Privacy preserving gate counting with collaborative Bluetooth scanners

Nelson Gonçalves[1], Rui José[2], and Carlos Baquero[3]

[1] Universidade do Minho, Portugal,
[2] Centro Algoritmi, Universidade do Minho, Portugal,
rui@dsi.uminho.pt
http://www.dsi.uminho.pt/~rui/
[3] DI/CCTC, Universidade do Minho, Portugal,
cbm@di.uminho.pt
http://gsd.di.uminho.pt/cbm/

Abstract. Due to its pervasiveness and communication capabilities, Bluetooth can be used as an infrastructure for several situated interaction and massive sensing scenarios. This paper shows how Bluetooth scanning can be used in gate counting scenarios, where the main goal is to provide an accurate count for the number of unique devices sighted. To this end, we present an analysis of several stochastic counting techniques that not only provide an accurate count for the number of unique devices, but offer privacy guarantees as well.

Keywords: Privacy, Gate Counter, Bloom Filters, Hash Sketches

1 Introduction

As Bluetooth becomes more and more pervasive, there is a growing potential to leverage on the possibilities offered by Bluetooth scanning as a flexible infrastructure for situated interaction and a general purpose platform for massive sensing and actuation in urban spaces. Bluetooth sensing is based on a discovery process through which a device can inquire about the presence of other nearby devices. If those devices are in *discoverable* mode, they will respond with their address, and possibly additional information, such as the device name, the device type (e.g. cellphone or computer) and available services. A Bluetooth scanner is a device that periodically scans nearby devices, registering and timestamping the observations and making them available to other applications and systems. Multiple Bluetooth scanners spread all over a city could thus serve as collection points for Bluetooth sightings, providing a major tool for observing, recording, modelling and analysing the city, physically, digitally and socially [8].

A large scale infrastructure of that nature is not likely to emerge as a single-domain initiative. There would not be a killer application that could by itself justify such huge investment. However, there is already a large number of Bluetooth scanners in urban environments. They are owned by many entities and

they serve very diverse purposes, such as proximity marketing, device localization or OBEX-based interaction. These same scanners could be used, without any additional cost, as nodes in a large scale collaborative sensing infrastructure. Each node would still scan for its own purposes, but it would also share part of the generated data with a central service that would then be able to produce aggregate information of mutual interest. This collaborative path could enable large scale Bluetooth sensing to quickly enter mainstream.

A major obstacle, however, is how to enable this type of large scale sensing without creating an overwhelming privacy threat. A Bluetooth scanner registers the Bluetooth addresses, 48-bit MAC, of the devices that have been sighted. Some devices are able to switch between multiple Bluetooth addresses, but for most cases this will be a reliable and permanent unique identifier for sighted devices and by extension to the respective owners. While a single scan of proximate devices is not in itself much of a problem, a systematic registration of Bluetooth sightings, especially when done at multiple locations, would have the potential to become a large scale tracking system. Relatively simple processes could be put in place to detect the presence, movements and patterns of individuals as well as co-location patterns between people. For privacy, a precautionary measure should avoid permanent storage and dissemination of Bluetooth addresses.

1.1 Bluetooth-based collaborative gate counting

In this study, we address one of the most common forms of urban sensing: counting unique visitors across multiple gate counters to measure the flow of people across the urban setting. In this scenario, a potentially very large set of heterogeneous and autonomous nodes support the counting process by acting as Bluetooth-based gate counters.

Conceptually, a gate is a virtual line across a street, and gate counting is the process of counting the number of people crossing that line. Each Bluetooth node is modeled as a gate counter that counts the number of unique Bluetooth addresses observed during a certain period. The use of Bluetooth as an enabling technology for gate counting has been extensively explored in [8] to establish the flows of people at sampled locations within a city over the course of a day. A Bluetooth-based gate counter does not really count all the persons passing-by, but only those who are carrying discoverable Bluetooth devices. Still, this is enough to make a reasonable correlation, using baseline data, to estimate the overall traffic.

