we assume that GPSR is used to route packets, although other geographic routing protocols can be used. Note also that in some cases GPSR may fail to deliver packets. An analysis of this problem and possible solutions can be found in [3].

M. Albano, S. Chessa, F. Nidito and S. Pelagatti are with Dipartimento di Informatica, Università di Pisa, Pisa, Italy

M. Albano is also with Instituto de Telecomunicações (IT), Aveiro, Portugal

S. Chessa is also with Istituto Scienza e Tecnologia dell' Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy

B. DCS-GHT

DCS-GHT [2] associates a meta-datum k to each datum d and it offers to the sensors' applications the two primitives `put(`d`,`k`)` to store the pair (d,k) and `get(`k`)` to retrieve all data whose meta-data matches k.

The `put(`d`,`k`)` primitive first computes the hash of the meta-datum k to obtain a pair of coordinates (x, y), then it routes a packet containing the pair (d,k) towards (x, y) by means of GPSR [1]. In general, GPSR will not be able to reach (x, y) since in that location there will not be any sensor with high probability. However GPSR will identify the home node for (x, y) and the home perimeter around (x, y). At this point, instead of discarding the packet, DCS-GHT stores a copy of the pair (d,k) in all the sensors in the home perimeter.

Data retrieval is performed by means of the `get(`k`)` primitive. This primitive computes the hash of k to obtain the coordinates (x, y), then, by means of GPSR, it sends a request for any data matching meta-data k towards (x, y). The request eventually reaches a sensor in the home perimeter that, in turn, replies to the request by sending its stored data that match k.

III. Load unbalance in DCS-GHT

In this section, we analyze the behavior of DCS-GHT. DCS-GHT performs data storage by replicating a datum in all the sensors belonging to a home perimeter, which in turn is obtained from the hashing of the meta-datum. The hash function used by DCS-GHT assumes Uniform distribution of meta-data and sensors. Moreover, the size of the home perimeter is determined by GPSR and it depends on topological properties of the WSN, thus its size is not controlled by DCS-GHT. We claim this behavior causes a serious load unbalance in the storage of data with consequent data losses on overloaded sensors. The following simulation experiments assess such unbalance and investigate its main causes.

In the simulations, we consider a WSN deployed in a square area of $200 \times 200 m^2$, and a transmission range of the sensors of $10m$. In all these simulations, we consider a network density (expressed as the average number of neighbors per sensor) in the range $[7, 30]$, and we consider both Uniform and Gaussian distribution of the sensors. The simulator implements DCS-GHT with GPSR and planarization with GG and RNG. In

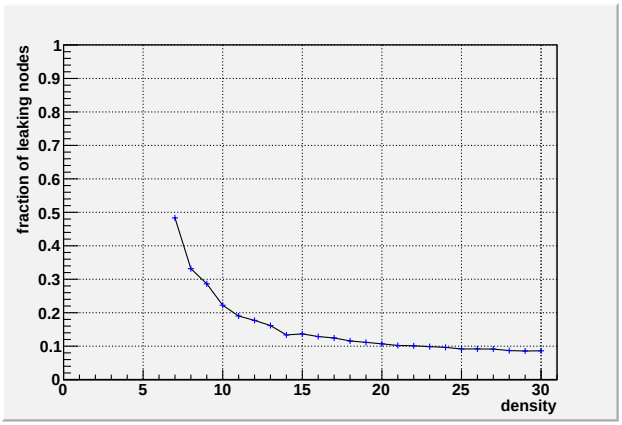


Fig. 1. Fraction of sensors that leak some data for GHT.

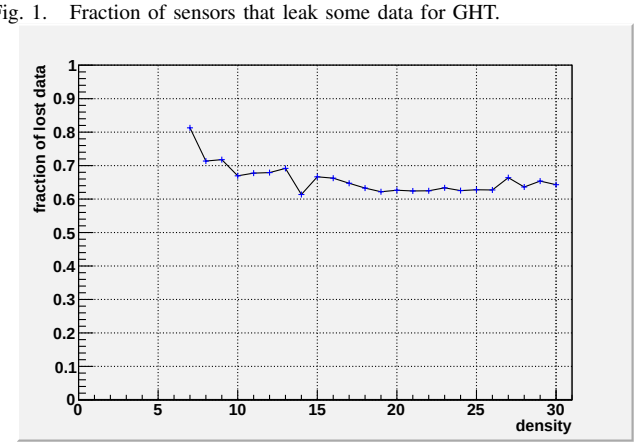


Fig. 2. Fraction of data that is leaked out of sensors for GHT.

each simulation run, the simulator generates a set of network topologies (according to the chosen network distribution). For each topology, the simulator generates a number of `put` operations and it computes the average storage load of the sensors and the average size of the home perimeter. The simulator iterates the runs until the output reaches a 99% confidence interval that is less than 1%

The results presented in the next sections refer only to the case where GPSR uses GG for planarization, however the results obtained with RNG are similar.

