

# Collaborative and privacy-aware sensing for observing urban movement patterns

Nelson Gonçalves<sup>1</sup>, Rui José<sup>2</sup>, and Carlos Baquero<sup>1</sup>

<sup>1</sup> HASLab, INESC Tec & Universidade do Minho, Portugal  
goncalvesnelson@gmail.com, cbm@di.uminho.pt

<sup>2</sup> Centro Algoritmi, Universidade do Minho, Portugal  
rui@dsi.uminho.pt

**Abstract.** The information infrastructure that pervades urban environments represents a major opportunity for collecting information about Human mobility that would be very important across many application domains. However, this huge potential has been undermined by the overwhelming privacy risks that are associated with such forms of large scale sensing. In this research, we are concerned with the problem of how to enable a set of autonomous sensing nodes, e.g. a Bluetooth scanner or a Wi-Fi hotspot, to collaborate in the observation of movement patterns of individuals without compromising their privacy. We describe a novel technique that generates Precedence Filters and allows probabilistic estimations of sequences of visits to monitored locations and we demonstrate how this technique can combine plausible deniability by an individual with valuable information about aggregate movement patterns. The results provide a promising step towards the application of new stochastic techniques in large scale sensing.

**Keywords:** Privacy, Algorithms, Mobility Traces, Bloom Filters, Vector Clocks

## 1 Introduction

The ubiquity of the information technology infrastructure that increasingly pervades urban environments constitutes a major opportunity for sensing Human activity. The large scale collection of such data may give new insights into the dynamics of city life and the digital fingerprint of the urban environment.

Wi-fi and Bluetooth hotspots are particularly interesting for that purpose. Given their widespread presence and their inherent communication with personal devices, they can easily be leveraged as general purpose platforms for massive sensing and actuation in urban spaces. However, this huge potential has been undermined by the overwhelming privacy risks that are associated with such forms of large scale sensing. Given that many detectable devices would be personal devices, their presence at a particular location is a reliable representation of the presence of the respective owner. Consequently, a record of Bluetooth or Wi-Fi sightings holds the potential to become a large scale tracking system capable of detecting the presence, movements and patterns of individuals.

In this research, we are concerned with the problem of how to enable a set of autonomous sensing nodes, e.g. a Bluetooth scanner or a Wi-Fi hotspot, to collaborate in the observation of movement patterns of individuals without compromising their privacy. We aim to understand to what extent it would be possible to use these ubicomp infrastructures to generate valuable information about macro-level movement behaviors while dealing with privacy challenges.

Our approach is based on a stochastic technique that can characterize the sequence of presences in the monitored areas, with a parameterized fidelity that protects user privacy. Our Precedence Filters algorithm combines properties found on counting bloom filters and vector clocks, providing a solid approach to the detection of sequences of presences. It enables us to provide probabilistic answers to questions such as: “*Are the individuals in this art gallery likely to have visited a given art museum first?*” or, in a shopping mall, “*Which shops are visited most likely after the movie theater? and before the theater?*”. The data collection system can be calibrated in a way that each individual (actually its MAC address pseudonym) has a given chance, say 50%, of not having been in a reported location or done a given reported transition, thus protecting its privacy and supporting plausible deniability.

To support the evaluation of the approach, we compare the accuracy of the causality traces estimated by the application of the algorithm with the ground truth corresponding to the causality traces generated from perfect information about node transitions. The evaluation confirms the expectations that a higher fidelity on collective movement patterns can be supported by lower fidelity in the individual traces, thus further improving individual privacy protection.

The contributions of this paper are twofold: we describe a novel technique that shows how causality principles can be applied to support collaborative and privacy-aware sensing of macro-level movement patterns between a set of autonomous nodes; we demonstrate that those techniques can be parameterized in a way that is able to combine plausible deniability by an individual (50% or more false positives) with valuable information about aggregate movement patterns (little more than 10% false positives). The results provide a promising step towards the application of new stochastic techniques in large scale sensing.

## 2 Related Work

**Anonymity** Tang et. al [27] describe a sensing method through which personal devices can become anonymous sensors reporting the number of nearby devices without compromising their own or other people’s privacy. In this case, individuals do not need to be sensed, and this work demonstrates how given a specific sensing goal, it can be possible to devise a technique that limits the collected information to those goals and thus significantly improves the privacy vs utility tradeoff. The use of pseudonyms is perhaps the most obvious way to achieve anonymity when individuals need to be identified in subsequent observations. However, using the same pseudonym for a long time makes it easy for an attacker to gather enough history on an individual to infer its habits or true identity. This is particularly true for spatial data. Previous work has shown how anonymous location traces can easily be re-identified by considering the likely home address of a person [20] or the Home/Work location pair [16].

Montjoye et. al have shown that even coarse datasets provide little anonymity [9]. Based on fifteen months of human mobility data for one and a half million individuals, their study reveals that four spatio-temporal points are enough to uniquely identify 95% of the individuals. To try to mitigate this issue, Beresford and Stajano in [5] proposed an idea which relied upon pseudonym exchange. They introduced two new concepts: *mix zone* and *application zone*. The aim is to conceal information in the *mix zone* so that users can safely

change pseudonyms when getting in and out of these zones. Based on a different concept, *k-anonymity*, Gruteser and Grunwald [17] were the first to investigate anonymity as a method to attain location privacy. According to them, a subject is considered to be *k*-anonymous with regard to location information, if and only if she is indistinguishable from at least  $k - 1$  other subjects with respect to a set of *quasi-identifier* attributes. Bigger values of  $k$  correspond to higher degrees of anonymity. They proposed a middleware architecture and an algorithm capable of adjusting location information resolution in spacial and/or temporal dimensions in order to comply with a specific *k-anonymity* requirement. Geddik and Liu’s *ClickCloak* algorithm [13, 14] allows each user to define her own minimum acceptable value for  $k$  ( $k_{min}$ ), as well as the maximum acceptable values for temporal and spacial resolution.