A Bluetooth-based gate counter needs to recognize subsequent sightings of the same entity. Repeated sightings of the same device can be very common, not only because people can be passing-by multiple times, but also because of persistent devices. Instead of scanning through a line, discovery is actually performed in an area and thus any device in that area, possibly in nearby buildings, would be repeatedly discovered. Results from empirical studies with known static and transient devices suggest that a transient device typically appears for up to 90 seconds while it crosses a gate [8]. A proper gate counting process would thus needs to account for this and filter persistent devices.

A gate counter should also be able to answer questions like "how many different people were seen in a given gate in the last 24 hours ?" or "what was the number of visitors of an amusement park during visitors peak hours?". To comply with this requirement, gate counters should be able to distinguish its readings over time.

The main challenge, however, is how to enable collaborative counting between multiple independent gate counters, while providing appropriate privacy guarantees as well as low communication costs. To be able to count unique entities, we need to identify multiple counts of the same entity at different nodes, to make sure that the same device sighted at two different gates will be counted only once. Imagine, for example, a city festival with multiple gate counters operating at various locations to count the number of visitors to the festival. The simple sum of individual gate counts would clearly overestimate the number of people since many of them would be spotted at multiple gates. We thus need some technique that works across multiple gate counters and is able to provide an aggregate count of the unique device addresses that have been spotted in the entire set of gate counters. Comparing the plain addresses observed at different gates would immediately solve this problem, but is not appropriate for privacy preservation.

1.2 Objectives

In this paper, we explore the use of stochastic summarizing techniques as a privacy preserving approach to enable Bluetooth-based gate counting of unique entities across multiple nodes. The objective is to assess to what extent these techniques are able to address the specific requirements of this distributed gate counting model and inform the design of large scale Bluetooth sensing systems.

We have identified the sensing requirements for this scenario and established a number of key criteria for assessing the various alternatives. We have then conducted an experimental study in which we compared how multiple types of stochastic summarizing techniques would behave across multiple variants of our gate counting scenario. The results provide a strong foundation for the development of these large scale Bluetooth sensing infrastructures, identifying major trade-offs and the implications of key factors such as cardinality and support for merge operations.

2 Probabilistic counters for privacy-enhanced gate counting

A common strategy in privacy-enhancing techniques is to reduce the data collected to what is absolutely needed for a particular purpose. In this gate counting scenario, we need to count devices and the ability to check if a particular device has already been counted before. Moreover, we need to be able to do this across a set of autonomous nodes. This means that we are not interested in information about the number or duration of sessions that each device generates at each gate.

Probabilistic counters provide an interesting solution to this problem. They are flexible enough for estimating the overall number of unique sightings with some controllable accuracy, without ever keeping the plain Bluetooth addresses.

In this section, we make a brief description of the algorithms that were implemented and tested for the Gate Counter scenario.

2.1 Bloom Filters

Bloom Filters were created in 1970 [2] by Burton Howard Bloom. They are a simple and memory efficient probabilistic data structure for set representation where membership queries are allowed. A Bloom Filter consists in a bit array (initially set to 0) and k consistent hash functions. Those hash functions are used to map an element into several array positions. The bits at those positions are then either set to 1 (in order to add the element to the Filter), or checked to see if all are set to 1 (in case they are, the element is considered part of the set). In short, the main properties of Bloom Filters are:

- Small memory footprint in comparison with the memory needed to represent the actual set,
- The add and check membership operations have $O(k)$ complexity, where k is the number of hash functions, therefore independent from the number of elements represented in the set.
- False positives are possible, but their occurrence can be controlled,
- No false negatives.

In this particular Gate Counter Scenario we used both the standard version of Bloom Filters[2] and Scalable Bloom Filters [1]. Scalable Bloom Filters are a variant of Bloom Filters that can dynamically adapt to the number of elements stored while respecting a maximum false positive probability, which is chosen at the beginning. Even though it's not their main goal, Bloom Filters (and Scalable Bloom Filters) can also be used to estimate the cardinality of multisets. This is accomplished using the ratio of bits set to 1 in the bit array. The size m of the bit array supporting the bloom filter is linear, $O(N)$, with the number N of elements to count.