A. Average data leakage

The first set of simulations are aimed at evaluating the average load of the sensors and the leakage of stored data. Data leakage occurs when a sensor serves too many `put` operations and it saturates its memory. To study such phenomenon, we consider sensors with storage capacity of $512KB$ (as it is the case of the Crossbow Mica family [4]). For each network generated in the experiments, the simulator executes a number of `put` operations for each sensor, each one accounting for 8 bytes (i.e. each `put` operation requires the storage of 8 bytes of data in each sensor in the home perimeter). In these simulations, we assume that each sensor produces during its lifetime a total of $2,100$ `put` operations, i.e. each sensor produces an amount of data to be stored that corresponds approximatively to $1/30$ of its memory capacity. We assume that, once a sensor is requested to store a new datum but its memory is full, it drops an older datum. This means that once a sensor reaches its maximum storage capacity it starts dropping data whenever it is requested to store new data.

For each simulation experiment, the simulator computes the number of sensors that do not leak data and the total amount of data lost.

Figure 1 shows the fraction of sensors that leak at least one datum and Figure 2 shows the fraction of overall data that is actually lost. In both figures, the x axis reports the density of the simulated networks. The result is that a substantial fraction of sensors leak data, since around 60% of data may get lost, regardless of network density. This is an important issue since all lost data could have been successfully stored.

B. Effect of WSN topology on load balance

The data leakage of DCS-GHT highlighted in the previous subsection is due to the way DCS-GHT chooses the sensors where to store the data, and in particular, in the fact that it does not control the size of the home perimeters.

Figures 3 and 4 show the mean and standard deviation of the size of the home perimeters obtained with Uniform distribution of sensors.

The figures show that, as the network density increases, the average number of nodes in a perimeter decreases. However, the actual number of nodes is extremely variable. One reason for such variability is that in some cases the `put` selects a point (x, y) that lies outside of the network boundary, and in such cases the home perimeter is very large since it includes all the sensors on the network outer border. In order to reduce this phenomenon, a simple solution is to restrict the area where (x, y) can fall. This solution was investigated in [5] where it was shown that, even cutting away the 5% of the WSN area from each border (assuming a WSN deployed on a square region this accounts for about 19% of the total WSN area) the variability of the size of the perimeters remains high.

Figures 5 and 6 show the mean and standard deviation of the size of the home perimeters obtained with Gaussian distribution of the sensors. It is clear that in this case the behavior of DCS-GHT is much worse than that with Uniform distribution, because DCS-GHT uses a uniform hash function to select the perimeters, independently of the distribution of the sensors in the WSN area.

IV. Periodical Refresh of replica set in LB-DCS

This section discusses the use of a Periodical Refresh Protocol (PRP) in LB-DCS which is an adaptation of the PRP protocol introduced in the original DCS-GHT paper [2]. The reason for using a PRP protocol is that the topology of the network may change for different reasons, such as mobility of sensors or sensor failures. Such changes may result in unavailability of some data, either because the sensors storing a datum may all fail, or because they are not accessible by the `get` anymore. Furthermore, changes in the network distribution may condition the hash function used in the storage and retrieval protocol, thus making the stored data unavailable.

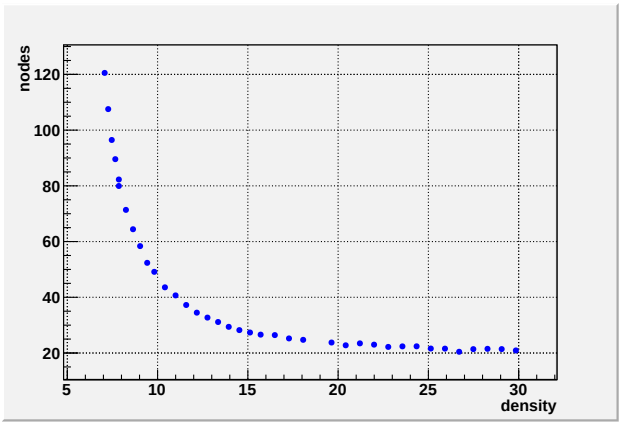


Fig. 3. Mean of perimeters (number of nodes) measured for different densities with GG planarization.

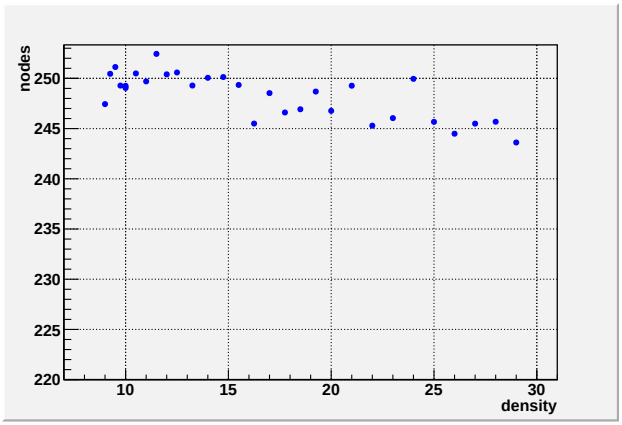


Fig. 5. Mean of GHT perimeters for different WSN densities, Gaussian network distribution, GG planarization.