Mokbel et al. in [26] use the *k-anonymity* concept as well. They presented the Casper framework which consists of two main components, a location anonymizer and a privacy-aware query processor. The location anonymizer blurs the location information about each user according to that user’s defined preferences (minimum area  $A_{min}$  in which she wants to hide and minimum value for  $k$ ). The query processor adjusts the functionality of traditional location-based databases to be privacy-aware. It does so by returning cloaked areas instead of exact points when queried for location information. Unlike the previous *k-anonymity* based approaches that require a centralized trusted server (Anonymizer) in order to compute the cloaking regions, Ghinita et al. in [15] use a decentralized peer-to-peer approach. This fixes two issues inherent to the centralized server approach: Fault Tolerance – the anonymizer is a single point of failure and the system cannot work without it. Security – all requests must go through the anonymizer, so in case it becomes compromised it would result in a serious security threat.

Our technique can be described as a form of anonymity, but because we never register any individual identifier, the technique is not prone to re-identification attacks that could compromise the entire trace of previous locations of an individual.

**Obfuscation** Obfuscation based techniques usually degrade the “quality” of the information in order to provide privacy protection. Even though this may seem comparable to what *k-anonymity* based techniques do, there is a key difference: obfuscation based techniques allow the actual identity of the user to be revealed (thus making it suitable for applications that require authentication or offer some sort of personalization [22]). Duckam and Kulik [10] were the ones who introduced the idea of obfuscation for location privacy. They talk about three distinct types of imperfection that can be present in spatial information:

- Inaccuracy - lack of correspondence between information and reality. E.g. “Paris is in Spain”
- Imprecision - lack of specificity in the information. E.g. “Paris is in Europe”
- Vagueness - special type of imprecision that concerns the existence of indeterminate borderline cases [11]. E.g. “Paris is in Western Europe”. This is a vague statement since there is no clear definition about the its borders.

Any of these types of imperfection can be used to obfuscate an individual’s location. In this particular case, the authors use imprecision to degrade the quality of the location information. They adopt a discrete model for space representation through the use of graphs. Another example of an obfuscation based approach was shown by Ardagna et al. in [3]

and later improved in [2, 4]. Their obfuscation process allows users to express their privacy preferences using another concept of theirs called *relevance*. Relevance is a value in the range  $]0, 1]$  and it represents the relative accuracy loss in location measurement. It tends to 0 when the location measurement is highly inaccurate and it is equal to 1 when the location measurement achieves the best accuracy allowed by the location technique in use. The higher the relevance the lower will be the location privacy. There are other examples of obfuscation approaches that are not only oriented to location data but to the degradation of data in general [1, 18]. In these approaches, privacy is obtained through the generalization of the information. This procedure is done resorting to generalization trees [1] or generalization graphs [18]. Even though there are several techniques that allow reducing the quality of raw location information, we found no approach that allows for the characterization of device sighting sequences with adjustable fidelity.

### 3 System Model

The model in which we base our work assumes the existence of a network of heterogeneous and autonomous nodes that collaborate in the tracking process. While we may consider various types of sensing nodes, for the remainder of this paper we will assume the use of Bluetooth devices. In this case, our sensing node would be some sort of Bluetooth scanner with the ability to discover nearby devices and obtain information about their MAC address, the timestamp of the sighting, the type of Bluetooth protocols supported by the devices, among others.

Our model does not impose restrictions on the type of information collected by the scanners or how it is used by the local node. We do, however, want to limit the information that each node is going to share with the rest of the system to the minimum information possible that is still able to support the detection of movement patterns. We need to be able to detect the same device on different nodes, but for this specific purpose and to minimize the amount of information collected, we can ignore information such as the duration of the sightings, the timestamp in which they occurred, the device name, supported Bluetooth protocols, among others. Our algorithm will only use the device’s MAC address.

In their everyday life and depending on their specific needs, people visit several different places. For instance, a person  $P_1$  wants to buy a new laptop. To do so, she visits store  $S_1$  which does not have the model she wants. She then visits store  $S_2$  which is out of stock and afterwards store  $S_3$  where the price is a little steep. She ends up buying the laptop in store  $S_4$ . To represent this behavior we introduce the concept of *mobility traces*. A mobility trace is simply the representation of the places visited in the order by which they were visited. In this specific case, the mobility trace of  $P_1$  is  $MT_{P_1} = \{S_1, S_2, S_3, S_4\}$ . Our mechanism, *Precedence Filters*, allows the recording of information relative to the individual traces of people, in a manner compatible with plausible deniability. That information can later be processed/mined to obtain more accurate data about the habits of the aggregate of all individuals. For instance, in this example, the order in which the stores were visited might be an indicator of their reputation/popularity.

Whenever a device is sensed, the sighting node records that event locally. This information is then used in the computation of device transitions between the system’s multiple nodes. The place where that computation occurs depends on the system’s architecture. Our

mechanism can be deployed in either centralized or decentralized architectures. In a centralized system configuration, the computation has to be done in the server since only it has enough information to do so. Each node only shares its local information with the server. On the other hand, with a decentralized architecture, nodes can do the processing locally. The local information each node possesses is shared with the other nodes, thus allowing all the nodes to have access to the data. Both models have advantages and disadvantages. For instance, the centralized approach is not fault tolerant, if the server crashes the tracking system stops working. This does not happen in the decentralized approach given that the same information is stored in multiple nodes (redundancy). The system can keep on working even if some of the nodes fail. Compared to the centralized version, the decentralized model also has greater availability as a result of the information redundancy. However, as a consequence of the exchange of information between all the nodes, the decentralized scenario has a bigger burden on the network when compared to the centralized one.

In order to achieve the goal we set ourselves, and taking into account the constraints presented by our model, our solution is based upon the following set of assumptions:

- Even though we cannot make assumptions about how each individual node will handle the observed Bluetooth addresses, our solution should never require the Bluetooth address or any other information that could uniquely identify individuals to ever leave the sensing node.
- No system element should, at any given time, have all the information necessary to accurately determine the path of a single individual.
- The aggregate result that a node can create about the set of all visiting devices should be accurate enough to be useful in human mobility observation scenarios, e.g. most common paths, similarity level between places.
- There are no communication failures in the system and the exchange of information between any two nodes is faster than the time it takes for a person to move between them. This ensures that the order in which the sightings are recorded is correct, i.e., when a person goes from node  $A$  to node  $B$ , node  $B$  must already have the information that she was in  $A$ .