2.2 Hash Sketches

Hash Sketches are a simple probabilistic data structure with which we can obtain the cardinality of sets. Much like Bloom Filters there are several variants of this algorithm. Despite being different, all of these variants have at least one bit array and use some kind of hash function to map elements to positions in the aforementioned array(s).

Hash Sketches have a small memory footprint, the ability to estimate the cardinality in a single pass over the set, as well as $O(1)$ complexity to add a new element and $O(m)$ complexity to estimate the cardinality (where m is the size of the bit array).

In the Gate Counter Scenario, we tested several versions of sketches: LogLog Sketches [3], HyperLogLog Sketches [7], Linear Counting Sketches [9], Robust In Network Aggregation Linear Counting Sketches (RIA-LC) [5, 4] and Robust In Network Aggregation Dynamic Counting Sketches (RIA-DC) [4].

LogLog Sketches are similar to the Probabilistic Counting algorithm presented in [6] since both use several small bit arrays (called buckets) instead of a single bit array. The main difference is that LogLog Sketches are much less memory consuming at the expense of some accuracy. Their name derives from the fact that each small bit array has size close to $log(log(N))$, being N the number of distinct elements. The estimate of the cardinality is obtained using the average of the several small bit arrays.

HyperLogLog Sketches are an improvement over LogLogSketches. Using the same number of bits as LogLog Sketches, HyperLogLog Sketches are able to provide more accurate results. According to the authors in [7], this improvement is accomplished by using *harmonic means* instead of *geometric means* in the evaluation function.

Linear Counting Sketches use only a single bit array. Their name comes from the fact that they have $O(N)$ size, meaning their size grows linearly with the number of distinct elements N. When compared to LogLog Sketches, in Linear Counting Sketches the size is a drawback, but they work better for small cardinality sets.

Both RIA-DC and RIA-LC sketches are based on the Linear Counting Sketches. While RIA-LC can be seen as a slightly improved/simplified version of Linear Counting Sketches, RIA-DC Sketches have the unique ability to merge sketches of different sizes. However, this ability comes with a price. RIA-DC Sketches assume that there is no overlap of elements belonging to different sketches, meaning that if we merge two different RIA-DC sketches with elements in common, those elements will be counted twice in the final aggregate.

3 Comparative Analysis

The presented techniques, should now be evaluated against the specific requirements of gate counting scenarios. Considering the requirements identified in subsection 1.1 we will now analyze the proposed techniques against the following criteria: **accuracy**, **size** and **aggregation**.

3.1 Criteria

Accuracy With this criterion we want to evaluate the accuracy of the techniques, their ability to count multiple sightings of the same device only once, and the quality of their estimators. Setting the maximum standard error to 5%, $\sigma = 0.05$, we measured for all techniques the relative error (root min squared error) for a range of cardinalities, whose average of 100 runs is shown in Fig 2.

Size The size of the techniques is an important factor. The less space the technique requires, the lower will be both the costs of communication between BT scanners and their memory requirements.

Regarding this criterion, we have 2 fundamentally different types of techniques: dynamic techniques which consist of Scalable Bloom Filters and static techniques comprising the rest. Static size techniques are techniques whose size is set at the time of creation and cannot be changed afterwards. This means that we must know the maximum number of unique devices to count before hand, or at least, we must be able to assume an upper bound for that number. Dynamic techniques on the other hand don't have this drawback because they can adjust to withstand arbitrarily large cardinalities. In practical terms this means that for static size techniques, once we create an instance with a certain capacity it is not possible to change that capacity afterwards, while for dynamic techniques there is no such constraint.

To further help us in our analysis, we can look at Figures 3(a) and 3(b) which respectively depict the number of bits required by each unique element and the size spent by each technique for a range of cardinalities.

