Privacy-Preserving Environment Monitoring in Networks of Mobile Devices

Lorenzo Bergamini, Luca Becchetti, Andrea Vitaletti

Privacy-Preserving Environment Monitoring in Networks of Mobile Devices*

Lorenzo Bergamini, Luca Becchetti, and Andrea Vitaletti

DIS - Sapienza University of Rome - Via Ariosto 25, 00185 - Rome Italy,
bergamini,becchett,vitale@dis.uniroma1.it

Abstract. Small portable devices carried by users can be exploited to provide useful information through active cooperation. To guarantee privacy, the system should use the information on the device without tracing the user. In this work we address this scenario and we present a privacy-preserving distributed technique to estimate the number of distinct mobile users in a given area. This could be of crucial importance in critical situations, like overcrowded airports or demonstrations where monitoring the number of persons could help organizing emergency countermeasures. In our envisioned scenario users periodically transmit a sketch (i.e. a summary) of the users they met in the past obtained applying suitable, duplicate-insensitive hash functions. Sketches are based on hash functions, so they also provide some kind of privacy, as it is not possible to hardware to show that it works on real devices.

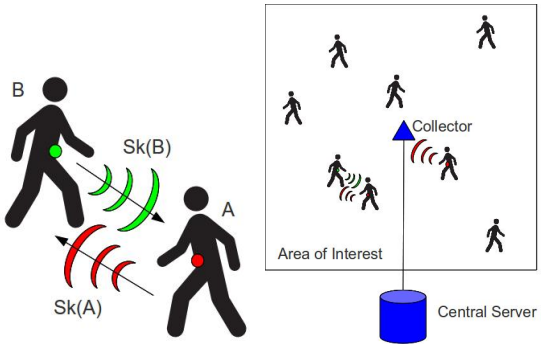
Keywords: Distributed systems, Privacy-preserving, Distinct counting

1 Introduction

In a recent report by IBM (http://www-03.ibm.com/press/us/en/pressrelease/33304.wss), the use of smart phones as mobile sensors in envisioned to be one of the top five innovations that will change our lives in the next five years. Projects such as the Reality Mining (http://reality.media.mit.edu/), Senseable city (http://senseable.mit.edu/) and the Human Dynamics Laboratory (http://hd.media.mit.edu/) at MIT are already exploiting such technologies to have a better understanding of the behavioral patterns of users in their environment. Such approach is providing an unprecedented amount of high quality data that will actually help in better understanding the social behavior of users and their "use" of the space. However, since personal information is managed, security and privacy support is a primary concern which is often neglected in current systems.

Contribution of the paper. In this paper we present a fully distributed approach to generate a map of the utilization of an area of interest over time by a set of mobile users, without disclosing any sensitive information about the users themselves. The only assumption we made is that each participant is identified by a unique id, either a meaningful one, such as the IMEI of a mobile phone or even a random one. When two users meet, they exchange their sketches (see figure 1a) , namely a compact summary which allows the involved nodes to estimate the number of distinct nodes they have interacted with so far. This information, is then occasionally communicated to a central

* Partially supported by EU STREP Project ICT-215270 FRONTS and EU STREP Project ICT-257245 VITRO



(a) When two user meet, they exchange their sketches.

(b) Users moving in an area of interest deliver their sketch to a collector.

Fig. 1: The reference scenario

server through a collector (see figure 1b). The Server displays the number of users in an area over time, and can be used by decision makers in architecture and urban planning or to calibrate/validate predictive models about pedestrians' behaviors. We stress that while this technique has been previously used in other settings [11,17,14], to the best of our knowledge this is the first study that considers and provides experimental evidences of the suitability of this technique in mobile environments.

The proposed technique has been designed to run on a network of resource constrained mobile devices, such as smart phones and/or wireless sensor nodes[1]. In this context, the limited resources available at the sensing devices, namely finite and non-replaceable energy source, limited computational power and limited storage capacity, require the adoption of optimized solutions both in terms of complexity, memory requirements and amount of exchanged information. These requirements make the trivial approach based on bit-masks unfeasible, since it requires an amount of memory that is linear in the size of the monitored population and the explicit notification of the id of the nodes encountered in the past has to be communicated with all the other nodes. Moreover, the techniques we propose entail a degree of privacy, since the counting task is performed locally by each involved entity and only on the basis of the aggregate information exchanged among the nodes. This implies that the central server only knows the number of distinct nodes in an area, but not their identity and only a limited amount of memory (i.e. logarithmic in the size of the population) is necessary to store the relevant information. We stress that the small size of the necessary information is important for two reasons: a) it reduces the amount of necessary memory, b) it reduces the size of data exchanged among nodes.