## 4 Precedence Filter Algorithm

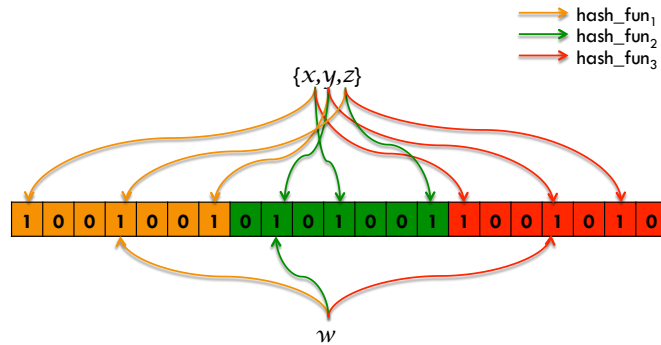
This section describes the behavior of Precedence Filters. However, to do so, we must first do a brief overview of the techniques in which they are based, namely Counting Bloom Filters [12, 25] and Vector Clocks [19, 24].

### 4.1 Bloom Filters

Bloom Filters (BFs) were created in 1970 [6] by Burton Howard Bloom. They are a simple and space efficient data structure for set representation where membership queries are allowed. Bloom Filters allow false positives but do not allow false negatives, i.e, when querying a Bloom Filter about the existence of an element in a given set, if the answer is no, then the element is definitely not in the set, but if the answer is yes, the element might be in the set.

A Bloom Filter for representing a set of  $n$  items  $S = \{x_1, x_2, x_3, \dots, x_n\}$  is traditionally implemented using an array of  $M$  bits, all initially set to 0. Then,  $k$  independent hash

functions are used  $\{h_1, h_2, \dots, h_k\}$ , each one mapping the element of the set into a random number uniformly distributed over the range  $\{1, \dots, M\}$ . For each element  $x$  of the set ( $x \in S$ ) the bits of the positions  $h_i(x)$  are all set to 1 for  $1 \leq i \leq k$ . A location can be set to 1 multiple times. Due to the independence of the hash functions, nothing prevents collisions in the outputs. In extreme cases it is possible to have  $h_1(x) = h_2(x) = \dots = h_k(x)$ . To prevent this, we use the variant of Bloom Filters presented in [7] which partitions the  $M$  bits among the  $k$  hash functions, creating  $k$  slices of  $m = M/k$  bits. This ensures that each item added to the filter is always described by  $k$  bits. Given a Bloom Filter  $BF_S$ , checking if an element  $z \in BF_S$ , consists in verifying whether all  $h_i(z)$  are set to 1. If they aren't, then  $z$  is definitely not present on the filter. Otherwise, if all the bits are set to 1, then it is assumed that  $z$  belongs to  $BF_S$  although that assumption might be wrong. This false positive probability exists because the tested indices might have been set by the insertion of other elements. Figure 1 illustrates such an example.



**Fig. 1.** Bloom Filter with  $k = 3$ ,  $M = 21$  and  $m = 7$ , containing elements  $\{x, y, z\}$ . Querying for the presence of element  $w$  yields a false positive.

The false positive probability  $P$  can be obtained using equation (1), where  $p$  is the ratio between the number of set bits in the slice and the slice size  $m$ . The fill ratio  $p$  can be obtained through equation (2).

$$P = (1 - p)^k \quad (1)$$

$$p = 1 - \left(1 - \frac{1}{m}\right)^n \quad (2)$$

Furthermore, given a maximum false positive probability  $P$ , and the number  $n$  of distinct elements to store, equations (3) and (4) can be used to estimate the optimal number of bits required by a Bloom Filter to store those  $n$  elements,  $M = m * k$ .

$$k = \log_2 \left( \frac{1}{P} \right) \quad (3)$$

$$m = \frac{n * |\ln P|}{k * (\ln 2)^2} \quad (4)$$

**Counting Bloom Filters** Counting Bloom Filters (CBFs) were presented in [12] although they were only given this name later in [25]. They were introduced in order to allow the deletion of elements from the Bloom Filters. Suppose that we have a set that changes over time, where elements are inserted and deleted. The insertion of elements can be easily done using a normal Bloom Filter, hash the element  $k$  times and set the bits to 1. Unfortunately the delete operation cannot be accomplished simply by reversing the process. The resulting positions from the hash functions cannot be set to 0, because each position may be hashed by some other element from the set.

In a Counting Bloom Filter, each position is a small counter rather than a single bit. When an item is inserted the corresponding counters are incremented, and when the item is deleted the same counters are decremented. We just have to choose sufficiently large counters, in order to avoid counter overflow.

## 4.2 Vector Clocks

Broadly speaking, a distributed system consists in a set of processes connected through a network. Each of these processes' internal state can only change through the occurrence of events. There are three kinds of events: message sending, message reception and state transition events deriving from the normal execution of the process. In a distributed system, processes can only communicate using message passing as there is no shared memory. In this context and considering the ordering of events in an asynchronous model (since no assumptions should be made about timing), several strategies to cope with this asynchronicity were devised. One of the most prominent is vector clocks.

In order to better understand how Vector Clocks work, we must first comprehend the concept of causality. Causality is a relation through which we can connect two events, a first event (known as the *cause*) and a second one (the *effect*).

In the context of Distributed Systems, causality is expressed using the *happens-before* relation [21] denoted by the  $\rightarrow$  symbol. For instance, given 2 events,  $x \rightarrow y$ , reads as  $x$  happened-before  $y$ , and means  $y$  might be a consequence of  $x$ .