Aggregation The ability to merge counts is crucial for scenarios with multiple gates. It is a key ingredient for obtaining the *aggregate number* of individuals in a set of gates. The merge operation consists in either a bitwise OR operation (Bloom Filters, Linear Counting, RIA-DC and RIA-LC sketches) or in a max operation (HyperLogLog and LogLog sketches) of structures that make each gate's counter. In order to merge several gate counters, there are 2 conditions that must be met: all counters must be instances of the same technique and every instance must have the same parameters and capacity (equally sized bit arrays). Meeting these conditions ensures that the same unique device will mark the same positions in the several gate counters it crosses. Therefore after merging (bitwise OR or max operation) the counters (Fig.1), it is possible to obtain the aggregate number of unique elements without counting the same device repeatedly.

With the exception of Scalable Bloom Filters and RIA-DC sketches, all the techniques presented here have the ability to merge, and therefore will not count the same device more than once in aggregate counts. Scalable Bloom Filters lack the ability to merge because their size varies dynamically with the number of unique elements, therefore we cannot guarantee that the same unique device will set the same positions for different filters. RIA-DC sketches might not provide accurate aggregate results since the estimator considers there is no overlap of elements between the different counters.

Aggregation is also important as a mean for answering time related questions like "how many different people were seen in a given gate in the last 24 hours ?" or "what was the number of visitors of an amusement park during visitors peak hours?". To answer these types of questions, it must be possible to distinguish/segment counter readings over time. This can by accomplished by sensor nodes periodically making a copy of their counters followed by a reset. Those copies will keep the information about the unique devices sighted during

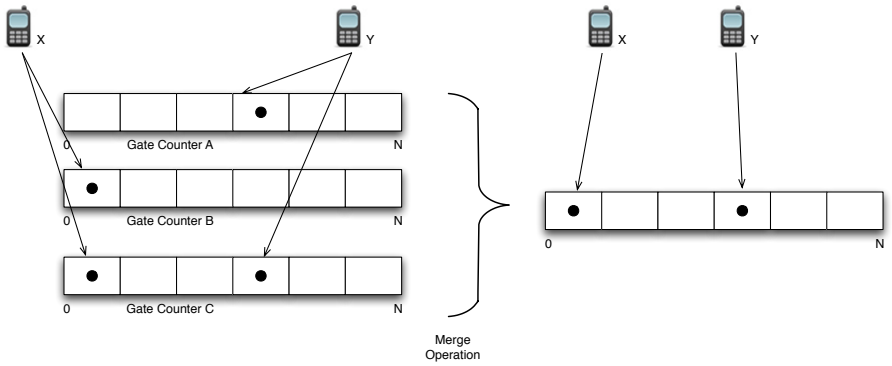


Fig. 1. Merge Operation

a certain time period. For example, considering that the rate at which counters are saved and reset would be 1 hour, the former question could be answered by merging the 24 last saved counters. To answer the latter we would need to merge the copies made during peak hours at the various nodes in the park.

To save some space we can use different time granularities, for instance, we can merge all unique counters saved during a day and obtain the aggregate count for the day, merge the counters from the last 7 days and get the aggregate count for the week, and so forth. We just need to keep in mind that because of the merge restrictions, the size of the counter that stores the unique number of devices sighted during an hour has to big enough to fit the number or unique devices seen during the entire week.