[1] These two devices are traditionally seen as separate islands, but it is possible to bridge them with a simple XBee Programmable USB stick such as that implemented by the WSNRG. http://www.sensor-networks.org/

Structure of the paper. In section 2 we provide an overview of similar works, which are in some sense connected with our paper; in section 3 we provide a short summary of the works in [11] and [14] and we explain how we adapted them to the WSN context; in section 4 we report and discuss some simulation results; in section 4.3 we describe the results we obtained with a real implementation on sensor nodes and the techniques we used to cope with the constraints imposed by the devices and finally in section 5 we remark our main results and we propose some future works.

2 Related Work

Our main motivating applications in this work arise in the field of mobile networks of nodes with communication and/or sensing capabilities. The monitoring scenario we consider is complicated by the fact that nodes can be in movement, so that we cannot assume an underlying routing or data aggregation infrastructure. We review below literature that is more closely related to the problem we consider.

Aggregation techniques in sensor networks. The authors in [12,10,19] report the deployment of such networks in a wide range of scientific, security, industrial and business applications. Examples include climatological and environmental monitoring, traffic monitoring, smart homes, fire detection, seismic measurements, structural integrity monitoring and habitat monitoring to name a few. In the scenarios outlined above, single individual values are usually not of great relevance. In fact, users are more interested in the quick extraction of succinct and useful synopses about a large portion of the underlying observation set. Consider, for example, the case of a temperature sensor network. We would like to be able to continuously monitor the entire infrastructure and efficiently answer queries such as "What was the average temperature over the entire terrain during the last 12 hours?", or "Are there any specific clusters that have reached dangerously high temperatures?". Trying to collect all data monitored by the sensors would be unrealistic in terms of bandwidth, power consumption and communication intensity. So, the canonical approach is to compute statistical *aggregates*, such as max, min, average, quantiles, heavy hitters, etc., that can compactly summarise the distribution of the underlying data. Furthermore, since this information is to be extracted and combined across multiple locations and devices, repeatedly and in a dynamic way, *in-network aggregation* schemes [16] must be developed that efficiently merge and quickly update partial information to include new observations. Also notice that, computing aggregates instead of reporting exact observations, can leverage the effect of packet losses and, generally, network failures, which are common phenomena in wireless networks of tiny artifacts.

Streaming algorithms Streaming algorithms have received considerable attention in the recent past, due to increasing gap between the rate of data generation and the computational, storage and communication resources available to process them. An excellent overview of the area is [17]. In a centralized setting, there has been a large body of work aimed at providing provably accurate and resource efficient heuristics for the computation of aggregate statistics of common interest. The resulting body of literature is vast and we only provide pointers to key references in the area that are more closely related to the subject of this study.

While some statistics, such as max, min and sum, can be easily maintained efficiently with small (constant) memory in a centralized setting, for others this is not the

case. One such statistic that received early attention is maintaining the number of distinct items that have been observed over a data stream. It turns out that this is a special case of maintaining the frequency moments over a stream. The basic tool we consider in this paper is a *counting sketch*, i.e., an approximate but accurate, composable and duplicate insensitive counter of the number of distinct items appearing in a stream. The counting sketch was first proposed in the seminal paper of Flajolet and Martin [11] and then reviewed and modified in subsequent work [5,6]

Streaming algorithms in networks of sensing devices. The extension of streaming techniques to distributed settings has mainly been motivated by applications in sensor networks [8,18], both assuming the presence of a reliable aggregation tree and in the more realistic scenario of *multi-path* routing protocols, where the partial information transmitted by each node is aggregated across many different paths towards the base-station. [18] proposed a formal framework to study multi-path aggregation, called *summary diffusion* and they also gave formal necessary and sufficient conditions for the correct estimation of order insensitive statistics. [7] and [18] proposed multipath aggregation techniques based on the counting sketch by Flajolet and Martin [11]. We emphasize that, differently from the above contributions, our emphasis is on networks of mobile agents.