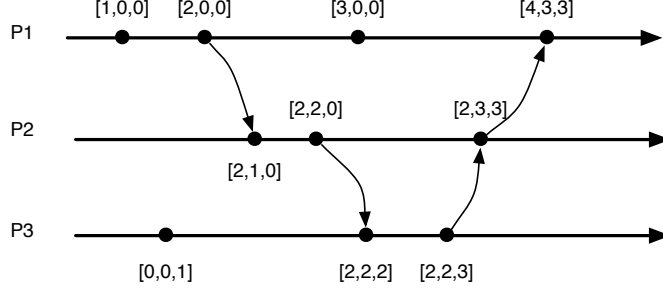
The *happens-before* relation has the following properties over any given event:

- $\forall a, b, c$  if  $a \rightarrow b$  and  $b \rightarrow c$ , then  $a \rightarrow c$  (transitivity);
- $\forall a, a \not\rightarrow a$  (irreflexivity)
- $\forall a, b$  if  $a \rightarrow b$  then  $b \not\rightarrow a$  (antisymmetry)

Vector Clocks were introduced by Colin Fidge [19] and Friedemann Mattern [24] in 1988 and are a practical implementation of the *happens-before* concept. In this algorithm, each process  $P_i$  has a vector of integer values  $VC_i[1..n]$  where  $n$  is the number of processes, maintained by the following set of rules:

1. In the beginning, all the positions from the vector are set to 0
2. Each time the state of a process  $P_i$  changes (send, receive or internal event), it must increment the value  $VC_i[i]$ , i.e,  $(VC_i[i] = VC_i[i] + 1)$ .
3. Each time a process  $P_i$  sends a message, its vector  $VC_i$  should be enclosed in that message.
4. When a process  $P_i$  receives a message  $m$ , it must update its vector using the formula:  $\forall x : VC_i[x] = \max(VC_i[x], m.VC[x])$ , where  $m.VC$  symbolizes the vector clock attached to  $m$ .

Figure 2 shows a concrete example of the above rules in a distributed system with 3 processes.



**Fig. 2.** Vector Clocks

Vector Clocks are able to accurately represent the causality relation and the partial order it defines. Given any two distinct events  $x$  and  $y$ :

$$\forall(x, y) : (x \rightarrow y) \iff (VC_x < VC_y)$$

Where  $VC_x < VC_y$  stands for:

$$(\forall k : VC_x[k] \leq VC_y[k] \wedge (\exists k : VC_x[k] < VC_y[k]))$$

An important property of this mechanism is the result by Charron-Bost [8] proving that Vector Clocks are the most concise characterization of causality among process events.

### 4.3 Precedence Filters

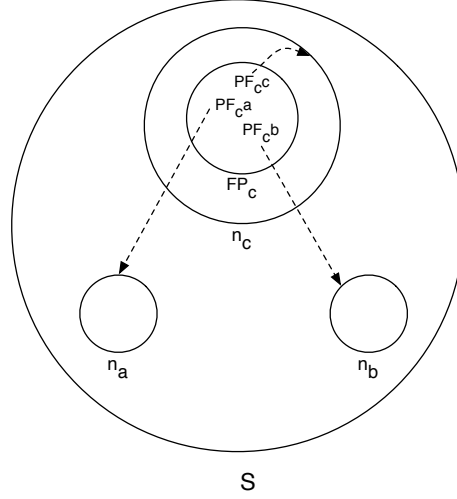
By applying some of the previously mentioned general constructs of distributed systems to the mobility sensing scenario, Bluetooth scanners can be treated as processes and device sightings as state transition events. Precedence Filters are based upon this idea and provide accurate mobility information, at a macroscopic level, without neglecting individual privacy. Precedence Filters can be seen as a vector clock [19, 24] implementation, whose difference is the use of Counting Bloom Filters [12, 25] (one for each node in the system) by the PFs instead of integers (one per process) used by vector clocks.

With that in mind, Precedence Filters work as follows: supposing we have a set of Bluetooth scanners (nodes)  $S$ , each node  $n \in S$  has a Precedence Filter  $PF_n$ . That PF is in turn composed of a map of *Counting Bloom Filters*, one for each node  $z \in S$ . We use notation  $PF_n^z$  to refer to the CBF for scanner  $z$  belonging to  $PF_n$ , as depicted in Figure 3.

All CBFs are initially set to 0, use the same set of hash functions  $K$  and have the same size  $M = m * k$  ( $k$  is calculated with equation 3 and  $m$  is calculated with equation 4). This ensures that the same device is correctly identified across the several nodes (upon detection it will be mapped to the same indices). Precedence Filters can also be seen as a matrix where the number of rows is equal to the number of nodes in the system and the number of columns is equal to  $M$ .

Each time a node  $n$  detects a device  $d$ , its Precedence Filter  $PF_n$  is updated according to the following set of rules:





**Fig. 3.** Diagram of relevant entities and data structures

1. Using the set of hash functions  $K$ , the node  $n$  calculates the set of indices  $I_d$ .  $I_d$  consists on the output from the  $K$  hash functions regarding device  $d$ ,  $I_d = \bigcup_{f \in K} f(d)$ .
2. Node  $n$  sends the set of indices  $I_d$  to all other nodes in  $S$ .
3. Each one of the  $z$  nodes belonging to  $Z$  ( $Z = S \setminus \{n\}$ ) replies with a set of tuples  $R_z^{I_d}$ .  $R_z$  contains the previously required  $I_d$  along with the set of values that each of the CBFs belonging to  $FP_z$  had stored in those indices,  $R_z^{I_d} = \{(i, PF_z^z[i]) \mid \forall i \in I_d\}$ .
4. Upon the reception of the replies from the other nodes, node  $n$  updates its own indices  $I_d$  on the CBFs relative to the other nodes with the maximum value received,  $PF_n^z[i] = \max(R_z^{I_d}[i]), \forall z \in Z, \forall i \in I_d$ , where

$$R_z^{I_d}[i] = v \rightarrow (i, v) \in R_z^{I_d}$$

5. Lastly,  $n$  updates the indices  $I_d$  on its own CBF ( $PF_n^n$ ). For each index  $i \in I_d$ ,  $PF_n^n[i] = \max(PF_n^s[i]) + 1, \forall s \in S$ . By adding 1 to the maximum value stored in the other nodes, the current node “dominates” them in the operation that returns the causality between the visited places. In other words, this is the key to obtaining the order in which the places were visited.