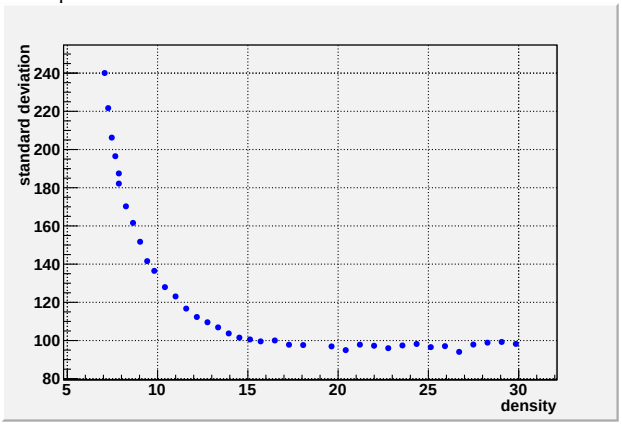


Fig. 4. Standard deviation of perimeters (number of nodes) measured for different densities with GG planarization.

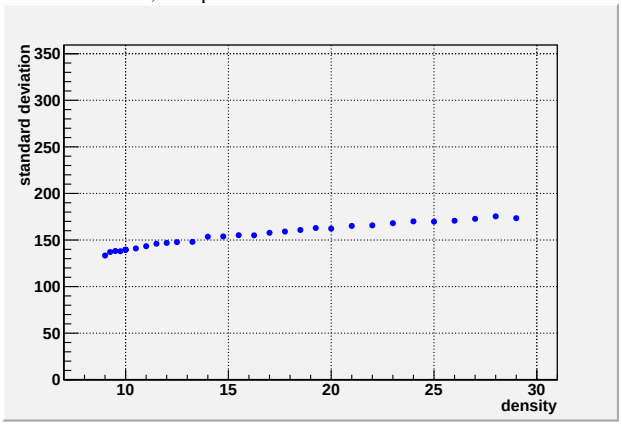


Fig. 6. Standard deviation of GHT perimeters for different WSN densities, Gaussian network distribution, GG planarization.

In the PRP protocol, the sensors in the replica set for a given key k (these sensors store all the data associated to this key) periodically generate *refresh packets*. The refresh packets contain k and all the stored data associated to k, and they are routed as in the `put` operation. Thus, if the topology of the network is not changed (and there have been no sensor failures), a refresh packet originated by a sensor in a replica set will identify (and thus refresh) the same replica set. Otherwise, if the topology has changed (or there have been sensor failures), the refresh packet will identify a new replica set the data associated to k will be stored.

In order to reduce the number of refresh packets, the sensors in the replica set use different timers to generate such packets. If a sensor in the replica set receives a refresh packet before generating its own packet then it restarts its timer (and thus it delays its own refresh packet). The timers are tuned in such a way that no sensors in the same replica set generate a refresh packet at the same time.

Each sensor has another timer associated to each different key it stores. This timer is used to let a sensor delete all the data associated to a key k when it is not in the replica set for k anymore. This timer is reset whenever the sensor receives a new `put` packet for the key k or when it receives a refresh packet for the same key. Since its expiration time is larger than the timer for the generation of refreshes for k, once it expires it means that the sensor has not received any refresh packet and thus it is no longer in the replica set for k.

The actual tuning of all of these timers is beyond the scope of the paper as it can be achieved as in [2], for this reason we disregard this issue.

We evaluate the cost of the PRP in a scenario where the WSN topology changes slowly with respect to the frequency of generation of the refresh packets. In this scenario, most of the sensors belonging to the original replica set of a meta-datum k are still in the replica set when a refresh packet is generated. When the first sensor in the replica set performs a `put` operation, the other sensors receive the refresh packet and hence they do not perform a `put` operation on their own.

Let us assume that, over the lifetime of the WSN, the data corresponding to a given meta-datum k are refreshed only once. The simulations considered the execution of the refresh protocol in a similar scenario to that considered in the other simulations (squared WSN area with a $200 \times 200m^2$, communication range of 10 meters, mean density of the WSN in the range $[7, 30]$, and QoS parameter of LB-DCS set to $q = 7$). The simulator performs a `put` operation per sensor, and reports the mean number of MAC level send and receive operations performed by sensors in the WSN for each meta-datum.