3 Problem and model

We describe below , i) the problem we want to solve, ii) the model of communication we assume and iii) the algorithm we use to solve it.

Problem. The basic problem we are interested in is estimating the overall number of distinct nodes in the network. Our goal is to make this process faster, by exploting the fact that nodes moving in the network occasionally come close enough to exchange information. The idea is having each node maintain an estimate of the number of distinct peers encountered so far. We assume that the area of interest contains a special, fixed collector node, which receives data from any agent occasionally moving in its vicinity and forwards them to a central server. Intuitively, in this way the collector will need to come in contact with a smaller number of mobile nodes to perform its task. Our experiments show that in this way convergence to an accurate estimate can be extremely fast.

Communication Model. We consider a set V of n distinct nodes, each equipped with a sensing device tagged with a unique ID within the network. After initial deployment, at time 0 nodes start moving randomly in a given area of interest. At any time t, only a subset of the node pairs can communicate, this subset depending on a number of factors, e.g., pairwise distance, wireless channel quality etc. We assume that the subset of nodes that can communicate at each round t is modeled by an undirected graph $G = (V, E_t)$, where the set of vertices is fixed and $(u, v) \in E_t$ if and only if u and v can exchange information during round t. This model is pretty general and can account for multiple aspects, such as communication range and collision resolution protocols in wireless networks. In Section 4, we assume that nodes move essentially according to the random way-point model [15] and we consider the simple model in which $(u, v) \in E_t$ if and only if their distance in round t is at most a given radius R. We make the minimal assumption that in a round t of communication a node can only broadcast a message to the set of its immediate neighbors, as defined by E_t. Our model has an immediate prac-

tical counterpart in wireless networks in which communication occurs over a broadcast channel, while in other cases a round of communication may actually involve multiple transmissions along point-to-point connections.

Overview of the approach. Each node in the network meets other nodes (possibly multiple times) over time and keeps a *sketch* of the set of *distinct* nodes encountered so far, i.e., a data structure that in "small" (polylogarithmic) space allows to accurately estimate the number of distinct nodes encountered so far. Considered a specific node u, we denote by Sk^u the sketch maintained by u. At a high level, each node u performs the following actions: i) when u meets another node v, it receives Sk^v and updates Sk^u accordingly; ii) after updating its local sketch, u broadcasts it to all nodes v, such that $(u, v) \in E_t$. In order to limit the amount of messages circulated in the network, in step ii) a node broadcasts its local sketch only if the new estimation of the number of distinct nodes observed so far exceeds the previous estimate by more than a given threshold. We next describe the data structure maintained by each node to keep track of the set of other distinct nodes encountered during its movement in the area. Clearly, not all sketches are suitable for our application. In particular, the presence of multiple data paths opens the possibility that a node receives the same information multiple times, which can dramatically affect the accuracy of the estimation. Following previous work, we consider sketches that are *composable* and *duplicate insensitive* [18,8,13]. Consider an order insensitive statistic of interest (i.e., whose value does not depend on the order in which events are observed), such as the number of distinct nodes that we consider in this paper. A sketch Sk is *composable* if the following holds: let $Sk^u(S_1)$ be the sketch maintained by u at time t if it so far met the subset S_1 of nodes in the network. Assume u receives sketch $Sk^v(S_2)$ from v and let $merge(Sk^u(S_1), Sk^v(S_2))$, be the sketch resulting from the aggregation of $Sk^u(S_1)$ and $Sk^v(S_2)$, where $merge(\cdot)$ is a suitable sketch aggregation function that depends on the statistic of interest and the sketching algorithm used to maintain it. Sk is composable if it is always the case that $Sk^u(S_1 \cup S_2) = merge(Sk^u(S_1), Sk^v(S_2))$. I.e., the sketch obtained by u after aggregating the sketch received from u is the same as the one u would have computed, had it encountered the whole set $S_1 \cup S_2$. The basic tool we consider is a *counting sketch*, i.e., a composable and duplicate insensitive counter of the number of distinct items (nodes in our case) observed in a stream of data. In the remainder, we consider the approach of Flajolet and Martin [11,9,5] and we use the phrase "FM sketch" (from the initials of the two authors) to refer to any implementation of the original counting sketch of [11]. The use of composable and duplicate insensitive sketches has been considered previously for restricted and static distributed settings, especially for data aggregation at the sink of a sensor network [18,8,13]; in this paper we are considering a dynamic network of moving wireless devices. FM sketches use a simple bitmap-based approach: every node ID (regarded without loss of generality as an integer value) is hashed onto a bit of a bitmap of length $k = O(\log M)$ bits, where M is an upper bound on the size of the universe (the number of nodes in the network in our case). In practice, $M = 2^k$, e.g., $k = 64$). The hash function $h(\cdot) : [n] \rightarrow \{0, \ldots, \log_2 M - 1\}$ used to this purpose is such that the probability of hashing onto the r-th bit is 2^{-r}. The bit to which the ID under consideration is hashed becomes 1 if it was not already. Considered any time t, let r denote the position of the least significant bit that is still 0 in the bitmap: it turns