This set of rules allows the precedence filters to record information about the precedence of the locals visited by a device. Given a set of indices  $I_d$  for device  $d$  and any pair of scanners  $x$  and  $y$ , we say that the sighting of  $d$  in  $x$  precedes the one in  $y$ ,  $x \rightsquigarrow y$  if:

$$x \rightsquigarrow y \iff PF_x[I_d] < PF_y[I_d]$$

Where  $PF_x[I_d] < PF_y[I_d]$  stands for:

$$\forall i \in I_d : PF_x^x[i] < PF_y^y[i]$$

Mobility traces, used in our model to describe the behavior of individuals, characterize a total order between the places visited. This means that it is always possible to establish an order between any two places in the mobility trace. However, being based upon the

happens-before relation [21], Precedence Filters represent partial orders. In this particular case, for each of the nodes/locations, they can only “remember” the last time each device was sighted in a given place. For instance, given the mobility trace

$$MT_P = \{S_1, S_2, S_1, S_3, S_2, S_4, S_1\}$$

where scanners  $S_1$  and  $S_2$  are visited more than once, in the best case scenario PFs can obtain  $CT_P = \{S_3, S_2, S_4, S_1\}$ , which we will refer as a *causality trace*. This is a consequence of the irreflexivity and antisymmetry properties from the happens-before relation. However, we can look at this as a feature of Precedence Filters, a sort of automatic data degradation. It ensures that the length of the record of sightings for any given device has an upper bound equal to the number of Bluetooth scanners in the tracking system.

The level of privacy offered by Precedence Filters can be further customized by adjusting the CBFs’ false positive ratio (Equation 1). The higher the ratio, the greater the inaccuracy of the PFs. The occurrence of false positives in the CBFs results in the appearance of *fictitious transitions*, i.e., the causal trace obtained from querying the filters, contains transitions which are non-existent in the original trace. Both these properties are what allow individual users to plausibly deny the fidelity of the data extracted from the Precedence Filters.

Ignoring constant and logarithmic factors, the communication and space storage scalability of the technique is dominated by the number of scanners  $S$  and the expected number of devices to monitor,  $D$ . Each scanner needs to store state that is linear with the number of devices  $O(D)$ . One should note however, that due to lossy compression only a fraction of the bit size of a MAC address is needed, with smaller fractions for higher privacy (and lower fidelity). As a whole the system stores state  $O(SD)$ , and this would be the server state for a centralized setup. Each time a device is sighted at a given scanner, its network link will have a communication load of  $O(S)$  and induce  $O(1)$  communication in other scanner links, as it collects logarithmic information on a constant number of positions  $k$  at all other scanners. Thus, the maximum communication load induced per sighting is linear on the number of scanning sites.

## 5 Metrics and Data Sets

To assess the estimation quality of Precedence Filters we compared the set of transitions obtained from querying the Precedence Filters with the set of transitions obtained from the causality traces (baseline), which were themselves obtained from mobility traces. For instance, given the mobility trace  $MT_P = \{S_1, S_2, S_2, S_1, S_3\}$ , we calculate its causality trace according to the happens-before relation (that only contains last sighting in each place),  $CT_P = \{S_2, S_1, S_3\}$ . Then we extract the set of transitions, denoted as  $\mathcal{T}$ , from that causality trace,  $\mathcal{T}(CT_p) = \{(S_2, S_1), (S_2, S_3), (S_1, S_3)\}$ . Each transition is a two location tuple where the first location causally precedes the second. In our scenario, that means the device was seen in the first location before being sighted at the second location. This set of transitions is then finally compared to a similar set of transitions obtained from the PFs,  $\mathcal{T}(PF)$ .

**Metrics** To support the evaluation of Precedence Filters, we used two different metrics. The *individual* metric which measures the false probability of statements like the following

– “individual  $X$  visited location  $S_1$  before visiting location  $S_2$ ”. For each user  $u$ , this is done by calculating the cardinality of the difference between the transitions belonging to the causality trace ( $\mathcal{T}(CT_u)$ ) and the transitions extracted from the Precedence Filter ( $\mathcal{T}(PF_u)$ ), according to Equation 5.

$$\frac{\#((\mathcal{T}(CT_u) \cup \mathcal{T}(PF_u)) \setminus (\mathcal{T}(CT_u) \cap \mathcal{T}(PF_u)))}{\#(\mathcal{T}(PF_u))} \quad (5)$$

The individual metric calculates the relative amount of incorrect information (information that is on the CTs and not on the PFs and *vice-versa*) returned by the Precedence Filters. However, given the assumption that the exchange of information between nodes is faster than people, our system never forgets information, i.e.,  $\mathcal{T}(CT) \subseteq \mathcal{T}(PF)$ . Therefore, Equation 5 can be simplified, resulting in Equation 6.

$$\frac{\#(\mathcal{T}(PF_u) \setminus \mathcal{T}(CT_u))}{\#(\mathcal{T}(PF_u))} \quad (6)$$

The *global* metric quantifies the inaccuracy of information regarding the relative weight of specific transitions. This enables us to establish the relative importance of each type of transition, i.e., to know the inherent error in statements like - “2% of the transitions are from Restaurant Y to Cafe Z”. Assuming that,  $U$  represents the universe of all users,  $\mathcal{A}_{PF} = \biguplus_{u \in U} \mathcal{T}(PF_u)$  and  $\mathcal{A}_{CT} = \biguplus_{u \in U} \mathcal{T}(CT_u)$  are respectively the multiset union of all transitions in the Precedence Filters and in the Causality Traces and that  $\mathcal{A}[t]$  is multiset composed only of the  $t$  transitions in  $\mathcal{A}$ , the global metric for each transition  $t \in \mathcal{A}$  is calculated according to equation 7. For each transition, we calculate the absolute difference between its relative weight in the Precedence Filter and its relative weight in the actual Causality Traces. Then we divide that number by its weight on the Precedence Filters. This gives us the relative error of the relative weight of the transition.

$$\frac{\left| \frac{\#\mathcal{A}_{PF}[t]}{\#\mathcal{A}_{PF}} - \frac{\#\mathcal{A}_{CT}[t]}{\#\mathcal{A}_{CT}} \right|}{\#\mathcal{A}_{PF}[t]} \quad (7)$$

**Real Data set** To evaluate the PFs’ performance we used a real data set with information about Bluetooth sightings by static nodes. This data set was taken from Leguay et al.’s work [23]. To collect this information, the authors handed out a set of Bluetooth enabled devices called *iMotes* to a group of users who carried them in their day-to-day. Additionally, the authors installed Bluetooth scanners in several places with the purpose of registering the sightings of *iMotes*. The dataset contains 18 static nodes and 9244 distinct device IDs, 6439 of which have been sighted only once and were therefore removed. This leaves us with 2805 devices, whose average mobility trace size is approximately 4 and maximum size is 11. Figure 4(a) shows the distribution of total and distinct sightings for all scanners. As expected, not all places have the same popularity, some are more visited than others, thus the bigger number of Bluetooth sightings.