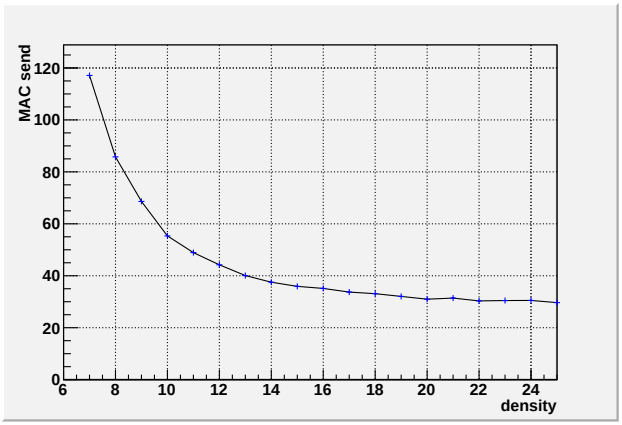


Fig. 7. MAC-level sends for the refresh protocol of LB-DCS.

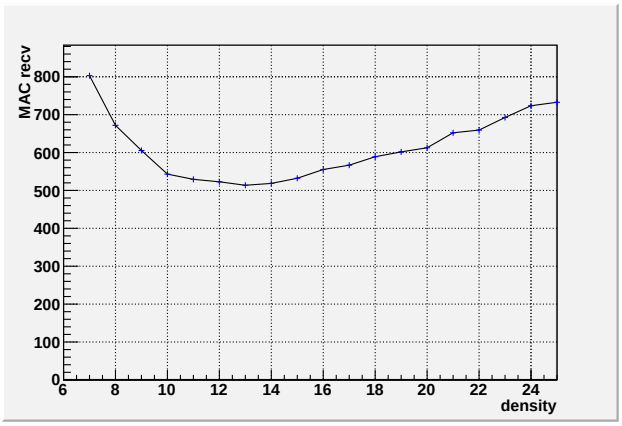


Fig. 8. MAC-level receives for the refresh protocol of LB-DCS.

Figure 7 reports the number of MAC-level send for the PRP protocol, while Figure 8 reports the number of MAC-level receives. From the simulations, it results that the cost of PRP is slightly less than the original `put` operation, since it is not necessary to traverse the WSN to reach the region where the home node is located.

V. Simulating density sampling: some insights on errors and costs

This section presents the simulation results on the error in the approximation of the network density computed by LB-DCS and on the cost of Broadcast, Stripes and FatStripes protocols.

For each simulation run, 100 WSNs are randomly generated, and the three protocols (Broadcast, Stripes and FatStripes) are run on them to evaluate the error in the density estimation and the messages exchanged by these protocols. The simulator iterates the runs until the outputs reach a 99% confidence interval that is less than 1%. We report here the main results obtained with 200 sensors with a transmission range of 25m in a WSN area of $100 \times 100m^2$, and network density (expressed as the average number of neighbors per sensor) equal to 39. Note that we obtained similar results for different sizes of the WSN area and/or network density.

In figures 9, 10 and 11, we report the error due to rebuilding a Gaussian distribution using 10×10 regions. We measure the error using the Mean Square Error (MSE), a scalar quantifying the distance between the real distribution and the distribution computed by the rebuilding algorithm. If we denote with $D^R = (d^R_{ij})_{n \times n}$ the matrix of real region densities (and recalling that $D = (d_{ij})_{n \times n}$ is the matrix of the estimated region densities), the MSE is defined as the average of the square of the errors on each region, $MSE(D) = \frac{\sum_{ij}(d_{ij} - d^R_{ij})^2}{n^2}$. Figure 9 shows the real distribution, that is the actual number of nodes present in each region. Figure 10 shows the distribution approximated by our algorithm, which is quite close to the real one. The error measured (Figure 11) is under 0.00018, and generally much lower.

In Figure 12, we show the errors measured by approximating a Uniform, Gaussian and Hill distribution when varying the number of regions used. The error is under 0.0035 with 9 regions and falls under 0.0001 with 25 regions.

Figure 13 compares the behavior of Broadcast, Stripes, and FatStripes in terms of the average number of messages generated, with respect to different numbers of sensors that query the sentinels. Since Broadcast is proactive, its performance is independent of the number of sensors' requests: all the sensors receive the sentinels' data ahead of time, so no real "request" is generated. Stripes has a good behavior when a small number of sensors ask for WSN density, but it suffers from its "unicast" communication when most of the sensors request this information. On the other hand, FatStripes gets the best of both worlds, since it is reactive and hence it is cheap when a few sensors query the sentinels, but it fully exploits the broadcasting characteristic of the physical medium to disseminate density information as much as possible. Hence, when many sensors request density information, most of them already have it because of the communication performed by past requests.

Figures 14 to 19 analyse the case in which all the sensors query the sentinels, in order to evaluate the worst case scenarios of the protocols. The histograms show the average number of sensors that send/receive a given number of MAC-level frames, including the frames necessary to elect the sentinels for the regions.

The histograms in figures 14 and 15 show the simulation results in the case where the sentinels broadcast the information about the density sampling. In particular, Figure 14 displays the average number of sensors that send a MAC level frame, while Figure 15 displays the average number of sensors that receive a MAC level frame. Figures 16 and 17 report the behavior of the Stripes protocol in the same scenario, in particular Figure 16 reports the average number of sensors that send a MAC frame with Stripes and Figure 17 reports the average number of sensors that receive a MAC frame with Stripes. Finally, figures 18 and 19 report the average number of sensors that send and receive a MAC frame in FatStripes.