out that 2^r is a good estimator for N_t, the number of distinct nodes encountered up to time t. To improve accuracy, we define a sketch as a collection of m bitmaps built as described above, each using an independently chosen hash function. If r_s denotes the least significant bit that is still 0 in the s-th bitmap, then $\frac{1}{m}\sum_{s=1}^{m} r_s$ is an accurate estimator of $\log_2 N_t$ if m is large enough (see theorem below). Clearly, the proposed sketch is composable and duplicate insensitive: Considered two sketches $Sk^u(S_1)$ and $Sk^v(S_2)$, $Sk^u(S_1)\texttt{OR}Sk^v(S_2)$ is clearly the sketch corresponding to $S_1 \cup S_2$ and it is the same for both u and v. The scheme outlined above achieves excellent bounds in terms of resource efficiency and precision, as stated by the following

Theorem 1 ([9,11,6]). *Let $0 < \epsilon, \delta < 1$. At any point in time, every node u maintains an estimate $\hat{C}$ of the number N_t of distinct nodes encountered so far using $O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$ memory words, such that:*

$$\mathbf{Prob}\Big[|\hat{C} - N_t| > \epsilon N_t\Big] \leq \delta.$$

Remarks. Some remarks are in order. First of all the naive strategy consisting in having nodes communicate their identities directly (and exclusively) to the sink when they eventually come close enough to it might require a long time till the sink has an accurate estimate. Furthermore, we assume for simplicity that the aggregate of interest has to be estimated at a special sink node, but the techniques we propose allow to address more general settings in which there is no distinguished sink. Finally, this approach can be immediately extended to estimate other aggregates using the same communication paradigm, for example sum or average (see [7]). Last but not least, one might argue that using bitmaps instead of the (approximate) sketches we propose could allow to exactly count the number of nodes in the network. This approach is not scalable, since it implies bitmaps of size proportional to the number of nodes in the network. On the contrary, the sketches we propose have size that is polylogarithmic in the network size and the same holds for the body size of the messages exchanged between nodes.

4 Experiments and Simulation Scenario

In this section we describe the experimental scenario we use to evaluate the accuracy of our algorithm. We imagine a situation in which a set of users is inside an area of a given size (we made experiments on two different sizes), moving according to the random way point model, and, after a given period of time, we want each node to estimate the total number of users inside the considered area and thus to communicate this information to a central entity. Please note that the central entity is used here just for collecting the results, and is never involved in any computation; we enforce the fact that the counting algorithm is completely distributed. We assume a very basic communication model in which in every round of communication each node of the network performs the following actions:
1. if a message is received, the node merges its local_sketch with the sketch it has just received from its neighbors (thanks to properties discussed in 3)
2. the node updates its estimate of the number of distinct users encountered

3. if the updated value exceeds the most recently propagated value by more than a given threshold the current sketch is transmitted

We consider a set of N distinct users, each of them equipped with a sensor device tagged with a unique ID inside the network. After the initial deployment, at time T users start moving randomly. In this section we describe how we set the paramenters for the simulations, and the metrics we want to evaluate. The simulation experiments have been performed using the Shawn simulator [1] and we ran 50 simulations for each set of parameters.