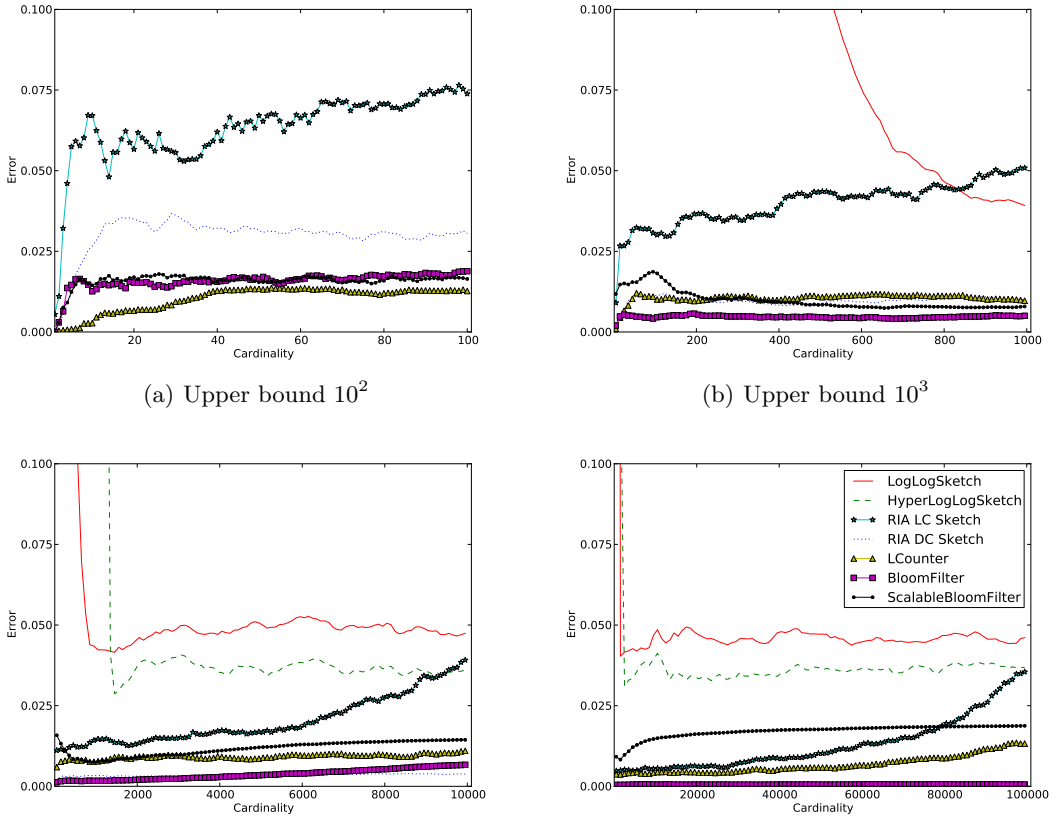
As we can see both *time segmentation* and *aggregate counting* are in fact variations of the same problem, which can only be solved with techniques that support merging.

3.2 Analysis

Using the results from our benchmark[4] shown in figures 2, 3(a) and 3(b) and having explained the different criteria we can now analyze each one of the techniques.

- **Standard and Scalable Bloom Filters** provide good accuracy for all the scenarios presented in Fig.2, never surpassing the stipulated relative error. However, they are the most expensive techniques regarding processing time and the number of bits required per element.
- **LogLog and HyperLogLog Sketches** have very low accuracy for small cardinalities, that is the reason they are not visible in Fig.2(a). Apart from

[4] Our Benchmark was built in Python, including the implementation of the several algorithms, with the exception of Bloom Filters where we used Jay Baird's implementation.



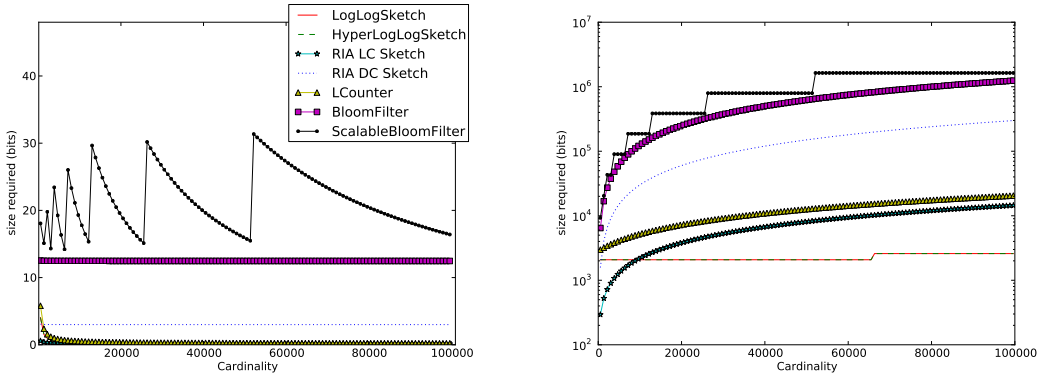
(a) Upper bound 10^2

(b) Upper bound 10^3

(c) Upper bound 10^4

(d) Upper bound 10^5

Fig. 2. Relative error of the several techniques using various upper bounds



(a) Number of bits per element used by each technique at different cardinalities

(b) Size of the several techniques for different cardinalities

Fig. 3. Size benchmarks

this issue, both these techniques are the best suited for large cardinalities due to their logarithmic growth in size (for cardinalities above 10000 these are the techniques that require less space). Between the two, HyperLogLog sketches have the advantage of being more accurate and of having a smaller variability.