From these figures it is evident that with FatStripes most of the sensors do not have to send any MAC frame (on the average more than 150 in this setting), while with Stripes the number of sensors that do not send MAC frames is much less (around 50 in this setting), and with Broadcast most of the sensors send on the average 4 MAC frames. Regarding

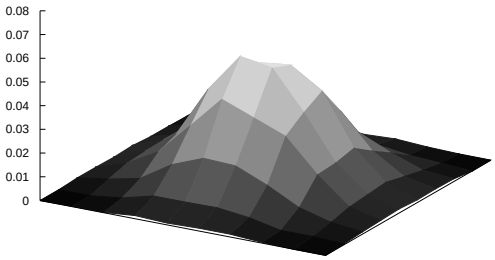


Fig. 9. The real density D^R of sensors in the regions.

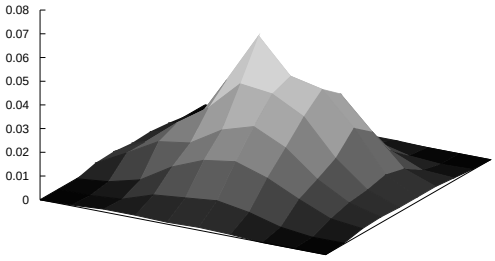


Fig. 10. The density D computed by the rebuilding algorithm.

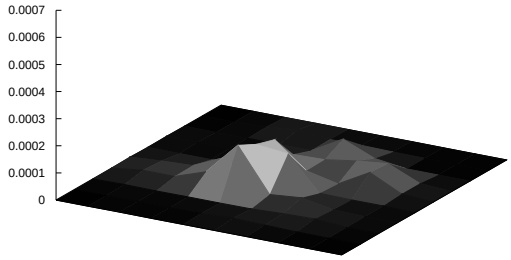


Fig. 11. Mean Square Error between D and D^R.

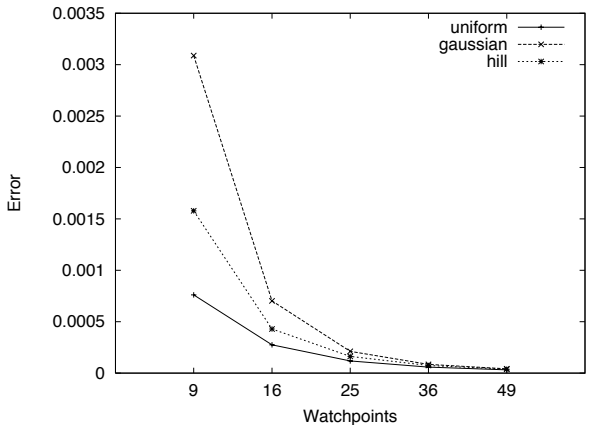


Fig. 12. Mean Square Error of estimated distribution (a scalar) measured for different values of p.

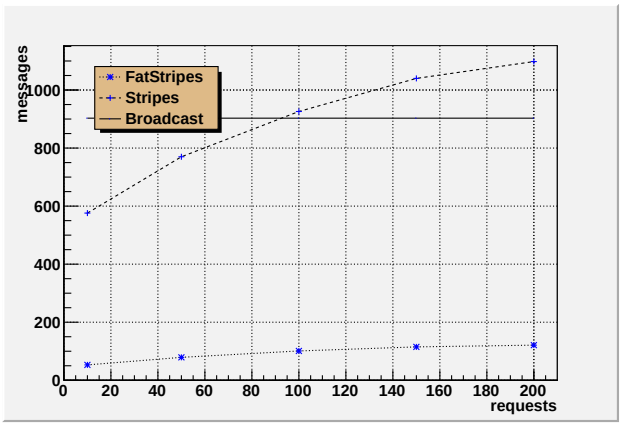


Fig. 13. Number of messages sent, against number of sensors that requested WSN density, $100 \times 100m^2$ WSN area, 200 sensors.

received frames, by comparing figures 15, 17, and 19 it results that with FatStripes the sensors receive on the average up to 50 frames, while this number raises to around 500 with Stripes and to around 180 with Broadcast.

VI. Data availability of LB-DCS

This subsection discusses data retrieval capabilities of LB-DCS in a scenario where the WSN is still connected but some of its nodes have failed. Let us assume that a datum d, whose meta-datum is k, is stored in a replica set of q sensors, let O be the destination $(x, y) = h(k, f)$, and let H be the home node for k located in (x_H, y_H). Let also assume that $q - 1$ sensors in the replica set fail and that sensor A is the only surviving sensor in the replica set. Let also assume that the whole network is still connected after the faults. Let now assume that a sensor issues a `get` request to retrieve the data that regard the meta-datum k. This request traverses the network and finds a new perimeter P' surrounding the point (x, y) (we refer to [1] for details on how P' is generated). If sensor A is in the new perimeter then it will respond to the `get` request, and LB-DCS protocol is able to provide data availability for data regarding meta-datum k.