Simulation Parameters		
World Size 25x25		World Size 100x100
Number of Nodes $\{25, 50, 100, 200\}$		Number of Nodes $\{25, 50, 100, 200, 500, 1000\}$
Number of Rounds $\{20, 50, 100\}$		Number of Rounds $\{25, 50, 100, 200, 500, 1000\}$
Threshold $\{0, 10, 25, 40, 50, 60, 75, 90, 100\}$		Threshold $\{0, 10, 25, 40, 50, 60, 75, 90, 100\}$

Table 1: Simulation Parameters

We change the size of the area and the number of nodes to verify the behavior of the algorithm under different situations of density (that strongly impacts on the accuracy of the distinct counting). We used different sets of nodes and rounds between the two area sizes, because in 100x100 the network density is so low that in the 20 rounds experiment no node was meeting any other node in the network, and thus the experiment was senseless. The number of rounds is related to the maximum number of messages that can be generated by each node: we assumed that in every round a node has the possibility to send a message depending on the value of the threshold: the higher the number of rounds, the higher the possibility for nodes to communicate. The threshold parameter is of the utmost importance as it limits the total number of messages sent by each node (and thus reduces the total amount of messages in the network), but also because it allows a node to send or not an update message. For example, when the threshold is 0 we expect the maximum number of messages in the network as each node transmits a message at every round, but we also expect the maximum accuracy on the estimate, as the information is continuosly refreshed. When the threshold is increased, it is possible that, especially in sparse networks, a node is never able to collect sufficient information to pass the threshold and thus to send its updated information, as we recorded in some experiments on the 100x100 area. We measured 2 fundamental metrics: **Error (in percentage)** on the estimated number of distinct elements compared to the real number of distinct users (e.g. considering a network of 200, if the estimated number is 190, we have an error of 3%); **Number of messages** managed in average by each node; we remark that this parameter takes into account both the messages sent and those received by each node in average. Please note that, implicitly, the number of messages sent in average by each node gives us an estimate of the energy consumed.

4.1 Results - World size 25x25

20 Rounds. In the first graph reported in figure 2a the number of simulation rounds is 20, while the threshold and the number of nodes in the network varies. On X axis

we reported the different values of threshold we used; on left Y axis we reported the number of messages, while on the left Y axis, we reported the error on the estimate in percentage. (calculated as $1 - accuracy$, where $accuracy$ is defined as $ESTIMATE\_DISTINCT/REAL\_DISTINCT$). The full lines thus represents how the number of messages varies with the threshold, while the dotted line represents how the error varies with the threshold.

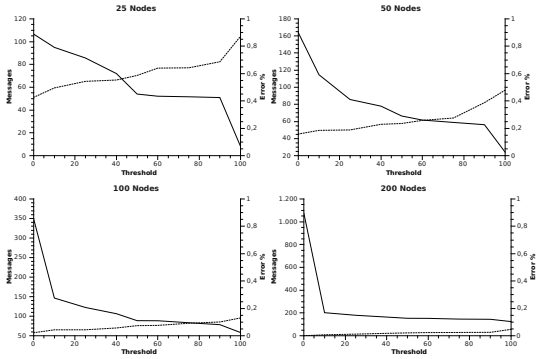
As can be observed in figure 2, the error on the estimate is high for low network densities (25, 50 users), while it is lower as the density increases (100, 200). This is an expected behavior, and it is due to the fact that our algorithm spreads the information between nodes; with an higher density the information is diffused faster and to a greater number of nodes per round. The second aspect to underline is the importance of the threshold. A low threshold (e.g. 0) means that in every round each node sends an update messages, then the error is reduced, since the information is refreshed continuosly but the number of messages is extremely high. Having a look in particular at the 100 and 200 nodes scenarios, it is evident how even a very small threshold value, like 10 reduces the number of messages by a factor of about 10. Increasing the value of the threshold has a significant impact on the error of the estimate when the density is low, while it doesn't seem to disrupt too much the values obtained in a dense network. Thus, we can infer that when the density of the network is sufficiently high the performance in terms of estimate accuracy is extremely good. Putting together the two needs of minimizing the number of messages by increasing the threshold, and limiting the error on the estimate by reducing it, it is possible to identify on the graph the "optimal" value threshold* when the 2 lines intersect. This means, for example, that threshold* for the 50 users case is 60, while the threshold* for the 200 users case is the maximum, 100, as the two lines doesn't intersect. In this case we can obtain an error of $\approx 5\%$ with an average of ≈ 80 messages.