– **Linear Counting Sketches** have the same good all around accuracy as Bloom Filters spending only a little more memory than LogLog sketches, therefore drawing the best from each technique.
– **RIA-LC and RIA-DC Sketches**, being based on Linear Counting Sketches, it is no surprise these techniques have good accuracy results. Furthermore and like their ancestor, they also achieve good results in the bits per element ratio.

Taking into account all that has been said, there are a few conclusions to be drawn. For scenarios where we cannot make assumptions on the maximum number of elements to count, we have to use Scalable Bloom Filters. For scenarios where there is a big discrepancy between the cardinalities of different counters and where the existence of repeated elements outside each counter can be neglected, RIA-DC Sketches are probably the best choice. For scenarios with very large cardinalities HyperLogLog Sketches are probably the correct choice since they are the most space efficient technique. Considering the expected most common scenarios, where we need accurate aggregate counts and where there is the possibility of counters with low cardinalities, the choice falls either within Linear Counting Sketches or RIA-LC Sketches. We prefer the latter since it is a slightly simplified version of the former.

As a final remark, we should emphasize the fact that all the techniques discussed in this article are more efficient (check Fig.3(a)) in terms of space than storing each unique device MAC address(48 bits).

4 Conclusion

Bluetooth devices are pervasive in most societies and the number of unique Bluetooth sightings is an adequate proxy for the number of actual individuals present. The trivial approach of collecting and counting the set of detected Bluetooth MAC addresses is not adequate, in most settings, both in terms of privacy concerns, system scalability and adequacy to devices with limited memory.

In this article we described and benchmarked a set of stochastic summarizing techniques that can be applied to the gate counting problem. By using these techniques our approach ensures the privacy of the users since Gate Counters don't store any extra raw information, i.e, the raw information that they keep at any given moment is also present in the Bluetooth network.

Furthermore, the analysis of these techniques and their trade-offs should help to determine the most adequate solutions for a specific gate counting scenario. We also hope to have motivated the community to the relevant role of stochastic counting techniques in privacy preserving gate counting.

Acknowledgments

This research has received funding from FCT under the Carnegie Mellon - Portugal agreement. Project Wesp (Grant CMU-PT/SE/028/2008).

References

1. Almeida, P.S., Baquero, C., Preguiça, N.M., Hutchison, D.: Scalable bloom filters. Information Processing Letters 101, 255–261 (2007)
2. Bloom, B.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
3. Durand, M., Flajolet, P.: Loglog counting of large cardinalities. Algorithms-ESA 2003 pp. 605–617 (2003)
4. Fan, Y.C., Chen, A.L.P.: Efficient and robust schemes for sensor data aggregation based on linear counting. IEEE Trans. Parallel Distrib. Syst. 21, 1675–1691 (2010)
5. Fan, Y., Chen, A.: Efficient and robust sensor data aggregation using linear counting sketches. IPDPS 2008. IEEE International Symposium on Parallel and Distributed Processing, 2008 pp. 1–12 (2008)
6. Flajolet, P., Nigel Martin, G.: Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences 31(2), 182–209 (1985)
7. Éric Fusy, G, O., Meunier, F.: Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In: In AofA '07: Proceedings of the 2007 International Conference on Analysis of Algorithms (2007)
8. O neill, E., Kostakos, V., Kindberg, T., Schiek, A., Penn, A., Fraser, D., Jones, T.: Instrumenting the city: Developing methods for observing and understanding the digital cityscape. UbiComp 2006: Ubiquitous Computing pp. 315–332 (2006)
9. Whang, K., Vander-Zanden, B., Taylor, H.: A linear-time probabilistic counting algorithm for database applications. ACM Transactions on Database Systems (TODS) 15(2), 229 (1990)