Since A is the only surviving sensor that was in the original ball $B_{(x_H,y_H)}(\overline{r})$, A does not to belong to P' *if and only if* there is an edge connecting two sensors in the perimeter P', say B and C, that intersects the segment connecting O to A. There are three possible cases in which this may happen, that are shown in figures 20, 21 and 22. In the first case (Figure 20) B and C both belong to the ball of H which was computed during the data dispersal phase; in the second case (Figure 21) only one of these sensors (say C) belongs to the ball around H, and in the third case (Figure 22), neither B nor C belong to the ball around H. In the first case, due to the property of

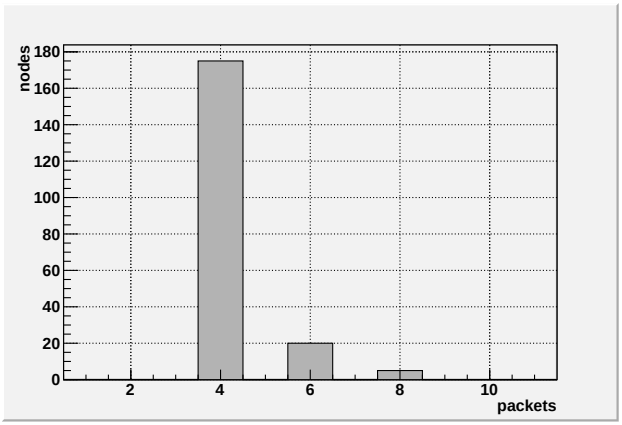


Fig. 14. Number of nodes against number of sent frames, density sent in Broadcast, $100 \times 100m^2$ WSN area, 200 sensors.

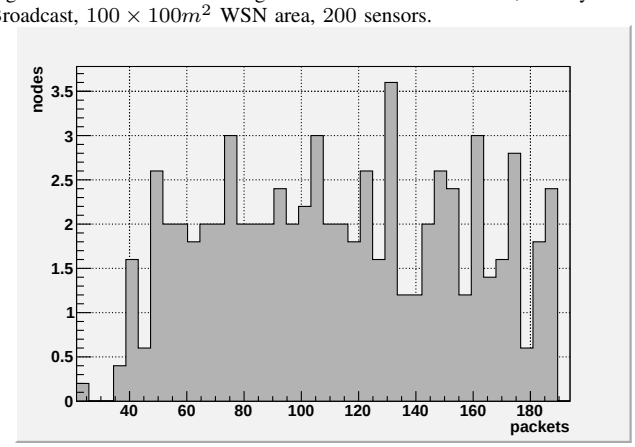


Fig. 15. Number of nodes against number of received frames, density sent in Broadcast, $100 \times 100m^2$ WSN area, 200 sensors.

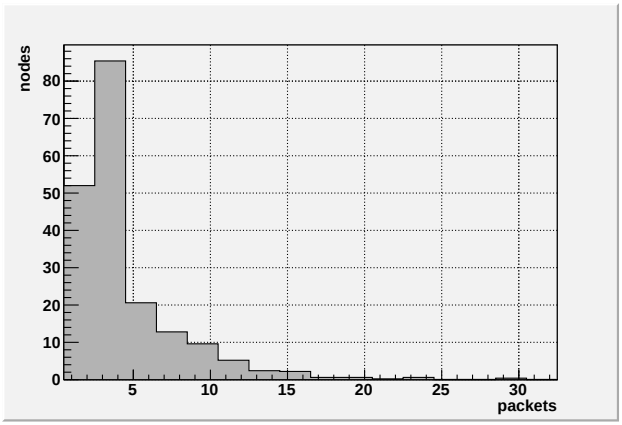


Fig. 16. Number of nodes against number of sent frames, density sent using Stripes, $100 \times 100m^2$ WSN area, 200 sensors.

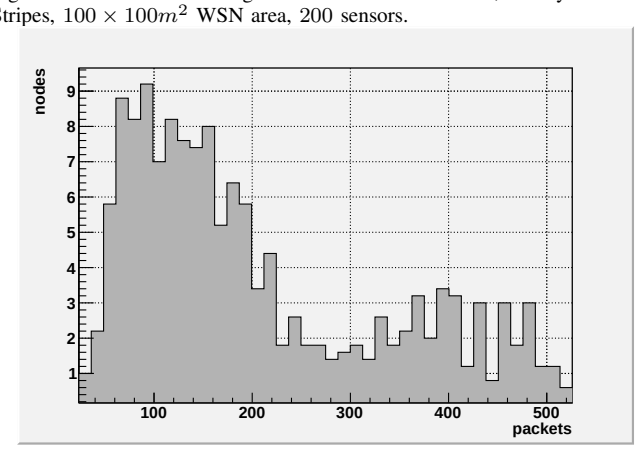


Fig. 17. Number of nodes against number of received frames, density sent using Stripes, $100 \times 100m^2$ WSN area, 200 sensors.

the dispersal protocol both B and C must be farther than A from O, in the second case only C must be farther than A from O, while in the third case there is no such a constraint. In any case, the link BC must be comprised in the planarized topology of the WSN, B and C must be close enough to communicate ($|BC < r|$), and both B and C must be farther from O than H, since H was the closest node to O.