50 rounds and 100 rounds.We can immediately note how increasing the simulation rounds has a positive effect on the accuracy in sparse networks: in these scenarios, even the 25 and 50 users experiments are very accurate when the sending threshold is 0. On the other hand, we record a significant increasing of the number of messages in the 100 rounds 200 users scenarios, confirming the intuition that when the network is dense, the threshold must be kept high. Looking at the same figure 1c, by the way, we can also notice that the two lines never intersect, thus indicating that with an high density we can use the maximum as threshold* without disrupting the accuracy of the estimate!

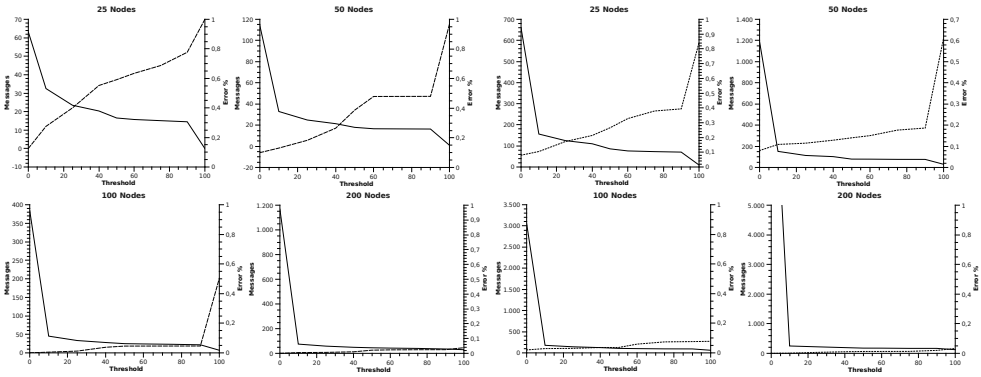
4.2 Results - World size 100x100

In the second set of experiments we increased the size of the monitored area, thus reducing the density of users. This has a strong impact on the performance of the algorithm, as we will show.

50 Rounds.The first figure refers to a monitoring of 50 rounds. As we can easily notice looking at figure 3, with a low number of users the error is constantly around 1, meaning that no user is able to communicate its sketch to others and this is due to the fact that the monitored area is so large that if the monitoring is performed for a very limited period, users almost never meet, and thus they cannot update their sketches. This intuition is confirmed by the fact that when the number of users increases $(500, 1000)$, with a very



(a) 20 Rounds



(b) 50 Rounds

(c) 100 Rounds

Fig. 2: Area size 25x25

low threshold we obtain an estimate, whose error is however high ($\approx 20\%$ for 1000 users and threshold 0), but at least some interactions between users happen.

100 Rounds.In this second case we monitored the same area for 100 rounds. We obtained that with a threshold $= 0$, for all the number of users we obtain an estimate with an high error, but different from 1, thus indicating that some interactions took place even in the 25 users case. Once again, when the density of the network is high, we obtain good estimations; in particular for a network of 1000 nodes, the error on the estimate is around 30% for the maximum value of the sending threshold (100). In all the other cases, the value of the threshold must be kept as low as possible, in order to reduce the error.