**Synthetic Data Set** Still in the context of evaluating the PF’s performance, we built a synthetic trace generator. Our motivation came from the need to simulate scenarios with arbitrary number of locations and users.

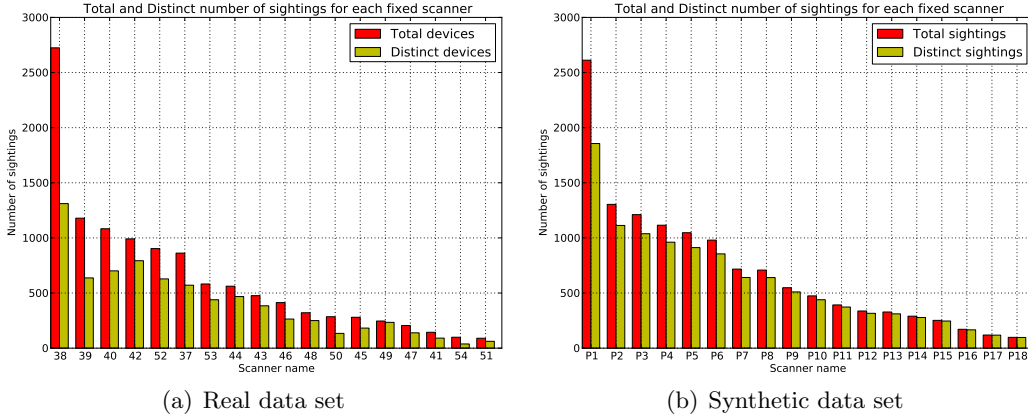


Fig. 4. Number of total and distinct device sightings across the scanners

In a first approach, we tried fitting the statistic distribution of the real data set using a negative exponential distribution. This would have allowed us to choose an arbitrary number of sensors, users and trace length. However, after evaluation with Pearson’s Chi-Square test, it turned out to be a bad fitting. This might be solvable by switching to a more complex distribution function. Instead we chose to work with the empiric distribution. On this second approach, it was decided to use the same number of sensors as the real data set, i.e. 18. This allowed us to simulate the popularity of each sensor/place using the number of sightings from the real data set as weights. The larger the number of sightings at a sensor, the bigger its weight is, and the more likely it is to be chosen. Each node is defined by two parameters, unique sightings and total sightings. We only made use of the latter. The use of *replication*<sup>3</sup> enabled us to create a simpler and less error prone simulator, capable of producing as many users as we want, as well as mobility traces with arbitrary length. The downside of not using the unique number of sightings is that we are assuming that even though places have different weights, they are the same for everyone, i.e. everyone has the same probability of choosing a given place, everyone is an “average” person.

Figure 4(b) shows the synthetic distribution obtained using the approach mentioned above. As expected the results are very similar but not a perfect match. There is a correlation between the number of total and unique sightings which stems from the use of the “average” person model. Also, the curve from the synthetic data set is smoother, it does not suffer from the “noise” inherent to raw real data.

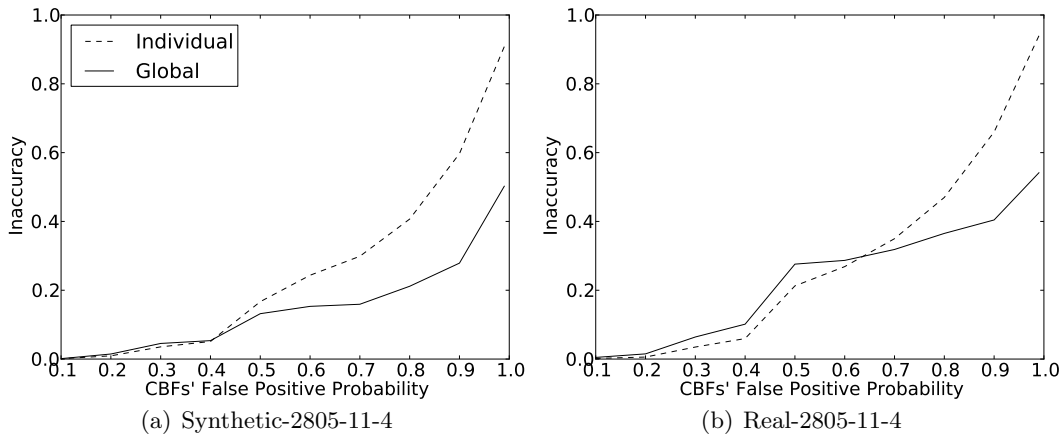
## 6 Evaluation

Using the metrics and data sets previously mentioned, we tested the Precedence Filter’s inaccuracy across several scenarios, varying both the number of devices and the length of the mobility traces. Furthermore, each of the scenarios was tested with multiple different settings for the Counting Bloom Filter’s maximum false positive probability (see Equation 1). A good performance is reflected through high inaccuracy values for individual information together with low values for global inaccuracy.

<sup>3</sup> In statistics, replication is the repetition of an experiment or observation in the same or similar conditions.

As can be seen in Figures 5, 6, 7 and 8, by increasing the false positive probability of CBFs, inaccuracy increases as well. Inaccuracy is manifested via the occurrence of fake (visible on PFs only) transitions, i.e., *fictitious transitions*. As the false probability increases, so does the percentage of fake transitions. This is easily explainable. In Bloom Filters, false positives denote elements wrongfully considered as belonging to the set. Given that in PFs Bloom Filters are used to record device sightings, the occurrence of false positives generates fake device sightings, which in turn give origin to fictitious transitions.

As previously explained, both data sets use 18 scanning nodes, what differs is the number of devices and the length of the mobility traces. To describe the parameters of each scenario, the following notation is used in the captions: Synthetic/Real-[*number of devices*]-[*maximum trace length*]-[*average trace length*].