We set up a simulation to evaluate the number of cases in which these configurations may occur. To this purpose we simulated $10,000,000$ data storage scenarios with a WSN area of $200 \times 200m^2$, communication range of the sensors set to 10 meters, density of the WSN in the range $[7, 30]$, and QoS parameter of LB-DCS set to $q = 7$. Each simulation run executes a `put` operation, simulates $q - 1$ sensor failures in the replica set, and, finally, it reports the occurrences of the three cases showed in figures 20, 22 and 21. The results of the simulations showed that these three cases of failure never occurred. Thus the simulation shows experimentally that data is lost with a probability that is negligible for most applicative scenarios.

Michele Albano received his BSc degree in Physics in 2003, and his BSc, MSc and PhD degrees in Computer Science in 2004, 2006 and 2010 respectively, all of them from the University of Pisa, Italy. He was visiting researcher at Universidad de Malaga in 2007, at Stony Brook University in 2009, and before being a researcher he worked as software engineer and wireless technologies specialist in private companies in the period 2001-2006. In 2006 and 2007 he was involved in EU funded projects SMEPP and XtreemOs, and he is now a postdoc researcher at the Instituto de Telecomunicações - Pólo de Aveiro, Portugal, working on EU funded projects C2POWER and PEACE. He has co-authored more than 20 papers published on international journals and conference proceedings, he is reviewer for several international journals and conferences. His main research interests are in the areas of wireless networks and peer-to-peer networks.

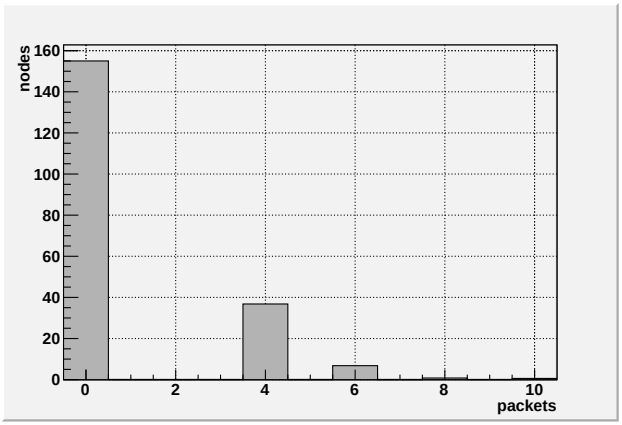


Fig. 18. Number of nodes against number of sent frames, density sent using FatStripes, $100 \times 100m^2$ WSN area, 200 sensors.

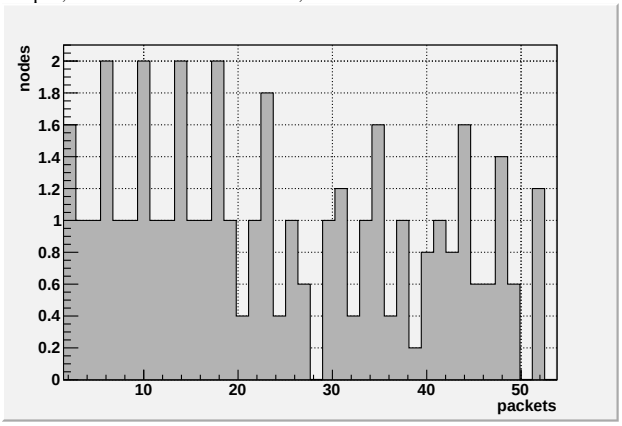
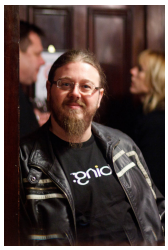


Fig. 19. Number of nodes against number of received frames, density sent using FatStripes, $100 \times 100m^2$ WSN area, 200 sensors.

Stefano Chessa received his MSc and PhD degrees in Computer Science from the University of Pisa, Italy, in 1994 and 1999, respectively. Since 2000 he is Assistant Professor at the Computer Science Department of the University of Pisa and since 2003 he is also Research Associate at the ISTI/CNR Institute (Information Science and Technology Institute). He has been involved in many national and European projects. In particular he has coordinated the CNR project Management of Data in Wireless Sensor networks(MaD-WiSe). He has also participated to the EU FP6 SatNex, SMEPP, InterMedia, PERSONA projects and to the EU FP7 universAAL project. He has co-authored more than 80 papers published on international journals and conference proceedings, he is reviewer for several international journals and conferences, and he has been member of several international program committees of conferences and workshops. His research interests are in the areas of wireless ad hoc networks, and wireless sensor networks, security in wireless communications, and system-level diagnosis.