1000 Rounds.This long-running experiment is the most meaningful for the 100x100 scenario, as we obtain results similar to those discussed for the 25x25 case. With a very small threshold (say 10 or even less), we are able to obtain a reasonable (error below $\approx 30\%$) estimation for "sparse" networks $(25, 50, 100)$ managing less than 1000 messages for each user; if a greater accuracy is required, then a threshold of 0 must be used, causing an average number of $\approx 1000 - 1500$ messages (which is, anyway,

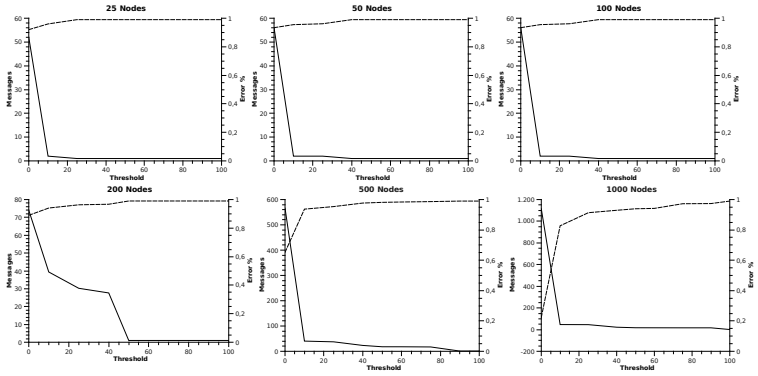


Fig. 3: 50 Rounds 100x100 World

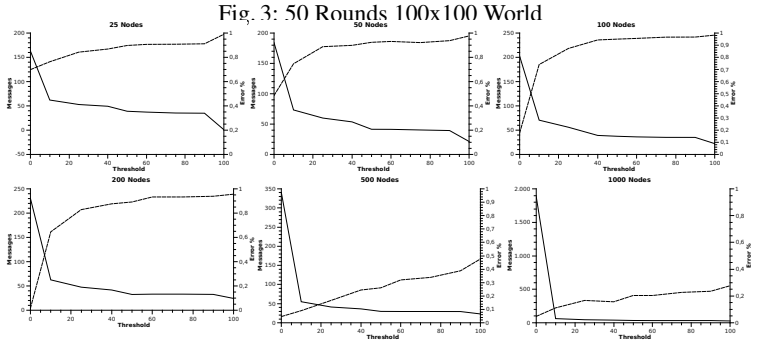


Fig. 4: 100 Rounds 100x100 World

a reasonable amount of traffic for devices like TelosB, as we discussed in section 4.3). It is interesting to notice how the number of messages dramatically drops when a very small threshold is used; this means that in a dense network the vast majority of messages carry redundant information, and thus can be avoided. Looking for example at the 1000 users case, it's easy to note that when the threshold is 0 the number of messages is out of scale (it is around ≈ 26000), while when a threshold of 10 is adopted, this number drops to less than 1000 messages, and the error on the estimate increases just to $\approx 4\%$. In conclusion, we can say that for sparse networks, a longer monitoring period and a low threshold are needed to obtain an accurate estimate of the population; as the density increases, to keep the number of messages low, the sending threshold must be increased, but this doesn't impact on the accuracy of the estimation, as we have shown in both the area sizes.

4.3 Experiments on real hardware - proof of concept

To assess the feasibility of our approach in practice, we implemented the proposed algorithm on state-of-art commercial devices, namely TelosB sensor nodes [4] running TinyOS [2]. We experimentally assessed the performance of the proposed techniques, in terms of computational, storage and communication resources, using 30 static nodes from Motelab testbed [3]. We decided to make experiments on tiny devices rather than on more powerful smart-phones, following the idea that if it these techniques work on