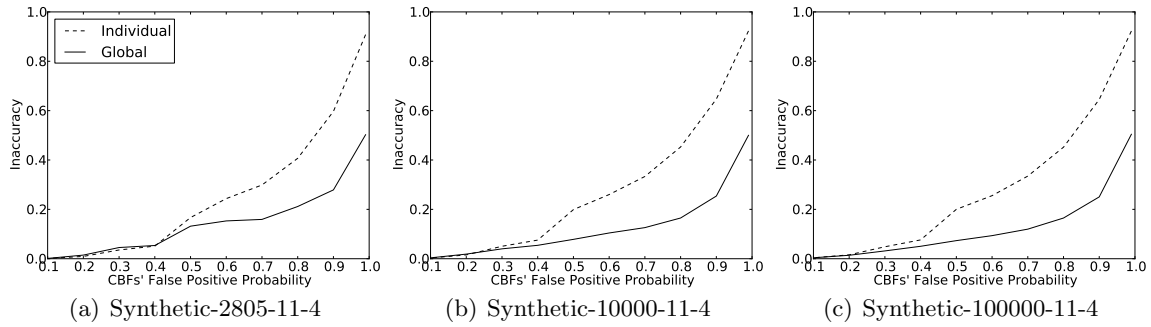


**Fig. 5.** Comparison between real and synthetic data set, with same parameters

Figure 5 shows the comparison between a synthetic data set and the real data set. This synthetic data set will serve as the baseline for all tests because it simulates both the number of devices and trace length found in the real data set. Our technique performs better with data from the synthetic data set (Figure 5(a)) than it does with data from the real one (Figure 5(b)). This is a consequence of the average person simplification we did for the synthetic data sets. Figure 4 shows that while the number of total sightings is approximately equal in both data sets, the number of unique sightings is usually bigger in synthetic data set. This means that the real data set has a greater number of repeated sightings by user, which in turn means that the average length of the causality traces is smaller, i.e., even if both data sets have the same number of users and similar sized mobility traces, the causality traces in the real data set are smaller, explaining the worse performance of our technique.

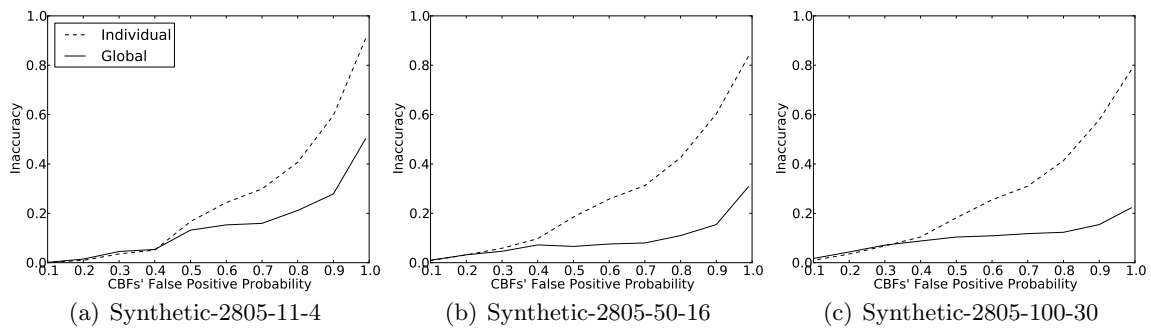
For both these data sets, there is a point where the individual accuracy is greater than the global one, however, when the individual inaccuracy is approximately 50%, the global inaccuracy is higher than what is desirable. This is a result of the low number of individuals and small mobility trace sizes of both data sets, as supported by Figures 6, 7 and 8.

Keeping the length of mobility traces constant, Figure 6 shows that Precedence Filter's global inaccuracy drops by increasing the number of devices to  $10^4$ , and then again, although



**Fig. 6.** Synthetic data sets with increasing number of devices

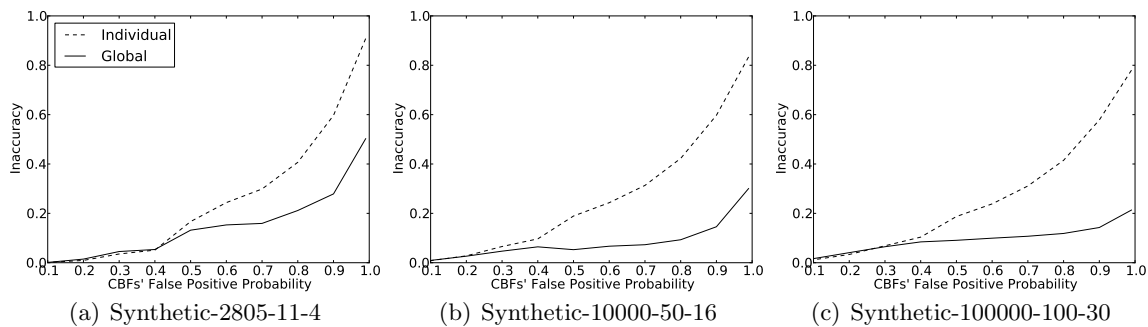
little, by increasing that number to  $10^5$ . In both scenarios, for a false positive probability of 0.8, PFs provide global inaccuracy below 20% while ensuring that in average, 50% of the information about any given individual is incorrect.



**Fig. 7.** Synthetic data sets with increasing trace sizes

Keeping the same number of devices, while increasing the length of the mobility traces also improves the Precedence Filter's global accuracy, as depicted in Figure 7. For instance, Figure 7(c) shows that by increasing the average and maximum values for mobility traces respectively to 30 and 100, our technique offers a global error of about 15%, while providing 50% of inaccuracy for individual information. Increasing both the number of devices and the length of the mobility traces yields the better results, which is not surprising given both the previous results.

In a scenario with  $10^5$  users whose maximum and average mobility trace sizes are respectively 100 and 30, depicted in Figure b8(c), our technique has a global inaccuracy a little higher than 10% while providing an individual inaccuracy of 50%. This means that, on average, half of the information about individual transitions is wrong, which we consider to be a value compatible with plausible deniability. To recap, our technique registers information user transitions. Each user has a set of transitions, and the individual metric measures, in average, the percentage of those which are fictitious/false. The global metric, on the other hand, returns the average error regarding the information about the popularity of specific transitions/paths. Increasing the false positive probability of CBFs increases the



**Fig. 8.** Synthetic data sets with increasing number of devices and trace sizes

total number of occurrences of transitions, which is why the individual accuracy drops; yet the relative weight of each transition remains more stable, thus the behavior of the global inaccuracy lines.