Francesco Nidito received his PhD in 2008 from the University of Pisa, Department of Computer Science. He performed his research activity at the same University and as a visiting scholar at the Northeastern University, Boston (MA) collaborating with Prof. Stefano Basagni and Andras Farago. After the Ph.D., he has been working in the search engines field, first at Ask.com and then at the Microsoft Search Technology Center of London (UK).

Susanna Pelagatti received her MSc and PhD degrees in Computer Science from the University of Pisa, Italy,in 1987 and 1993, respectively. In 1995, she joined the University of Pisa, Dipartimento di Informatica as an Assistant Professor. Since 2002 she is an Associate Professor in the same department. Her main research interests are in the areas of parallel computing, wireless and ad Hoc networks, and sensor networks. She has been involved in national and international projects and she has co-authored more than 50 papers published in international journals and conference proceedings. She has been member of international program committees of conferences and workshop.

REFERENCES

[1] Karp, B., Kung, H.T.: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In: ACM MobiCom 2000, Boston, 243–254 (2000)
[2] 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. MONET **8**(4) (2003) 427–442
[3] Frey H., Stojmenovic I.: On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In: ACM MobiCom, Los Angeles, 390-401 (2006)
[4] Crossbow Technology: http://www.xbow.com
[5] Albano, M., Chessa, S., Nidito, F., Pelagatti, S.: Q-NiGHT: Adding QoS to Data Centric Storage in Non-Uniform Sensor Networks. In: MDM 2007, Mannheim, Germany (2007).

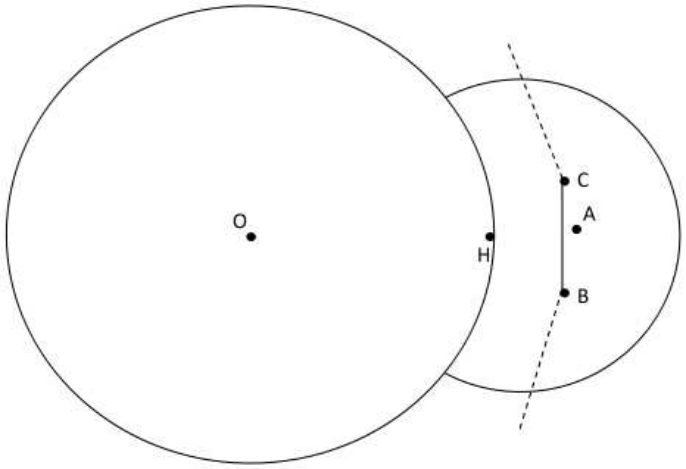


Fig. 20. Node A is cut out of the perimeter around O, when B and C are both inside the ball around H.

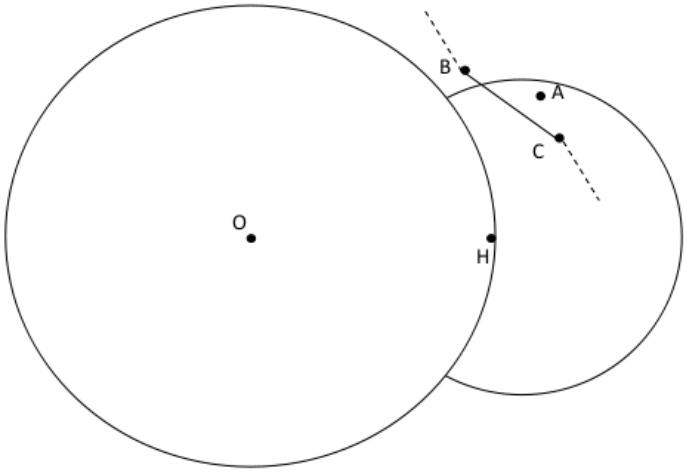


Fig. 21. Node A is cut out of the perimeter around O, when B is outside the ball around H, and C is inside the ball.

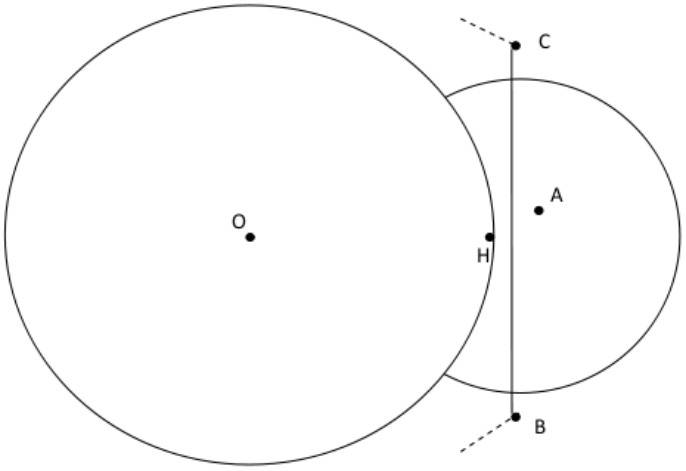


Fig. 22. Node A is cut out of the perimeter around O, when B and C are outside the ball around H.