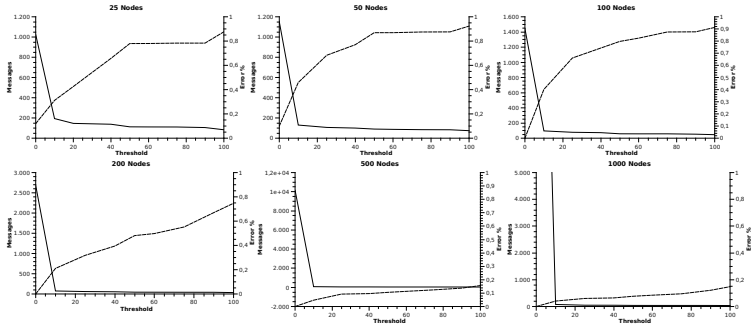


Fig. 5: 1000 Rounds 100x100 World

challenging resource-constrained devices, then it's even likely that they work fine on more powerful hardware. The main difference between simulations and real implementation is in how sketches are managed. In TinyOS we must also cope with strong limitations on the size of messages, thus we decided to proceed as follows: each node stores some (20 in our implementation) uint32_t variables, named local_sketch1,2.....n representing sketches; when sending a message these values are transmitted and converted in binary form by the receiver; then all the operations described in section 4 are performed and finally the sketches obtained in this way are converted back in decimal form for the next transmission. The rest of the algorithm is the same. The main difference between simulator and real nodes is that in the simulations the sketch is transmitted as a bitmap (represented as a NUMHASH-dimensional matrix), while on real nodes we exchange decimal values that are then converted to binary bitmap for update on the receiver-side. This procedure is completely transparent to the final user of the algorithm. With this assumption, we can build up a packet consisting of 82 bytes at the application layer (4 bytes * 20 sketchs, plus 2 bytes of node ID), plus 17 bytes of header.

The maximum size of a TinyOS packet is 128 bytes including header, thus our 99 bytes packets can be transmitted with no problems. We will not present detailed results on this scenario, as our aim is just to show that our technique meets the strong requirements of low-power devices, but in any case we will discuss some interesting aspects. First of all, the error we recorded was perfectly compatible with the one we obtained from the simulation: in a scenario like the one we tested, using 30 nodes we obtained an estimate of 33 nodes with a threshold of 10, thus the error was 10%. Second point, the binary image is $\approx$ 36 Kb, where the programmable memory of Telosb nodes is 48 Kb, thus the memory is sufficient for our algorithm. Last point, the computational power of the Telosb device is sufficient to perform the calculations required by the algorithm we presented: when receiving a packet a node is able to perform all the required operations in a few milliseconds.

5 Conclusions and future work

In this work we presented a fully distributed privacy-preserving counting technique to estimate the number of users in a given area. The privacy issue is also taken into account, as there is no way to retrieve the users from the summary information (the

sketch). Our simulations showed that depending on the density of the network and on a parameter we named sending threshold, it is possible to estimate with a low error the real number of users, with each device managing a limited amount of messages. We also showed in section 4.3 that these techniques can be successfully implemented on real devices (TelosB), and the preliminary results we collected from a real testbed are comparable to the simulation result. As a future work, our idea is to implement these techniques on smartphones and to run large-scale experiments, to confirm the results of the simulations and the testbed. Additionally, we want to implement algorithms to calculate other statistical aggregates (discussed in section 2) to produce a complete privacy-preserving application for distributed monitoring.

References

1. http://www.swarmnet.de/shawn/.
2. http://www.tinyos.net/.
3. http://motelab.eecs.harvard.edu/.
4. Tmote sky datasheet. http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf.
5. N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
6. Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 1–10, Cambridge, Ma, USA, 2002. Springer-Verlag.
7. J. Considine, M. Hadjieleftheriou, F. Li, J. Byers, and G. Kollios. Robust approximate aggregation in sensor data management systems. *ACM Trans. Database Syst.*, 34(1):1–35, 2009.
8. J. Considine, F. Li, G. Kollios, and J. W. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, pages 449–460, 2004.
9. G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006*. IEEE Computer Society, 2006.
10. D. Culler, D. Estrin, and M. Srivastava. Overview of sensor networks. *IEEE Computer*, 37(8):41–49, 2004.
11. P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.
12. M. M. Gaber. Data stream processing in sensor networks. In *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer, 2007.
13. M. Hadjieleftheriou, J. Byers, and G. Kollios. Robust sketching and aggregation of distributed data streams. Technical Report 2005-011, CS Department, Boston University, 2005.
14. J.Considine, F.Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. 2004.
15. D. B. Johnson and D. A. Maltz. *Mobile Computing - Dynamic source routing in ad hoc wireless networks*, chapter 5, pages 153 – 181. Kluwer Academic Publishers, 1996.
16. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.
17. S. Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.
18. S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 250–262. ACM, 2004.
19. S. Subramaniam and D. Gunopulos. A survey of stream processing problems and techniques in sensor networks. In C. Aggarwal, editor, *Data Streams: Models and Algorithms*, Advances in Database Systems, chapter 15, pages 333–352. Springer, 2007.