## 7 Conclusion

Vast amounts of collective movement patterns could be harnessed and put to good use in city planning, ad placement, and traffic congestion prevention. The limitations are no longer of a technological nature and suitable infrastructures are often in place that could allow mass sensing. Our stance is that privacy is a key limiting factor in this area.

We have presented a technique that provides Precedence Filters. Through the aggregation of probabilistic information about sequences of visits to monitored locations, this technique is able to reveal information about the relative frequency of transitions. In practice, frequent transitions can lead to optimizations in transportation systems, discount policies in businesses and museums, and many other potential applications.

To evaluate the technique we first had to define and propose new metrics to analyze trace estimation quality. Evaluation was based on a robust trace driven simulation, from a real mobility dataset, complemented with simulations over longer synthetic traces that allowed a comprehensive analysis of the long term properties, for higher numbers of users and longer traces.

The resulting technique is highly adjustable in the degree of privacy and degradation of fidelity that is required in each potential setting. An important overall property is the ability to give good quality collective traces from lower quality individual traces. This brings the best of both worlds, high individual privacy and good collective statistics.

## References

1. Anciaux, N., Bouganim, L., Van Heerde, H., Pucheral, P., Apers, P.: Instantdb: enforcing timely degradation of sensitive data. In: Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. pp. 1373–1375. IEEE (2008)
2. Ardagna, C., Cremonini, M., Damiani, E., De Capitani di Vimercati, S., Samarati, P.: Location privacy protection through obfuscation-based techniques. *Data and Applications Security XXI* pp. 47–60 (2007)
3. Ardagna, C.A., Cremonini, M., De Capitani di Vimercati, S., Samarati, P.: A middleware architecture for integrating privacy preferences and location accuracy. *New Approaches for Security, Privacy and Trust in Complex Environments* pp. 313–324 (2007)

4. Ardagna, C.A., Cremonini, M., De Capitani di Vimercati, S., Samarati, P.: An obfuscation-based approach for protecting location privacy. *Dependable and Secure Computing, IEEE Transactions on* 8(1), 13–27 (2011)
5. Beresford, A., Stajano, F.: Location privacy in pervasive computing. *Pervasive Computing, IEEE* 2(1), 46 – 55 (jan-mar 2003)
6. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* 13(7), 422–426 (Jul 1970), <http://portal.acm.org/citation.cfm?doid=362686.362692>
7. Chang, F., Chang, F., chang Feng, W.: Approximate caches for packet classification. In: *IEEE INFOCOM*. pp. 2196–2207 (2004)
8. Charron-Bost, B.: Concerning the size of logical clocks in distributed systems. *Information Processing Letters* 39(1), 11–16 (1991)
9. De Montjoye, Y.A., Hidalgo, C.A., Verleysen, M., Blondel, V.D.: Unique in the Crowd: The privacy bounds of human mobility. *Scientific Reports* 3 (2013), <http://www.nature.com/srep/2013/130325/srep01376/full/srep01376.html>
10. Duckham, M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. *Pervasive Computing* pp. 243–251 (2005)
11. Duckham, M., Mason, K., Stell, J., Worboys, M.: A formal approach to imperfection in geographic information. *Computers, Environment and Urban Systems* 25(1), 89–103 (2001)
12. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: A scalable wide-area web cache sharing protocol. In: *IEEE/ACM Transactions on Networking*. pp. 254–265 (1998)
13. Gedik, B., Liu, L.: Location privacy in mobile systems: A personalized anonymization model. In: *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*. pp. 620–629. *IEEE* (2005)
14. Gedik, B., Liu, L.: Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *Mobile Computing, IEEE Transactions on* 7(1), 1–18 (2008)
15. Ghinita, G., Kalnis, P., Skiadopoulos, S.: Prive: anonymous location-based queries in distributed mobile systems. In: *Proceedings of the 16th international conference on World Wide Web*. pp. 371–380. *ACM* (2007)
16. Golle, P., Partridge, K.: On the Anonymity of Home/Work Location Pairs. *Pervasive Computing* 5538, 390–397 (2009), [http://dx.doi.org/10.1007/978-3-642-01516-8\\_26](http://dx.doi.org/10.1007/978-3-642-01516-8_26)
17. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: *Proceedings of the 1st international conference on Mobile systems, applications and services*. pp. 31–42. *ACM* (2003)
18. Heerde, H., Ancaux, N.: Data degradation to enhance privacy for the ambient intelligence (2006)
19. J.Fidge, C.: Timestamps in message-passing systems that preserve the partial ordering. *Australian Computer Science Communications Vol.10* (1988)
20. Krumm, J.: Inference Attacks on Location Tracks. *Pervasive Computing* 10(Pervasive), 127–143 (2007), <http://www.springerlink.com/index/TG64551RW2716103.pdf>
21. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21, 558–565 (July 1978), <http://doi.acm.org/10.1145/359545.359563>
22. Langheinrich, M.: Privacy by design—principles of privacy-aware ubiquitous systems. In: *UbiComp 2001: Ubiquitous Computing*. pp. 273–291. *Springer* (2001)
23. Leguay, J., Lindgren, A., Scott, J., Friedman, T., Crowcroft, J.: Opportunistic content distribution in an urban setting. In: *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*. pp. 205–212. *ACM* (2006)
24. Mattern, F.: Virtual time and global states of distributed systems. *Workshop on Parallel and Distributed Algorithms* (1988)
25. Mitzenmacher, M.: Compressed bloom filters. *IEEE/ACM Trans. Netw.* 10, 604–612 (October 2002), <http://dx.doi.org/10.1109/TNET.2002.803864>
26. Mokbel, M., Chow, C., Aref, W.: The new casper: query processing for location services without compromising privacy. In: *Proceedings of the 32nd international conference on Very large data bases*. pp. 763–774. *VLDB Endowment* (2006)
27. Tang, K.P., Keyani, P., Fogarty, J., Hong, J.I.: Putting people in their place: An anonymous and privacy-sensitive approach to collecting sensed data in location-based applications. In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. vol. 8, pp. 93–102. *ACM* (2006), <http://portal.acm.org/citation.cfm?id=1124772.1124788>